REPUBLIC OF CAMEROON

Peace - Work - Fatherland

UNIVERSITY OF YAOUNDE I

Faculty of Sciences

POSTGRADUATE SCHOOL FOR SCIENCE, TECHNOLOGY AND GEOSCIENCE

RESEARCH AND POSTGRADUATE TRAINING UNIT FOR MATHEMATICS, COMPUTER SCIENCE, BIOINFORMATICS AND APPLICATIONS

REPUBLIQUE DU CAMEROUN

Paix - Travail - Patrie

UNIVERSITÉ DE YAOUNDÉ I

Faculté des Sciences

CENTRE DE RECHERCHE ET DE FORMATION DOCTORALE EN SCIENCES, TECHNOLOGIES ET GÉOSCIENCES

UNITÉ DE RECHERCHE ET DE FORMATION DOCTORALE MATHÉMATIQUE, INFORMATIQUE, BIO-INFORMATIQUE ET APPLICATIONS

LABORATORY OF COMPUTER SCIENCE AND APPLICATIONS

LABORATOIRE D'INFORMATIQUE ET APPLICATIONS

Distributed approach driven Dynamic Service Composition in a System Scalabity

Thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy in Computer Science
Option: Information Systems

Presented by:
BESSALA BESSALA Célestin Parfait
Registration Number: **01U249**

Thesis Director:
ATSA ETOUNDI Roger
Professor

Academic Year : 2021/2022

REPUBLIQUE DU CAMEROUN
Paix – Travail – Patrie
- . - . - . -
UNIVERSITÉ DE YAOUNDÉ I
Faculté des Sciences
Département d'Informatique
B.P. 812 Yaoundé

REPUBLIC OF CAMEROON
Peace – Work – Fatherland
- . - . - . -
UNIVERSITY OF YAOUNDÉ I
Faculty of sciences
Department of Computer Science
P.O.Box 812 Yaoundé

Unité de Recherche et de Formation Doctorale Science, Technologie et Géoscience
Laboratoire d'Informatique
DÉPARTEMENT D'INFORMATIQUE
DEPARTMENT OF COMPUTER SCIENCE
ATTESTATION DE CORRECTION DU MÉMOIRE DE THESE PhD EN INFORMATIQUE

Nous soussignés, KOLYANG DINA Taiwe, **Président de jury,**

TSOPZE Norbert **Membre examinateur**, du Jury de la soutenance de THESE PhD en Informatique tenu le 25 janvier 2024 de l'étudiant **BESSALA BESSALA Célestin Parfait**, Matricule **01U249,** sous le thème **« Distributed approach driven Dynamic Service Composition in a System Scalability »**,

attestons que le mémoire de thèse a été corrigé conformément aux observations et recommandations comme l'atteste le tableau ci-dessous :

Observations et recommandations du jury	**Travail effectué par le candidat**
Mettre à jour la liste protocolaire	Une nouvelle liste mise à jour été a intégrée (au début du document)
Réduire le résumé	le résumé a été réduit à une page (page i, ii)
Corriger les fautes et les coquilles	Le document a été relu et toiletté de ses coquilles (pages 38, 50,75, 110-113, 116,136)
Corriger les formules de la complexité	Les formules présentant la complexité ont été ajustées (pages 103,104, 117)
Bien présenter les références bibliographiques	Les références ont été bien présentées (page 170, 171,)
Améliorer la description des figures	L'amélioration de la description des figures a été observées (pages 39, 118,119)

En foi de quoi le présent document lui et établi et délivré pour servir et valoir ce que de droit.

Président du jury **Examinateur**

KOLYANG DINA Taiwe, Pr

TSOPZE Norbert, MC

Distributed Approach driven Dynamic Service Composition in a System Scalability

BESSALA BESSALA Célestin Parfait 01U249

Director: Pr ATSA ETOUNDI Roger

8 avril 2024

Dedication

To my dear Father **BESSALA OKALA Célestin**...

To my dear Mother **NKE ODILE** ...

To all those who contributed but did not see it ...

Acknowledgments

My sincere gratitude goes to Prof. ATSA ETOUNDI Roger, Coordinator of Computer Science Laboratory, for heartily welcoming me into his team and for supervising this work. "Grand Prof", all my gratitude for the many discussions and fruitful exchanges I have had with you throughout this work. You have supervised my Master's work and now, thanks to you, I can see the end of this long course which is the thesis.

I am very grateful to Prof. TCHANA Alain, Professor at the Ecole Normale Supérieure de Lyon in France for his great availability, support and constant encouragement.

I would like to thank Prof. SOSSO Aurélien, Rector of the University of Yaoundé I for allowing me to continue my research work.

My thanks also go to all the teachers of the Department of Computer Science in the Faculty of Sciences of the University of Yaoundé I, for the training and the teachings they gave me.

My thanks also go to the managers of the INPT-ENSEEIHT (National Polytechnic Institute of Toulouse) in France and in particular the researchers of the Institut de Recherche Informatique de Toulouse (IRIT) for the warm welcome they have always reserved tome during my research trips and seminars to which I was invited throughout this work. I particularly thank Prof. HAGIMONT Daniel, Dr. TEABE Boris, Dr. EKANE Brice for their availability, support and encouragement.

I thank all the members of the Computer Science Laboratory coordinated by Prof. ATSA ETOUNDI Roger more particularly Dr. ABESSOLO, Dr. MOUAFFO Laetitia for the important exchanges maintained during the multiple seminars.

I would like to thank especially Mr. BENE NTOUDA Sylvain, Mr Ndima Cyril and Mr. NGUELE Serge, who read this thesis again and did me a great favor to put it in the better lines.

I am so thankful to my wife Marcelle BESSALA BESSALA who has continuously supported me and for the understanding and patience she has shown throughout this work.

I hope everyone who contributed directly or indirectly to the realization of this work may find here my gratitude and may not feel frustrated for any forgetting.

Finally, I want to thank all my beloved ones and my friends for the shared moments, as well as my family, without whom I would be nothing.

Abstract

Dynamic Composition of Services (DSC) consists of providing at the runtime of a request (at each request from an end user) a new complex service by combining or composing a set of atomic services. It is a process of complex steps (service discovery, service selection, composition plan generation and selection) which poses difficulties when the data set becomes large. Scalability in the dynamic composition of services has therefore become a major challenge for its adoption in the real world. The research lists three causes of scalability: (i) the number of requests, (ii) the complexity of requests, and (iii) the number of services. Several works have made it possible to optimize the dynamic composition process to manage scalability mainly by making each of its steps efficient. Our main concern is therefore to address the scalability problem in dynamic service composition based on its three known sources.

This preoccupation leads us to evaluate the impact of current solutions on the three sources of known compositions. And subsequently take into account the neglected cause of scalabity with a distributed approach.

The modeling of an e-government based on this model was the experimental phase which made it possible to verify its adaptation to the environment of a developing country like Cameroon, faced with infrastructural difficulties and the energy crisis. .

Keywords: SOA, scalability, dynamic service composition, distributed system, e-government

Résumé

La Composition Dynamique de Services (CDS) consiste à fournir au moment de l'exécution d'une requête (à chaque demande d'un utilisateur final) un nouveau service complexe en combinant ou en composant un ensemble de services atomiques. C'est un processus d'étapes complexes (recherche de service, sélection de service, génération de plan de composition et sélection) qui pose des difficutés lorsque la taille des données devient importante. La scalabilité dans la composition dynamique des services est donc devenue un défi majeur pour son adoption dans le monde réel. Les recherches énumèrent trois causes de scalabilité : (i) le nombre de requêtes, (ii) la complexité des requêtes et (iii) le nombre de services. Plusieurs travaux ont permis d'optimiser le processus de composition dynamique pour gérer la scalabilité principalement en rendant performante chacune de ses étapes.

Notre principale préoccupation est donc de traiter le problème de scalabilité dans la composition dynamique de service sur la base de ses trois sources connues Cette préoccupation, nous a conduits à évaluer l'impact des solutions actuelles sur les trois sources de compositions connues. Et par la suite prendre en compte la source de scalabité negligée en appliquant une approche distribuée ?

La modélisation d'un e-government basé sur ce modèle a été la phase expérimentale qui a permis de vérifier son adaptation à l'environnement d'un pays en voie de développement comme le Cameroun, confrontés à des difficultés infrastructurelles et à la crise énergétique.

Mots clés : SOA, scalabilité des systèmes, composition dynamique des services, système distribué, e-government

Contents

List of Figures

List of Tables

Part I

GENERAL INTRODUCTION

A man of honor is one who has dreamed of honors; a man of money is one who has dreamed of money; a fool is one who sleeps to dream.

C.BESSALA OKALA

1 Introduction

Contents

1.1 Introduction to research

Between 2014 and 2017, people's access to ICT has improved considerably, from 0.356 to 0.476 in Cameroon(ANTIC, 2017). In the same period, the IDI (ICT development index) increased significantly from 2.03 to 3.85(ANTIC, 2017). This index had a positive impact on the use of ICT over the same period, from 0.37 to 3.58. This use of ICTs and the internet paved the way for Cameroon to advance its e-government project with the support of the Republic of South Korea(**?**).

It should be noted this kind of project involves a significant number of actors. This can lead to a proliferation of computer systems that need to cooperate. Also, one of the major challenges of this cooperation between different IT systems is to facilitate their interoperability. This requires thinking and facilitating since the design, on the architectural means to make automatic transactions and data exchange between the different computer platforms despite their multi-aspect heterogeneity . In this perspective, service-oriented architectures (SOA) can provide an effective response to this concern.

In fact, the SOA as architectural model of the information systems guarantees the autonomy of the cooperating actors and facilitates the development of the systems in a technical and philosophical independence. The research has already proved its effectiveness in environments with a high level of cooperation, particularly in the implementation of e-government.

By proposing to set up an e-government by the SOA in Cameroon, this study aims to present the different advantages of this approach in an evolving, multi-stakeholder and unstable environment like those of the developing countries. Moreover, as an important advantage of SOA, the automatic and dynamic service composition is the main subject of this study. In this direction, this work proposes a new approach of automatic and dynamic service composition able to deal with scalability issue, which is often an obstacle to its adoption despite its relevance in evolving environments.

1.2 Research context

This part will allow the reader who does not follow constantly the situation of computer development projects in Cameroon to learn and especially to get an idea of the state of implementation of its e-government project.

Cameroon has adopted a Strategy Document for Growth and Jobs (SDGJ) in 2010. On the basis of this plan, the Ministry of Posts and Telecommunications adopted in 2005 a sectoral strategy document in the field of telecom-

munications and ICT(**?**). This led reforms in the legislative and regulatory framework have followed. We can mention mainly the adoption of the laws relating to cybersecurity and cybercriminality, the laws relating to electronical transactions and e-commerce(ANTIC, 2017).

In 2014, the sectoral strategy for the telecommunications and ICT was updated to aligning it with current technological changes. Also, to promote the creation of wealth through private initiatives including startups, a strategic plan for the development of digital economy "Digital Cameroon 2020", was developed and adopted by the government in 2016(**?**).

In order to meet the objectives of this Strategic Plan, the government has initiated projects for the strengthening of high and ultra high speed telecommunications infrastructure with the construction of landing points for submarine fiber optic cables:

— SAT3 (South Africa Transit 3) in Douala with a capacity of 280Gbps put into service on February 18, 2002;

— WACS (West Africa Submarine Cable system) in Limbé with 2x140 Gbps capacity commissioned July 1, 2015

— NCSCS (Nigeria – Cameroon Submarine Cable System), extension of Main One to Kribi with a capacity of 40 Gbps put into service on January 25, 2016

Two more cables are planned:

— ACE, whose will enable Cameroon to benefit from 48.9 Gbps capacity as soon as construction is completed;

— SAIL (South Atlantic Inter Link) planned to land in Kribi, a linear of nearly 6,000 km across the Atlantic Ocean to directly connect Cameroon and Brazil.

The strategic plan also aims to extend the national fiber-optic backbone to allow national terrestrial fiber-optic transmission of approximately 12,000 kilometers between the 10/10 Regions, 52/58 Departments and 209/360 Districts. It also launched the implementation of the NBN (National Broadband Network) program, and the construction of urban optical loops as well as the construction of IXPs (Internet Exchange Points) built in 2016 in Yaoundé and Douala, to enable the conservation of national traffic at the local level without using equipment outside national borders.

In addition to the infrastructure, there are other projects that reveal the permanent concern to have effective e-government. We can mention among others:

— drawing up a master plan for the development of e-government with the technical and financial support of KOICA (Korea International Cooperation Agency). Here, the goal is to have a global plan of realization in the next five years of an e-government;

— the design, development and hosting of an e-government web portal for the information of the e-government committee;

— design and development of a government web portal in Cameroon with technical and financial assistance from the World Bank to centralize all services and administrative procedures available at the government web portal;

— the establishment of a government intranet permitting fiber optic interconnection of all the central and deconcentrated services of the Public Administrations.

Others innovative projects include:

— a feasibility study with technical and financial support from the IUT for the establishment of a national digital library;

— A series of measures taken at the level of the NAICT (National Agency of ICT) as well as important investments made in order to make the ".cm" viable and credible, making it possible to increase from 650 domain names in 2009 to 63,059 in 2016;

— implementation of tools allowing a better knowledge and enhancement of Cameroon's tourist and cultural heritage;

— the acquisition of a telemedicine site at the Yaoundé University Hospital hosting the telemedicine platform, acquired as part of the pan-African online services project;

— the establishment of a distance learning platform, through the Virtual University of the Central Africa subregion, located at the University of Yaoundé I and the National Virtual University at the Yaounde ENSPT, as part of the pan-African online services project;

— the payment of online tuition fees through different money transfer platforms belonging to mobile operators.

All these projects and initiatives seem to sufficiently reveal the option of electronic services of public and private organizations. Their cooperation and the transactions they may wish to make should impose on them a service-oriented architectural model. And in view of the dynamism of the technological environment, the dynamic and automatic composition will be presented as the outcome by which it should pass.

1.3 Research problem background

Collaboration between organizations has become a major challenge dictating their performance. Meaning; an organization's capability to produce, manage, process and analyze information quickly and efficiently collaboratively with its stakeholders gives a competition. Even for public administration, the rapid interactions between computer systems of various departments reduce delays in processing files and brings an effective mean to fight against the inertia(West, 2004)(Peña-López et al., 2012). This requirement has imposed the establishment of information systems dealing with heterogeneity, interoperability and ever changing requirements; and the advent of a new paradigm of computer development called Service Oriented Architecture (SOA)(Pulparambil and Baghdadi, 2019).

Service composition has thus become a fundamental area in service-oriented modeling as it solves complex problems by combining basic services available to satisfy an initial purpose. It is therefore one of the most active areas of research on web services(Blake et al., 2010) because it is complex and involves several activities such as discovery, sorting and selection and execution of services.

Internet, standards and web technologies have significantly contributed to the rise of SOA. The Internet has thus positioned itself beyond a mere vector of data exchange, that is to say, into a more constructive platform for the transport and exchange of self-describing, modular, easily integrable components weakly coupled called services(Papazoglou, 2003). Web services have thus given organizations the possibility to open technically to others in order to materialize their different lines of collaboration. They have also made application development faster by reusing services and reducing the time to develop new applications. Early on, as individual service were limited in their capabilities, putting together a set of services proved to be essential for creating more complex services. We are talking about the composition of services.

Service composition has thus become a fundamental area in service-oriented modeling as it solves complex problems by combining basic services available to satisfy an initial purpose. It is therefore one of the most active areas of research on web services(Blake et al., 2010) because it is complex and involves several activities such as discovery, sorting and selection and execution of services.

The service composition can be static or dynamic depending on when it is performed. If the sequence is described at the time of the coding of the application, one speaks of static composition but if this scheduling is carried out during the execution, one speaks about dynamic composition. The composition can also be automatic when performed by a machine. It can be semi automatic if it is performed by a machine assisted by the human interven-

tion: It can also be manual if it entirely relies on a human action. Given its inefficiency mainly due to the permanent changes in technological environments, manual and static service composition are given way to dynamic and automatic service composition.

Also, the integration of cloud computing that promotes outsourcing of IT resources in organizations; in SOA has created a new concept that visualizes the composition of services as a whole *service*. Then, the proposed Composition as a Service (CaaS) approach attempts to combine collaborative software engineering principles with latest innovations in service-oriented computing(Blake et al., 2010). This practice has proven its efficiency in such a way that it is currently promoted in other domains such as Function as a Service (FaaS). But the latter has caused many concerns. Among these, scalability has become a major problem(Baryannis and Plexousakis, 2010; Yu et al., 2008). In fact, the dynamic composition methods that provide the responses related to the dynamics of the environments are yet to effectively answer the problem of the large number of available data. For example, in the case of CaaS, these large and available number of data are due to the ever growing number of services and the exponential number of system users(Yu et al., 2008). Thus, in one hand, the discovery, the selection of a service to be part of a chain of the execution of several others become complex operations. And in the other hand, the number of requests received by the composition server that provides the composition plans is also a factor that can deteriorate his performance(Yu et al., 2008). All these aspects increase the response time of the system and can have a negative impact on the processing of customer requests(Baryannis and Plexousakis, 2010).

It is therefore in this context that this work is set to provide a solution to scalability issue to reduce the request processing time during the process.

1.4 Problem statement

The scalability problem is still exploring because it delays the implementation and adoption of automatic and dynamic service composition as standard(Baryannis and Plexousakis, 2010)(Lécué et al., 2008a). There are many recent and repeat recommendations that large number of data is the main cause of scalability(Yu et al., 2008)(Baryannis and Plexousakis, 2010)(Rostami et al., 2014). Researchers argue that there are three causes of scalability in a DSC namely: (i) the large number of service composition requests, (ii) the complexity of the request and (iii) the large number of services which compose the service repository(Baryannis and Plexousakis, 2010).

The current approach for addressing this challenge consists in performing separately each stages of the composition process: user's request analysis, service discovery, services selection and composition(Rao and Su, 2005; Kopecký et al., 2007; Baryannis and Plexousakis, 2010; Blake et al., 2010; Medjahed and Atif, 2007). But put together their results didn't make easy to adopt DSC at the industrial level by resolving the scalability issue(Baryannis and Plexousakis, 2010).

Therefore, to perform the dynamic service composition, it seems important to make an holistic analysis of the problem. Specifically, it seems important to propose a solutions related the three known sources of scalability in DSC.

1.5 Research relevant to information systems

As an application area, E-government is a multidisciplinary issue of interest to managers, economists and computer scientists(Lofstedt, 2012a). Scholl(Scholl, 2004) argues that the complex relationship between information technology and e-government has become a major focus of academic research in several fields such as public administration, organizational behavior, information science, and technology innovation. That is why researchers who have chosen e-government as an application area might have their theoretical starting points in several other disciplines like organization theory, social science, informatics, computer science, public administration, business administration, economy, political science, law, government professionals, library science etc(Lofstedt, 2012a). Thus the starting point of this work concerns service oriented architecture which is a branch of software design which is also a part of information system in computer science.

1.6 Research goal

The goal of this work is to define a new approach to deal with the scalability issue in automatic and dynamic service composition. This will lead us to improve the implementation of e-government based on SOA in Cameroon. In order to achieve this goal, the following objectives are constructed:

— To study and evaluate the impact of the current solutions on the three known causes of scalability;

— To propose a new approach based on the above study, which can deal with the scalability issue in DSC;

— To propose an e-government based on SOA to improve the effectiveness of cameroonian administration.

1.7 Main research question

The primary research question of this research can be formalized as follow:
How can we deal with the scalability issue in the dynamic service composition process based on its 3 known causes?
This question can be turned into secondary questions.

1.8 Secondary research questions

To ensure that the primary question is answered, the following secondary questions were addressed:

1. Are the three factors causing the scalability taken into account in the current solutions to deal with it?
2. How can a new approach deal with scalability problem in dynamic service composition by taking into account the negleted cause?
3. How Cameroon can implement his adapted e-government based on service oriented architecture?

1.9 Research justification

This work can be justified in its thematic application and in its scientific issue.

In the developing countries, the debate on the relevance and contribution of ICT in the development process compared to the basic social infrastructure has found a better balance in the views (Osterwalder, 2003). Indeed, ICT do not solve the basic social problems arisen but is rather a tool that can facilitate the access, the availability and the proper functioning of the services responsible to provide the answers to these priority needs of populations in these areas. Some therefore believe rightly that, it is an instrument to facilitate the achievement of the MDGs(of Science and Technology, 2006) and a better tool for good governance (Bertot et al., 2010).

One of the best ways to make visible the social and economic impact of ICTs is to make use of them in the different approaches to public governance of the states. Electronic government called e-government or E-Gov(Relyea,

2002), is since many years seen as an important aspect in accelerating the growth in developing countries by creating a lot of opportunities among which the cost reduction and efficiency gains, the quality of service delivery to businesses and customers, the transparency, the anti-corruption, the accountability, the increase of the capacity of government, the network and community creation, the improvement of the quality of decision making, and they promote the use of ICT in other sectors of the society(Ndou, 2004).

But different applications and platforms that cover the overall range of the e-government implementation area need to interoperate in order to provide integrated governmental services to the citizens and businesses(Yan and Guo, 2010)(Peristeras et al., 2009). So researchers have proposed in the area of enhancing e-government interoperability to use common models and/or ontologies (Peristeras et al., 2009). And others propose the use of Service Oriented Architecture to potentially address more those needs and provide a modern application architecture for the interaction of existing and new distributed systems(Yan and Guo, 2010).At the same way, to overcome lack of interoperability's situation, researchers propose the use of a designed semantic platform to easy public administration to cooperate and expand the accessibility of services in a broad sense(Sabucedo and Anido-Rifón, 2006). The authors argue that Service oriented architectures, methodologies and tools, together with the conceptual and empirical framework of web services have a high potential to assist public administrations in ongoing e-government innovation processes(Marchese, 2003).

Thus, in view of the particular context of developing countries(Ndou, 2004)(Yan and Guo, 2010), the SOA-oriented e-government solution must be able not only to improve existing methods of collaboration between public administrations information systems, but also to provide an appropriate response to the environments of these special countries. The problems of scalability that show down the adoption of dynamic and automatic service composition as a model of composition(Baryannis and Plexousakis, 2010) must above all find an effective response.

1.10 Research contribution

This study contributes in two folds to the body of knowledge:

1.10.1 Practical

This study contributes practically as an analytical guide for the implementation of service-oriented information systems. The study indicates the structural approach to implement a service-oriented e-government that can support scalability by providing multiple levels of composition. The local level that makes it possible to take advantage of the activities of the remote server, and the external level which makes it possible to entrust the tasks of compositions to a dedicated server. This architecture is perfectly suited to the structural organization of public administrations that respect a certain hierarchy between the systems and especially allows to build an autonomy in time that generates fault tolerance.

1.10.2 Theoretical

The study contributes to:

— a global analysis of the scalability problem of dynamic composition of services on the basis of the three causes indicates by researches:

  Our solution presents scalability as a consequence of the actions of an entire system that it is necessary to resolve by involving all the end-users of the system;

— a formalization of the problem:

  This formalization allows us to better appreciate the problem and the objective of the scientific approach. Our formalization presents the results of the static composition model as the desired goal of the optimization of dynamic composition;

— a proposition of a distributed and participatory approach using end-users' infrastructures:

  Based on the rules of the distributed systems, the study therefore proposes way to decentralize this process of services composition by considering the composition path as the shared files, the composition server as a main tool and the clients as the local servers able to refer to themselves even before referring to the server dedicated.

The automatic and dynamic service composition process is as a centralized client-server shared files system where the composition path is a shared file. Thus, this study presents the distributed approach as an effective mean to deal with scalability problem.

1.11 Thesis structure

This work is organized in 3 parts. The first part which is the General Introduction of the thesis is composed of 2 chapters. This introduction (Chapter 1) and the chapter 2, where we present the methodology of our research and the philosophical means used to deal with the scalability in the automatic and dynamic service composition.

The second part is the state of the art and composed of 2 chapters too: chapter 3 and chapter 4. The chapter 3 presents an overview on service-oriented architectures. This chapter also presents the place and purpose of automatic and dynamic service composition in SOA. In chapter 4, the focus will be on methods and algorithms for dynamic composition of services and their limits when scaling up. Existing and planned solutions are explored to provide effective responses to this concern.

The part third part of this work is the Research implementation. He is composed of 3 chapters: chapter 5, chapter 6 and chapter 7. Chapter 5 presents the principles and concepts of distributed approach in which the contribution is based on. Chapter 6 presents the distributed approach to deal with the scalability in automatic and dynamic service composition. The model is exposed with its benefits and its conceptual organization. In chapter 7, we present the application of the emanating framework and the results of its assessments. The technical environment of the work is also presented and simulations results in case study of e-government based SOA in developing countries as Cameroon are also evaluated.

The last part of our work is a General conclusion and is composed of one chapter 8 which is the conclusion. In this chapter, we analyse the advances that this new solution brings before coming up with logical perspectives that can help to refine this research. Before that, the contribution's points are summarized and some limits of our work are exposed.

All this can be summarized by figure 1.1:

The following chapter is focused to present the methodology of our research and the philosophical means used to deal with the scalability in the automatic and dynamic service composition.

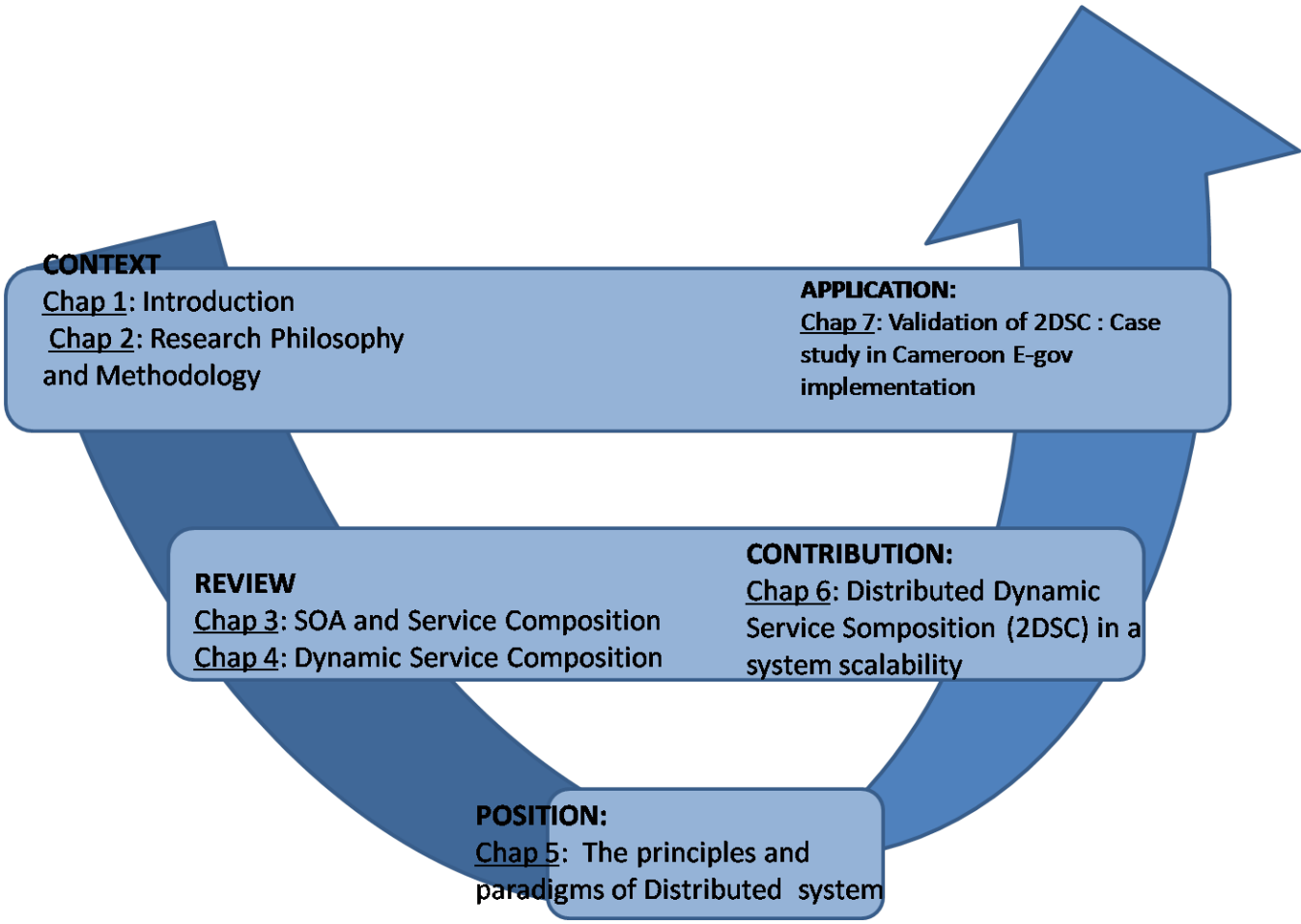

Figure 1.1 – Thesis blue print

If we want everything to continue, everything must first change

G. TOMASI DI LAMPEDUSA

2

Research philosophy and methodology

Contents

2.1 Introduction

The term research is often linked to academic activity, however several writers link research to everyday life and see it as a fundamental activity of everyday living so they suggest that everyone is engaged in the research process through attempting to find solutions to problems which are perceived (Johnston, 2014). The research can have several objectives. It can allow an author to explore a phenomenon, solve a problem, question or refute results provided in previous work. It can also be experimenting with a new process, a new solution or a new theory. Research can also include applying a practice to a phenomenon, describing it, or simply explaining it. Scientific research is therefore a systematic and dynamic process or a rigorous rational approach that allows the construction of new knowledge (Quinlan, 2011).

The generation of new knowledge or its extension in the context of academic research must be integrated and reliable for their validity (McGregor and Murnane, 2010). This credibility is acquired through the reliability of the author's approach. Scientific rigor is guided by the notion of objectivity that demonstrates that the researcher only deals with facts, within a framework defined by the scientific community. Academic research work must therefore be based on a philosophy and methodology of research. To define the research philosophy, most of researchers use the term paradigm which is a set of assertions, concepts, values and practices that constitute the path to shape a reality (Scotland, 2012). A paradigm consists of the following components: ontology, epistemology, methodology, and methods (Scotland, 2012). Philosophically, a paradigm aims to outline the basics assumptions and beliefs about and technically, it permits to outline the methods and techniques used to conduct the study (McGregor and Murnane, 2010). That is why beyond the research philosophy, the research methodology of this study will also be presented in this part.

Simply put, research methodology refers to how each of logic, reality, values and what counts as knowledge inform research (McGregor and Murnane, 2010). According to Crotty(Crotty, 1998) methodology is the strategy or plan of action which lies behind the choice and use of particular methods. Thus, methodology is concerned with why, what, from where, when and how data is collected and analyzed (Scotland, 2012). Guba and Lincon explain that methodology asks the question: how can the researcher go about finding out whatever they believe can be known(**?**). Saunders and al. consider research methodology as the principle of how research will be undertaken including the theoretical and philosophical assumptions at which the study is grounded and the implications of techniques adopted (Saunders, 2011).

Thus, guided by research onion 2.1 as proposed by Saunders (Saunders, 2011), this part will present the philosophical and methodological research choices of the study. This includes the research design for the study, the research philosophy, the approach, the research strategy, choices, the time horizons and techniques and procedures.

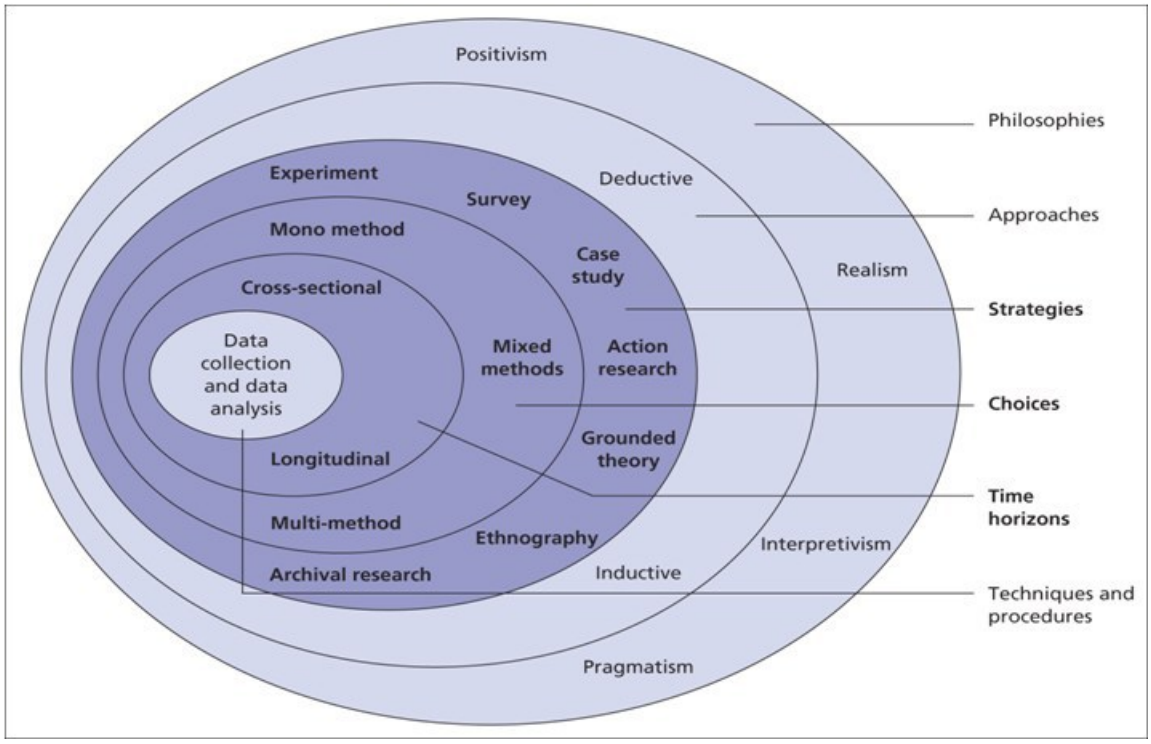

Figure 2.1 – Research onion (Saunders, 2011).

2.2 Research process

Scientific research is a process. It has a definite purpose, the steps, with a beginning and an end. According to Saunders and al., it is a set of procedures, methods and steps followed by the researcher to conduct the research (Saunders, 2011). This work followed the research process as presented by Saunders and al. (Saunders, 2011). It begins with a presentation of the literature review on service-oriented architectures, the resulting process of service composition. An extensive literature review on the automatic and dynamic service compo-

sition is followed to better present the problem of scalability that results and especially the factors that generate it.

This chapter presents the technical and theoretical details on which the study is conducted. The research design is the next section.

2.3 Research design

A research design is a plan that specifies how the researcher plans to carry out a research project and how he expects to use its evidence to answer their research question. In fact, when building a house, it is necessary to define the type of house and its use before drawing up a plan and other necessary administrative procedures. And even in a research process, you have to design and structure the process before collecting the data, for example, and analyzing them.

According to De Vaus and al., research design can be seen as the logical task adopted to assist the researcher in collecting evidence that could enable him to answer the research question (De Vaus, 2001). Flannely and al. explained that the main aim of the research design is to reduce the likelihood of incorrect conclusions being drawn from the data by the researcher as a result of relevant data not being collected (Flannelly and Jankowski, 2014). Thus, the research model of this study is based on research onion 2.1 presented by Saunders and al (Saunders, 2011).

2.4 Research philosophy

The research onion describes 4 main paradigms: positivism, realism, interpretism and pragmatism (Saunders, 2011).

2.4.1 Positivism

The positivist approach concludes that the world is external and therefore should be measured by objective methods (Johnston, 2014). Positivism comes up with the research questions and hypotheses that you can test. With positivism, you can find the explanations measuring the accepted knowledge of the world. It is the type of body of research that other researchers can also take to find the same outcome. Here, you give importance to get the quantitative results. It may lend you to statistical analysis.

2.4.2 Realism

Realism says that you can revise every theory. It further explains that you can't find the reality without continuous research. Realism says that you don't need to hesitate in using new methods of research. This way realism allows you to use many types of research methods. It will help you to come up with a reliable outcome. Realism and positivism are quite similar. It says that social reality and you are not dependent on each other. It will not let them give biased results. However, realism says that scientific methods are improper. It separates the realism and positivism.

2.4.3 Interpretivism

Interpretivism helps you in interpreting how people participate in the social and cultural life. In other words, you can learn what people understand about their own and others actions. It can help you in understand a culture. Furthermore, you can also learn about the cultural existence and change through learning about the ideas and valuables. Some meanings may also prove helpful for you in the same.

2.4.4 Pragmatism

Pragmatism: Constructivism and Objectivism are the ideal ways to conduct a research in the views of pragmatism. A topic can also judge from one or both viewpoints about the impact of the social actors. You can use these views to create a practical approach to research. It is essential in the research onion. You can use it to come up with the solution of the problems.

This study is based on positivism. This philosophy fits better with the projected strategy. In fact, this strategy consists in studying the mechanism of automatic and dynamic service composition in Service Oriented Architectures, evaluating the sources of scalability in this process, studying the existing solutions to deal with the problem and highlight their limits before proposing an approach that will be evaluated on a statistical basis.

2.5 Research approach

After the discovering of research aim and limitations it is important to choice an approach. Deductive and inductive are the two approaches that the second layer of the research onion includes (Saunders, 2011).

2.5.1 The deductive method

The deductive method aims to find the answer to the question at the start of the research. In the deduction process, the theories are scanned to the research question. It further leads to gather data and ultimately the confirmation or rejection of the question. The revision of the theory can be done.

2.5.2 The inductive method

Inductive method aims to create a new theory. So, this point in the research onion works in the opposite way of the deductive method. It means researcher doesn't need to give a thought to an existing theory. The hierarchy of the research runs from research question to observation. Next come the observation, description, analysis and the researcher comes up with her own theory in the end. Hence, it is better to use inductive method for the research if the research requirements are not big.

The study follows a deductive approach. Because there is a research question at the start of the research and there is a review of literature already done in the field. This study will therefore build on this previous work to address the problem of scalability on the automatic and dynamic service composition by investigating the causes of their lack of efficiency, before propose a new distributed approach.

2.6 Research strategy

A research style is used to gather and analyze data like grounded theory are the choices. Every choice has its benefits and limitations. Thus, it is a must for a researcher to give a thorough thought to each of it. Furthermore, he should explain and balance the choice well in the work. The researchers have the option to choose more than one beyond the following strategies.

2.6.1 Experiment

Experiment: the experimental designs can be find very scientific and complex in their structure. It may make it tough for others to replicate your research. Experimental designs test the casual effects of the phenomena on a group of people. It comprises the group of people who are not under the effects of the phenomena. The independent variables can be called on the dependent variable as causal effects. Moreover, experimental strategies can give you such data that you can analyze statistically.

2.6.2 Survey

Survey strategy of the research onion is often linked with the deductive approach. It is one of the finest and economical research strategies. It allows the researcher to gather huge data to answers the who, what, where, when and how of your research. Rich and reliable data can be collected through this method.

2.6.3 Case study

A case study design helps researcher in doing a study on one or more people. This can use the same to do research on real life cases. Researcher should gather many types of data while studying a case: take a look at the peoples' behavior, consider the settings, interview the people and search the records.

2.6.4 Action research

Action research is the method in the research onion that tries to find and solve a problem or an issue. For example, an organization makes the inquirer part of it if they ask him to do a research for them. In other words, him and the organization work in collaboration on the topic. The following process helps to do things: have an objective, find the diagnosis of the issue and make the list of the actions to deal with the problems.

2.6.5 Grounded theory

The grounded theory builds a theory after predicting and explaining the behavior with the use of the inductive methods. In this method, researcher collects the data through observation. Next, he makes predictions and theories with the use of this data. Finally, he tests the predictions. Although this theory comes up with new theories, yet is it grounded. It is because of the existing theory and literature on the topic.

2.6.6 Ethnography

Ethnography can be found on the roots of the anthropology. Anthropology is the study that allows you to study others in a detached way. But, to research with the ethnography method, researcher has to stay in the community or situation. He may find this research method time-consuming. It is because things can take time to change or get in her head.

2.6.7 Archival

Archival research goes as the name suggests. In other words, researchers do research with the use of the archive documents and existing information. Archival research allows to explore and explain the changes happening over a long span of time. The researchers can also do a descriptive analysis of it but, it is possible that the information may have some fault. It may lead to not reaching anywhere.

This study is based on deductive approach. So the survey strategy of the research onion is more adapted to lead our research. In addition, the research question is open and there is a large literature review on which this study is conducted.

2.7 Research choice

In this layer, researchers discover the ways to use quantitative and qualitative methods for the research purposes. Here, they ensure whether both methods are ideal to use or one is fine. Furthermore, the also decide whether use one method more than other. Or the frequency will stay equal. In quantitative research, researchers consider the quantity and measurements. Whereas qualitative research allows them to explore personal accounts, descriptions and opinions. You can use three types of methods according to the fourth layer of the research onion diagram: mono-method, Mixed-methods and Multi-methods.

2.7.1 Mono-method

In the mono-method, the inquirers gather only one type of information from qualitative and quantitative. They may have to do it due to the demands of the philosophy. Philosophical choices and used strategies may also demand it.

2.7.2 Mixed-methods

Mixed-methods allow to use both qualitative and quantitative methods for the study purposes. It is possible when researcher combines both these methods, to offset their limits by finding and filling the gaps in the information easily.

2.7.3 Multi-methods

Multi-methods helps in researching with the use of the qualitative as well as quantitative information. However, researcher focus stays on the one source only. This way he analyzes both types of data, but with the same point of view.

This study is a quantitative research because it will consider the quantitative measurements of execution time of requests which are coming from the clients. The mono-method will be his way to analyze the effectiveness of the new approach.

2.8 Time horizon

Layer five of the research onion has two-time horizon methods. First is cross-sectional that can be used to conduct a short time study. Another one is longitudinal that can be used while doing a long-term study.

2.8.1 Cross-sectional

Cross-sectional: qualitative and quantitative research can be used in the cross-sectional method. It may help researcher in observing the behavior of a group of people or an aspect. He can also use the same method to do study on an individual at one point of time.

2.8.2 Longitudinal

Longitudinal method also allows to use qualitative and quantitative research methods. But, this method is used to study behavior and events with focused samples over a longer time.

This study use cross-sectional time horizon methods. The framework generated by the new approach will be observed and its performance evaluated during the study.

2.9 Data collection and analysis techniques

Layer six of the research onion gives a fine idea of the practicalities of data collection and analysis. In this stage, the researchers have to make a decision of which data may prove best for their studies. Moreover, they also have to discover the analysis to use to find the desired results. In this section of the

research onion, inquirers decide the questionnaire content and sample groups. They also give a thought to the questions you will ask in the interviews and many others things.

The study will be done on a dataset of services. A service will be mainly a resource characterized by its input and output. A composition plan will therefore be a path that can exist between services. The composability of two services is essentially the possibility of matching them. The processing time of a request is therefore the time taken by the composition server to return a composition plan to a client having sent him a request clearly showing its input and the output which is the desired result.

2.10 Synthesis

Research philosophy and methodology is a good way to design academic studies. It is a must to make sure that all the decisions and tools used sync with the objectives of the research. The researchers must keep the same thing in mind for the philosophical stances, strategies, choices, time-horizons, data collection and analysis. It may help you reach the valid results. We can summarize the research philosophy and methodology of this research as follows.

2.10.1 Research philosophy

This research is based on positivism. This philosophy fits better with the projected strategy. In fact, this strategy consists in studying the mechanism of automatic and dynamic service composition in Service Oriented Architectures, evaluating the sources of scalability in this process, studying the existing solutions to deal with the problem and highlight their limits before proposing an approach that will be evaluated on a statistical basis.

2.10.2 Research approach

Our research follows a deductive approach. Because there is a research question at the start of the research and there is a review of literature already done in the field. This study will therefore based on this previous work to address the problem of scalability on the automatic and dynamic service composition by investigating the sources of their lack of efficiency, before proposing a new distributed approach.

2.10.3 Research strategy

This research is based on deductive approach. So the survey strategy of the research onion is more adapted to lead our research. In addition, the research question is open and there is a large literature review on which this study is conducted.

2.10.4 Research choice

Our research is a quantitative research because it will consider the quantitative measurements of execution time of requests which are coming from the clients. The mono-method will be our way to analyse the effectiveness of the new approach.

2.10.5 Time horizon

This study uses cross-sectional time horizon methods. The framework generated by the new approach will be observed and its performance will be evaluated during the experimentations.

2.10.6 Data collection and analysis techniques

The research will be done on a dataset of services. A service will be mainly a resource characterised by its input and output. A composition plan will therefore be a path that can exist between services. The composability of two services is essentially the possibility of matching them. The processing time of a request is therefore the time taken by the composition server to return a composition plan to a client which has sent a request with the input and the output which is the desired result.

The following figure 2.2 summarises the research philosophy and methodology of this research.

The following part is focused on the state of the art. He will make the the presentation of Service-Oriented Architecture (SOA) and the current solutions proposed to deal with the problem of scalability during the dynamic service composition process.

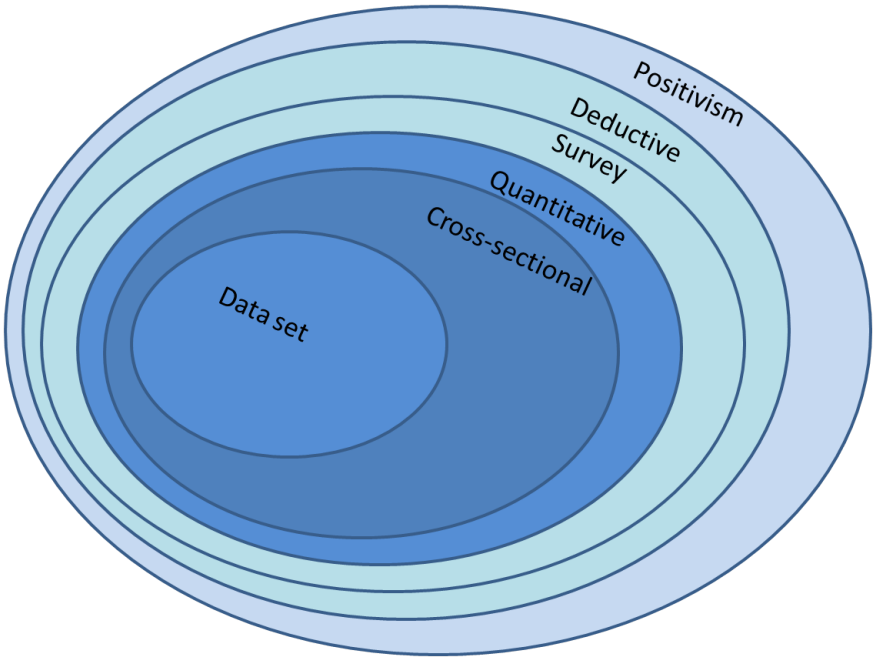

Figure 2.2 – Research onion of the research.

Part II

STATE OF THE ART

The questioning about our deepest existence, about the direction to be given to our existence must be the great business of our intellectual effort

M. TOWA

3

Service Oriented Architecture and Service Composition

Contents

This chapter will first present the SOA, its history, its characteristics, its layers and protocols before present one of the most advantage of this paradigm which is the service composition.

3.1 Introduction

Thomas Erl defines SOA as a terminology that represents a model in which automation logic is decomposed into smaller, distinct units of logic. Collectively, these units comprise a larger piece of business automation logic. Individually, these units can be distributed (Erl, 2005).

It is an approach to build distributed systems by integrating components that are independent from the technology platform, implementation languages, and operating systems. It delivers the functionality of an application as services that will be operated directly by the end user or used to form other services. A service-oriented architecture establishes a conceptual model that aims to improve the efficiency, agility, and productivity of a business by considering services as the primary means through which the logic of a solution is represented in the business support for the achievement of strategic objectives.

Ultimately, depending on the role of each in an organization, a SOA will have a definition accordingly. For an executive, a SOA can be defined as a set of services that the company wants to expose to their customers and partners, or other parts of the organization. When the architect views them as a vendor, requester, and service description-based architectural style, and supports the properties of modularity, encapsulation, decoupling, reuse, and composability. A developer will define them as a programming model with its standards, paradigms, tools and associated technologies. And for a solution integrator, it is a middleware offering functionality in terms of assembly, orchestration, monitoring and management of services.

The concept of ***service*** is at the basic concept of SOA. It is a standalone entity that encapsulates one or more features called service operations. There are several definitions for a service. It is a software component that provides a particular functionality and is accessible through its interface. A service is always accompanied by a description providing applications with the information necessary for its use. Specifically, the services are independent of any technology platform or implementation languages. In most cases, SOA are implemented using web technologies and the notion of *service* has become synonymous with that of the *web service*.

According to World Wide Web Consortium (3WC) which is a reference, a web service is a software system designed to support interoperable machine-

to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). These service operations are independent of the context or status of other services. It has a description of the functionality it performs in terms of operations and their input and output parameters. It specifies a communication model that governs interactions with other departments or clients. They expose their functionalities through a standard interface and communicate via message exchanges. Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

A service is represented by a tuplet $(Id, IO, Desc, Cond, QoS)$ where:

— Id is the identifier of each service;

— IO represents the input and output of the service interface descriptions;

— $Desc$ the basic information of the service (such as the URL, name and publisher);

— $Cond$ is a set of actions that must be verified before and after the service execution;

— QoS is the service quality set of non-functional descriptions, such as the cost, reliability, execution time, reputation, and availability.

The services are developed by the providers, which publishes their services descriptions in the form of files. These descriptions are centralized and stored in registries. Client applications send requests to registries to search for services. As soon as they found it, then download the descriptions of the selected services, and invoke them directly. The figure 3.1 illustrates these different transactions.

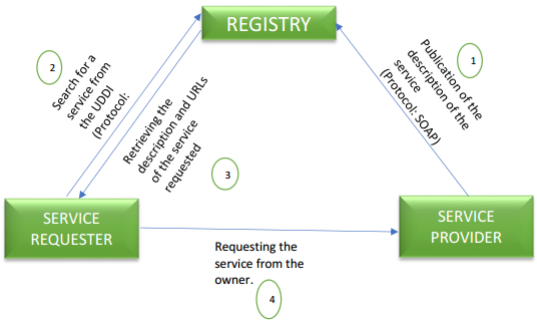

Figure 3.1 – SOA architecture.

3.2 History and philosophy

In most approaches to structuring complex systems, service orientation is becoming increasingly important. And yet, the concept of service is not new but given its transversality, it becomes difficult to give it an exact and precise content. Depending on the areas in which it appears, a definition will be attributed to it. In software engineering, the notion of service is still evolving, but its origins can be traced back to existing programming techniques. Indeed, we have moved from the procedural programming model (functions) to the object model where the encapsulation principle allows data to be grouped together in an object and associated processing. We then moved to the component model that allowed us to expose the functionality of an application or information system through a set of entry points. The service orientation is a logical follow-up to these advancements and consists of presenting access to a set of treatments related to certain functionality, without providing the details of the treatments. The data is visible only through the messages exchanged. This orientation, which has mainly benefited from web technologies accessible to all stakeholders, is therefore a continuation of these methodologies and sharing technologies. Intuitively, the concept of service represents an abstraction of the functionalities of an entity that can be as simple as a piece of data or an object on the web or as complex as an adaptable information system of an organization.

3.3 Characteristics

According to Thomas Erl (Erl, 2005), there are 7 principles of service orientation which are applied on SOA:

— Loose coupling Services maintain a relationship that minimizes dependencies and only requires that they retain an awareness of each other;

— Service contract Services adhere to a communications agreement, as defined collectively by one or more service descriptions and related documents;

— Autonomy Services have control over the logic they encapsulate;

— Abstraction Beyond what is described in the service contract, services hide logic from the outside world;

— Reusability Logic is divided into services with the intention of promoting reuse.

— Composability Collections of services can be coordinated and assembled to form composite services;

— Statelessness Services minimize retaining information specific to an activity;

— Discoverability Services are designed to be outwardly descriptive so that they can be found and accessed via available discovery mechanisms.

SOA have many advantages for business and for development engineering.

— For organizations, they improve the agility and flexibility of the business while facilitating the management of its business processes. They also offer the ability to break down organizational barriers and reduce the time of product development cycle. On the economic field, organizations are improving their return on investment and increasing their revenue opportunities through service oriented architecture.

— On the technological front, service oriented architecture reduces the complexity of solutions. And taking advantages to the reusability, we can build services once and use them frequently. They ensure standardized integration and heterogeneous customer support. They also facilitate systems maintenance.

The three fundamental interactions of the functioning of the SOA require a good description of service to facilitate its publication in the repository, its identification so that it is found easily and a mean of communication on for its invocation. It makes possible to define to them four layers of functionalities:

— The publication layer, which allows the centralization, storage and dissemination of service descriptions;

— The description layer, which groups together necessary details to invoke services in a document;

— The message layer, which ensures the uniform structuring and exchange of messages;

— The transport layer, which allows messages to be conveyed through the network.

3.4 SOA layers and basic protocols

The adoption of SOA in the computer world has been facilitated by several aspects including Web standards and technologies that have widely spread their implementation. They are used to provide broad support for SOA in the publication layer, the description layer, message layer and transport layer.

— Publication layer

The publishing layer is based on the Universal Description, Discovery, and Integration (UDDI) protocol, which consolidates, stores, and distributes service descriptions. It is the seat of the service registry. EbXML will also be noted as an alternative to provide the same functionalities in the case of e-business (Castillo et al., 2011).

— Description layer

The description layer is supported by the Web Service Description Language (WSDL), which describes the functionality provided by the service, the messages received and sent for each feature, as well as the protocol adopted for the communication. The types of data contained in messages are described using XML Schema language. There are several others methods of describing a service like Semantic Annotations Web Service Description Language (SAWSDL); Web Service Modeling Ontology (WSMO); Ontology Web Language for Service (OWL-S) which aim to enrich semantically the service description.

— Message layer

The message layer uses XML-based protocols because its unique syntax resolves syntactical conflicts when encoding data. Currently, Simple Object Access Protocol (SOAP) (Box et al., 2000), XML-Remote Procedure Call (XML-RPC) (Merrick et al., 2006) and Representational State Transfer (REST) (Erl et al., 2012) are the protocols used for this layer. Even though SOAP is still the predominant protocol, REST seems technically recommended (Castillo et al., 2011).

— Transport layer

For the transport layer, HyperText Transfer Protocol (HTTP) has become the de facto standard. This ubiquitous protocol on the Internet is generally tolerated firewalls. Making it particularly suitable for communications between organizations. However, other protocols can be used, such as Simple Mail Transfer Protocol (SMTP) or File Transfer Protocol (FTP), allowing web services to remain independent from the transport mode used.

The main advantage of SOAP and UDDI protocols as well as WSDL language is based on the Extensible Markup Language (XML) which is a language of data description that has huge benefits:

— a unique syntax that allows its adoption by various operating systems;

— a tree structure that facilitates its readability;

— Extensive scalability, because it imposes no restrictions of use apart from its syntax.

Thus, the exclusive use of languages and protocols based on XML language, as well as current Internet standards such as HTTP protocol, promotes interoperability between distributed information systems. Services remain independent from operating systems and development platforms, which facilitates interactions with client applications, but also across multiple departments involved in the process. That is why SOA are adapted to improve e-government (Heeks and Bailur, 2007). But one of the key features of developing a services oriented e-government is to structure and design composite services to gain real benefits for existent services. That is why the research community is mobilizing to find more effective ways to exploit this advantage. The next section will present the service composition in general and its different forms.

3.5 Service composition

One of the most important aspects of service oriented architectures is the possibility to combine many services to provide a new complex one which can give response to a new request.

3.5.1 Introduction

Sometime it is necessary to combine several services into a workflow to solve a complex problem. The service composition is an ability to provide a new functionality obtained from a combination of several Web services offered by various providers (working group et al., 2004). It is the way to take advantage of the two main characteristics of SOA which are service reusability and service composability. They respectively allow, to make use of a service without worrying about its execution environment and to combine in the form of an execution chain, a set of services for solving a complex problem (Thang et al., 2010).

A Web Service Composition (WSC) problem, in a general environment setting, is defined as (I, G, S) where:

— I is an initial interface, provided by a user in its request, indicating the starting point;

— G is a goal interface, provided by a user in its request, indicating the ultimate interface the user wants to obtain;

— S is a set of candidate web services.

Given a web service composition problem (I, G, S) a solution to this problem, called a composition plan π, is a sequence of totally ordered web services such that $\pi \subseteq S$. By applying each service in π, the resulting interface is a superset of G.

The service composition can also be define as an AI classic planning problem (Bevilacqua et al., 2011) (Carman et al., 2003) with a deterministic transitional model, represented as the 5-tuple:(S, S_0, S_f, A, Γ), where:

— S is a finite set of states;

— A a finite set of available actions;

— S_0 the initial state;

— and S_f the desired final state.

According to Bevilacqua and al. (Bevilacqua et al., 2011) and Carman and al. (Carman et al., 2003), a state can be either the set of necessary conditions for the execution of a service operation (pre-conditions) or the set of the effects produced by the execution (post-conditions). Problem inputs and desired outputs can be expressed as predicates in the initial state S_0 (describing available knowledge) and predicates in the S_f final state (describing additional knowledge produced by service execution), respectively. Two states are connected by a transition if an operation exists that can be performed to transit from the first state into the second one. An action, generated by a service operation, triggers a transition from its pre-conditions to its post-conditions. $(\Gamma \subseteq S \times A \times S)$ represents this transition relationship. Service goal G (or simply goal) the pair (S_0, S_f), denoted as $S_0 \rightarrow S_f$ in the following, representing the desired state transition from the initial state S_0 to the final state S_f

There are many types of service composition methods. These types are based either on the composition scheme elaboration time, or on the author of this composition.

3.5.2 Manual and automatic service composition

The composition of services can be characterized by its level of automation. According to this criterion of automation it can be manual, automatic and semi-automatic (Foster et al., 2003). The composition is manual when the user of the services programs himself, and by hand all the steps of the composition process of the composite services he needs, using a text editor without the help of dedicated tools. This method has the disadvantage that it requires extensive knowledge and considerable effort from the user who does

not always have the technical credentials. This task remains arduous and sensitive to errors caused by the dynamism and flexibility of proprietary service environments. Thus, when changes have been made even in one service, the operation of the resulting composite service may be affected and require changes in the composition process that must be restarted.

Automatic composition is an approach that supports the entire composition process and performs it automatically, without any user intervention. The composition process in this case is a generic process for different users who want to search and compose services that respond to their specific requests. Therefore, this process needs to be flexible and agile to accommodate the unique requirements and preferences of each user.

This agility criterion adds to the exploration of a multitude of possible Web-based solutions and the ability to interpret the operation of existing web services to make the automatic composition process a complex processEndong (2020); Hatzi et al. (2013). Automatic composition is considered, in fact, to be a very complex task because of the diversity of the elements (properties) of Web services that it has to examine to identify those appropriate and compatible with suitable composition plans and to verify their composability. This complexity and subsequently the cost of composition are further increased because of the rapid proliferation of Web services available on the Internet and the difficulty of semantically and correctly interpreting the properties of these services to meet user expectations To cope with some of the difficulties posed by the automatic service dialing process, a third interactive approach which is semi-automatic composition has been proposed. Here, the user keeps a supervision control in the composition process envisaged even if he does not have a deep knowledge of programming. It can make use of graphical tools to model and design composite services; select appropriate services based on semantic suggestions from dedicated tools such as METEOR-S, specify preferences, and intervene continuously during the composition processPatil et al. (2004); Verma et al. (2005).

3.5.3 Dynamic and static service composition

Depending on the time of its implementation, the service combination to create a new composite service can also be done statically or dynamically. The service composition is static when her implementation workflow is designed during the conception of the application. In the static approach, aggregation of services is performed at the design phase of the composite service. Services are determined, selected, aggregated and linked together before being

deployed. This type of composition is more suitable for stable environments where services previously identified by the user are not likely to change quickly. However, changing a component service or substituting another service also causes changes in the design of the composite service.

Unlike static composition where the number of services provided is limited and the services to be composed are specified beforehand, dynamic composition is initiated by a request from the user. The dynamic web service composition creates process model and selects atomic services automatically at runtime. It is a process which allows the services to interact in an intelligent way, to discover other services, to negotiate between them, and to compose themselves in more complex services (Domingue et al., 2005). It enables you to discover, select and dynamically combine services from Web service registries while taking into account the specific constraints of the user. It is an approach that brings flexibility and adaptability in the composition process. It makes it possible to generate composition plans adapted to each request from the services available at the time of the composition. This dynamic approach has several advantages over the static approach, in particular its flexibility and adaptability. The figure 3.2 and the figure 3.3 illustrate the difference between the two methods.

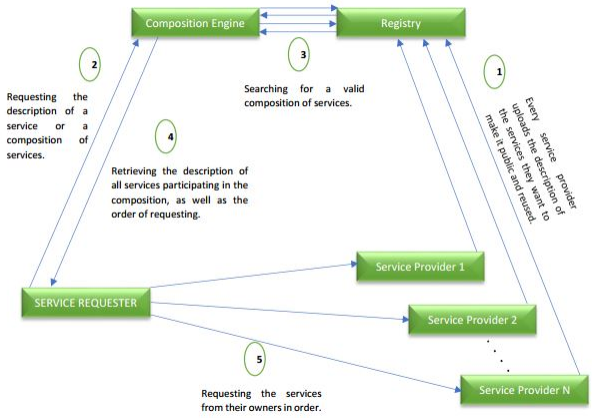

Figure 3.2 – Dynamic service composition.

3.6 Synthesis

Service oriented architecture has made an important contribution to the interoperability of information systems. In this chapter, some of the fun-

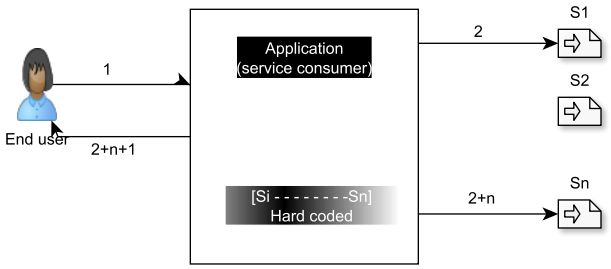

Figure 3.3 – Static service composition.

damental standards, protocols, and key web technologies on which they are based have been exposed. The service composition which is one of the major strengths of this paradigm is a vast and interesting area of research. The different composition approaches were presented. Manual and static service composition are time-consuming and hard task when more and more services have been deployed .The automatic and dynamic composition of services is a present trend (Pulparambil and Baghdadi, 2019; Endong, 2020; Garriga et al., 2016; Alwasouf and Kumar, 2019) and this approach is the most adapted for its flexibility and adaptability in relation to the dynamism of technological environments and especially in relation to the proliferation of services and stakeholders in the web.

But this approach raises a number of problems related to its effectiveness that research seeks to provide solutions. So, achieving dynamic composition in an environment where the number of services provided is constantly changing and the number of users is always increasing is not a simple task. In the next chapter, there will be a review of the literature on the dynamic service composition and the limitations that underlie this research.

Intelligence is defeated as soon as the expression of thoughts is preceded implicitly or explicitly by the little word "we".

S. WEIL

4

Dynamic Service Composition

Contents

4.1 Introduction

According to the service composition life cycle 4.1Hatzi et al. (2013); Alwasouf and Kumar (2019), researches to improve the dynamic service composition process and deal with scalability are focused on their main activities: service description, user request specification, service location, and request matching and service (Rostami et al., 2013). These activities can be mainly grouped in service discovery, service selection and composition plan generation.

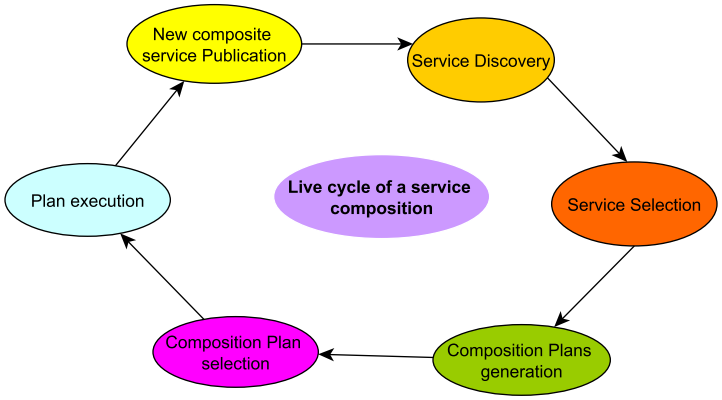

Figure 4.1 – Service composition live cycle.

4.2 Service description

The description of a service consists in defining an interface exposing operations performed by the service and linking each operation to its realization. There are models of syntaxic description and semantic approaches to enrich with semantic annotations and ontologies the operational interfaces of a service. There are also methods of behavioral description of a service.

4.2.1 Syntactic models

The Web Service Description Language (WSDL) (Christensen et al., 2001) is the current standard for defining services. Through an XML document, it describes a service through an interface presenting a set of operations and their entries and exits parameters respectively. The WSDL interface exposes the functionality accomplished by the service that is, what it does, without

specifying the conditions of completion of this feature understood how the service does it. A WSDL document is an XML file consisting of five main elements: the *types*, the *message*, the *portType*, the *binding*, and the *service*.

— ***types*** is a container defining the data within messages exchanged by the service. It defies through an XML schema the types of these data. These data can be simple or complex.

— ***message*** defines the messages exchanged by the service. It is composed of a set of *part*. A *part* is associated with a data described in *types*.

— ***portType*** defines a set of operations. An *operation* is an abstract description of an action performed by the service. It consists of an *input* and an *output* describing respectively the input and output parameters of the operation. Each of these elements has a message attribute designating a *message* element.

— ***binding*** resumes element operations *portType* and associates a protocol of transfer of message to them that the *binding* describes how the messages will be exchanged. It specifies the exchange protocol but it does not specify the addresses between which the messages will be exchanged. The WSDL describes the *binding* with the GET and POST methods of the HTTP protocol. The *binding* element has two attributes. The name attribute defines the name of the binding and the type attribute indicates the *portType* for which the *binding* takes place.

— ***service*** specifies the address or addresses where the service is located. A *service* is a set of *port*. A *port* specifies an address for a given *binding*.

4.2.2 Semantic models

The WSDL standard is not precise in the description of a service. This is due to the low level of expression of the syntactical description as it focuses on a sort of operations enumeration and the description of their types of parameters. It does not provide information on the meaning of the function performed by the service (Berners-Lee et al., 2001) (D.Fensel, 2002). To overcome WSDL lack of semantics, several approaches propose to add a semantic layer above WSDL to complete its syntactic description.

— **SAWSDL**

The Semantic Annotations for Web Services Description Language (SAWSDL) and XML Schema (Kopeckỳ et al., 2007), recommended by W3C in April 2007, is an approach that presents a mechanism for semantically enriching the services described with WSDL and their associated XML

schemas. SAWSDL does not specify a language to represent semantic models. It provides rather a mechanism through which concepts belonging to existing semantic models can be referenced from a WSDL.2 document. SAWSDL provides additional annotation extensions for XML schemas. The main extensions for annotating a WSDL document are the following attributes: modelReference, liftSchemaMapping, and loweringSchemaMapping.

- The modelReference attribute is used to annotate all WSDL elements. In particular, it is presented as an attribute of *interface*, *operation* and *fault*. He points to the equivalent concept by adding his address.
- The liftingSchemaMapping and loweringSchemaMapping attributes are used to associate a concept diagram or an element with a concept in an adopted reference ontology.

Many work environment prototypes implementing SAWSDL are emerging.

- **Web Services Policy (WS-Policy)** (Vedamuthu et al., 2007) , is a W3C standard that is used to describe policies adopted by the provider of a Web service as well as the expectations of a customer. Indeed, beyond its functional capabilities, a Web service has non-functional capabilities including its level of security and quality of service. A service provider can provide the same functional web service according to different policies in terms of non-functional aspects. For example, a given web service may be provided with different levels of security depending on the transport protocols and encryption algorithms used. In this case, the choice of these protocols and algorithms forms a policy of the Web service in terms of security. It uses a simple and extensible syntax grammar to describe and communicate the strategies of a Web service. WS-Policy provides particular basic constructs that can be extended by other Web services specifications. It is based on XML for the purpose of conveying Web services policies in an interoperable mode.

According to WS-Policy, the policy of a Web service can be expressed by several alternative policies. Each alternative policy consists of a set of assertions. Each assertion expresses a characteristic (described by the provider) or a requirement (required by the client) of a subject that may be an Endpoint, a message, an operation, etc. For example, the "*Encrypted*" assertion associated with an endpoint indicates that the

messages exchanged must be encrypted. Operators are also used to express policies. We quote for example *ExactlyOne* or *All*.

The *All* operator specifies the fact that all assertions constituting an alternative policy are or must be satisfied to accept the alternative policy proposed. The *ExactlyOne* operator specifies that at least one of the alternative policies is or must be satisfied to accept this policy. In addition, the use of the *WS-PolicyAttachment* element allows you to attach a policy to a topic as an intrinsic part of its definition (in the WSDL file or at the UDDI level for example) or as part of an independent document (apart) used for this purpose (Vedamuthu et al., 2007). The *EncryptedParts* and *SignedParts* assertions allow you to specify the parts of the message (exchanged by the web service) which are respectively concerned by a protection of confidentiality (encryption) and integrity (digital signature).

The assertions used to describe the policies of a Web service can relate to different non-functional areas such as security or quality of service. Some areas are currently covered by specifications defining all the assertions relating to it. In particular, these specifications make it possible to describe, in a standard manner, the properties of Web services relating to a specific non-functional domain. As an example, we mention the *WSPolicySecurity* specification covering the security domainBelouadha et al. (2012). One can nevertheless point out the absence of a semantic description of the assertions at WS-Politics. It is a limitation that does not allow the correct interpretation of non-functional assertions relating to domains that correspond to user contexts and not covered by standard specifications such as WSPolicySecurity.

— **WSDL-S**

This is a common specification for IBM and the LSDIS lab (Akkiraju et al., 2005) . It was submitted to W3C in 2005. Its main purpose is to provide a semantic annotation process compatible with existing technologies. For example WSDL-S *Meta- model* extends the WSDL by adding three major elements: *category*, *precondition*, *effect* and two attributes: *modelReference* and *schemaMapping*. The elements introduced make it possible to add information that was not taken into account in WSDL such as the preconditions and the effects of an operation while the attributes make it possible to link concepts in reference ontology.

— The *category* element is a sub-element of *portType*. It specifies the

category of a service when publishing in a directory or registry.

- The *precondition* element is a sub-element of *operation* that specifies the preconditions to be checked for the operation to run as expected.
- The *effect* element is a sub-element of ¡operation¿, indicating the effects of executing the operation.
- The modelReference attribute can be added to an *xs: element* in an XML grammar and to the *operation*, *precondition*, and *effect* elements to indicate their matches in an given ontology.
- The schemaMapping attribute can be added to an *xs: element* in an XML grammar to describe the mappings between the annotated grammar and the reference ontologies.

— **METEOR-S** project launched by the LSDIS laboratory (Patil et al., 2004) aims at adapting service technologies to those of the semantic web. It does not provide a new service representation model, but proposes to generate semantic annotations for WSDL. METEOR-S is based on the METEOR (Managing End To End OpeRations) project which deals with the management of large-scale process workflows in heterogeneous environments.

4.2.3 Ontologies

The Semantic Web (Berners-Lee et al., 2001) aims to provide, for each Web resource, a semantic that can be interpreted by the machine. In this context, ontologies present as the key element of the semantic Web. They aim to describe concepts and their interrelationships, as well as the rules of deduction that govern them. From this fact, they are seen as a way that allows the user to perform more relevant searches on the Web, as he becomes able to access not only those resources syntactically related to its requests, but also to those which are semantically related to it and that can be deduced from domain ontologies. Ontology of services represents the various aspects related to the description of the services and their use through a set of concepts, properties and relations between them. OWL-S and WSMO are the main models used in SOA.

— **OWL-S**: Ontology Web Language for Services (DAMLS in earlier versions) (Martin et al., 2004), is an ontology for describing web services, whose objectives are to resolve ambiguities and to make the description of a service understandable by a machine to automate the discovery,

invocation, composition and monitoring of service delivery. It was submitted to W3C in November 2004 as a proposal and is based on the concept of OWL classes. Its tripartite structure represented by the figure is composed of a service profile, a service grounding and a process model. These three concepts describe the functionality performed by the service, link this description with the WSDL description of the service, and describe the behavior of a service when interacting with other services.

- The service profile mainly contains a description of the service and its provider, the functional behavior of the service and a set of attributes such as the category of the service. The service provides a high-level description of a service and its provider.
- The process model defines several types of processes that describe the scheduling of operations and the behavior of a service in an interaction.
- The service grounding describes how to communicate with a service by specifying the protocol, the format of the messages, the serialization and the addressing. It is the result of a mapping between the OWL-S process describing the service and the original WSDL description.

— **WSMO**: Web Service Modeling Ontology - The WSMO ontology was initiated by the ESSI WSMO workgroup (Domingue et al., 2005). She is based Web Service Modeling Framework WSMF proposed by Fensel and Bussler(working group et al., 2004).

Like the OWL-S, WSMO aims to automate service-related tasks with the essential aspect of separating the description of semantic web services from executable technologies. To do this, WSMO provides an ontology description model that is independent of the language used to describe them. The central concept is the *WSMO element* which specializes in *ontology* or *webService* or a *goal* or a *mediator*. The elements *ontology* and *webService* aim to describe the services. The *goal* and *mediator* elements deal with their discovery.

- An *ontology* element denotes a reference ontology imported by an element of the WSMO ontology in order to describe its properties. The ontology provide a terminology of semantic description of elements belonging to specific domains. They consist of concepts, relationships, functions, instances and axioms, which allow him to describe all the information used by the actors of a Web service in a

comprehensible and interpretable way by a machine. WSMO allows to import directly an ontology in another in the case of absence of conflicts, or through a mediator when conflicts of incompatibility are noted to facilitate the use of different ontology.

— A *webService* element is a computing unit capable of responding to a user request. WSMO *webServices* are defined in a consistent way with the following: service functionality, description interfaces, non-functional properties, imported ontologies, and mediators used. Service functionality, or capability, is described in terms of operations associated with preconditions, assumptions, post-conditions, and effects. Pre-conditions and the post-conditions are the requirements on the data before the execution of the service and the changes that they undergo after execution; the assumptions and effects describe the requirements and changes to the services that interact with the described service. The description interfaces describe the behavior of the service, in other words the scheduling of the operations. The non-functional properties of a *webService* describe the attributes that do not relate directly to the processed data, such as the execution time of an operation. The imported ontologies are used to referencing the elements of the service description. Mediators are used if two interacting *webServices* matter two different ontologies.

— **Mediators** solve the structural, semantic, or conceptual incompatibilities that can be detected at the data or process level, in order to connect WSMO heterogeneous resources. In the case of incompatibility at the given level, mediation serves to establish the correspondence between the different terminologies. In the second case (incompatibility at the process level), the mediation analyzes the execution of the two services and covers any disparities.

— **Goals** define exactly the objectives that a web service user is trying to satisfy. Whether human or software, this user specifies his search criteria by describing the interface and expected service features through the description of the goal. Customer requests and services are strongly decoupled in WSMO. Requests are based on goal descriptions, and mediators must look for matches between these goal-level requests and existing Web services.

Observations: the service description stage is important in the dynamic service composition. It is the basis of any operation but, its optimization has a direct impact only on the number of services.

4.3 Service discovery

Service discovery consists in detecting services that meet the specifications of a given request (Benatallah et al., 2005). It is an essential step in a process of composition of services. It not only allows to find a service that can respond to a solicitation, but in the process of composition, it also identifies services capable of cooperating to form a composite service that responds to a complex request. Service recovery is in general as the process that takes a user request as input and returns a list of resources or services that may eventually fill the described need (Toma et al., 2005a) (Toma et al., 2005b). Another definition of this process has described it as the operation of locating a machine-comprehensible description of a service possibly unknown beforehand and corresponding to certain functional criteria (Hassina Nacer Talantikite, 2009) (Bucchiarone and Gnesi, 2006).

4.3.1 Main aspects

There is a search directory that supports the descriptions of available services and facilitates their discovery. This directory called registry receives the information relating to the technical specifications of the interfaces of the available services. It is structured in white pages, yellow pages and green pages (Hammami et al., 2018). The white pages contain the information on the service providers. The yellow pages describe the services and the green pages give their technical specifications.
In the literature, several architectures (Benatallah et al., 2003a) (Toma et al., 2005b) (Keller et al., 2005) have been proposed for realizing Web services discovery. Most of these works specified three major steps which constitute service discovery: the specification of the user request, the location of service description interfaces, and the matching of the search request and the services found. In the definitions above, we can highlight three important aspects in the discovery of services:

— The location of services which consists of finding the address to which the description of a service is provided.

— The data processing that indicates the degree of automation of the discovery.

— The matching that performs the comparison between the user query and the listed services.

4.3.2 Service publication and location

4.3.2.1 Centralized approaches

The Universal Description Discovery and Integration (UDDI) model has established itself as the reference model for centralized service publishing and discovery approaches before becoming an OASIS standard in July 2004Rompothong and Senivongse (2003). A UDDI registry is a set of catalogs that provide mechanisms for classifying and managing services to facilitate discovery and invocation. A UDDI may belong to a public domain such as the internet or any other network accessible to an unlimited number of users, as it may belong to a restricted domain such as the intranet of a company or a group of companies. The data structures of a UDDI are as follows:

— ***businessEntity***: contains information describing an organization providing services.

— ***businessService***: Describes a collection of services provided by an organization. A *businessService* is contained in a *businessEntity*.

— ***bindingTemplate***: Define the technical information needed to invoke a service. A *bindingTemplate* is contained in a *businessService*.

— ***tModel***: Define a concept representing a type of service, a protocol used by the services or a category of services. A *tModel* element is reusable because it can be referenced by more than one *bindingTemplate*.

— ***publisherAssertion***: Define a link between the *businessEntity* that contains it and another *businessEntity*. When two *businessEntity* elements are referencing the same *publisherAssertion*, we talk about relationship between these *businessEntities*

4.3.2.2 Distributed approaches

Distributed approaches consist of extracting service descriptions from decentralized repositories. These approaches assume that services are stored in the provider sites of these services. Thus, the user request is executed directly at the provider server or through agents or robots for collecting service descriptions (Song et al., 2007) (Zhou and Huang, 2008) (Guan et al., 2008). The first distributed approaches to service discovery were to set up a federation of UDDI registries (Rompothong and Senivongse, 2003). This federated registry was to provide a service that acts as a layer of abstraction linking several public access UDDI instances. Each instance of UDDI is responsible for the data for the services to which it refers.

There are several distributed registries models (Meshkova et al., 2008). (Rompothong and Senivongse, 2003) present an expandable peer-to-peer infrastructure, directories for the publication of semantic services and their discovery entitled METEOR-S Web Service Discovery Infrastructure (MWSDI) (Verma et al., 2005). This infrastructure, which is not limited to the UDDI model, makes possible to link several models of proprietary directories. It is a four layer structure:

— The data layer containing a set of UDDI directories pointing to the services.

— The communication layer is the underlying peer-to-peer network hosting directories, services and customers. The nodes are categorized into four categories: Operator Peer, Peer Gateway, Auxiliary Peer and Peer Client.

— The operator services layer contains all the services provided by the Operator Peers. If they imply a different directory of UDDI, they must provide the meta-data and APIs to query them.

— The semantic specifications layer contains a set of domain ontologies and the relationships between them. The elements of the semantic layer specifications annotate registries and services to automate interaction and make discovery more accurate.

(Sivashanmugam et al., 2004) is based on the MWSDI model and proposed the establishment of a federation of public and private registries at the same time. They assume that an enterprise can be brought to share its directory infrastructure with its partners to demonstrate and share its services. They present the XTRO (Extended Registries Ontology) ontology, which brings together registry ontologies and federations of registries and domains, thus allowing the relationships between these different elements to be described. The particularity of this work resides in the fact that it takes into account both the semantic aspect of the publication - discovery process and the possibility of interaction between public and private registries.

The UDDI 3.0.2 (Clement et al., 2004) specification has incorporated this principle into a UDDI directory federation. It presents an UDDI registry as a set of UDDI nodes. Nodes in a registry collaborate to manage a set of UDDI data structures. A node is part of a single registry. Each node has a replicated copy of the overall schema of the registries data structure to allow better management of requests addressed to it.

Another distributed approach for the services discovery is also described in (Papaioannou et al., 2006). The services are supposed to be described by OWL-S ontology services. The peer-to-peer protocol adopted for information retrieval is Gnutella. By combining these two models, OWL-S and Gnutella, the authors wanted to combine the precision of description of the OWL-S approach with the efficiency of the Gnutella search and localization technique.

(Schmidt and Parashar, 2004) proposes a peer-to-peer indexing and storage system. The indexing approach is based on the Hilbert Space-Filling Curve matching function. After partitioning the data space, they use the CHORD (Stoica et al., 2001) protocol to distribute the data in a virtual network to facilitate the search. Their model has search and localization capabilities in real time and it allows finding all the elements corresponding to a need if they are described on the network. This is done with a limited number of messages and cost.

However, it can be noted that collecting and managing information in a distributed environment is a difficult and complex task. Because it can face the problem of heterogeneity of presentation and the description of services that the target sites can present. That is why (Liao et al., 2019) propose a flexible paradigm for intelligently discovering, aggregating and processing big distributed data is a crucial requirement in large content-centric Internet. They present a scalable Semantic Concast service on Named Data Networking (NDN) being considered as a promising paradigm for the future Internet. The service enables cooperation between data discovering, aggregating and processing among intermediate nodes for a user's Interest that contained a hierarchical name and semantic constraints. Specifically, multiple types and strategies of data aggregation and processing for combining and processing the positive data and suppressing the negative, futile data, as well as a determination of response completeness are introduced for enhancing relevant results recall and sharing.

With the growing number of services in the repositories and the challenges of quickly finding the right ones, the need for clustering related services becomes obvious to enhance search engine results with a list of similar services for each of it. That is why Platzer and al. (Platzer et al., 2009) proposed a statistical clustering approach that enhances an existing distributed vector space search engine for Web services with the possibility of dynamically calculating clusters of similar services for each hit in the list found by the search engine. Recently, Rostami and al. (Rostami et al., 2014) proposed a novel architecture for semantic web service composition using clustering and Ant

colony algorithm to perform the service discovery. They used clustering for categorizing the web services to allow web service consumers to find relate services easily and an ant colony algorithm used for finding the best set of web services that have high combining ability. The proposed system can find the optimal length of web service composition with the different challenge set for different number of services.

4.3.3 User request specification

To enable users to express their needs and requirements easily even without technical knowledge of service specifications, service dialing systems have created the query languages. A user can usually describe his request by specifying its functional aspects sought and the non-functional constraints required. Thus, the discovery becomes a matching between the functional properties of the published services and the functional description of the query. Non-functional constraints are often used in the selection phase of discovered services.

Request languages are distinctly different from service specification languages. There are several techniques for representing requests languages among which we can cite textual approaches, logic-based approaches, graphical approaches or specific languages to semantic web service models.

— Textual approaches (Bosca et al., 2005) (Cremene et al., 2009) use natural language to express a request. But this requires a later processing which consists of formalizing and rewriting the request before being able to process it automatically. Neng and al. propose a method to obtain relevant services accurately with a keyword query by exploiting domain knowledge about service functionalities (i.e., service goals) mined from textual descriptions of services(Zhang et al., 2018). This later firstly extracts service goals from services' textual descriptions using an NLP-based method and clusters service goals by measuring their semantic similarities.

— Logic-based approaches and graphs (Lazovik et al., 2005) (Karakoc and Senkul, 2009) use formal models to describe the query. Logic-based approaches express queries as predicates. This method takes advantage of the techniques of artificial intelligence in terms of intelligent management of knowledge. Above all, it gives the possibility of breaking complex requests into simple and targeted sub-requests.

— Graph-based approaches (Ceri et al., 1999) express requests in the form of start nodes representing the composite service inputs and arrival nodes

representing the outputs of the service. The departure and arrival nodes delimit any sub-graphs satisfying the request.

— Semantic approaches are extensions based on semantic description models. We can cite WSML (Web Service Modelling Language) (de Bruijn et al., 2006) in the case of WSMO and OWL-QL(OWL Query Language) (Fikes et al., 2004) can be cited in the case of OWL-S.

Observations : the service discovery is an important step in the dynamic service composition. However, its optimization has a direct impact only on the number of services and also on the specificities of the user's request.

4.4 Service selection

To dynamically select a service, a matching is made between the user's request and the available service. Then, the best service capable of satisfying the user needs is chosen. In the context of a composite service, a study of the composability of the services to be involved in the composition process is still needed.

4.4.1 Request matching and Service

To identify similarities and possible bridging between the properties expressed at the query level and those specified in the description of the service identified, the matching is carried out. Its purpose is to evaluate the level of correspondance between the user's request and the description of the localized service. The functional properties constitute the main elements of analysis of this comparison. This property allows discovering the services that can meet the functional needs of the user without taking into account the quality of this service, its preferences and some other user's constraints. Matching mechanisms are categorized into two major categories: syntactic matching and semantic matching.

— Syntactic matching

The syntactic matching of services is the lexical analysis to evaluate the level of correspondence between textual structures describing the properties of the service and those expressed at the level of the user request. Syntax matching can be based on matching keywords (Kreger et al., 2001) or based on WSDL matching (Wang and Stroulia, 2003). The first is to compare the keywords in the query with those describing the Web service at the service directory level. The second assumes that given

a textual description of the desired service, a traditional information-retrieval method is used to identify the most similar service description files, and to order them according to their similarity. Next, given this set of likely candidates and a (potentially partial) specification of the desired service behavior, a structure-matching step further refines and assesses the quality of the candidate service set.

— semantic matching

This requires the requester to specify several constraints including the dependency of atomic, the user's preference and so on (Toma et al., 2005b). Some researchers affirmed that the first step toward this interoperation is the location of other services that can help toward the solution of a problem (Paolucci et al., 2002). In this direction, they claim that location of web services should be based on the semantic match between a declarative description of the service being sought, and a description of the service being offered. Furthermore, they claim that this match is outside the representation capabilities of registries such as UDDI and languages such as WSDL and proposed a solution based on DAML-S, a DAML-based language for service description. They show how service capabilities are presented in the Profile section of a DAML-S description and how a semantic match between advertisements and requests is performed.

After that, Michael and al proposed an algorithm, which ranks the matching degree of service descriptions according to OWL-S(Jaeger et al., 2005). Different matching degrees are achieved based on the contra variance of the input and output types for requested and advertised services. Furthermore, additional elements of the service description, such as the service category, are either covered by reasoning processes or, such as quality of service constraints, by custom matching rules. Contrary to mechanisms that return only success or fail, ranked results provide criteria for the selection of a service among a large set of results. With such a discovery mechanism additional Web services can be found that might have normally been ignored. Klust and al. (Klusch et al., 2006) present an approach to hybrid semantic Web service matching that complements logic based reasoning with approximate matching based on syntactic IR based similarity computations. The hybrid matchmaker, called OWLS-MX, applies this approach to services and requests specified in OWL-S.

Also using ontologies, researchers proposed a novel matching framework for Web service composition (Medjahed and Atif, 2007). This framework combines the concepts of Web service, context, and ontology. They

claim that the context-based matching for Web services requires dealing with three major research thrusts: context categorization, modeling, and matching. So they first propose an ontology-based categorization of contextual information in Web service environments. We then define a two-level mechanism for modeling Web service contexts. In the first level, service providers create context specifications using category-specific Web service languages and standards. In the second level, context specifications are enveloped by policies (called context policies) using WS-Policy standard. Finally, they present a peer-to-peer architecture for matching context policies. The architecture relies on a context matching engine, context policy assistants, and context community services. Community services implement rule-based techniques for comparing context policies. An another context-based semantic service matching approach named 'Process-Based service MatchMaker'(PBMM) is proposed to select the suitable services for the process from candidate services through taking the dependencies of related services process into consideration (Huang et al., 2019).

4.4.2 Service composability

The composability of Web services is a process which verifies that it is possible to compose one or more services. This is a fundamental step in the building phase of the service composition plans because it makes it possible to ensure that one or more services, or even service operations, can be connected to interact with each other and cooperate to meet the functional need of a given request (Medjahed and Bouguettaya, 2005). It also makes it possible to generate effective composition plans. Most of the studies on the composition of services were based on the syntactic or semantic matching of the inputs and outputs of service operations to evaluate their composability. Others have proposed an extended analysis of all the functional, non-functional, contextual, data-driven and technical properties of services to evaluate their composability (Omrana et al., 2012) (Belouadha et al., 2012).To this end, they proposed more detailed service description models to facilitate the study of their composability (Omrana et al., 2010).It is therefore a fundamental element in the selection of services in the composition process before generating the composition plans. And these plans have to be reliable.

4.4.3 Composite service reliability

In services composition, reliability is an important aspect. The reliability of a service can be defined as the probability of a service that will perform a required function without failure under stated condition for a stated period of time. Reliability in service composition is important to improve performance through fault tolerance of the system (Tiwari and Mishra, 2018).

Fan and al. (Fan et al., 2013) proposed an approach to constructing the reliable service composition. The underlying formalism is Petri net, which provides means to observe behaviors of basic component, and to describe their interrelationship. The transaction attributes, reliability and failure processing mechanisms are articulated. The composition mechanism systematically integrates these schemas into a transaction mapping model. Based on this, a reliable composition strategy and its enforcement algorithm are proposed, which can verify the behaviors of service composition at design time or after runtime to repair design errors. The operational semantics and related theories of Petri nets help prove the effectiveness of the proposed method.

Recently, Tiwari and al. (Tiwari and Mishra, 2018) proposed a CPN based Reliability in Composite Web Service (CPN-RCWS). They compared the reliability of composite web services between local level recoveries and replicated level recoveries. The Local Level Recovery is a sort of fault tolerance mechanism where failed service is recovered on the concept that the failed subtask of the service will be again rescheduled the same node for execution. But the problem in this approach is that there is no possibility of successful execution as the node is failure prone node. The benefit in this approach is that the communication overhead is minimal. But oppositely, the Replicated Level Recovery is a type of fault tolerance approach where the failed task of service is replicated to other nodes to resume the execution after failure occurs. The benefit in this approach is that the probability of success of service execution of web service is higher than in local level recovery.

Observations : the service selection stage is very important in the dynamic service composition. However, its optimization has a direct impact only on the number of services and also on the specificities of the user's request.

4.5 Composition plan generation approaches

During manual and static service composition, the services to be combining are determined by the designer of the composite service. But during the automatic and dynamic service composition it is necessary automatically,

upon receipt of a complex request, to find possible solutions by composition of services before discovering among the available services those which make it possible to reach the objectives of the request. Each of the way to combine a set of abstract services to respond to a given user's request is a composition plan or a composition path.

Constructing composition plans is a very complex task that is a problem that has been extensively dealt with in several research projects (Rao and Su, 2005). The main approaches dealing with this issue are based on workflows, artificial intelligence planning techniques and dependency graphs. Other works also propose hybrid approaches.

4.5.1 Workflow approach

Workflows are activities that involve the coordinated execution of multi tasks performed by different processing entities (Atsa Etoundi and Ndjodo, 2005). Workflows are business processes that are run in an IT environment using a tool such as IBM MQSeries Workflow (Kreger et al., 2001). Workflow tools allow businesses to define each of their business processes as a series of activities carried out by individuals or applications, and vary the sequence through the activity series depending on the output data from each individual activity. They are a means of specifying and composing activities in order to form a chain of processes performing any management process in application engineering.

Workflow-based service composition techniques have benefited from the work of the research community in the area of workflow management and the maturity of their approaches. It should be noted that a workflow is composed of activities that are then replaced by services. Researchers believe that a composite service can be conceptually similar to a workflow (Casati et al., 2000a). As a result, the execution of a composite service appears to be comparable to the execution of a Worfklow whose mechanisms ensure the flexibility, the adaptation and the integration of automatic processes. But the services are different from the activities in the case of the composition by their distributed aspect, their autonomy and their heterogeneity (Benatallah et al., 2003b) that must be taken into account when managing or operating the elaborated Workflow.

Composition frameworks based on workflows were the first solutions proposed for the composition of services especially in the context of a static composition. The construction of static type composition schema is usually based on BPML(Business Process Modeling Language) and WS-BPEL(Web Service

for Business Process Executive Language) behavioral description languages that are used for the orchestration of web service as well as WS-CDL(Web Service Choreography Description Language) (Kavantzas et al., 2005) and WSCI(Web Service Choreography Interface) (Arkin et al., 2002) which are used instead to specify their choreography. To create dynamic compositions, there are other frameworks using workflows such as eFlow (Casati et al., 2000b) (Casati et al., 2000a),PAWS (Process for Adaptative Web Service) (Ardagna et al., 2007) and the framework proposed by Majithia and al. (Majithia et al., 2004).

— eFlow (Casati et al., 2000b) (Casati et al., 2000a) is a system that supports the specification, enactment, and management of composite e-services, modeled as processes that are enacted by a service process engine. Composite e-services have to cope with a highly dynamic business environment in terms of services and of service providers. In addition, the increased competition forces companies to provide customized services to better satisfy the needs of every individual customer. Ideally, service process should be able to transparently adapt to changes in the environment and to the need of different customers with minimal or no user intervention. In addition, it should be possible to dynamically modify service process definitions in a simple and effective way to manage cases where user intervention is indeed required.

— The PAWS (Processes with Adaptive Web Services) (Ardagna et al., 2007) framework facilitates flexible and adaptive execution of managed Web-service-based business processes. The framework coherently integrates several service-adaptation modules and uniquely couples design-time and runtime mechanisms for process specification and global framework execution. During design, PAWS achieves flexibility through a number of mechanisms: it identifies a set candidate services for each process task, negotiates QoS, specifies quality constraints, and then identifies mapping rules for invoking services with different interfaces. At runtime, PAWS exploits the design-time mechanisms to support adaptation during process execution: it selects the best set of services to execute the process, reacts to service failures, and preserves execution when a context change occurs. Results show that it can reduce design-time efforts to create a flexible process, while ensuring a good trade-off between user and provider requirements.

— (Majithia et al., 2004) present a framework to facilitate automated service composition in Service-Oriented Architectures using Semantic Web technologies. The main objective of the framework is to support the

discovery, selection, and composition of semantically-described heterogeneous services. This framework has three main features which distinguish it from other work in this area. First, it proposed a dynamic, adaptive, and highly fault-tolerant service discovery and composition algorithm. Second, it distinguished between different levels of granularity of loosely coupled workflows. Finally, the framework allows the user to specify and refine a high-level objective.

4.5.2 Planification techniques

Planning is an Artificial Intelligence (AI) strategy that allows you to choose and organize actions based on a specific goal. This strategy considers a solution to a given problem as being a sequence of actions to be taken from an initial state to a target state, having knowledge of the possible actions and states, and the conditions of application of these actions. Then, each problem to be solved is represented by a tuplet (S, S_0, G, A, Γ) where:

— S is the set of possible states of the world;

— S_0 is the set of initial states such that $S_0 \subset S$;

— G is the set of target states such as $G \subset S$;

— A is the set of actions;

— Γ is the translation relation that specifies the change from state S_1 to state S_2 after execution of action $A(\Gamma \subseteq S \times A \times S)$.

In order to resolve dynamically constructing service composition plans using the planning techniques derived from the AI, most of the work assimilates this problem to a tuple of which:

— actions represent services;

— the set Γ denotes all the preconditions and effects of the services;

— the set S_0 indicates the initial conditions;

— the set G designates the objective or the customer request to be satisfied.

Thus, by representing the services by actions and by choosing a given goal to be achieved through a predefined set of services, the planner is tasked with generating an ordered collection of services that achieves the stated goal. Several planning techniques have been applied to the dynamic service composition domain: situation computation, hierarchical task network (HTN), theorem proofs, and rule-based systems. The use of these techniques is often associated with the use of ontologies to make them more efficient through semantic enrichment.

4.5.2.1 Situation calculation

McIlraith and al. firstly proposed the markup of Web services in the DAML family of Semantic Web markup languages. This markup enables a wide variety of agent technologies for automated Web service discovery, execution, composition and interoperation (McIlraith et al., 2001). And they present one such technology for automated Web service composition by augmenting version of the logic programming language, Golog provides a natural formalism for automatically composing services on the Semantic Web (McIlraith and Son, 2002). To this end, researches adapted and extended the Golog language to enable programs that are generic, customizable and usable in the web context. Further, they propose logical criteria for these generic procedures that define when they are knowledge self-sufficient and physically self sufficient. To support information gathering combined with search, they propose a middle-ground Golog interpreter that operates under an assumption of reasonable persistence of certain information. This approach combines online execution of information-providing web services with offline simulation of world altering Web services, to determine a sequence of Web Services for subsequent execution.

Lécué and al. introduced a framework for performing dynamic service composition by exploiting the semantic matchmaking between service parameters (i.e., outputs and inputs) to enable their interconnection and interaction (Lécué et al., 2008b). The basic assumption of the framework is that matchmaking enables finding semantic compatibilities among independently defined service descriptions. They also developed a composition algorithm that follows a semantic graph-based approach, in which a graph represents service compositions and the nodes of this graph represent semantic connections between services. Moreover, functional and non-functional properties of services are considered, to enable the computation of relevant and most suitable service compositions for some service request.

In the same trend Phan and al. (Phan and Hattori, 2006) proposed a formal approach to translate OWL-S web service descriptions into primitive and complex actions of ConGolog. In addition, in order to support information gathering with search in an open world initial database, they proposed an extended version of the middle-ground ConGolog interpreter which relies on a theorem-prover with prime implicates.

Recently, researchers proposed a new architecture for atomic service discovery, composition and automatic plan generation for the proper execution of its candidate services. The proposed architecture takes the advantage of adductive event calculus that uses adductive theorem prover to generate a sound

and complete plan for the proper order of execution of the atomic services. (Paulraj et al., 2016)

4.5.2.2 Hierarchical tasks network

Using DAML-S and SHOP2 which is an Hierarchical Task Network (HTN) planner well-suited for working with the Process Model, researchers have proven the correspondence between the semantics of SHOP2 and the situation calculus semantics of the Process Model (Wu et al., 2003). Then, Wu and al. have also implemented a system which soundly and completely plans over sets of DAML-S descriptions using a SHOP2 planner, and then executes the resulting plans over the Web.

After, Klusch and al proposed OWLS-Xplan converts OWL-S 1.1 services to equivalent problem and domain descriptions that are specified in the planning domain description language PDDL 2.1, and invokes an efficient AI planner Xplan to generate a service composition plan sequence that satisfies a given goal (Klusch et al., 2005). Xplan extends an action based Fast Forward-planner with a HTN planning and re-planning component. Others researches proposed a mapping a given set of process models and preferences into a planning language for representing Hierarchical Task Networks (Lin et al., 2008). They then present SCUP, the new web service composition planning algorithm that performs a best-first search over the possible HTN-style task decompositions, by heuristically scoring those decompositions based on ontological reasoning over the input preferences.

4.5.2.3 Proofs by theorems

According to Manna and al. (Manna and Waldinger, 1983) (Abadi and Manna, 1986)The deductive approach is a formal program-construction method in which the derivation of a program from a given specification is regarded as a theorem proving task. To construct a program whose output satisfies the conditions of the specification, we prove a theorem stating the existence of such an output. The proof is restricted to be sufficiently constructive so that a program computing the desired output can be extracted directly from the proof. The program we obtain is applicative and may consist of several mutually recursive procedures. The proof constitutes a demonstration of the correctness of this program.

In order to enable markup and automated reasoning technology to describe, simulate, compose, test, and verify compositions of Web services, Narayanan and al. (Narayanan and McIlraith, 2002) take as a starting point the DAML-S

DAML+OIL ontology for describing the capabilities of Web services. After, they define the semantics for a relevant subset of DAML-S in terms of a first-order logical language. With the semantics in hand, they encode their service descriptions in a Petri Net formalism and provide decision procedures for Web service simulation, verification and composition. They also provide an analysis of the complexity of these tasks under different restrictions to the DAML-S composite services we can describe. The implementation of this approach takes as input a DAML-S description of a Web service, automatically generates a Petri Net and performs the desired analysis. Such a tool has broad applicability both as a back end to existing manual Web service composition tools, and as a stand-alone tool for Web service developers.

Raoa and al. (Rao et al., 2004) (Rao et al., 2006) introduced a method for automatic composition of semantic Web services using Linear Logic (LL) theorem proving. The method also uses semantic Web service language (DAML-S) for external presentation of Web services, while, internally, the services are presented by extra logical axioms and proofs in LL. They use a process calculus to present the composite service formally. The process calculus is attached to the LL inference rules in the style of type theory. Thus the process model for a composite service can be generated directly from the proof. The sub typing rules that are used for semantic reasoning are presented with LL inference figures. The approach proposes a system architecture where the DAML-S translator, the LL theorem prover and the semantic reasoner can operate together to fulfill the task.

4.5.2.4 Rule-based system

The rule-based systems model the aspects of web services with particular reference to preconditions and effects when executing queries. Thus, each service is linked to a rule created at the system level to specify that the effects that are achieved when the pre-conditions are verified. A composite service is further specified by an initial state and a final state (Rao and Su, 2005).

Rule-based systems are being applied to tasks of increasing responsibility. Deductive methods are being applied to their validation, to detect flaws in these systems and to enable us to use them with more confidence (Waldinger and Stickel, 1992). Thus, each system of rules is encoded as a set of axioms that define the system theory and the operation of the rule language and information about the subject domain are also described in the system theory. Validation tasks, such as establishing termination, unreachability, or consistency, or verifying properties of the system, are all phrased as conjectures.

Ponnekanti and al. (Ponnekanti and Fox, 2002) address a particular subset

of this problem with SWORD, a set of tools for the composition of a class of web services including *information-providing* services. In SWORD, a service is represented by a rule that expresses that given certain inputs, the service is capable of producing particular outputs. A rule-based expert system is then used to automatically determine whether a desired composite service can be realized using existing services. If so, this derivation is used to construct a plan that when executed instantiates the composite service. As our working prototype and examples demonstrate, SWORD does not require (but could benefit from) wider deployment of emerging service-description standards such as WSDL, SOAP, RDF and DAML. This SWORD's method is different to some other plausible existing approaches, especially information integration. They show that although SWORD's expressive capabilities are weaker, the abstractions it exposes capture more appropriately the limited types of queries supported by typical Web services and thus result in simplicity and efficiency.

4.5.3 Dependency Graph approach

Several works have designed the services composition by dependencies between services using different types of graphs. Sometimes one has used oriented, undirected, weighted or acyclic graphs. Any service dependency graph consists of a set of nodes and arcs that translate inter-node relationships derived from the descriptions of these services. Nodes can represent different elements such as services, operations, parameters, preconditions, effects, etc. according to the description formalism chosen. Two nodes are connected by an arc if there is a syntactic or semantic matching which reveals their dependence. Thus, the response to a complex query during service composition consists of searching the service graph for all possible paths that from the request entries lead to the expected outputs. The path of these graphs can be done by search algorithms such as Forward chaining, Backward chaining, A *, Floyd-Warshall. They also allow you to select the best plan based on the defined preference criteria. These criteria may include customer constraints and requirements, often in terms of quality of service and context. Several dependency graph approaches have been exploited in the dynamic composition of services.

Hashemian and al. proposed a technique takes advantage of graph structures and also a particular formalism called interface automata to construct the graph when executing the client's request (Hashemian and Mavaddat, 2005).The method aimed at searching among Web services in order to find those whose composition provides a specific behavior. Those Web services found after this search are incrementally composed together to build a new service that realizes that behavior.

To reach the true potential of such a distributed infrastructure, Gekas and al. (Gekas and Fasli, 2005) proposed to combine such autonomic services together as parts of a workflow, in order to collectively achieve combined functionality. They proposed an automatic workflow composition among web services with semantically described functionality capabilities. For that purpose, they are using a set of heuristics derived from the connectivity structure of the service repository in order to effectively guide the composition process. The methodologies described have been inspired by research in areas such as citation analysis and bibliometrics. Its execution prior to the execution of requests to optimize the composition time is faced with the problem of updating the graph at each change which is a complex and time-consuming operation.

To meet the user's requirement to compose automatically services at runtime, Li and al. (Li et al., 2011) proposed a new architecture of service composition based on cloud computing technology to achieve the mapping between the cloud service and abstract service, and the service parameter relationship graph based on semantic was put forward to achieve automatic real-time service composition at runtime. They use an oriented graph structure representing dependencies between services and their inputs and outputs. This graph is built before the composition phase in order to optimise search times when executing client requests. The nodes represent services or its input or output parameters. Each node is weighted using an overall QoS weight previously calculated and associated with the service in question. Here, service selection is performed using backward chaining backhaul techniques and the first depth-of-search search based on QoS.

Ying and al. (Ying, 2010) propose a *Gmax* chaining technique preferably forward chaining to build a maximal service composition graph which initialized from the entries of the user request and with the starting node of the *Gmax*, the node whose outputs are equivalent to the input parameters of the query. The two stages of service discovery and matching of the inputs and outputs of these services are subsequently altered until the output parameters of the user request are obtained from those of the discovered services. All sub graphs representing composition plans that respond to the client request are included in the graph *Gmax*.Thus, based on the detection of shared nodes and by eliminating redundant nodes, we can extract a *Gmin* from a *Gmax* and propose as different composition plans satisfying the user request.

To provide a good automatic service composition algorithms that not only synthesise the correct work plans from thousands of services but also satisfy the quality requirements of the users some researches (Jiang et al., 2010) have designed and implemented a tool QSynth to use QoS objectives of service requests as the search directives. This approach effectively prunes the search

space and significantly improves the accuracy of the search results. Evaluations show that compared to the state of the art, QSynth achieves superior scalability and accuracy with respect to a large variety of composition scenarios.

An another method of automatic composition plan creation that relies on automatic extraction of dependencies among services is proposed by Omer and al. (Omer and Schill, 2009). For automatic dependency extraction this approach makes use of semantic similarities between Inputs and Outputs parameters of services. Extracted Inputs/Output dependencies are represented using a directed graph. The approach recognizes when cyclic dependencies exist and proposes a way of dealing with it. Modified topological sorting algorithm is used for the execution plan generation showing execution order of candidate services.

An A* algorithm which solves the problem of semantic input-output message structure matching for web service composition and purpose to deal with different issues like performance, semantics or user restrictions is presented by Rodriguez-Mier and al. (Rodriguez-Mier et al., 2011). Given a request, a service dependency graph with a subset of the original services from an external repository is dynamically generated. Then, the A* search algorithm is used to find a minimal composition that satisfies the user request. Moreover, in order to improve the performance, a set of dynamic optimization techniques has been implemented over the search process. The generation of composition plans is an important step for the behavioral description of the new composite service created.

Observations : the dynamic generation and selection of composition plans is an important stage in service composition. However, its optimization only has a direct impact on the number of services and the specificities of the user's request.

4.6 Composite service description

The composite service is described by the behavioral aspect. The behavioral description consists in describing the order of invocation of the operations of a service. A service can have several behavioral descriptions such that each description is a view of the possible behavior of the service in a given interaction. It is a set of information about the internal and external behavior of the composite service that allows, on one hand, to execute its internal services in the expected order when this Web service is invoked, and on the other hand, to interact correctly with others external services. This type of property is particularly critical in the web services composition. This description pro-

vides details on an important aspect of this service and must necessarily be mentioned in its description. There are two ways to describe the sequence of activities that makes up a business process: service orchestration and service choreography (Peltz, 2003).

4.6.1 Orchestration

Service orchestration represents a single executable business process that coordinates the interaction among the different services, by describing a flow from the perspective and under control of a single endpoint (Sheng et al., 2014). Orchestration describes the sequence of services according to a predefined framework and executes them as a set of actions to be performed through web services (Sadiq and Racca, 2003). As described in 4.2, it is an external behavior of a composite service which refers to how the services invoked at a service level are aggregated to provide a more complex feature. Orchestration can therefore be considered as a construct between an automated process and the individual services that enact the steps in the process. This includes the management of the transactions between the individual services and the error handling, as well as describing the overall process. WS-BPEL is the standard for Web services orchestration designed by OASIS (Organization for the Advancement of Structured Information Standards) which is largely supported by the industry.

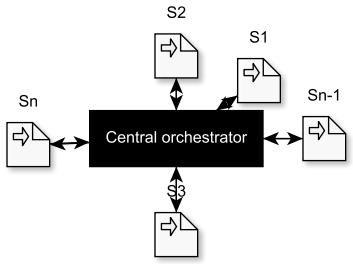

Figure 4.2 – Orchestration

In WS-BPEL, to describe the orchestration of web services, we distinguish between primitive and structured activities. The primitive activities, namely, *Receive*, *Reply*, or *Assign*, describe how the process performed by the composite web service should react to the messages being exchanged. *Receive* marks the expectation of receiving a message by blocking this process. *Reply* responds to a received message and *Assign* updates the values of the process

variables following the completion of the activities. Structured activities, such as *Sequence*, *Switch*, or *Split*, combine several primitive activities to describe different types of branching in a process.

4.6.2 Choreography

The description of the possible interactions between web services is named choreography. Choreography represents a global description of the observable behavior of each of the services participating in the interaction, which is defined by public exchange of messages, rules of interaction and agreements between two or more business process endpoints (Sheng et al., 2014). As described in 4.3 and according to WSMO, choreography represents the messages exchanged between a service and its user to fulfill the functionality of the service. WSMO differentiates between performance choreography and meta choreography: the first challenge is an interaction protocol, while the second challenges a negotiation and monitoring protocol execution. Choreography is typically associated with the interactions that occur between multiple Web services rather than a specific business process that a single party executes.

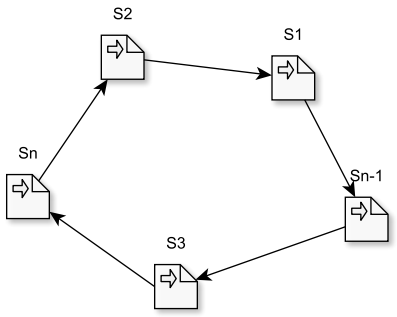

Figure 4.3 – Choreography

The choreography mechanism is supported by the standard WS-CDL (Web Services Choreography Description Language) (Kavantzas et al., 2005). This language is based on XML and has been recommended by the W3C consortium. This is a specification that allows you to model the choreography by describing the peer-to-peer collaboration between the participating web services, but from a global point of view (Kavantzas et al., 2005). It seems like an extension of WSDL web services because it gives the definition of the interactions between participants involved by describing their observable behaviors.

In this sense, WS-CDL provides a global model that ensures the coherence of interactions between cooperating services. In particular, the choreography, which he describes, guarantees a contractual behavior between multiple services without resorting to a complex interconnection tool. The execution of WS-CDL is independent of platforms and programming languages. Its main features (Arkin et al., 2002) are:

— the specification of a multi-party common contract which gives a description of the services' behavior and ensures their interoperability;

— the management of exceptions in the exchange of messages between web services;

— and the security, which aims to describe the exchanges of messages between web services.

The behavioral description of a composite service is fundamental because it makes it possible to describe the interactions between the atomic services involved into the process of composition, and to provide sufficient information on their cooperation contracts. Whether internal or external, it influences the architectural organization of service composition processes.

With the advent of cloud computing, several research studies have identified the composition as a new service.

Observations : the work carried out in the modeling of composite services aims to optimize the generation and selection of service composition plans. This work therefore relates to a single step of the process and only has a direct impact on the number of services and the specificities of the user's request.

4.7 Cloud computing and microservices

Dynamic service composition with microservice is likely to gain more popularity in the future according to the rapid adoption of the new Function as a Service (FaaS) cloud model (e.g., Amazon Lambda) and in case of Composition as a service (Caas). In fact, Cloud computing presents an efficient managerial, on-demand, and scalable way to integrate computational resources (hardware, platform, and software). According to Vaquero and al. (Vaquero et al., 2008) clouds are large pools of easily usable and accessible virtualized resources (hardware, development platforms, and services). These resources can be dynamically reconfigured to adjust to a variable load (scalability), and to allow optimal resource utilization. This pool of resources is typically exploited by a pay-per-use model. In recent years, there has been an increasing

interest in web service composition when cloud computing is gradually evolving as a widely used computing platform where many different web services are published and available in cloud data centers (Jula et al., 2014). Cloud manufacturing has been thus recognized as a transformative manufacturing paradigm to enable rapid production of highly customized products in a networked environment, through on-demand consumption of cloud-based manufacturing services (Lu and Xu, 2017).

As cloud based services become increasingly popular, service composition algorithms that are aware of the cloud selection have deep impacts to the overall efficiency improvement and cost saving for enterprises (Zou et al., 2010). In this base, Zou and al. proposed a framework of web service composition in multi-cloud base environments (Zou et al., 2010). Then, they presented three different cloud combination methods to help service requesters select cloud combination. Their proposed method based on artificial intelligence (AI) planning and combinatorial optimization can more efficiently and effectively find high quality service composition plans with minimal cloud expense.

But, existing cloud architecture lacks the layer of middleware to enable dynamic service composition. To enable and accelerate on-demand service composition, researchers explore the paradigm of dynamic service composition in the Cloud for Pervasive Service Computing environments and propose a Cloud-based Middleware for Dynamic Service Composition (CM4SC) (Zhou et al., 2012). In this approach, the authors introduce the CM4SC *'Composition as a Service'* middleware layer into conventional Cloud architecture to allow automatic composition planning, service discovery and service composition. This new paradigm of Composition as a service enhances the current centralization of the dynamic service composition process. Also, as the fundamental issue of on-demand manufacturing service provision is service composition, resources are mapped to personalized service requests and the research challenge in this is to explore a feasible service composition method that facilitates easy mapping between service requests and manufacturing resources based on restrictive rule sets in the cloud and availability information about a resource (Lu and Xu, 2017). This approach proposed system utilizes distributed knowledge for intelligent service composition and adaptive resource planning. However, to address several challenges when developing and deploying such large-scale systems such as reliability, reproducibility, handling failures on infrastructure, scaling deployment time as composition size grows, coordinating deployment among multiple organizations, dependency management, and supporting requirements of adaptable systems, researchers propose a flexible and extensible middleware solution named CHOReOS Enactment Engine, which is a robust middleware infrastructure to automate the deployment of large-scale service

compositions (Leite et al., 2014).

Observations : the advent of cloud computing to manage scalability in the dynamic service composition has introduced a new paradigm: Composition As a Service (CaaS).This practice is likely to gain more popularity in the future according to the rapid adoption of the new Function as a Service (FaaS) cloud model (e.g., Amazon Lambda) and in case of Composition as a service (Caas) with microservices. The latter entrusts the composition operations to a particular middleware from which each user would request composition plans in relation to his requests. It is a centralized approach which has the merit of offering a dedicated infrastructure but which cannot considerably solve the problem of scalability.

4.8 Decentralization in service composition

Although service composition invokes services distributed over several servers, the dynamic service composition is typically under centralized control. Because performance and throughput are major concerns in enterprise applications, it is important to remove the inefficiencies introduced by the centralized control. In this trend, many researchers have proposed decentralization of service composition as the main mean of eliminating the problems caused during scaling-up.

Nanda and al. (Nanda et al., 2004) proposed a distributed, or decentralized orchestration. In this approach, the BPEL program is partitioned into independent sub-programs that interact with each other without any centralized control. This decentralization increases parallelism and reduces the amount of network traffic required for an application. The method presents a technique to partition a composite web service written as a single BPEL program into an equivalent set of decentralized processes. It gives a new code partitioning algorithm to partition a BPEL program represented as a program dependence graph, with the goal of minimizing communication costs and maximizing the throughput of multiple concurrent instances of the input program. In contrast, much of the past work on dependence-based partitioning and scheduling seeks to minimize the completion time of a single instance of a program running in isolation.

Taking into account a number of factors which affect the QoS of an enterprise system, including availability, scalability and performance, and after evaluation of others distribution patterns, Barrett and al. (Barrett et al., 2006) propose a brand new approach which combines a Model Driven Architecture using UML 2.0 for modeling and subsequently generating Web service

compositions, with a method for achieving dynamic decentralized interaction amongst services with reduced deployment overheads. These approaches combined provide for the generation of dynamic Web service compositions driven by a distribution pattern model.

Falou and al.proposed a decentralized multi-agent approach (Falou et al., 2009). In fact, taking into account the weakness of classical planners based on Planning techniques which are no longer well suited to compose Web services in a reasonable time, they proposed a decentralized multi-agent approach to solve the Web services composition problem at runtime. This model consists of a set of Web service agents where each agent has a set of services organized in a graph. Responding to a request, agents propose partial plans which are partial paths in the graph, and then they coordinate their partial plans to provide the best global plan for the submitted request. The approach is capable of scaling up when compared to the of state-of-the-art techniques for automated web service composition.

Rapti and al. (Rapti et al., 2015) confirmed that traditional service composition approaches rely mostly on centralized architectures, which have been proven inadequate in pervasive Internet of Things (IoT)environments. In such settings, where decentralization of decision-making is mandatory, nature-inspired computing paradigms have emerged due to their inherent capability to accommodate spatiality, self-adaptability and resolvability. So that taking inspiration from natural metaphors they propose a decentralized service composition model which is based on artificial potential fields. In the proposed approach, artificial potential fields (APFs) lead the service composition process through the balance of forces applied between service requests and service nodes. APFs are formed considering the percentage of user requested services that can be offered by service provision nodes, as well as service node availability.

Observations : the decentralization approaches here are applied to certain stages of the dynamic service composition cycle. It can be a distributed or decentralized orchestration and choreography; it can be a decentralized multi-agent approach where each agent has a set of services organized in a graph; it also can be the use of the parallelism in composition plan generation.

4.9 Synthesis on scalability issue on dynamic service composition

The number of services, the complexity of the requests and the number of requests are the 3 main causes of scalability in the dynamic service composition

(Tanenbaum and Van Steen, 2007) (Yu et al., 2008). Being conceived as a series of linked steps, the researchers to deal with scalability problems in the dynamic service composition have provided solutions aiming to optimize each of its steps as we have presented above and summarized by the figure4.4.

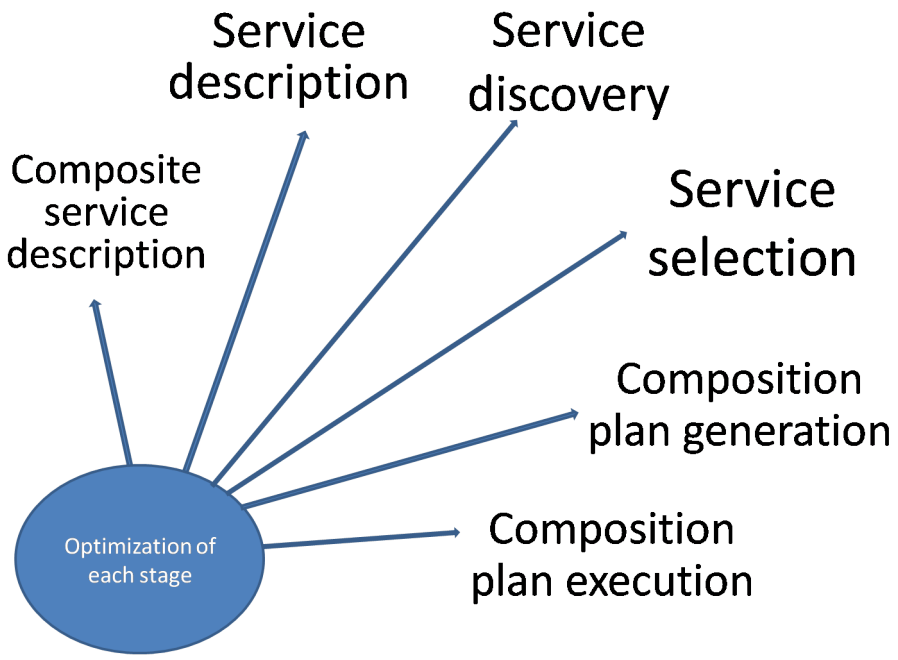

Figure 4.4 – Current approaches on dynamic service composition live cycle

However, despite the satisfaction of these results in isolation, when put together, the process does not return better results in a real work environment. To understand why, we will study the impact of the optimization of each of the process steps on the three referenced causes of scalability.

We present it in the following table

Table 4.1 – Solutions impact on the 3 causes

DSC stages	Number of services	Request complexity	Number of request
Service description	yes	no	no
Service discovery	yes	yes	no
Service selection	yes	yes	no
Composition plan generation	yes	yes	no

Table 4.1 – Solutions impact on the 3 causes

DSC stages	Number of services	Request complexity	Number of request
Composite service description	yes	yes	no
DSC and cloud cumputing	yes	yes	no
Decentralisation in DSC	yes	yes	no

This table shows the different stages in which research has focused to improve the process of dynamic service composition in order to optimize it and reduce the problems linked to scaling up. The table also shows the three known causes of scalability and the link between these solutions and their impacts on these enumerated causes. We can see that concurrent access to requests is the main cause which is not taken into account by the different solutions. This leads us to formulate our research question.

4.10 Research question

A major drawback in most if not all technical composition is the lack of scalability (Baryannis and Plexousakis, 2010) (Lecue and Mehandjiev, 2009). Despite the work done by the research community on the automatic and dynamic service composition, it remains a process that generates a scalability problem in case of large data (Rostami et al., 2014). This is the main obstacle to its adoption as an effective means of development instead of manual and static composition (Baryannis and Plexousakis, 2010). Even in e-government implementation based on SOA, scalability is the main issue in case of larger number of citizens and businesses served (Lofstedt, 2012a).

In fact, as presented by the figure4.5 among the 3 causes of scalability enumerated by to YU and al. (Yu et al., 2008) and Tanembaum and al. (Tanenbaum and Van Steen, 2007) (the number of services, the number of user's requests and the complexity of the requests), the literature review above shows that the proposed solutions deal only with 2 of them: the number of services and the complexity of the requests. Thus, concurrent access of user's request is not taken into account by these proposed approaches.

That is why to deal with this aspect, the question is: **how to take into account the concurrent access of customers to deal with scalability problems in the automatic and dynamic composition of services?**

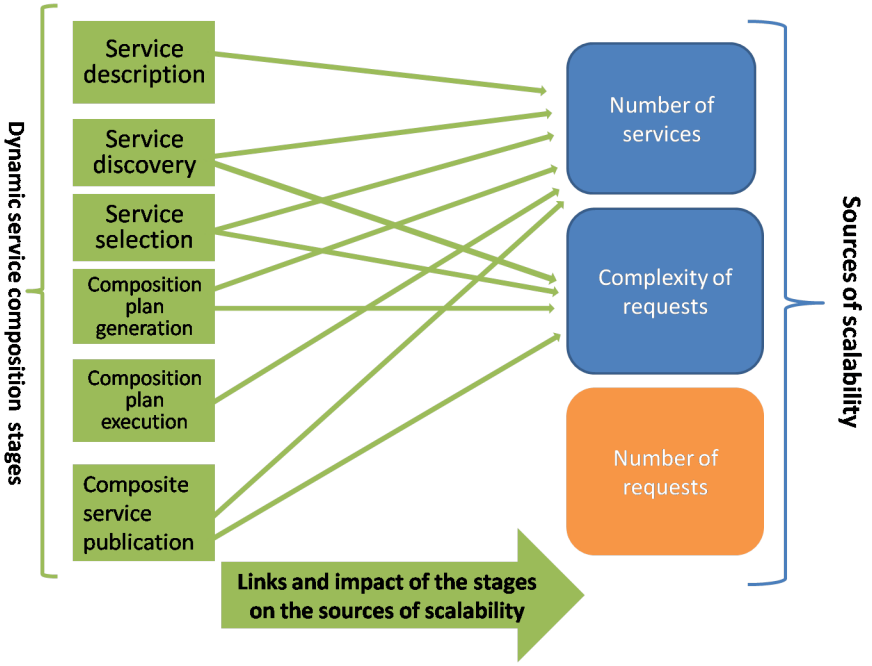

Figure 4.5 – Links and impact of the stages on the sources of scalability

4.11 Synthesis

This chapter presented the steps that constitute a dynamic service composition process. Its also presents the different approaches to solving the problems related to this trend. We note that to optimize this process, the activities of each of its stages must undergo improvements that the various works of the researchers have proposed. Some works even propose to begin by better describing and better locating the services to allow to refine and to optimize the process of their dynamic composition. Most of the approaches described to optimize the dynamic composition process have advantages that enhance this activity in each step of the life cycle of a composite service. But, the recurrent problem of scaling up always persists.

So, to deal with the recurring problem of scaling up, the distributed approach of the service composition process seems to be the best mean. Most of the researches proposing suggest parallelism in the processes of discovery, of selection, or of orchestration. But these approaches always organize the process of dynamic service composition around a central infrastructure. They describe the composition from the point of view of a centralized composition service or engine. They all implement a centralized model of composition which is the

main cause of this scalability when data becomes very large.

Also, among the 3 mentioned causes of scalability (Yu et al., 2008), only 2 are taken into account by the proposed solutions. Thus, the researches don't involve the clients of the composition to participate in the service composition process. All researches are done in the total forgetfulness of the possibility of bringing certain composition operations back to the level of the client's infrastructure. It is in this logic of participative collaboration that a distributed approach of dynamic service composition is proposed. This proposes to involve customer infrastructures in the dynamic service composition process to decongest the central server composition infrastructure by decentralizing the publication of composite service. The next chapter is devoted to present this new approach.

The next part will present the distributed approach on which this research is based, the implementation of our proposed framework and it validation.

Part III

RESEARCH IMPLEMENTATION

When the intelligence is uncomfortable, the whole soul is sick

S. WEIL

5

The principles and paradigms of Distributed Systems

Contents

5.1 Introduction

According to Tanenbaum and al.(Tanenbaum and Van Steen, 2007) various definitions of distributed systems have been given in the literature, none of them satisfactory, and none of them in agreement with any of the others. Thus, they propose to give a loose characterization: a distributed system is a collection of independent computers that appears to its users as a single coherent system. It is a system which is in contrast to the previous centralized systems (or single processor systems) consisting of a single computer, its peripherals, and perhaps some remote terminals.

The definition of distributed systems has several important aspects (Tanenbaum and Van Steen, 2007). The first one is that a distributed system consists of components that are autonomous. A second aspect is that users (be they people or programs) think they are dealing with a single system. This means that one way or the other the autonomous components need to collaborate. Therefore, the main question is how to establish this collaboration lies at the heart of developing distributed systems (Tanenbaum and Van Steen, 2007).

5.2 Distributed systems goals

Distributed systems consist of autonomous computers that work together to give the appearance of a single coherent system with many goals (Tanenbaum and Van Steen, 2007).

5.2.1 Making Resources Accessible

The main goal of a distributed system is to make it easy for the users (and applications) to access remote resources, and to share them in a controlled and efficient way. Resources can be printers, computers, storage facilities, data, files, Web pages, networks, costly resources such as supercomputers, high-performance storage systems, image setters, and other expensive peripherals to name just a few. Therefore, there are economics reasons for wanting to share resources (Raymond, 1995).

5.2.2 Distribution Transparency

Another important goal of a distributed system is to hide the fact that its processes and resources are physically distributed across multiple computers. A distributed system can be able to present itself to users and applications as a

single computer system is said to be transparent. The concept of transparency can be applied to several aspects: access, location, migration, relocation, replication, concurrence and failure (Raymond, 1995).

5.2.3 Openness

An open distributed system is a system that offers services according to standard rules that describe the syntax and semantics of those services (Tanenbaum and Van Steen, 2007). In distributed systems, services are generally specified through interfaces, which are often described in an Interface Definition Language (IDL). Interface definitions specify precisely the names of the functions that are available together with types of the parameters, return values, possible exceptions that can be raised, and so on (Raymond, 1995).

5.2.4 Scalability

Scalability is one of the most important design goals for developers of distributed systems because of Worldwide connectivity through the Internet is rapidly becoming as common as being able to send a postcard to anyone anywhere around the world (Raymond, 1995). A system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity (ord Neuman, 1994). Scalability of a system can be measured along at least three different dimensions (ord Neuman, 1994). First a system can be scalable with respect to its size, meaning that we can easily add more users and resources to the system. Second, a geographically scalable system is one in which the users and resources may lie far apart. Third, a system can be administratively scalable meaning that it can still be easy to manage even if it spans many independent administrative organizations.

5.3 Distributed systems types

According to Tanenbaum and al. (Tanenbaum and Van Steen, 2007), distributed systems are composed by distributed computing systems, distributed information systems, and distributed embedded systems.

5.3.1 Distributed Computing Systems

An important class of distributed systems is the one used for high-performance computing tasks. They are two subgroups: cluster computing and grid computing (Tanenbaum and Van Steen, 2007).

— In cluster computing, the underlying hardware consists of a collection of similar workstations or PCs, closely connected by means of a high-speed local-area network. In addition, each node runs the same operating system (Tanenbaum and Van Steen, 2007).

— The grid computing consists of distributed systems that are often constructed as a federation of computer systems, where each system may fall under a different administrative domain, and may be very different when it comes to hardware, software, and deployed network technology (Tanenbaum and Van Steen, 2007).

5.3.2 Distributed Information Systems

Another important class of distributed systems is found in organizations that were confronted with a wealth of networked applications, but for which interoperability turned out to be a painful experience (Tanenbaum and Van Steen, 2007). Therefore, Bernstein and al. (Bernstein, 1996) argued that many of the existing middleware solutions are the result of working with an infrastructure in which it was easier to integrate applications into an enterprise-wide information system. in fact, as applications became more sophisticated and were gradually separated into independent components (notably distinguishing database components from processing components), it became clear that integration should also take place by letting applications communicate directly with each other (Tanenbaum and Van Steen, 2007).

5.3.3 Distributed Pervasive Systems

As its name suggests, a distributed pervasive system is part of our surroundings (and as such, is generally inherently distributed) (Tanenbaum and Van Steen, 2007). An important feature is the general lack of human administrative control. At best, devices can be configured by their owners, but otherwise they need to automatically discover their environment and *'nestle in'* as best as possible (Tanenbaum and Van Steen, 2007). Computational power will be then available everywhere so mobile and stationary devices will dynamically connect and coordinate to seamlessly help people in accomplishing their tasks (Grimm et al., 2004). For this vision to become a reality, developers

must build applications that constantly adapt to a highly dynamic computing environment by following the following three requirements for pervasive applications: embrace contextual changes, encourage ad hoc composition and recognize sharing as the default (Grimm et al., 2004).

5.4 Distributed systems characteristics

Distributed systems are characterized by their architectures, processes, communication, naming, synchronization, consistency and replication, fault tolerance and security.

5.4.1 Architectures

Distributed systems can be seen as a set of complex pieces of software dispersed across multiple machines. To manage them, it is crucial that these systems are properly structured. There are different ways on how to view the organization of a distributed system, but an obvious one is to make a distinction between the logical organization of the collection of software components and on the other hand the actual physical realization (Tanenbaum and Van Steen, 2007).

5.4.1.1 Architectural styles

Architectural styles tell us how the various software components are to be organized and how they should interact.

— Layered architectures: components are organized in a layered fashion where a component at layer L is allowed to call only components which are their direct neighbors.

— Object-based architectures: each object corresponds to what we have defined as a component, and these components are connected through a (remote) procedure call mechanism (Tanenbaum and Van Steen, 2007).

— Data-centered architectures evolve around the idea that processes communicate through a common (passive or active) repository. For example Web-based distributed systems are largely data-centric because processes communicate through the use of shared Web-based data services.

— Event-based architecture processes essentially communicate through the propagation of events, which optionally also carry data. For distributed systems, event propagation has generally been associated with what are

known as publish/subscribe systems in which subscribers register their interest in an event, or a pattern of events, and are subsequently synchronously notified of events generated by publishers (Eugster et al., 2003).

5.4.1.2 System architectures

The system architectures instantiate and place software components on real machines. There are many different choices that can be made in doing it and the final instantiation of a software architecture is also referred to as a system architecture (Tanenbaum and Van Steen, 2007).

— Centralized architectures are the basic client-server model where processes in a distributed system are divided into two groups: a server and a client. A server is a process implementing a specific service and a client is a process that requests a service from a server by sending it a request and subsequently waiting for the server's reply. This client-server interaction, also known as request-reply behavior (Tanenbaum and Van Steen, 2007).

— Decentralised architectures are Multitiered client-server architectures which are a direct consequence of dividing applications into a user-interface, processing components, and a data level. The different tiers correspond directly with the logical organization of applications (Tanenbaum and Van Steen, 2007).

— Hybrid architectures are specific classes of distributed systems in which client-server solutions are combined with decentralized architectures.

5.4.2 Processes

The concept of a process originates from the field of operating systems where it is generally defined as a program in execution. From an operating-system perspective, the management and scheduling of processes are perhaps the most important issues to deal with (Tanenbaum and Van Steen, 2007).In distributed system, processes are modelized by threads and virtualization.

— Threads: Tanenbaum and Van Steen(Tanenbaum and Van Steen, 2007) note that to efficiently organize client-server systems, it is often convenient to make use of multithreading techniques. So a main contribution of threads in distributed systems is that they allow clients and servers to be constructed such that communication and local processing can overlap, resulting in a high level of performance.

— Virtualization: in recent years, the concept of virtualization has gained popularity (Tanenbaum and Van Steen, 2007). This allows an application, and possibly also its complete environment including the operating system, to run concurrently with other applications, but highly independent of the underlying hardware and platforms, leading to a high degree of portability. Moreover, virtualization helps in isolating failures caused by errors or security problems (Tanenbaum and Van Steen, 2007).

5.4.3 Communication

The communication between processes is at the heart of all distributed systems. It the ways that processes on different machines can exchange information. Communication in distributed systems is always based on low-level message passing as offered by the underlying network. There are three widely-used models for communication: Remote Procedure Call (RPC), Message-Oriented Middleware (MOM), and data streaming.

— Remote Procedure Call (RPC): When a process on machine A calls' a procedure on machine B, the calling process on A is suspended, and execution of the called procedure takes place on B. Information can be transported from the caller to the called in the parameters and can come back in the procedure result. No message passing at all is visible to the programmer and data streaming.

— Message-Oriented Middleware (MOM): Likewise, the inherent synchronous nature of RPCs, by which a client is blocked until its request has been processed, sometimes needs to be replaced by something else which is messaging.

— Data streaming: There are forms of communication in which timing plays a crucial role. Let us consider the next example (Tanenbaum and Van Steen, 2007). An audio stream built up as a sequence of 16-bit samples, each representing the amplitude of the sound wave as is done through Pulse Code Modulation (PCM). Also assume that the audio stream represents CD quality, meaning that the original sound wave has been sampled at a frequency of 44, 100 Hz. To reproduce the original sound, it is essential that the samples in the audio stream are played out in the order they appear in the stream, but also at intervals of exactly 1/44, 100 sec. playing out at a different rate will produce an incorrect version of the original sound.

5.4.4 Naming

According to Tanenbaum and Van Steen, a name in a distributed system is a string of bits or characters that is used to refer to an entity (Tanenbaum and Van Steen, 2007). Names have an important role in all computer systems because they are used to uniquely identify entities, share resources or to refer to locations. To allow a process to access the named entity, it is necessary to implement a naming system. The difference between naming in distributed systems and nondistributed systems lies in the way naming systems are implemented (Tanenbaum and Van Steen, 2007).

Anything can be an entity is a distributed system. A name of entity can then be an address or an access point to operate on it. An entity may change its access points in the course of time when moves to another location as a person moves to another city or country, it is often necessary to change telephone numbers as well. An address is therefore a special kind of name which refers to an access point of an entity.

Wieringa and de Jonge have determined some properties which identifier a name (Wieringa and Jonge, 1995):

— An identifier refers to at most one entity;

— Each entity is referred to by at most one identifier;

— An identifier always refers to the same entity (it is never reused).

There are structured, attributed-based and unstructured or flat names.

— Flat names are good for machines, but are generally not very convenient for humans to use. In case of flat names, identifiers are simply random bit strings which are conveniently refer to as unstructured names;

— As an alternative, naming systems generally support structured names that are composed from simple, human-readable names. Not only file naming, but also host naming on the Internet follow this approach.

— There are many ways in which descriptions can be provided, but a popular one in distributed systems is to describe an entity in terms of (attribute, value) pairs, generally referred to as attribute-based naming. In this approach, an entity is assumed to have an associated collection of attributes. Each attribute says something about that entity therefore, by specifying which values a specific attribute should have, a user essentially constrains the set of entities that he is interested in (Tanenbaum and Van Steen, 2007).

5.4.5 Synchronization

A problem in distributed systems is that there is no notion of a globally shared clock so that processes on different machines have their own idea of what time it is. It is important that multiple processes do not simultaneously access a shared resource, such as printer, but instead cooperate in granting each other temporary exclusive access (Tanenbaum and Van Steen, 2007). The synchronization can be implemented through various methods.

— Clock synchronization: It is a method where an accurate account of time is needed because time is unambiguous in a centralized system. So if a process wants to know the time, it makes a system call and the kernel tells it. If process A asks for the time and then a little later process B asks for the time, the value that B gets will be higher than (or possibly equal to) the value A got. It will certainly not be lower. In a distributed system, achieving agreement on time is not trivial.

— Logical clocks: According to Tanenbaum, clock synchronization is naturally related to real time so that it may be sufficient that every node agrees on a current time, without that time necessarily being the same as the real time.

— Mutual exclusion: In many cases in distributed systems, processes will need to simultaneously access the same resources. Mutual exclusive access by processes is a means through which the corruption of the resource can be prevent.

— Global positioning of nodes: When the number of nodes in a distributed system grows, it becomes increasingly difficult for any node to keep track of the others. Then global positioning of nodes may be important for executing distributed algorithms such as routing, multicasting, data placement, searching, and so on. In geometric overlay networks each node is given a position in an 111 dimensional geometric space, such that the distance between two nodes in that space reflects a real-world performance metric (Tanenbaum and Van Steen, 2007).

— Election algorithms: Many distributed algorithms require one process to act as coordinator, initiator, or otherwise perform some special role. In general, it does not matter which process takes on this special responsibility, but one of them has to do it. In general, election algorithms attempt to locate the process with the highest process number and designate it as coordinator. The algorithms differ in the way they do the location (Tanenbaum and Van Steen, 2007).

5.4.6 Consistence and replication

Data are generally replicated to enhance reliability and performance. One of the major problems is that when one copy is updated we need to ensure that the other copies are updated as well; otherwise the replicas will no longer be the same.

— Data are replicated to increase the reliability of a system: If a file system has been replicated it may be possible to continue working after one replica crashes by simply switching to one of the other replicas. Also, by maintaining multiple copies, it becomes possible to provide better protection against corrupted data.

— Data are replicated to increase the performance: Replication for performance is important when the distributed system needs to scale in numbers and geographical area. Scaling in numbers occurs, for example, when an increasing number of processes needs to access data that are managed by a single server. Scaling with respect to the size of a geographical area may also require replication. The basic idea is that by placing a copy of data in the proximity of the process using them, the time to access the data decreases (Tanenbaum and Van Steen, 2007).

Any distributed system that supports replication is to decide where, when, and by whom replicas should be placed, and subsequently which mechanisms to use for keeping the replicas consistent.

5.4.7 Fault tolerance

The fault tolerance in distributed systems is strongly related to availability, reliability, safety and maintainability.

— A system is available means that a system is ready to be used immediately. This property refers to the probability that the system is operating correctly at any given moment and is available to perform its functions on behalf of its users.

— A system is reliable means that a system can run continuously without failure. This property is defined in terms of a time interval instead of an instant in time. A highly-reliable system is one that will most likely continue to work without interruption during a relatively long period of time.

— Safety refers to the situation that when a system temporarily fails to operate correctly for only a very brief moment, the effects couldn't be disastrous and nothing catastrophic happens.

— A system is maintainable when it is easy to repair it when he is failed. A highly maintainable system may also show a high degree of availability, especially if failures can be detected and repaired automatically.

When a system has these characteristics above, then it can be considered as demonstrates fault tolerant. And that is a good attribute for a system in general, and distributed system in particular.

5.4.8 Security

Security in a computer system is strongly related to the notion of dependability and dependability includes availability, reliability, safety, and maintainability (Tanenbaum and Van Steen, 2007). But it is important to put our trust in a computer system, then confidentiality and integrity should also be taken into account (Tanenbaum and Van Steen, 2007).

— Confidentiality refers to the property of a computer system whereby its information is disclosed only to authorized parties.

— Integrity is the characteristic that alterations to a system's assets can be made only in an authorized way. In other words, improper alterations in a secure computer system should be detectable and recoverable. Major assets of any computer system are its hardware, software, and data. Another way of looking at security in computer systems is that we attempt to protect the services and data it offers against security threats (Tanenbaum and Van Steen, 2007).

5.5 Relevance of distributed approach in information system

Many platforms have adopted the distributed approach to improve the performance of their systems. This has led to the emergence of new architectures in several IT development contexts like distributed object based systems, distributed file systems, distributed web based systems and distributed coordination based system.

5.5.1 Distributed object based systems

The key feature of an object is that it encapsulates data called the state, and the operations on those data, called the methods. There is a separation between interfaces and the objects implementing these interfaces and this is

crucial for distributed systems because it allows to place an interface at one machine, while the object itself resides on another machine.

The notion of an object plays a key role in establishing distribution transparency in distributed object-based systems. This paradigm treats everything as an object and clients are offered services and resources in the form of objects that they can invoke. Distributed objects form an important paradigm because it is relatively easy to hide distribution aspects behind an object's interface and furthermore, because an object can be virtually anything, it is also a powerful paradigm for building systems (Tanenbaum and Van Steen, 2007). The principles of distributed systems are applied to a number of well-known object-based systems like CORBA, Java-based systems, and Globe.

5.5.2 Distributed file systems

To share data is fundamental to distributed systems so that distributed file systems form the basis for many distributed applications. They form an important paradigm for building distributed systems which are generally organized according to the client-server model, with client-side caching and support for server replication to meet scalability requirements. In addition, caching and replication are needed to achieve high availability. Distributed file systems allow multiple processes to share data over long periods of time in a secure and reliable way (Tanenbaum and Van Steen, 2007). As such, they have been used as the basic layer for distributed systems and applications. NFS and C0DA are the main examples of distributed file systems.

5.5.3 Distributed web-based systems

Since 1994, Web developments have been initiated by the World Wide Web Consortium and this consortium is responsible for standardizing protocols, improving interoperability, and further enhancing the capabilities of the Web (Tanenbaum and Van Steen, 2007). In addition, many new developments take place outside this consortium, not always leading to the compability one would hope for (Tanenbaum and Van Steen, 2007). By now with the introduction of Web services we are seeing a huge distributed system emerging in which services rather than just documents are being used, composed, and offered to any user or machine that can find use of them.

Service oriented architecture is a part of distributed web based systems then many concepts underlying Web technology are based on the principles discussed in the first chapter**??**.

5.5.4 Distributed coordination based system

The distributed coordination based systems is a generation of distributed systems that assume that the various components of a system are inherently distributed and that the real problem in developing such systems lies in coordinating the activities of different components (Tanenbaum and Van Steen, 2007). This means that instead of concentrating on the transparent distribution of components, emphasis lies on the coordination of activities between those components.

Key to the approach followed in coordination-based systems is the clean separation between computation and coordination and many conventional distributed systems are gradually incorporating mechanisms that play a key role in coordination-based systems (Tanenbaum and Van Steen, 2007).

5.6 Relevance of distributed approach in the present study

The present study is focused on service oriented architecture. It is a part of distributed web based system implementation. The main problem this study is resolving is the scalability. The distributed approach has among its goals the scalability so it is a good mean to deal with it.

Having discussed some of the scalability problems brings us to the question of how those problems can generally be solved. In most cases, scalability problems in distributed systems appear as performance problems caused by limited capacity of servers and network. There are now basically only three techniques for scaling: hiding communication latencies, distribution, and replication.

The techniques used in this study are the replication, the distribution and the synchronization. The study will propose the model for composition plan replication and the service registry clustering distribution. The study will also propose the mechanisms of coordination of the whole process.

5.7 Synthesis

Various aspects of real-life systems such as WebSphere MQ, DNS, GPS, Apache, CORBA, Ice, NFS, Akamai, TIBlRendezvous, Jini, and many more examples illustrate the thin line between theory and practice, which makes distributed systems such an exciting field.

Distributed systems have made an important contribution to the interoperability of information systems. In this chapter, some of the fundamental principles on which they are based have been exposed. The scalability which is one of the major strengths of this paradigm is a real way of motivation that can make the dynamic service composition more effective. The automatic and dynamic composition of services is a present trend (Benatallah et al., 2003a) and this approach is the most adapted for its flexibility and adaptability in relation to the dynamism of technological environments and especially in relation to the proliferation of services and stakeholders in the web.

The current dynamic service composition approach raises a scalability problems related to its effectiveness that research seeks to provide solutions. So, achieving dynamic composition in an environment where the number of services provided is constantly changing and the number of users is always increasing is not a simple task. In the next chapter, there will be a presentation of the dynamic service composition based on distributed approach to deal with the scalability issue.

A logical equivalence goes beyond a simple numerical comparison because it takes into account the entire operating environment of the models to evaluate their performance; whereas the numerical equivalence simply makes a comparison between the values of a single element of the systems.

ATSA ETOUNDI Roger

6

Distributed Dynamic Service Composition (2DSC) in a System Scalability

Contents

6.1 Introduction

Service-oriented architectures brought a great evolution in the interoperability of information systems. This progress relies on a set of web technologies and standards. The dynamic service composition, despite its advantages over the static composition, it is slow in its execution because of many operations it imposes during its process at runtime. This situation leads to scalability problems as data becomes larger and larger. To make these processes effective even when scaling up, a lot of work has been done.

The preceding chapter**??** has extensively presented the work done to improve each of these phases, including the interpretation of the request, the search for services and their selection, the composition plan's generation and the selection of the best, the execution and publication of the new service. The previous chapter has also swept away important work such as the description of services that may facilitate their handling in the other stages of the composition and also the description of a composite service. This preceding chapter also presented the means implemented to describe interactions between services.

This chapter is dedicated to the presentation of a Distributed Dynamic Service Composition (2DSC) to make this process effective. He will first present some notions needed to the next, the principle of our method, and the algorithm of the solution before its formalization and its theoretical validation. But the chapter begins with the preliminaries to well present our solution and with the reminder of the research question.

6.2 Preliminaries

This section is dedicated to the preliminarie notions. Then, some concepts will be presented. This will be the main knowledge which will allow us to better understand the solution that will be presented later. It will begin with a motivation example able to better present the treated issue.

6.2.1 Motivation example

It is always easier to make a problem perceptible through one or more examples. In the context of the presentation of the dynamic service composition problem, a good example would also be an important illustration mean.

Take an information system that establishes and generates, among other things, the criminal record of a citizen. Let us figure out he has to collaborate

with a number of other partner government systems. It is assumed that all these entities have implemented service-oriented based systems to facilitate and improve their exchanges and collaborations. This presupposes the existence of a service registry in which all available services are published. Let us imagine that there is a composition server that can generate the composition plans for a received request (Blake et al., 2010). Let us suppose also that the social security number is the identifier of each individual.

When the system receives a request from a user wishing to generate the criminal record of an individual A, the request is sent to the composition server to search for and classify the services that can contribute to the achievement of this objective. The composition server will then serve to the system a composition plan that the latter will undertake to execute and return the result to the user. But if a user requests the system again to generate another criminal record, the request is sent again to the composition server which will probably answer again with the same composition plan. However, these multiple and sometimes unnecessary solicitations have a negative impact on the performance of the composition server and on the client system. This is one source of scalability among the two others that researches have treated. Because the processing of each request is dedicated to the composition server exclusively.

And yet, it would be advantageous to generate a distributed approach, creating for the client system, a registry in which the composition plan executed successfully is saved. This would decentralize the dynamic composition approach and allow the remote composition server to be entered only after an unsuccessful search of the locally created registry. It is therefore this distributed solution that we propose to solve the problem of scalability in automated dynamic service composition activities in SOA. It is a decentralized manner and personalized service composite publication which seems like a mean to build a self composition and discovery mechanism (Zimeo et al., 2008). The appropriate mechanism is based on distributed architecture.

6.2.2 NFS architecture

NFS (network files system) is a feature of UNIX systems that consists of sharing files across a network. It makes it possible to decongest the central system by the use of the cache memories in its central memory and in the requesting systems.

In fact, when a remote user wishes to have a shared file, his request is first processed locally. This local processing consists in checking if the cache of its system contains this file to put it directly at the disposal of the user. If not,

its request is then transmitted to the central server. The latter searches first if the file is presented in its cache. If this is the case, the file is served to the user. If, on the other hand, the file is not available in the server cache, the server will immediately search for it on its disk and return it to the user.

This method allows the system to gain a lot of time during transaction file sharing procedures with its clients. It is a system that uses the clients' infrastructure as a backup point to give them what they need from their activities with the server. This prevents a client from making remote calls to find a file when it has become routine to solicit it regularly.

The following algorithm illustrates this mode of operation.

Algorithm 1 Basic NFS file system

```
1: FindLocalCache
2: if (CacheMiss) then
3:    FindServerCache
4:    if (CacheMiss) then
5:       FindDisk
6:    end if
7: end if
```

Algorithm 2 Basic2DSC algorithm

```
1: FindLocalComposition
2: if (LocalCompositionMiss) then
3:    FindRemoteServerCompositionCache
4:    if (RemoteServerCompositionCacheMiss) then
5:       FindRemoteServerComposition
6:    end if
7: end if
```

6.3 Distributed Dynamic Service Composition (2DSC)

This section aims to formalize the 2DSC system. This consists of formalizing some illustrations. It also consists of formalizing the main components and mechanism of the model. Among them, we cite local library generation, the local registry generation and the monitoring between the composition server and others local devices.

6.3.1 Basic idea

Our basic idea in this chapter can be exposed as follows. Indeed, we consider a dynamic service composition operation as a composition path-sharing system in which the composition paths are the searched information; the requests are the client requests that are addressed to the central composition engine being server. In this architecture, dynamic service composition is perceived as a distributed operation in which it is easier to contain the flow of requests sent to the same infrastructure using proxy servers. Thus, when the processing of a request has given a composition plan executed successfully, this path must be kept and saved at the client system that issued the request to probably serve in another request of the same type.

To illustrate the problem that the centralization of the dynamic composition process, let us take the example of a request Q of a client S addressed to the composition engine M. After processing this request, a composition plan P_S is returned to him for execution. Let us suppose that after a time T, S returns another similar request Q to M for a second time. We observe here that this treatment could be avoided if in the local system of S, there existed a library of composition plan in which the local compositor can refer to before the system S send it to M. It is therefore one of the major challenges of the participation of customer systems in the policy of dynamic service composition.

Even more importantly, if the client's composition plans are well executed, the services that it contents are sufficiently adapted to this client. That is, when selecting each service belonging to the plan, a set of descriptive information of this service has already been taken into account during this selection step. The QOS applied to these services make them therefore refined enough to make it possible to build a personal registry of services more appropriate to the client's activities. It is therefore both a method of clustering services adapted to customers and optimization of the dynamic composition that can be done by using the infrastructures of the customers.

The construction of this library of composition plans and local registry will give to the system the possibility to decongest the composition server, to build a personalized service for all the clients and to make a local composition to all clients of the system.

6.3.1.1 Decongest of the composition plan server

This will avoid similar treatments to the system by keeping locally the composition plans needed to process the requests already sent to the composition

system. This perspective based on the NFS mechanism avoids the composition server to receive repeated requests of the same nature and from the same customers. This will necessarily decongest this infrastructure and allow it to become more efficient for processing new queries. It also brings the customer closer to the information he often needs to provide effective responses to his users.

6.3.1.2 Service clustering

This model will facilitate a clustering of the service registry according to the precise requests of the users. Service clustering allows you to select and match a set of services with a high probability of participating in the same service composition activities. Taking into account the composition plans received from the composition server, each client locally builds its service register, which is already refined in terms of quality of service (QoS) and composability.

6.3.1.3 Local composition plan generator

This model can facilitate the establishment of a local composition engine based on the resources available in the local registry and the composition plan library. With the availability of a set of service descriptive information, a simple composition algorithm can be effective for locally producing service plans that can respond to requests from the customer's users. This algorithm will benefit from treatments already done during the preliminary selection of services during the remote composition stage.

To set up this mechanism, it would be necessary to define in a specific way the concepts that will be manipulated in the following for a better comprehension.

6.3.2 Concept's specification

This part gives a precise meaning to each of the notions used in the proposed approach.

6.3.2.1 semantically request similarity

When sharing similar demands, enterprises are using their specific vocabulary and structural representations for modeling business processes (Ehrig et al., 2007). The semantically request similarity aims to evaluate the degree of semantic correspondence between two or many requests.

According to Michael Richter (Richter, 1993), a real-valued function $Sim : S \times S \rightarrow [0, 1]$ on a set S measuring the degree of similarity between two elements is called similarity measure if $\forall Q_1, Q_2 \in S$ the basic axioms for similarity of Q_1 and Q_2 are:

— $Sim(Q_1, Q_1) = Sim(Q_2, Q_2) = 1$: **reflexivity**;

— $Sim(Q_1, Q_2) = Sim(Q_2, Q_1)$: **simmetry**;

The request similarity can be measured on its syntactic, linguistic and structural aspects. Its complete measure is therefore the combination of these three aspects. Thus, in order to compute the combined similarity Sim_{com} between concept instance names $c1$ and $c2$ let $c1$ be a particular concept instance name of Q_1 and $c2$ be a concept instance name of Q_2. Then the combined similarity is an aggregation of the degrees returned from the syntactical, linguistic and structural similarity measures as explained above:

$Sim_{com}(C_1, C_2)=$

$$(\frac{w_{syn} * Sim_{syn}(Q_1, Q_2) + w_{lin} * Sim_{lin}(Q_1, Q_2) + w_{str} * Sim_{str}(Q_1, Q_2)}{w_{syn} + w_{lin} + w_{str}})$$

Where w_{syn}, w_{lin} and w_{str} are respectively the weights of syntactical, linguistic and structural similarities. The weightings of each type of similarity can be given by each user according to the priorities of their environment.

The similarity between two semantic request Q_1, Q_2 is defined by semantic relationships, which we consider by the two sets of concept instances $C_1 and C_2$ of $Q_1 and Q_2$.

— Equivalence:$Sim(Q_1, Q_2) = 1$ if $C_1 = C_2$;

— Disjointedness:$Sim(Q_1, Q_2) = 0$ if $C_1 \cap C_2 = \phi$;

— Intersection: $Sim(Q_1, Q_2) \in]0...1[$ if $C1 \cap C2 = \{x|(x \in C_1) \wedge (x \in C_2)\} \wedge C_1 \neq C_2$.

6.3.2.2 Local request matching

The local matching of a request is the semantic study which allows the prior study of its similarity with other requests already processed by the system. This operation makes possible to associate to the new request the composition plan of the old one for its execution when there is a similarity between a new request and an old one. It is during this operation that the system chooses to send or not a request to the central composition engine to receive a plan that composition adapted for its processing. Let Q_1 an old request that has been processed by a composition plan P_1. Let Q_2 be a new request being processed.

If $Sim(Q_1, Q_2) = 1$ then, P_1 can be executed for Q_2. Instead of sending Q_2 to the central composition engine, we simply associate Q_2 to the composition plan P_1 and execute it. We can note:

$R_{Match}(Q, Rep)$=P if $Sim(Q, Q_1)$=1$\wedge P$ is the composition plan of Q_1. With R_{Match} the request matching function and Rep the set of composition plan composing the repository of matching;

6.3.2.3 Local library

The local library is a directory in which are saved all the composition plans whose execution was satisfied and their associated requests. Indeed, it is the main directory that permits the matching of new requests. The local library is therefore an essential tool in which updates must be made when:

— When the execution of a composition plan is well proceeded, this request and its semantic description is kept in local repository as well as the associated execution plan;

— if by associating a composition plan to a new request, if the execution of this plan is no longer well proceeded, and then a new matching must be carried out by removing this defective composition plan from these new matching operations.

6.3.2.4 Service monitoring

Monitoring consists of modifying a composition plan already registered in the local library. This operation is carried out to inform in the following cases:

— The owner of a service has modified the service: here, it is a question of updating the information relating to one or more services belonging to one or more composition plans of the local library according to its new description received from the registry through the central composition engine;

— The owner of a service has removed it: here, it is always a matter of updating a composition plan of the local library by replacing the services deleted by the new alternative services. These new services must be composed with the other services of the composition plan and have the same functional properties.

6.3.2.5 Local registry

The local registry makes it possible to build a local service cluster of the remote registry based on their participation in the efficient execution of re-

quests received by the system. This method of clustering is more efficient to give a response to a new user request because the services registered therein have been selected by a user based centric filter and on their quality of service preferences. A local composition engine can therefore be started to provide composition plans that can be used to give effective composition plan to new requests without referring to the central composition engine.

6.3.2.6 Local plan generator

Based on a local registry enriched by the system, the local composition plan generator makes it possible to generate the composition plans capable to give solution to requests received in the system without referring to the remote composition engine. This local composition operation is initiated when, after receiving a request, no composition plan from the local library is appropriate for processing the request. It should be noted that to be efficient, this local composition operation can be started if and only if there is a sufficient amount of services in the local directory.

After having defined these concepts, the description of the envisaged process will follow.

6.3.3 Process description

The request arrives in the system. It is sent to the local request matching. The latter verifies if there is a composition plan that can process the request in the local library. If this is the case, the request is processed directly and the response is sent to the user. If the local matching composer does not find an appropriate composition in the local plan composition library, the request is transmitted to the remote central composer which processes it by examining its composition cache. If this cache of the central composer does not find a path that can answer the request, the composer then starts the dialing operation by relying on directories. It finds the services corresponding to the request and places them in a satisfactory order of execution. It sends the composition plan to the requesting system. It then updates his central composition cache.

The requesting system executes the path passed to it. If the execution is good then it updates its local composition library. Otherwise, it informs the central composer which stimulates the composition and sends him a new plan. The process begins again until the satisfaction of the requesting system or until the composer has transmitted the entire possible plan. A monitoring system is set up to update the various registers according to the activities in the directories. For example, if a service involved in a composite service is

changed, the corresponding plan is updated in the composition cache of the central composer. This update is also done in the local library.

It is therefore a model adapted to this description which will be presented later.

6.3.4 Model

The model is the one that will bring optimization in the process of dynamic service composition based on the 2DSC's orientations described above. This presentation will begin with the logical approach illustration, the presentation of the model, the formalization of the system and the appropriated algorithms.

6.3.4.1 Logical approach illustration

In a static system, the services composition is done during development. The response time T_{SC} of a request Q that needs to involves n services is the sum of the response time of each service forming part of the composition. It can be noted:

$ST_{SC} = T_{EX}[S_1] + T_{EX}[S_2] + ...T_{EX}[S_n]$

$=\sum_{i=1}^{n} T_{EX}[S_i]$

But in case of dynamic composition, the response time T_{DC} of a request Q become the sum of response time of each service, added to the composition time procedures and the network time. This response time becomes:

$T_{DC} = T_{Comp} + T_{EX}[S_1] + T_{EX}[S_2] + ...T_{EX}[S_n]$
$= T_{Comp} + \sum_{i=1}^{n} T_{EX}[S_i] + \sigma$
$= T_{Comp} + T_{SC}.$

where:

— T_{SC} is the response time in case of static composition;

— T_{DC} is the response time in case of dynamic composition,

— T_{Comp} is the time of composition process;

— $T_{EX}[S_i]$ the execution time of each S_i service;

— σ is network time.

When the request Q_1 arrives to the composition server, this can be running to process another client request later Q_2. So the composition time T_{Comp} also depends on the number of requests the composition server receives. Then, beyond the number of services to be selected in the registry, this time

$T_{Comp} = f(Request, Service)$.

This means that, if the number of request is large, the response time will be long. Likewise, if the number of services in the registry is large, this response time also becomes as important. Our work aims to reduce this composition time substantially to the level of execution time during static composition.

Otherwise:
Let T_{SC}: the static composition time and
Let T_{DC}: the dynamic composition time. We noted that:

$T_{SC} = T_{EX}[S_1] + T_{EX}[S_2] + ...T_{EX}[S_n]$

$=\sum_{i=1}^{n} T_{EX}[S_i]$
$T_{DC} = T_{comp} + T_{EX}[S_1] + T_{EX}[S_2] + ...T_{EX}[S_n] + \sigma$

$= T_{comp} + \sum_{i=1}^{n} T_{EX}[S_i] + \sigma$

with:

— $T_{EX}[S_i]$: execution time of the service i

— σ : is the network time.

So we are trying to build our local composition's plan library, which will lead the system S after a number X of requests and after a time t, to have:
$T_{SC} \leq T_{DC} \leq T_{comp} + T_{SC}$.
We are also trying to build our local registry, which will lead the system S after a number X of requests, and after a time t, to have his own registry which will avoid some remote researches transactions in the network. S will locally build some service composition plan.

Our idea is to build a local library and a local service registry for the client service composition system S like in the case of NFS shared file system.

This approach can be described as follows.

6.3.4.2 Architectures of the current and dynamaic service composition approach

In the next figure you can see the general structure of the current **??** and our proposed system **??**. It is important to note, that Local Composition Engine, Local Registry and Application (to serve a Client) could be put as one component (even, for example, same machine or somewhere locally, on the client

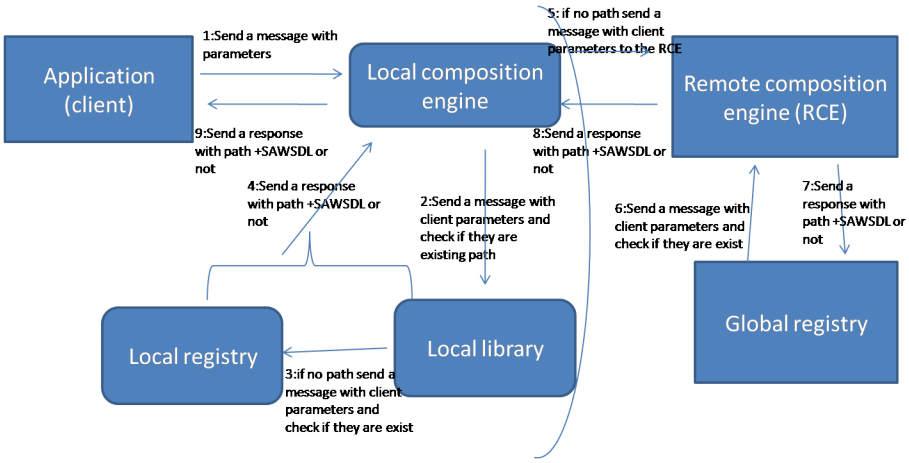

Figure 6.1 – Semantical representation of 2DSC

side) while Remote Composition Engine and Global Registry could be placed elsewhere. So that, according to our assumption, Client asks an Application for certain outputs, providing some inputs for it. At this point, Application sends a message of a client to the Local Composition Engine with inputs. Using its own algorithm, Local Composition Engine requests Local Registry for the information. After the communication is finished, Local Composition Engine makes a decision if it needs to send the same request to the Remote Composition Engine (in case we didn't find the path - send a message). And then Composition Engine makes the same operation with Global Registry and sends response to LCE (path with files). After receiving a response and has already downloaded the files, Local Composition Engine needs to send the response to Application (path with files) and send the same files and the information to the Local Registry to put data in it. As you can see, at the beginning, when the Local Registry is empty, the time for the response will be even bigger, than if use simple system from the first approach. But after the Client requested for the same outputs using already used inputs, we can see that finding the data in Local Composition Engine, which is much smaller, takes less time, so that this approach can reach the goal of the project.

The detailed architecture of the project can be seen below from Java point of view in Figure 6.3. Here, set of Interfaces are marked as blue, and classes also marked as yellow.

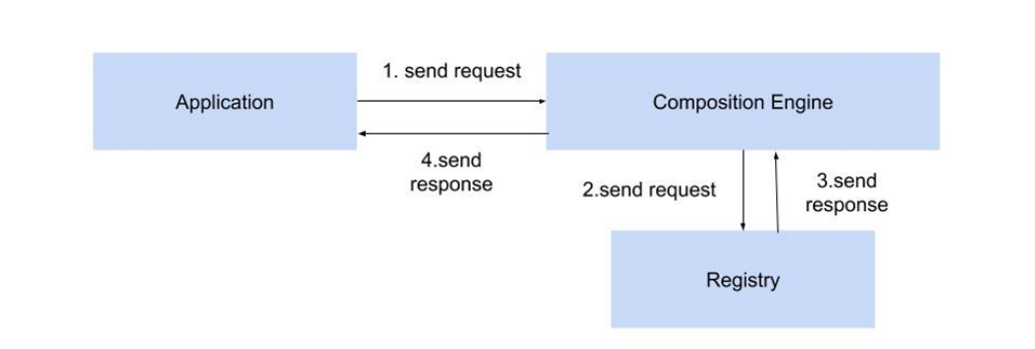

Figure 6.2 – Currrent approach

6.3.4.3 Model architecture

The design of the system is as described by the figure 6.4:

A local registry and a composition plan library are installed near each client. There will be saved all descriptive information for each service that is part of a successful executed composition plan. For each composition plan saved in the library, the associated request is also saved in the library. A monitoring tool is integrated into the composition server to allow the synchronization of updates between the components of the system.

Before any transaction with the remote composition server, the client should do a local research of the information in his own library or his own local registry.

6.3.4.4 Formalisation

The formalization here is to represent abstractly the major components of the model and to present the associated interactions.

The 2DSC is represented by a quintuplet (C, RC, Q, P, S) where:

— C is the set of possible client's information system;

— RC is the remote composition server;

— Q is the set of requests received by the client's information system;

— P is the set of composition plan generated by the remote composition server;

— S is the set of services;

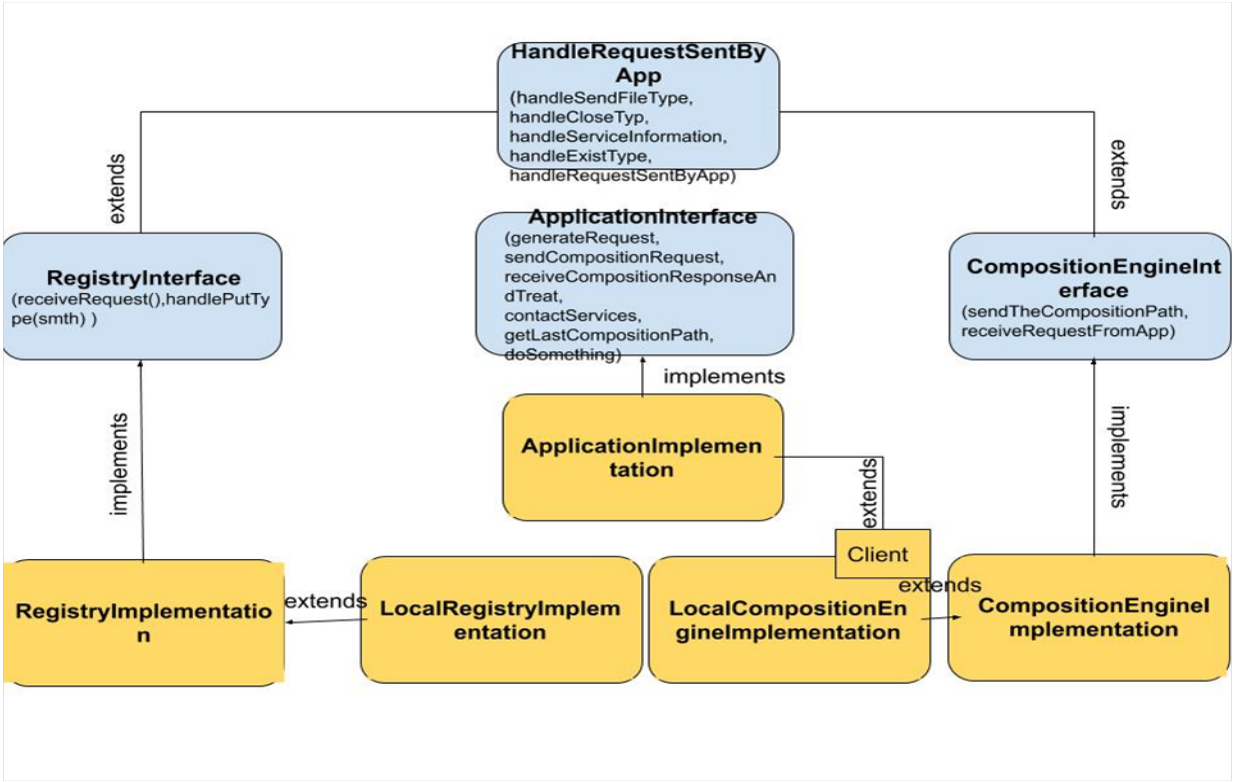

Figure 6.3 – Architecture in term of Java.

On the basis of the above notations, the construction mechanisms of the composition plan library, the local service registry and the monitoring responsible for updating the information can be defined as follows:

— Local registry generation:
This explains how the local registry is generated. In fact, for each client of service composition $c \in C$ of RC, there is a registry in which all the services $s_c \in S$ that are part of the execution of its composition plans $p_c \in P$ are saved. This directory is called local registry of the client c.

$$\forall c_i \in C, \forall s_{c_i}/s_{c_i} \in S \wedge s_{c_i} \in p_c, \exists A_c/s_{c_i} \in A_c$$

— Local library generation:
This explains how the local library is generated. In fact, for each client of composition service $c \in C$ of RC, there is a library noted L_c in which all composition plans $p_c \in P$ and the requests $q_c \in Q$ for which they

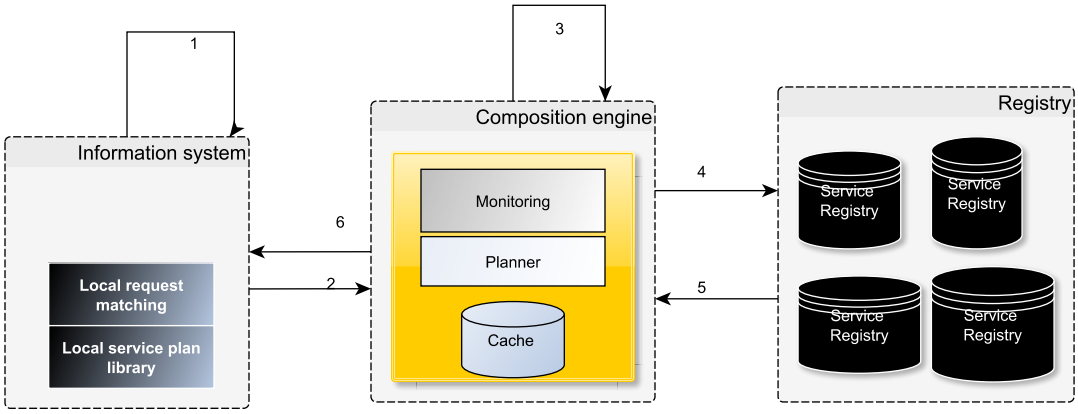

Figure 6.4 – Proposed architecture.

are generated are saved. This library is called local library of service composition plans.
$\forall c_i \in C, \exists L_{c_i}(Q \times P)/(p_{c_i}, q_{c_i}) \in L_{c_i}$.
L_c is the local library of c_i.

— Monitoring:
The monitoring of the system is a main function which will perform it. In fact; Information about changing of a service in the remote registry must be reflected on local registries that use it. These changes must also be reflected in the local libraries of the associated plans.
$\forall s \in S, \forall c_i \in C, \forall A_i(c_i), \forall L_{c_i}, desc(s(A_i)) = desc(s(A)) \wedge desc(s(L_{c_i})) = desc(s(A))$. With $desc$ as the description of the service s and $s(A)$ as the service which is a part of registry A.

We can describe these activities by the algorithms bellow.

6.3.4.5 Algorithm

The basic principle of this system is to set up a support infrastructure, external to the main composition engine, which will relieve it during multiple similar and repetitive solicitations. The model therefore requires within each client system, an infrastructure to build a library of composition plans exploitable to autonomously treat requests similar to those already processed through the composition server. It also requires within each client system, a local registry to save services taking part in each composition plan of a library.

This will bring a significant time saving when processing certain requests. This will bring back after a number of requests processing, the time of the dynamic composition to a level very close to the processing time during the static composition.

The appropriate algorithms are described as follows.

Algorithm 2DSC:

The client system receives a request noted Rq. It checks its path library with the function noted $LocalMatch$ if there is a similarity to an entry in this table. If this is the case, it executes with the function noted $ExecuteEngine$ the associated composition plan. If this execution goes well, it sends the answer to the user. If the execution goes bad, after marking the appropriate entry with the function $SetInvalide$, it will search again to look for another entry the similar request. He repeats this operation until he has gone through all the entries of the composition library. When this step is unsuccessful, it sends the request to the remote composition server with $RecomteCompos$ and waits for the plan it will execute with $ExecuteEngine$. It inserts into the library the new composition plan if the execution goes well with the function $InsertLibrary$ and upto date the system. This update also the function $SetVAlide$ which makes all the entries of the library available to new similarity test's operation.

These others functions called in the algorithm are described below:

Algorithm 3 2DSC

```
1: input: request noted Rq;
2: output: response noted Rp;
3: Vector Tab = Null
4: PathRq = LocalMa tch(Rq)
5: if (PathRq != NULL) then
6:    Rp = ExecuteEngine(PathRq)
7:    if (Rp != NULL) then
8:       Return(Rp)
9:    else
10:      repeat
11:         Add(Tab,PathRq)
12:         SetInvalide(PathRq)
13:         PathRq = LocalMatch(Rq)
14:         Rp = ExecuteEngine(PathRq)
15:      until (PathRq == NULL or Rp != NULL)
16:      if (Rp != NULL) then
17:         Return Rp
18:         SetValide(Tab)
19:      else
20:         repeat
21:            PathRq = RemoteCompos (Req)
22:            Rp = ExecuteEngine (PathRq)
23:         until (PathRq == NULL or (Rep != NULL))
24:         if (Rp != NULL) then
25:            Return Rp
26:            InsertLibrary (Rq,PathRq)
27:            Uptodate ()
28:         else
29:            Return Rp
30:         end if
31:      end if
32:   end if
33: end if
```

Algorithm 4 localMatch

```
1: input: Request noted Req;
2: input: Registry noted Reg;
3: output: Path noted PathReq;
4: Vector Tab = Reg
5: i = 1
6: repeat
7:    Sim(Tab[i],Rq)
8:    i=i+1
9: until Sim(Tab[i], Rq) or n ≤ i)
10: if i ≤ n then
11:    Return PathRq(Tab[i-1])
12: else
13:    Return NULL
14: end if
```

The function ***LocalMatch()*** takes a request Rq and returns its last corresponding composition path $PathReq$. This function performs a similarity test of the new request received with the entries in the plans library Tab.

Algorithm 5 ExecuteEngine

```
1: input: Vector noted PathRq;
2: output: Response noted Rep;
3: Vector Tab = PathRq
4: i = 1, Rep=NULL
5: repeat
6:    Rep= Rep + Result (Tab[i])
7:    i=i+1
8: until i = N+1) or Result (Tab[i])= NULL
9: if i ≤ N then
10:   Rep= NULL
11: else
12:   Return Rep
13: end if
```

The function ***ExecuteEngine*()** takes a composition plan *PathReq* associated to a request *Rq* and returns the result of his execution *Rep*. This result can be conclusive or not;

Algorithm 6 RemoteCompos

```
1: input: Request noted Req;
2: output: Vector noted PathRq;
3: Return PathRq
```

The function ***RemoteCompos ()*** is the central composition system. It takes a request *Rq* as parameter and returns a composition plan *PathRq* to the user information system;

Algorithm 7 SetInvalide

```
1: input: Vector noted Regis;
2: input:PathReq noted path;
3: output: Vector noted Regis;
4: Vector Tab = Regis
5: i = 1
6: if Tab[i]=path then
7:    state (Tab[i]=0)
8:    Tab = Regis
9: else
10:   repeat
11:      i=i+1
12:   until (i = N+1) or (Tab[i]= path)
13:   if Tab[i]=path then
14:      state Tab[i]= 0
15:      Tab = Regis
16:   end if
17: end if
18: Return Regis
```

The function ***SetInvalide()*** removes a composition path *PathReq* from the local composition processes. Precondition: the path cannot be empty.

Algorithm 8 SetValide

```
1: input: Vector noted Regis;
2: input:PathReq noted path;
3: output: Vector noted Regis;
4: Vector Tab = Regis
5: Vector Tab = Regis
6: i = 1
7: if Tab[i]=path then
8:    state(Tab[i]=0)
9:    Tab = Regis
10: else
11:    repeat
12:       i=i+1
13:    until (i = N+1) or (Tab[i]= path)
14:    if Tab[i]=path then
15:       state Tab[i]= 1
16:       Tab = Regis
17:    end if
18: end if
19: Return Regis
```

The function ***SetValide()*** takes in parameter a set of composition plans that it puts into treatment in the graph. This process makes them available to the new local matching operation.

Algorithm 9 Insert Library

```
1: input: Vector noted Lib;
2: input :PathReq noted path;
3: output: Vector noted Lib;
4: Vector Tab = Regis
5: i = 1
6: if Tab[i]=null then
7:    state (Tab[i]=path)
8:    Tab = Regis
9: else
10:   repeat
11:      i=i+1
12:   until i=N+1 or Tab[i]=null
13:   if Tab[i]=null then
14:      state Tab[i]= path
15:      Tab = Regis
16:   end if
17: end if
18: Return Regis
```

The function ***InsertLibrary()*** takes a composition plan *PathReq* and a request *Rq* that it saves to the local library. As precondition: this information must be new.

Algorithm 10 UpToDate

```
1: input:Vector noted Lib;
2: input:Request noted Req;
3: input:PathReq noted path;
4: output:Vector noted Lib;
5: Vector Tab = Lib
6: i = 1
7: if Tab[i]=Req then
8:    state(Tab[i]=path)
9:    Tab = Regis
10: else
11:    repeat
12:       i=i+1
13:    until i=N+1 or Tab[i]=Req
14:    if Tab[i]=Req then
15:       state Tab[i]= path
16:       Tab = Lib
17:    end if
18: end if
19: Return Lib
```

The function ***Uptodate ()*** makes to date the composition system by adding in his library, the new composition plan that can satisfy the current request and calling the *Setvalide* function.

6.4 Comparison between the current DSC and our Distributed and dynamic service composition (2DSC)

This section will evaluate the temporal complexity of our proposed approach, then evaluate the temporal complexity of the current approach before comparing the temporal resources of these two methods and enact a lemma followed by its proof.

The comparison of the complexities of the two approaches is done in the best case, in the average of cases and in the worst case.

6.4.1 Formalization of the global composition time

The global composition time is the sum of:

— The network time σ;

— The time taken for the execution of each stage of the composition (discovery, selection of services, generation of composition plans, selection of the best plan, execution of the plan, publication of the new service) T_i;

— T_{exec}: execution's time;

We can formalize the temporal complexity of the current and proposal approaches as follows:

Let S_1 be the current dynamic composition system without distributed approach (1);

Let S_2 be the proposed dynamic composition system with distributed approach (2);

(1) $\tau_{S_1} = T_{DC} + \sum_{i=1}^{n} T_i + \sigma$.

(2) $\tau_{S_2} = T_{2DC} + \sum_{i=1}^{n} T_i + \sigma$.

Let us calculate the complexities of the 2DSC and current approaches in the best cases, in the average cases and in the worst cases, and compare them.

6.4.2 Computation time evaluation of the current approach

In the current system, the best case is similar to the worst and the average case because the process does not vary. We repeat the same activities each time.

We can evaluate the temporal complexity τ_{S_1} of this approach as follow :

$$\tau_{S_1} = \begin{cases} T_{DC} + \sum_{i=1}^{n} T_i + \sigma & \text{in the best case} \\ T_{DC} + \sum_{i=1}^{n} T_i + \sigma & \text{in the average case} \\ T_{DC} + \sum_{i=1}^{n} T_i + \sigma & \text{in the worst case} \end{cases}$$

We observe that it is the time.

6.4.3 Dynamic service composition layer description

Our proposed dynamic service composition framework figure 6.5 is a hierarchical structure of three main levels. We have the local level, the intermediate

levels; the number of which depends the administrative or territorial structure of each organization, and the national level. Each level is composed of two components: a composition plan library, a service repository. A monitoring support makes it possible to establish coherent communication at the different levels and is responsible for updating the service registries and libraries.

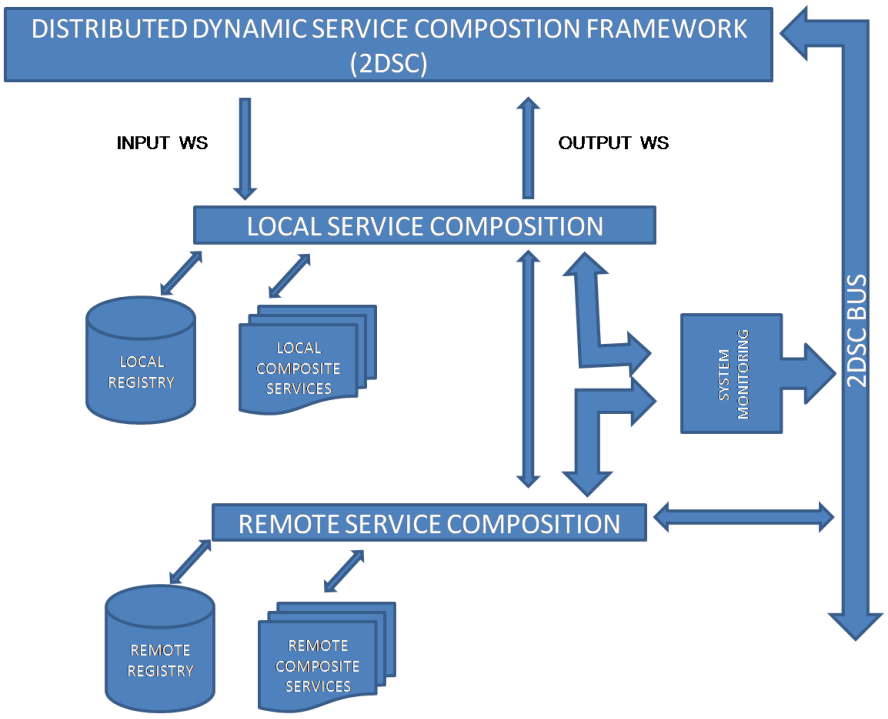

Figure 6.5 – Dynamic Service Composition Layer

In fact, the process begins by the sending of a user's request to the client system. The client system sends the request to his local composition engine. This local composition engine checks in its composition library if it has already processed such a request. If so, its composition path is then used to process the request without involving the remote composition server and by relying on the local registry. The composition path is then returned to the local composition which will return it to the client. If this request is new for the system, it is sent to the remote service composition which will repeat the same process at its level.

We can generate the following framework for a distributed multi layer collabartion 2DSC 6.6.

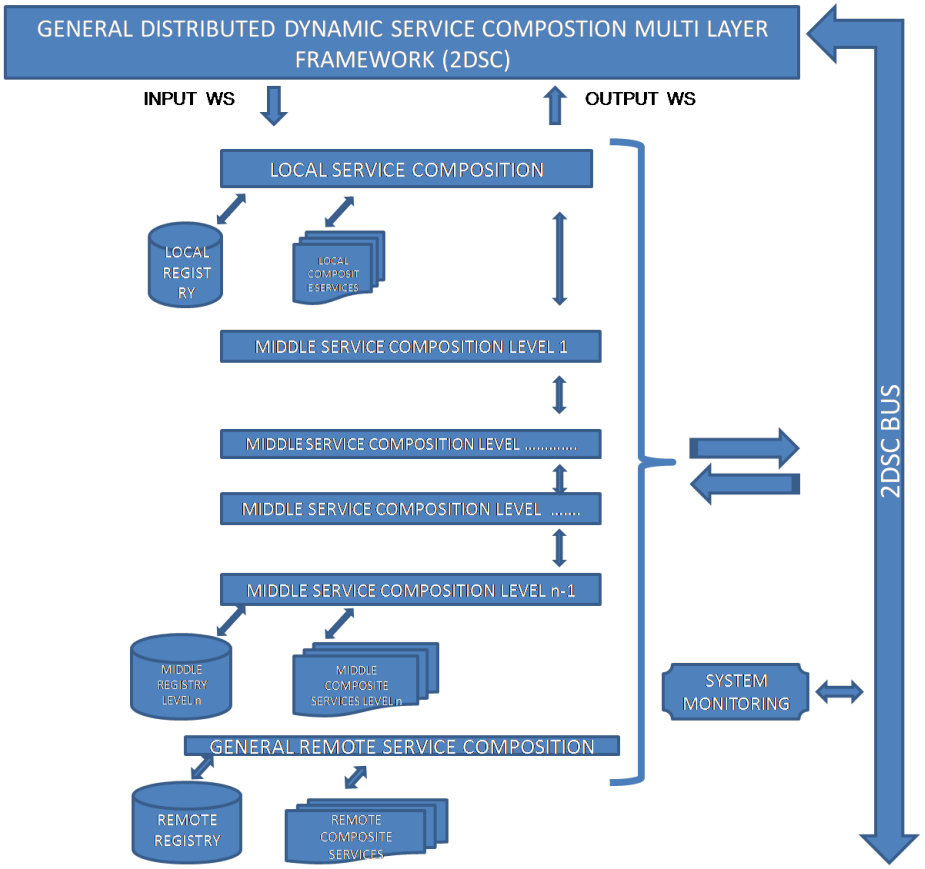

Figure 6.6 – Multi layer Dynamic Service Composition

6.4.4 Computation time evaluation of the 2DSC approach

In the 2DSC approach, the best case represents the case where the request is still processed by the local library. Middle case is the situation in which everything goes without support from the local library. This case is identical to the basic process. The worst case is to find a way and therefore the execution goes wrong. In this case, the composition engine is still requested for a new composition. Logically, this is a situation that can arise when updates have not been made by the monitoring system.

With the 2DSC, we can evaluate his complexity τ_{S_2} as follow:

$$\tau_{S_2} = \begin{cases} \sum_{i=1}^{n} T_i + \sigma & \text{in the best case} \\ T_{LC} + \sum_{i=1}^{n} T_i + \sigma & T_{exec} = T_{exec'} \\ T_{LC} + T_{DC} + \sum_{i=1}^{n} T_i + T_{LC} + \sigma & T_{LC} \text{ is the time of local research} \end{cases}$$

6.4.5 Best case comparison

The system works at its best when its local library finds the desired composition path for all requests. This allows the remote composition server to be completely free from this solicitation. This situation is favored by the enrichment of the local library of the client system over time taking advantage to the various activities during the operation of the system.

The complexity of the new model 2DSC in the best case is: $\tau_{S_2} = \sum_{i=1}^{n} T_i + \sigma$.

And the complexity of the current DSC model in the best case is: $\tau_{S_1} = T_{DC} + \sum_{i=1}^{n} T_i + \sigma$.

So we conclude that: $\tau_{S_2} \leq \tau_{S_1}$.

That means the new approach is better in this case.

6.4.6 Worst cases comparison

The worst case is when the local system always fails to produce the composition plan; the system cannot find the information in its local library. It will therefore first perform an unsuccessful search locally before calling on the remote composition server again. This situation is favored by the lack of activities intended to enrich the local library of the client system. It is clear that our approach is less efficient than the current model.

The complexity of the new model in the worst case is:

$\tau_{S_2} = T_{LC} + T_{DC} + \sum_{i=1}^{n} T_i + \sigma$

And the complexity of the current model in the worst case is:

$\tau_{S_1} = T_{DC} + \sum_{i=1}^{n} T_i + \sigma$

So we can conclude that: $\tau_{S_1} \leq \tau_{S_2}$

That means the current model is better in this case.

6.4.7 Average case comparison

We consider here as the average case in our approach the combination of both the worst and the best case. The complexity of the new model in the average case is:

$\tau_{S_2} = T_{LC} + \sum_{i=1}^{m} T_i$

And the complexity of the current model is:

$\tau_{S_1} = T_{DC} + \sum_{i=1}^{n} T_i + \sigma$

We can see that it depends of the value of T_{DC} and T_{LC}. But we observe that the remote server is freed. And that is our first priority.

So we can conclude that: $\tau_{S_2} \leq \tau_{S_1}$

That means that the current approach is most efficient.

6.4.8 Comparison analysis

In the worst case and the average case of the theoretical comparison, the complexity of our system shows that it is less efficient than the current model. These will happen when the client system is at the start of its operation and the tests it performs in its local composition cannot give it a positive response. Quite simply because his composition library is still empty. To avoid this situation, it is possible to carry out the first treatments in the system without having recourse to the local composition infrastructure. The main question after this analysis is to be sure that our system will still work in the best case condition to validate the performance of our approach on a theoretical level. As a result of these preoccupations, we can dictate the following lemma:

Lemma Let T be the computation time of a request processing.
Let S_1 be the current dynamic service composition and S_2 be the new system.
Let be R the set of request. $\forall r \in R, T(S_1(r)) \ggg T(S_2(r))$

Proof To prove that the new approach is better than the current one, let's show that the system will spend more time in the best case process. Let N be the number of functionalities of a system. After a system running time t, there will be a number n of composition plans if n distinct requests have been processed in the local library.

Then, when t tends to infinity, n tends to N. It means that the registry can already contain a number of plans able to satisfy all its functionalities locally. This would imply that the majority of process is done locally.

In other words, let $f(t)$ be the function that inserts the composition plans into the local library of composition paths. Let N be the finite number of

functions of a system. $f(t)$ tends to N when t tends to infinity.

$$f(t) = \begin{cases} t \to \infty & \text{when } t \text{ tends to infinity} \\ n \to N & n \text{ tends to } N \end{cases}$$

This lemma proves that our approach will allow to build in time, a local composition system for each client and this will free the remote composition server and make the model more efficient. The scalability due to the growing number of users and services will decrease, thus reducing the dynamic composition time to the same level as during static composition. This will also lead to reduce request response time for end-users. This shows that in the time, the system will situate in the best case.

This shows that in the time, the system would behave as being totally local composition. This would situate it in the best case.

6.5 2DSC systems caracteristics

As a distributed system, the 2DSC is also characterized by an architecture, a process, a communication, a naming, a synchronization, a consistency and replication, a fault tolerance and security.

6.5.1 Architectures

2DSC can be seen as a set of complex pieces of software dispersed across multiple machines which are properly structured. There are architectural styles which informs how all the various software components are to be organized and how they should interact. This system is a data-centered architecture because processes communicate through the use of shared Web-based data services. The system is also a decentralised architectures and multitiered client-server architectures.

6.5.2 Processes

In 2DSC system, processes are modelized by threads because clients and servers are constructed such that communication and local processing can overlap, resulting in a high level of performance.

6.5.3 Communication

The communication between processes is at the heart of the 2DSC systems. Different machines can exchange information by Remote Procedure Call (RPC) because when a process on machine A calls a procedure on machine B, the calling process on A is suspended, and execution of the called procedure takes place on B. Also, information can be transported from the caller to the called in the parameters and can come back in the procedure result.

6.5.4 Naming

The system is mainly based on two entities: the developed and published service, the user request for processing and the composition plans generated by the composition server. All these entities are named exclusive in the system. This means that the names of these entities are their addresses and there are some properties to identify these names:

— An identifier refers to at most one entity;

— Each entity is referred to by at most one identifier;

— An identifier always refers to the same entity (it is never reused).

The entities have structured and attributed-based names. Entities are described in terms of attribute and value. Thus, an entity is assumed to have an associated collection of attributes. Each attribute says something about that entity therefore, by specifying which values a specific attribute should have, a user essentially constrains the set of entities that he is interested.

6.5.5 Synchronization

The synchronization in the system is ensured by the composition server as usual. But there is also a monitor that alerts the client systems on any updates that can improve their databases and harmonize all the system. The update can be useful in case of new information received through the service registry or in case of new performed composition plan to process some hold request. This monitoring tool is in permanent but transparent communication with the client systems. It is a process required to act as coordinator, initiator, or otherwise performs some special role.

6.5.6 Consistence and replication

Data on service registry are generally replicated on local registries to enhance reliability and performance. One of the major rules of the monitoring

entity is that when one copy is updated we need to ensure that the other copies are updated as well; otherwise the replicas will no longer be the same. Data on such request and their effective composition plans are also replicated to increase the performance of the system. This replication for performance is important when the system needs to scale in numbers and geographical area. Scaling in numbers occurs, for example, when an increasing number of processes needs to access data that are managed by a single server. Scaling with respect to the size of a geographical area may also require replication. The basic idea is to place a copy of data in the proximity of the process using them to decrease the time to access the data. Thus the system supports replication and decides where, when, and by whom data should be placed, and subsequently which mechanisms to use for keeping the replicas consistent.

6.5.7 Fault tolerance and disaster recovery

The new distributed approach for dynamic service composition is supports availability, reliability, safety and maintainability.

— The 2DSC system is available because the system is ready to be used immediately. Theoretically it is proven that the probability that the system is operating correctly at any given moment and is available to perform its functions on behalf of its users.

— The system is reliable because it can run continuously without failure due to the scalability. The system will most likely continue to work without interruption during a relatively long period of time even in case of composition plan server failure.

— The system has a safety property because of the monitoring operations, if the system temporarily fails to operate correctly for only a very brief moment, the effects couldn't be disastrous and nothing catastrophic happens.

— The system is maintainable because it is easy to repair it when he is failed. In fact, we can observe that our distributed model creates the relocation of data which participate to the service composition. This allows the system to build a user's profile data clustering. In addition, this relocalisation approach also gives the system the ability to make a disaster recovery after a major incident with the monitoring function. En fact, the monitoring function up to date the subsystem by using the remote central composition server cache.

  This is the prove of the faults tolerance and the disaster recovery of our proposed approach.

6.5.8 Security

Security in the DSC system is strongly proven by their confidentiality and integrity.

— Confidentiality: the information is disclosed only to authorized parties. Any service clusters contains only their concerns information.

— Integrity is taken into account and improper alterations are detectable and recoverable. Major assets of the system are its hardware, software, and data.

6.6 Synthesis

Dynamic service composition is an important operation when handling user requests in a cooperative environment of service-oriented information systems. It starts with the discovery of services, the selection of services, the generation of composition plans, the selection of the best composition plan, the execution and the publication of the new composite service. It is therefore a heavy operation that puts the client system, the service-owning systems, the composition engine and the service registry into permanent activity and generates scalability. To deal with this issue, the new distributed approach to dynamically compose services has proposed.

It is an approach to publishing composite services during dynamic service composition to optimize service discovery under a distributed architectural model. We called it 2DSC. It allows making the process effective by building in time at each client system, a registry of composite services emanating from the results of its various requests. It is therefore a method that brings each client system closer to the composite services generated by its requests. This reconciliation also makes it possible to locally generate composition plans capable of responding to the concerns of its users without the intervention of a remote infrastructure.

By publishing compositing plans locally as new composite services, processing a new request that is similar to an earlier request reduces the response time by deleting transactions with the remote registry and those of the remote composition engine (server). In addition, this can generate new composition plans, if necessary, directly on the basis of the built library.

With this in mind, we have defined a general dynamic composition algorithm that integrates local composition plan, local composition plan search, central composition engine, local request matching and local composition control functions. Request matching is based on semantic similarity rules.

In addition, the publication that is above all dynamic composition activities is seen through our approach as an important aspect as it seems to be the basis for the execution of the entire dynamic service composition process. As an information-sharing moment, it can help improve the process of dynamic service dialing by leveraging customer infrastructures to decentralize information in ways that benefit customers.

In conclusion, the distributed dynamic services composition (2DSC) proposed in this chapter is based on the distributed approach and sharing of descriptive information about services. Regardless of the service description methods used, this approach simply captures the information resulting from the successful processing of a client request. In fact, the composition plan associated with a request that has been successfully executed can be exploited to handle similar requests.

In the next chapter, this approach will be evaluated in the case of a service-oriented E-government in a developing country like Cameroon. We will focus on the improvement of quality of service especially in developing countries where ICT infrastructure issues are important.

The passions of the heart are more lively but less constant than those of the mind

LAMENAIS

7

Validation of Distributed Dynamic Service Composition (2DSC): Case study on Cameroon e-government implementation

Contents

7.1 Introduction

This part aims to show how our proposed model based on 2DSC architecture can improve dynamic service composition especially in E-government in a developing country like Cameroon. Our study will start with the presentation of the context of our study in developing countries, the definition of E-government and the methods to implement it and some issues of the concept. Then we will propose an architectural model of information systems based on our proposed framework in developing countries particularly in Cameroon where the administration is structurally and territorially decentralized and where there are many problems of ICT's infrastructures. An illustration of our proposal will be focused on improving the collaboration between Cameroonian public administration information's systems.

7.2 E-government

This part will present the context of the experimentation of our model on E-government. Then we will present the concept of e-government, its definition and the problems it raises during its implementation. It will also be clearly exposed the precise problem that our solution brings in the service-oriented e-government both in a general way and in a particular way in the developing countries.

7.2.1 Context

In the developing countries, the debate on the relevance and contribution of ICT in the development process compared to the basic social infrastructure has found a better balance in the views (Osterwalder, 2003). Indeed, ICT do not solve the basic social problems arise but is rather a tool that can facilitate the access, the availability and the proper functioning of the services responsible to provide the answers to these priority needs of populations in these areas. Some therefore believe rightly that it is an instrument to facilitate the achievement of the MDGs (of Science and Technology, 2006) and a better tool for good governance (Bertot et al., 2010).

One of the best ways to make visible the social and economic impact of ICTs is to make use of them in the different approaches to public governance of the states. Electronic government called E-Government or E-Gov (Relyea, 2002), is since many years seen as an important aspect in accelerating the growth in developing countries by creating a lot of opportunities among which

the cost reduction and efficiency gains, the quality of service delivery to businesses and customers, the transparency, the anti-corruption, the accountability, the increase of the capacity of government, the network and community creation, the improvement of the quality of decision making, and they promote the use of ICT in other sectors of the society (Ndou, 2004).

But different applications and platforms that cover the overall range of the E-Government implementation area need to interoperate in order to provide integrated governmental services to the citizens and businesses (Yan and Guo, 2010) (Peristeras et al., 2009). So researchers have proposed in the area of enhancing e-government interoperability to use common models and/or ontologies (Peristeras et al., 2009). And others propose the use of Service Oriented Architecture to potentially address more those needs and provide a modern application architecture for the interaction of existing and new distributed systems (Yan and Guo, 2010).At the same way, to overcome lack of interoperability's situation, researchers propose the use of a designed semantic platform to easy public administration to cooperate and expand the accessibility of services in a broad sense (Sabucedo and Anido-Rifón, 2006). The authors argue that Service oriented architectures, methodologies and tools, together with the conceptual and empirical framework of web services have a high potential to assist public administrations in ongoing e-government innovation processes cite ref131.

In view of the particular context of developing countries (Ndou, 2004) (Yan and Guo, 2010), an SOA-oriented E-government solution must be able not only to improve existing methods of collaboration between public administrations information systems, but also to provide an appropriate response to the environments of these special countries.

In fact, developing countries are countries which are not yet developed. They are the countries of the south and present transitional infrastructural characteristics, ie they are moving from traditional to modern environment. To describe developing countries in the context of information and communication technologies, we can observe them on the infrastructural level, on the political level and on the human or cultural level (Motahari-Nezhad et al., 2018). In fact, developing countries are characterized by a low energy supply, the availability of which can sufficiently support the permanent commissioning of infrastructures dedicated to ICT (Mulugetta et al., 2019). There is low energy coverage which causes equipment to be switched off and this leads to the unavailability of several services accessible via computers. In terms of human capital, and culture, the use of tics has become real and it has spread through the use of cell phones. This reality is especially favored by the juvenization of the population (Hossain et al., 2019; Demissie et al., 2016). On the political

level, several states have evolved their legal orders to adapt to the new situation (Ngeminang, 2012). Cameroon is one of the countries of sub-Saharan Africa which has launched its electronic government project since 2000 with technical support from Korea (MINPOSTEL, 2020; ANTIC, 2017). it has also previously set up a far-reaching telecommunications and internet infrastructure.

It is in this context that the evaluation of the 2DSC in the new service-oriented E-government paradigm is presented. It will be the way to verify the improvement of the service composition process during the processing of requests despite the problem of physical interconnectivity of the information systems of public administrations and other takesholders.

7.2.2 Definition

E-government is about reinventing the way in which governments interact with citizens, governmental agencies, businesses, employees, and other takesholders; it is also about enhancing the democratic process and also about using new ideas to make life easier for citizens (Lofstedt, 2012b). E-Government is a powerful guiding vision for the transformation of public administration. It has many definitions according to the literature and all these definitions integrate the notion of public service, the facilitation of the collaboration between public administration and its partners, and the use of the technological infrastructure.

Thus, E-government refers to the delivery of national or local government information and services via the Internet or other digital means to citizens or businesses or other governmental agencies (Palvia and Sharma, 2007). According to World Bank, E-Government refers to the use by government agencies of information technologies (such as Wide Area Networks, the Internet, and mobile computing) that have the ability to transform relations with citizens, businesses, and other arms of government. It is an innovative way of the production and delivery of government services through IT applications, used to simplify and improve transactions between governments and citizens (G2C), businesses (G2B), and other government agencies (G2G) (Sharma, 2006).

The benefits of e-government usually include improved quality of citizen services, internal efficiencies, law enforcement, education and information, promotion and outreach activities, safety and security, health care services and management, and involvement of citizens in the democratic process (Sharma, 2006). The table 7.1 summarizes the characteristic differences between the traditional government and e-government organizations (Sharma, 2006).

Table 7.1 – characteristic differences between traditional government and e-government organizations

Traditional government	**E-government**
Bureaucratic controls, clear authority hierarchy	Client service and community empowerment, leveled/blurred hierarchy
Process centricity	Customer centricity
Isolated administrative functions and data collection	integrated resource service and knowledge focus
Functional specialization of units or geographic bias	Breakdown of unit barrier, government integration
Decision based on uniform rules and awkward reporting approvals	Decision based on negotiation and implicit controls and approvals
Isolated administrative functions	integrated resource services
Disjointed information technologies	Integrated network solutions
Time-consuming process	Rapid streamlined responses

7.2.3 E-government implementation approach

Many researchers have been done in the field of the implementation approach of E-government (Heeks and Bailur, 2007). Authors proposed four stages to offer a path for governments to follow and suggest challenges, both in terms of the organization and technical aspects (Layne and Lee, 2001): cataloguing, transaction, vertical integration, and horizontal integration described as below.

— cataloguing: in stage one of cataloguing, initial efforts of state governments are focused on establishing an on-line presence for the government. Many state governments' efforts on web development and forms-on-line initiatives belong to this stage;

— transactional: in this second stage, e-government initiatives will focus on connecting the internal government system to on-line interfaces and allowing citizens to transact with government electronically;

— Vertical integration: Vertical integration refers to local, state and federal governments connected for different functions or services of government;

— horizontal integration: horizontal integration is defined as integration across different functions and services. So systems in both agencies talk to each other or work from the same database.

These stages are mainly based on technical, organizational and managerial feasibilities and corresponding examples. They suggested that e-government is an evolutionary phenomenon and therefore e-government initiatives should be accordingly derived and implemented. Some researches proposed five stage-model of e-government framework which are emerging, enhanced presence, interactive presence, transactional presence and networked presence (Palvia and Sharma, 2007). This approach takes the above approach by simply exploding the first stage (cataloging) in 2 stages (emerging and enhance presence).

7.2.4 Our issue

Researches in the field of e-government are mainly focused on E-service, E-security, interaction, e-democracy and management and organization defined as follow (Palvia and Sharma, 2007).

— E-Services form an emerging field which is rapidly gaining attention and importance. Citizens expect and demand governmental services with a high degree of quality, quantity, and availability in a 24-hour, seven-days-a-week, and year-round fashion;

— E-Democracy is explored as a subset of the greater, and more important, philosophical topic of democracy itself. E-Democracy focuses on the use of information and communication technologies in supporting democratic decision-making processes and in allowing more effective and transparent engagement between government, business, and citizen;

— Organization and Management: There is a need to develop theories, models, and methods within the area of e-government. Thus far, the research has mainly involved descriptive studies, philosophical studies, theoretical research, and empirical studies;

— E-Security: E-Government services have to be secure with regards to all aspects, so that the government and the users trust the system and feel confident in using it. Security is critical since it can influence citizens' willingness to adopt the services offered;

— Interactions: there have been a number of categories identified for interaction within e-government: government-to-citizen (G2C), government-to-employee (G2E), government-to-government (G2G), and government-to-business (G2B). Each uses Internet technology to provide government services online.

Cameroon is making progress in the implementation of an e-governance system. In accordance with the step-by-step approach above, we can situate this implementation of its e-government's projet at the level of service's

transactions between administrations and between administration and others (citizen, employee, business). It is therefore at the level of interactive e-services that the challenge lies, i.e. the integration and the use of services belonging to another administration and others takesholders in the computer system of a given administration. These service's transactions must be dynamic with regard to the limits of static composition. That is why this work aims to improve dynamic service composition (the quality of interactions) in service-oriented architecture based e-government in general and particulary in the environments of the developing countries of which Cameroon is a part.

7.2.5 E-government research in developing countries

Research in the field of e-government in developing countries has made substantial progress and researchers had primarily focused on gaining an understanding of the adoption and usage of ICT in governments focusing on exploring the implications of transforming traditional governments to e-government, as well as the challenges and constraints to the implementation and advancement of e-government (Fonou Dombeu and Rannyai, 2014).

This paradigm is became one with research theories and some future research directions from a methodological point of view pay more attention to improve the quality of research (Wahid, 2012). Thus, the integration of heterogeneous and distributed services to achieve interoperability of Information and Communication Technology (ICT) is then became the real challenge for the e-government and the Service Oriented Architecture has emerged over the past several years as a preferred approach to deal with this issue (Das et al., 2010).

Many papers address the technical design of systems for successful implementation of e-government initiatives (Fonou Dombeu and Rannyai, 2014). Several researchers have proposed models and frameworks for the successful development, monitoring and implementation of e-government in African countries and the preconditions for successful implementation of these initiatives are also discussed (Wahid, 2012; Saleh et al., 2013).

Authors undertaking e-government research in Africa for example tackle certain issues varied from country to country as challenges and opportunities of e-government in Africa, the proposal of e-government strategies, best practices of e-government implementation, evaluation of government websites, models and frameworks for implementing e-government, and assessment of the state of e-government, the implication of e-government on public policy and citizen roles and participation in e-government were, the accessibility of e-government services, analysis of e-government readiness, and usage of ICTs

in e-government implementation (Lu and Xu, 2017).

Thus, several works have focused on interoperability and integration problems between systems in e-government (Saleh et al., 2013; Wahid, 2012). The service oriented approach has conquered space and the problems of application collaboration in e-government are approached from an architectural view (Fonou Dombeu and Rannyai, 2014; Alghamdi et al., 2011; Das et al., 2010). Several studies have focused on a specific sector of activity such as health and education by highlighting the ontologies of business fields (Saleh et al., 2013).

Compared to these different works, our study addresses an important and updated part of service-oriented architectures to fulfill its use in developing countries. Indeed, after admitting that the service oriented approach is the most suitable for building collaborative platforms dedicated to public administrations, one of the advantages of SOA, which is the services composition, must go beyond the manual and static principle to dynamic principle to cope with changes of environments. The dynamic service composition has therefore become an important research question where its optimization to deal with the resulting scalability becomes a necessity for its pratical adoption (Baryannis and Plexousakis, 2010; Lecue and Mehandjiev, 2009).

7.2.6 motivation

The reasons for this guidance are mainly related to the quality of the communication infrastructure and the availability of electrical energy. Indeed, the availability of computer systems is not assured when the electrical energy is not available all the time. And in a service-oriented architecture, the availability of the registry and each service owner is essential for the good running of the service composition and execution process.

Our solution which aims to reduce the number of interventions of the service composition server by using composite service local backups, gives a way to reduce considerably the rate of failures of the possible requests and also gives a better way to disaster recovery. It will allow the availability of service composition even when the composition engine and/or the registry is/are out of service.

7.2.7 Cameroon e-government's project structure

In November 2001, less than 10 percent of government agencies had a Web presence and the value of data provided was poor. By August 2002, nearly half of ministries have a Web presence, and other ministries and government agencies have defined plans and timescales for their Web presence. There has

been standardisation of Web site structure and appearance via the portal. There has been an increase in the volume of data available online, including budgets, government plans and regulations so businesses and citizens can get access to some tender information online.

The Cameroon e-government's project is a service orientated projet figure 7.1 with a transactional authority. To integrate various systems, both new systems, and existing systems there are two main strategies: the bottom-up and the top-down. The strategy's implementation based on a bottom-up approach when it needs to take into account several existing applications yet used by many takesholders. The new systems are build on top down strategy to take the service oriented vision at the beginning of the system design. The structure of the Cameroon e-government is centralized around a certification authority, which guarantees the security of transactions between the various IT platforms.

This architecture also supplies, as indicated in the figure 7.1, a public network which is open to transactions between all takesholders, and a private network infrastructure which is dedicated only to government agencies. This model started out around an idea of manual composition. Thus, the dynamism of the environment cannot be taken into account. This will quickly make the system become obsolete. This is why the dynamic service composition approach can deal whith this issue and the scalability which is therefore the crucial aspect of this process requires our interest.

We can observe that this architecture presents 4 major entities. Users or citizens who can request services from an administrative entity via the Internet form the first the entity. An another block is composed by government information systems's partners. The third block is composed by public administration systems that can interact with each other via a dedicated intranet network based on a service registration structure (registry). The fourth component is a certification authority; a public key infrastructure, which is at the center of all transactions between all the information systems and other entities.

7.3 2DSC algorithm for Cameroon's e-government project

The main problem that reduces the use of dynamic service composition despite its importance over static composition is scalability (Baryannis and Plexousakis, 2010; Lecue and Mehandjiev, 2009). The scalability has three causes but solutions are not taking into account the number of user's request

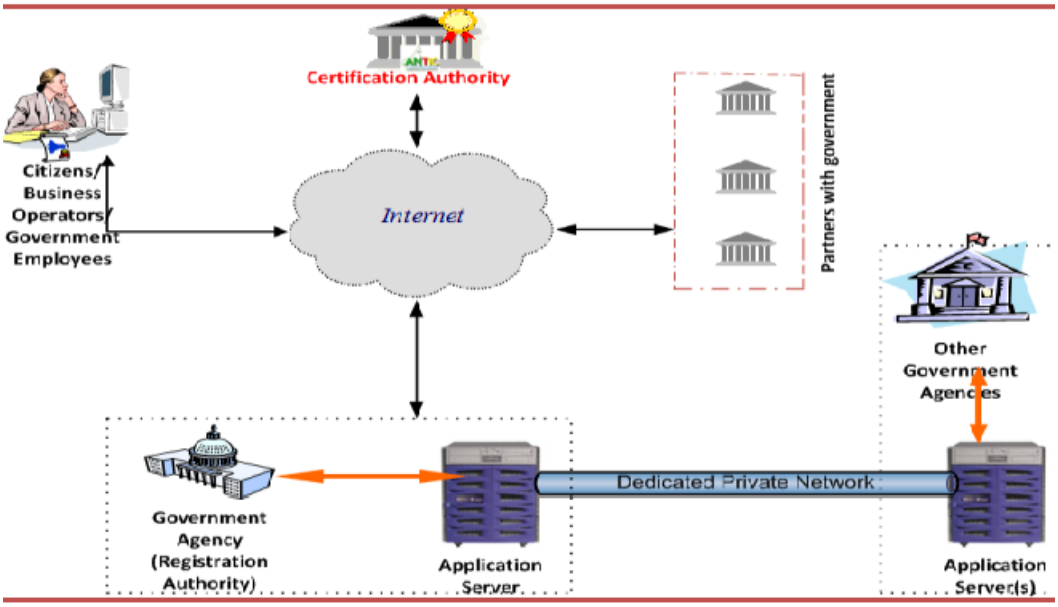

Figure 7.1 – Cameroon PKI architecture (**?**)

(Lecue and Mehandjiev, 2009). This is why it is on this level we focus our approach to deal whith this issue. We are proposing to improve the dynamic service composition layer and that will act to improve the extension of e-government, especially in developing countries.

We propose to extend application infrastructures with a local service composition system which will serve as a smart cache for the remote public composition engine. This middleware is provided as a plug-in for the application's infrastructure in such a way that the application developer has no addition task to perform. It privately and locally processes a number of service composition requests without soliciting the shared remote composition server. Algorithm 11 summarizes our idea.

The functions of our algorithm are described as follows:

— The function *LocalEgovComposition* takes a request and returns the result of the composition at local level. The result can be good or not;

— The function *LocalEgovCompositionMiss* informs for the result of this *LocalEgovComposition*;

— The function *RegionalEgovServerComposition* is the regional composition system. It takes a request parameter and returns a composition path to the information system;

Algorithm 11 Basic algorithm of Distributed Dynamic Service Composition (2DSC)

```
1: LocalEgovComposition
2: if LocalEgovCompositionMiss) then
3:    RegionalEgovServerComposition
4:    if RegionalEgovServerCompositionCacheMiss then
5:       NationalEgovDynamicServiceCompositon
6:    end if
7: end if
```

— The function *RegionalEgovServerCompositionCacheMiss* is a sub-function of *RegionalEgovServerComposition* which verifies if it exists a past composition plan that can be used to treat the new request;

— The function *NationalEgovDynamicServiceCompositon* launches the whole process of dynamic service composition at the national level.

7.4 DSC layer for Cameroon's e-government system with 2DSC approach

This part aims to build and evaluate a distributed simulator for 2DSC approach into an e-government in a developing country like Cameroon. The goal is to simulate the operation of such a technological infrastructure, evaluate system performance and draw observations. A global architecture will first serve to present the general vision of this integration.

7.4.1 Simulation specifications

According to previously considered scheme of SOA, the architecture of the simulator consists of Application (the Service consumer, or the other words - client), Registry of services (which includes the list of services with the corresponding links) and Composition Engine (Service provider which is collecting needable services) with its own algorithm of searching the outputs according to inputs. Here on the scheme you can see how it works. Below we are describing what exactly each component is doing in the architecture.

— Application part:

  — Loading properties to communicate with Composition Engine (port, hostname and so on);

  - Generating requests or using the text file with the list of requests to send it to Composition Engine with input, output parameters;
  - Receiving the response from the Composition Engine as a path from input to output;
  - Sending a request to Composition Engine to download corresponding WSDL files;
  - Creating the files with the content of WSDL files.

- Registry:
  - Loading properties to connected to database stored information about services.
  - Receiving the request from the Composition Engine with Client's message:
    - 1. Checking if input parameters exist;
    - 2. Searching in database outputs correspondant to input parameters.
  - Repeating 1 and 2 while find (or not) corresponding output;
  - Send the response to Composition Engine about existing of the parameters and their communications .
  - send the WSDL file to Composition Engine.

  Thus our data base is in the form of table, that contains the name of the service, the name of the input and the output as well as the link where the WSDL file exist in the file system. We can see the example of our MySQL database as follows 7.2:

  .

  We can transform this data base to a directed graph, where arcs present the name of the services that take a parameter as input and give another parameter as output. The nodes are these parameters. The directed graph that we can imagine is in the form shown in the figure 7.3: bellow:

  .

- Composition Engine:
  - Loading the information to communicate to a Registry of services;
  - Receiving the message from Application;
  - Sending parameters to register after splitting client's message;

```
mysql> SELECT * FROM Webservices.webs;
+-------------+------+------+-----------------------------------------------+
| servicename | INP  | OUTP | linkWSDL                                      |
+-------------+------+------+-----------------------------------------------+
| s1          | P1   | P2   | /home/ayman/eclipse-workspace/WSDLs/service2.wsdl |
| s2          | P4   | P6   | /home/ayman/eclipse-workspace/WSDLs/WS.wsdl       |
| s3          | P2   | P3   | /home/ayman/eclipse-workspace/WSDLs/WS.wsdl       |
| s4          | P4   | P5   | /home/ayman/eclipse-workspace/WSDLs/WS.wsdl       |
| s5          | P3   | P1   | /home/ayman/eclipse-workspace/WSDLs/WS.wsdl       |
| s6          | P2   | P4   | /home/ayman/eclipse-workspace/WSDLs/WS.wsdl       |
| s7          | P3   | P4   | /home/ayman/eclipse-workspace/WSDLs/Service2.wsdl |
| s8          | P6   | P1   | /home/ayman/eclipse-workspace/WSDLs/Service2.wsdl |
| s9          | P1   | P4   | /home/ayman/eclipse-workspace/WSDLs/Service2.wsdl |
| s10         | P2   | P5   | /home/ayman/eclipse-workspace/WSDLs/Service2.wsdl |
| s11         | P5   | P7   | /home/ayman/eclipse-workspace/WSDLs/WS.wsdl       |
| s12         | P7   | P3   | /home/ayman/eclipse-workspace/WSDLs/WS.wsdl       |
| s13         | P7   | P8   | /home/ayman/eclipse-workspace/WSDLs/WS.wsdl       |
| s14         | P9   | P3   | /home/ayman/eclipse-workspace/WSDLs/WS.wsdl       |
| s15         | P0   | P7   | /home/ayman/eclipse-workspace/WSDLs/WS.wsdl       |
| s16         | P9   | P1   | /home/ayman/eclipse-workspace/WSDLs/WS.wsdl       |
+-------------+------+------+-----------------------------------------------+
16 rows in set (0.03 sec)
```

Figure 7.2 – MySQL Data base

— Receiving the response from registry, start searching composition with an algorithm(build a tree to find correspondence output);

— Composing the servers and send to client (if the composition is found, otherwise send a message with corresponding result);

— Downloading the WSDL files from registry and send it to client.

The composition engine will create the root node of the tree representing the input parameter requested by the application and create three tables that will help to create the tree. The composition engine will, each time, request the name of services as well as the name of the output parameters for a given input in the aim of creating a tree to find the output parameter requested by the application.

So, at the beginning there is a set of clients which are requesting the Application to invoke Composition Engine for the services at the same time. They are doing it using the messages consists of the information of which parameters should be considered as an input and the output we want to receive. When the message is delivered, Composition engine starts the searching algorithm

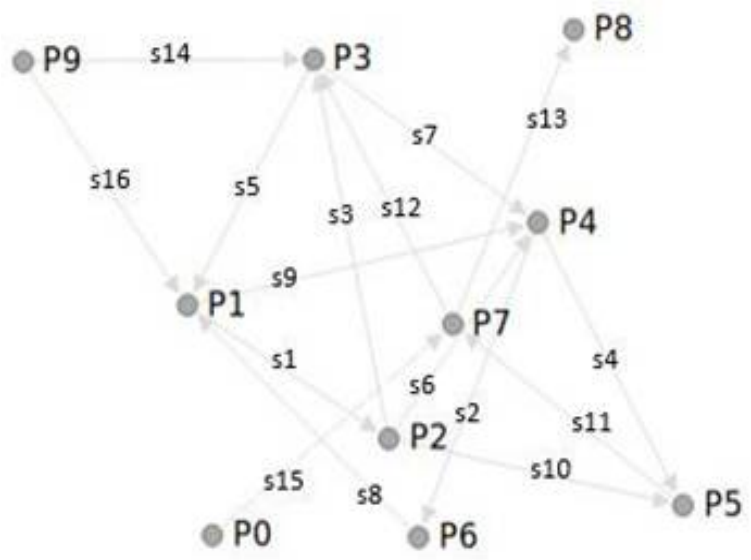

Figure 7.3 – Transformation of the database to the Graph

which is sending the messages to the Registry step by step checking if the parameters are exist and which outputs they are produced. This approach is based on searching in a tree algorithm.

At the Database there is set of rows described the relationship between the 2 parameters: one with inputs and one with the outputs. So that we can build a graph for simple representation of the relationship between parameters. In this graph it is possible to find the path from each vertex to the others. Checking all the neighbors of a vertex we are checking their neighbors and so on while the simulator can find (or we sure it cannot) the corresponding output. After the Composition Engine complete the composition services path, it sends that path to the client, and afterall, sends the WSDL files corresponding to the path.

Now imagine that Global Registry consists of millions of rows, meanwhile in your application, for example, you are using not more than thousand (the most popular ones). Or, for example, you are using several registries of services and to make a composition you have to consider all rows in all registries, which is, obviously, takes a lot of time. So that the searching time would be huge each time while searching.

Now the proposed solution is considering adding to the simulator Local Composition Engine (LCE) and Local Registry (LR). In this case the LCE and LR will be stored locally with the Application, and so, according to our

assumption, could help us to decrease the time to receive the response. To improve the summary time to receive response the proposition is a solution based on dividing the structure of our architecture by 5 components:

— Application (to serve a client);

— Local Registry which contains just needable information while;

— Global Registry would contain all the information about the services;

— Local Composition Engine which would work directly with application and;

— Global Composition Engine to communicate with Global Registry.

Here you can see a description of each component, all the procedures they do and how they are connected to each other. Here you can see the general structure of the code. It is important to note, that Local Composition Engine, Local Registry and Application (to serve a Client) could be put as one component (even, for example, same machine or somewhere locally, on the client side) while Global Composition Engine and Global Registry could be placed elsewhere.

So that, according to our assumption, Client asks an Application for certain outputs, providing some inputs for it. At this point, Application sends a message of a client to the Local Composition Engine with inputs. Using its own algorithm, Local Composition Engine requests Local Registry for the information. After the communication is finished, Local Composition Engine makes a decision if it needs to send the same request to the Global Composition Engine (in case we didn't find the path - send a message). And then Composition Engine makes the same operation with Global Registry and sends response to LCE (path with files). After receiving a response and has already downloaded the files, Local Composition Engine needs to send the response to Application (path with files) and send the same files and the information to the Local Registry to put data in it.

As you can see, at the beginning, when the Local Registry is empty, the time for the response will be even bigger, than if use simple system from the first approach. But after the Client requested for the same outputs using already used inputs, we can see that finding the data in Local Composition Engine, which is much smaller, takes less time, so that this approach can reach the goal of the project. The detailed architecture of the project you can see below from Java point of view you can see below. Here we have set of Interfaces, market as blue, and also classes (yellow) 7.4.

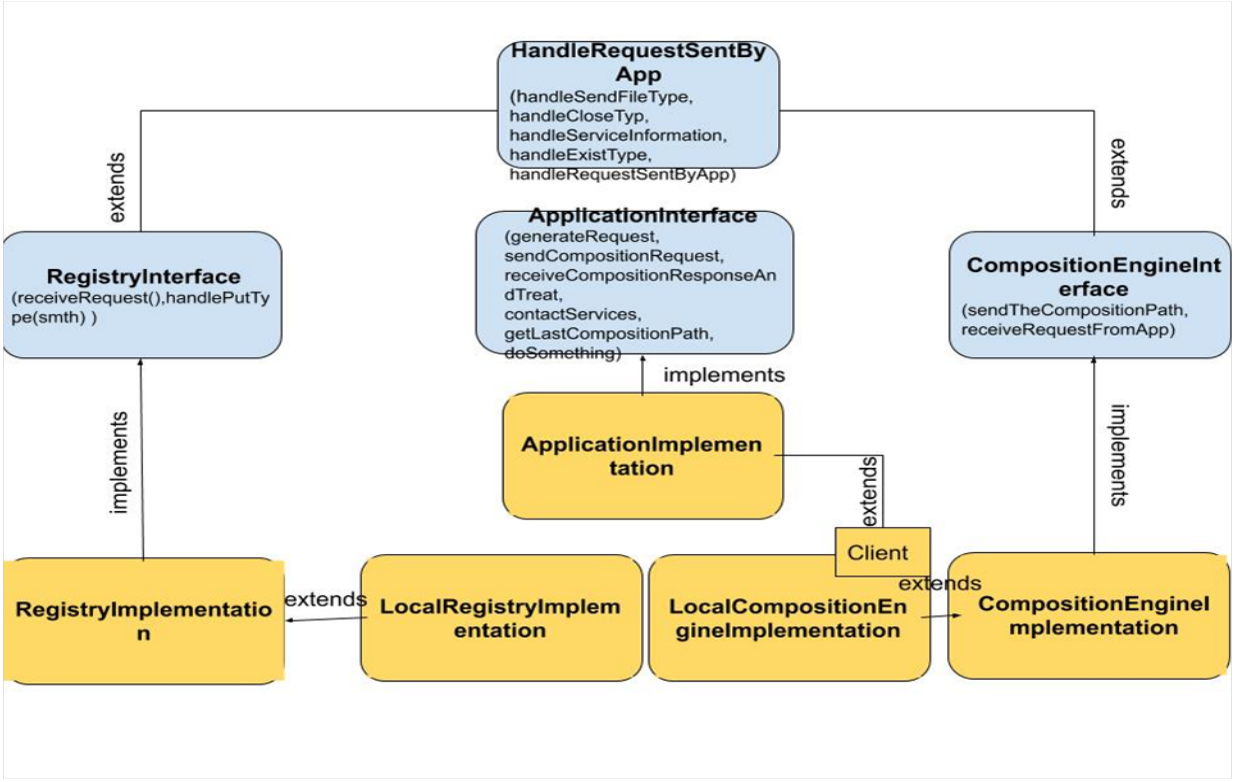

Figure 7.4 – Architecture of the Simulator

7.4.2 Distributed Dynamic Service Composition framework generated

Our proposed dynamic service composition approach leads to a generated dynamiccomposition framework presented in figure 7.5 is an hierarchical structure of three main levels. We have the local level, the intermediate our middle levels which the number depends on the administrative or territorial structure of the organization, and the national level. Each level is composed of two components: a composition plan library, a service repository. A monitoring support makes it possible to establish coherent communication at the different levels and is responsible for updating the service registries and libraries.

In fact, the process begins by the sending of a user's request to the client system. The client system sends the request to his local composition engine. This local composition engine checks in its composition library if it has already processed such a request. If so, its composition path is then used to process the

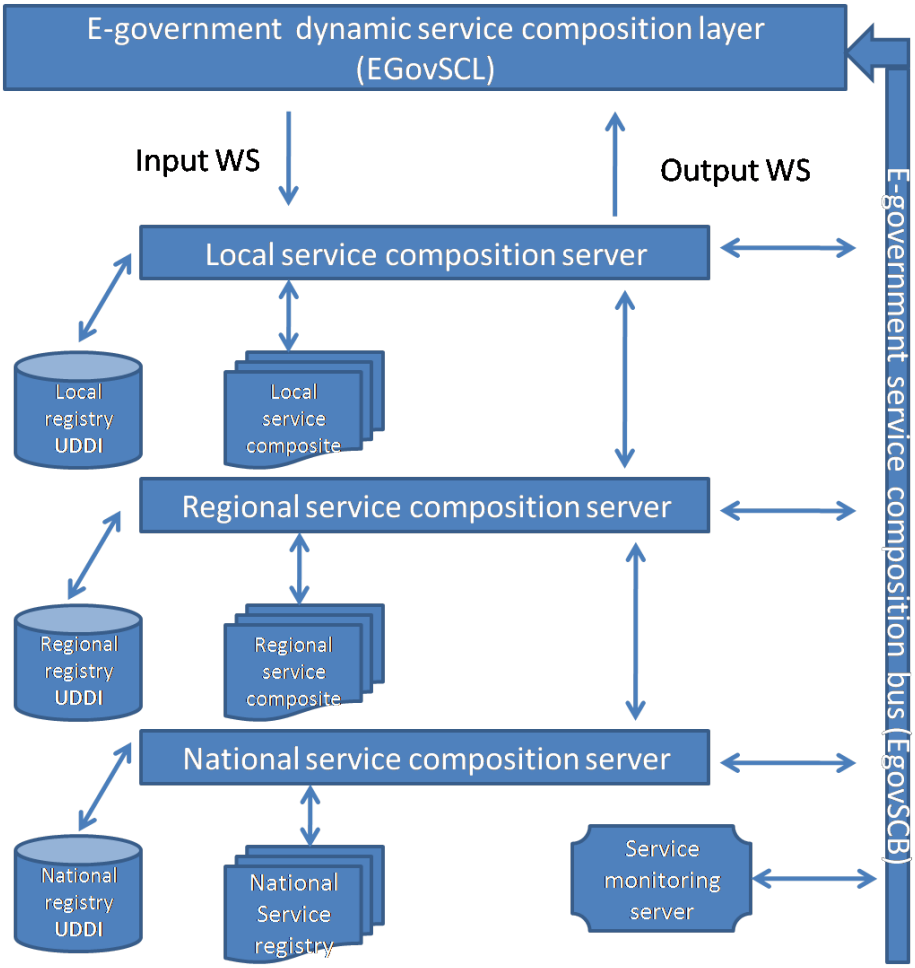

Figure 7.5 – DSC layer for Cameroon SOA based e-government

request without involving the remote composition server and by relying on the local registry. The composition path is then return to the local composition which will return it to the client. If this request is new for the system, it is sent to the remote service composition which will repeat the same process at its level.

7.4.3 Architecture

Based on the the figure 7.1 we can deduce the following figure 7.6 as the old approach architecture of the the centralized e-government's project.

with our 2DSC we can deduce the following decentralized architecture 7.7

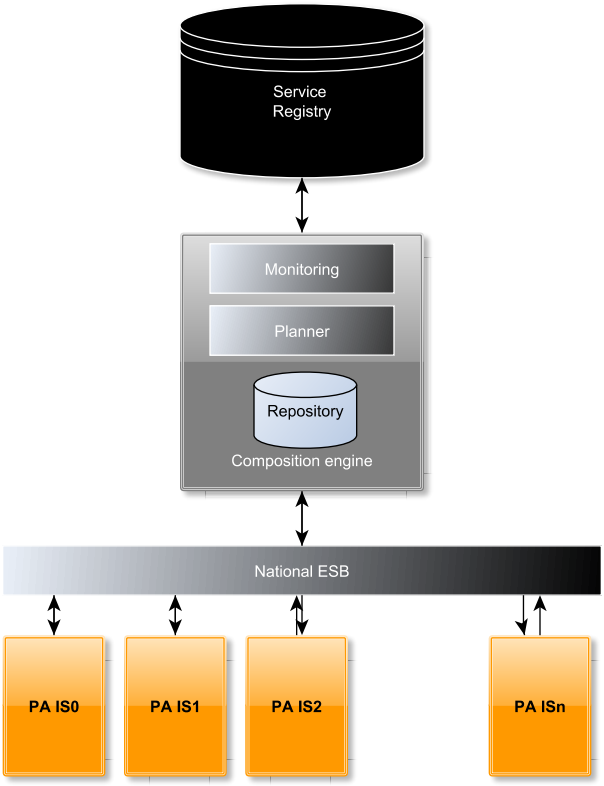

Figure 7.6 – Current DSC for e-government

as the new architecture.

7.4.4 Technical environment

— Environment Intellij IDEA: it is a Java integrated development environment. It is User-Friendly and easy to integrate with database. It is also GIT-supportive;

— Programming languages: we used Java. It is a popular object-oriented programming language with code reusability. It is used to build the simulator which is divided into blocks (Application part, Composition Engine part, Registry part);

— Data base: we used MySQL which is a relational database management system. It is used to store the information about services;

— Operating system: Our OS was Linux (Ubuntu). It is an open-source

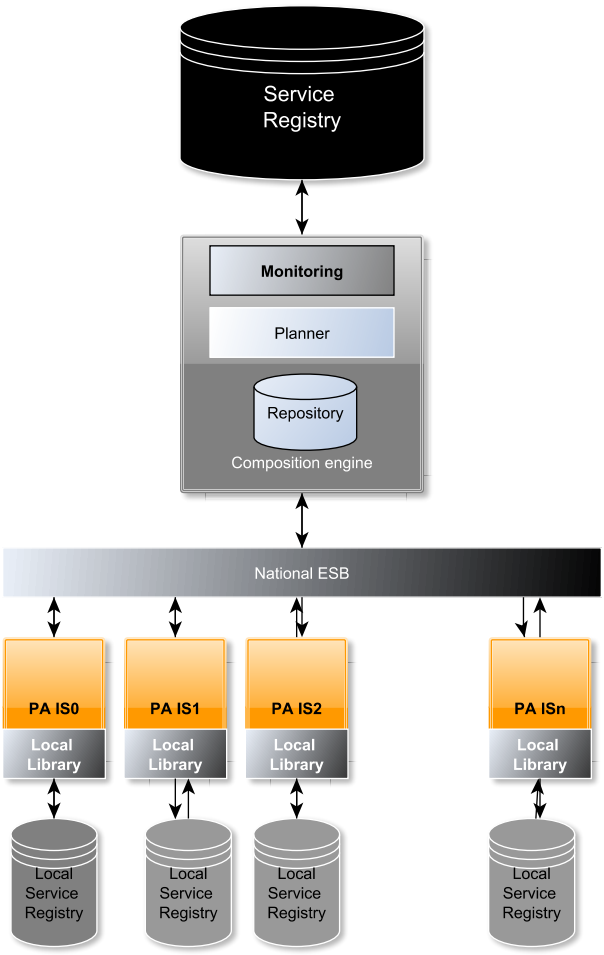

Figure 7.7 – 2DSC for Service oriented e-government

software operating system, fast for development, highly secured.

7.4.5 Test's results

This section is reserved for the presentation of the results of the tests carried out to test the theoretical demonstrations made around our model. We will launch dynamic composition processes without the distributed architecture by gradually increasing the data to observe the behavior of the device. Then we will resume this process in an environment integrating our approach. A comparison will then be made to highlight the performance gaps of our

2DSC model. At the end, we will simulate a power failure by successively shutting down the remote composition engine and the service registry. We will thus draw the beneficial consequences of our approach in an e-government environment in developing countries.

It should be noted that in the modeling that follows, each application is considered a computer system of a government structure. Thus, the experiments made between several applications represent the cooperation between these different state entities.

7.4.5.1 Basic test results

Let us consider the public agency A that send P_4 as input and P_1 as output. we need services s_2, s_8. Let us consider the public agency B that send P_1 as input and P_5 as outputs. we need services s_1, s_6 and s_4. Let us consider the public agency C that send P_4 as input and P_3 as outputs. we need services s_2, s_8,s_1 and s_3. After making some experiments we receive the time to reach the response in the 2DSC and the approach without 2DSC noted in the table7.2.

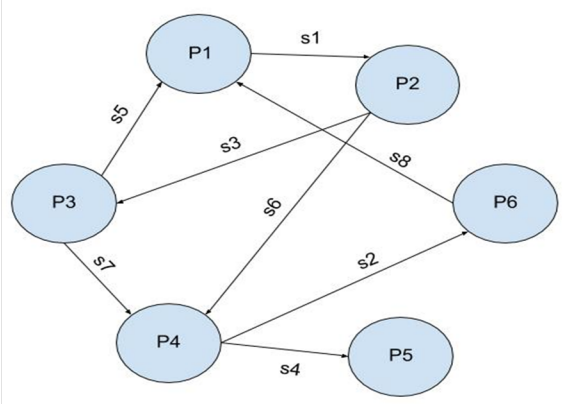

Figure 7.8 – Basic test results

Table 7.2 – Basic test's results with estimation time in seconde

Requests	Services	Without 2DSC	With 2DSC /first time	With 2DSC/ second time
$P_4 \rightarrow P_1$	S_2, S_8	0,214846138	0,486304783	0,166178036
$P_1 \rightarrow P_5$	S_1, S_6, S_4	0,416030016	0,860169026	0,253286912
$P_4 \rightarrow P_3$	S_2, S_8, S_1, S_3	0,416891526	0,919836546	0,325509476

So the request column in the table 7.2 means the inputs and outputs given by user. Here each P consists of a set of some parameters. The next column is dealing with the services used to reach the output parameters. You can see the response time with a new approach, performed for the first time, when the registry is empty and the request needs to pass all the way to the local composition engine, remote composition engine and both registries. And at the end, you can see results when the simulator puts already something in the local registry, so that there is no need to make the big circle. As we can observe, the time of response to reach the application is decreased by 20-30%.

One of the challenges was also to check how the simulator will be work both with big amount of requests from one Application and when the amount of Applications is decreased. For this purpose some amount of requests (10, 25, 100, 1000) was considered and checked how the simulator is doing in case of 1 application is requesting for the response and if there are 2 or 4. So, after tasting we obtained such results:

7.4.5.2 Case of dynamic service composition without 2DSC approach

Table 7.3 – Simulation without 2DSC approach with estimation time in seconde

Requests	1 application	2 applications	4 applications
10	14.95	15.05	17.69
25	36.29	36.85	49.27
100	154.19	158.10	235.08
1000	1537.06	1583.76	2354.13

As we can see in the table 7.3, the response time for 4 application increased almost 1.5 times compared with the results of the work of 1 application. This increase in the response time we observe is consistent with our approach which considers that this response time of a request depends both on the amount of data (number of services available) in the service registry and the number of composition plan clients requesting the composition server. Thus, as the number of clients increases, the response time becomes as long. Just as it gets long when the number of service increases in the registry. This is one of the problem our approach have to resolve.

7.4.5.3 Case of dynamic service composition based on distributed approach

Table 7.4 – Simulation with 2DSC approach with time estimation in seconde

Requests	1 application	2 applications	4 applications
10	11.55	13.55	13.45
25	36.14	34.02	36.22
100	143.99	151.88	162.52
1000	1266.81	1492.51	2062.95

The table 7.4 shows that at the same time, the Simulator presents good results with our distributed approach. Certainly this time always increases when several applications solicit the composition plan server, but it does not

believe with the same ampleness. Also, according to the initial example 7.2, during the next execution, our approach will still allow to reduce the response time. This can leed us to the next comparisons.

7.4.5.4 Comparison of results

In the figure 7.9, the figure 7.10 and the figure 7.11, blue line is performance with current approach and red one is with 2DSC approach.

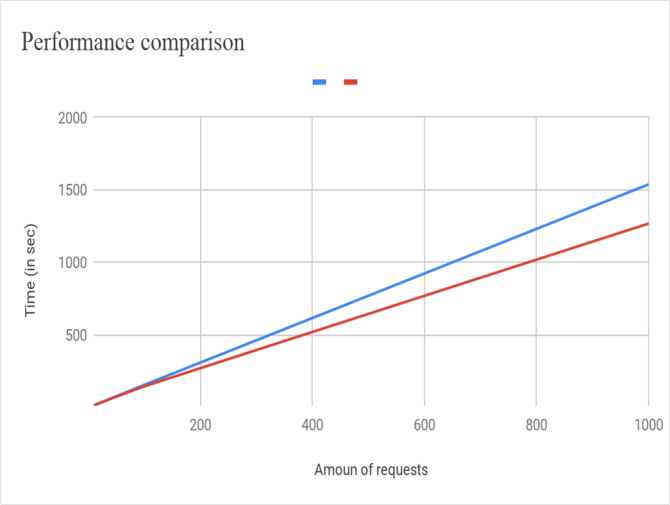

Figure 7.9 – Graphic representation of performance on current and new approaches in case of 1 Application

As we can see, the 2DSC approach is decreasing the time for response to a client. It should be noted that performing a request takes a long time. For the case of 1 or 2 applications, the performing time is very similar, but if to compare with the case of 4 applications, it takes approximately 1.5 times more to perform requests come from 4 applications.

7.4.5.5 Latency and Throughput

So, to measure the latency we count how big is the tail latency, and to achieve this goal we counted the amount of different requests (for which we suppose to call remote composition server), choose the max from it and the size of a registry and divide on the amount of requests. Throughput is counted by the amount of request done per second. As you can see it increased 30% compared with the current approach.

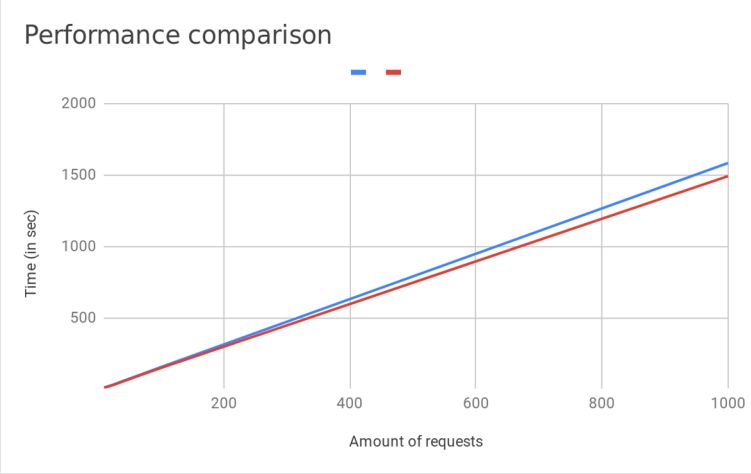

Figure 7.10 – Graphic representation of performance on current and new approaches in case of 2 applications

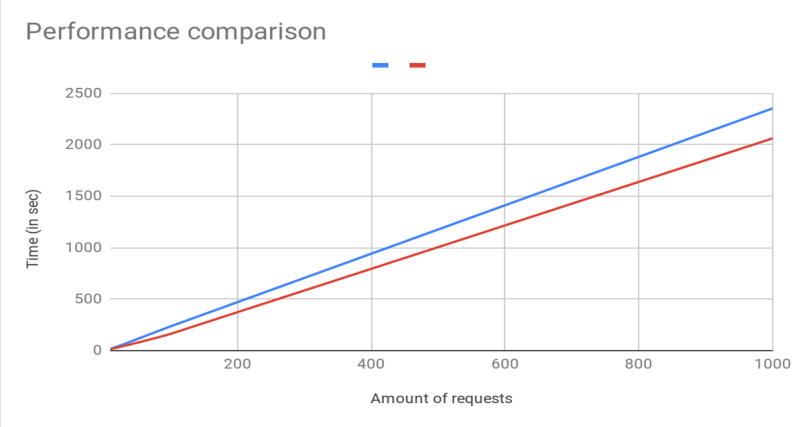

Figure 7.11 – Graphic representation of performance on current and new approaches in case of 4 applications

7.4.5.6 Case of power failure

Developing countries are mainly characterized by the lack of permanent and stable electrical energy. To assess the ability of our approach to better integrate in this type of country like Cameroon, we will decommission the remote composition engine and the service registry after a certain number of operating hours and launch the experiments and evaluate the failure rate of the

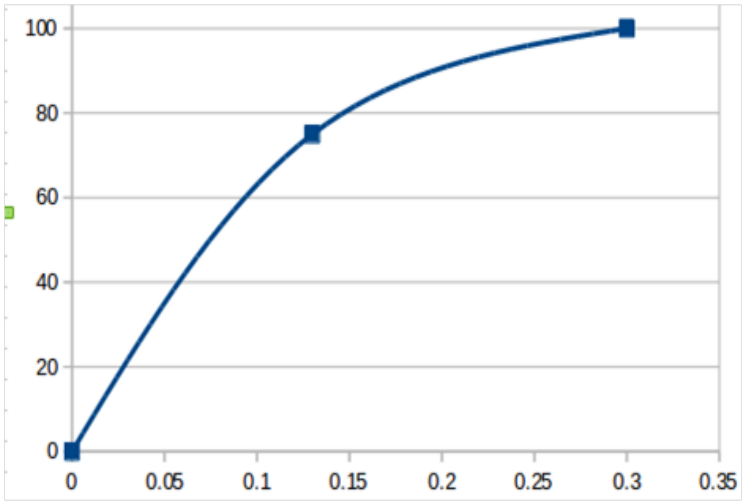

Figure 7.12 – Representation of tail latency

requests in one application. It is obvious that the current approach does not allow the operation of the system when one of the components is unavailable. For this reason, the automaton of each system is one of the priorities in the environment of dynamic service composition in developing countries.

Table 7.5 – Simulation of power failure

Time in hour	Rate of failure
10	50.45
25	24.22
100	15.52
1000	4.95

It is observed that the number of requests unable to find an answer locally when the remote composition server is unavailable is reduced over time. It is the same with the unavailability of the registry. This proves that our approach allows to build in time a certain autonomy with the client of composition. This may allow the running of a system, although some entities may not be able to collaborate for multiple reasons. What is important in developing countries like Cameroon.

7.4.5.7 Conclusion

The above results that emerge from the experiments have sufficiently shown the effectiveness of the dynamic service composition model based on the 2DSC architecture. These results allowed observing a net saving of time during dynamic service composition processes when the client applications of compositions have the capacity to store composition plans that are served to them during their requests. These plans can also be used to build local registries that can be used to generate local composition plans. This autonomy is therefore observed within these client applications over time with respect to the remote composition server. This ability to empower composition plan customers not only frees the composition engine, but also gives them the opportunity to deal with certain constraints related to infrastructural failures, particularly those related to electrical energy, which is fundamental to the proper functioning of computer systems.

7.5 Synthesis

In the previous chapter an distributed dynamic service composition framework has been presented. It is an approach of decentralization of the dynamic service composition based on personalized publication of composite services. Theoretically, this approach makes the process more effective than the current centralized method. This chapter is focused on the experimental validation of our new model and especially its adaptability to improve the performance of e-government in developing countries like Cameroon.

It started by revisiting the general concept of e-government, the stages of its implementation and its issues for its proper functioning. The chapter presented some issues of implementation of e-government based on service-oriented architecture in developing countries. We finally presented the SOA based e-government that takes into account our approach and evaluated its effectiveness in the context of developing countries like Cameroon.

In general, the 2DSC approach has enormous experimental performance. There is a marked improvement in the response time during transactions in the system. To take into account some infrastructural problems such as lack of electrical power that can disturb availability of some resources, we have simulated the system by decommissioning some infrastructures such as the composition plan server or the registry to observe a satisfactory rate of successful transactions.

Also, this new approach is sufficiently adapted to decentralised and hierar-

chical architectures like the public administration. It is thus possible, to place each information system as a composition plan server or registry for others systems which are situated in the lower administrative level. And this is very effective.

The next part will conclude our research process.

Part IV

GENERAL CONCLUSION

Ignorance more frequently breeds self-confidence than does knowledge.

C. Darwin

8 Conclusion

Contents

8.1 Context

Cameroon has launched for several years a major project to modernize its administration. It has developed technological infrastructure to support this great ambition aimed at collaboration between its various information systems. Also, advances in web technologies and standards have strongly contributed to the adoption of service-oriented architectures as a new architectural paradigm for interoperable application development. One of the major advantages of this model is service composability. But, it is sometimes necessary to collaborate several services in order to fill a more complex task. This is called service composition. It can be static or dynamic, manual or automatic. Taking advantages to the advent of the semantic web, dynamic service composition has taken over the static model which presents many gaps on the evolutionary level of applications.

In an another plan, dynamic service composition has a life cycle. It begins with the description of the services, their location, their selection, the generation of the composition plans capable of giving an effective response to the processed request, the selection and execution of the best composition plan, and the publication of the composite service. Thus, to optimise the dynamic service composition and deal with scalability issue, the researchers proposed methods to make each of these steps less expensive.

But with the development of many services and the multiplicity of actors in the cyberspace, dynamic composition optimisation methods lose their efficiencies and must deal with the scalability issue. So these algorithms are inefficient in real world. It is in this context that this work was done.

8.2 Problem

The main problem of this work consisted in defining on the basis of the known sources of scalability, an approach able to optimize the dynamic service composition process. To ensure that the main problem is answered, the following secondary issues were addressed:

1. Analysis of current solutions to deal with scalability in dynamic service composition related its three known sources;
2. On the basis of this analysis, the second issue was to propose a new approach which can deal with the scalability problem in dynamic service composition process;
3. The final issue was to propose a model more adapted to implement the Cameroon's e-government project based on service oriented architecture.

These concerns have been addressed during this work and the results have been clearly laid out.

8.3 Methodology

Among its many advantages, scalability is one of the most important design goals for developers of distributed systems because of Worldwide connectivity through the Internet is rapidly becoming as common as being able to send a postcard to anyone anywhere around the world (Raymond, 1995). A system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity (ord Neuman, 1994). This is why the distributed approach seemed relevant to us to provide a solution to the concerns observed during our analysis of the solutions in the literature review.

8.4 Contribution

Over the course of these, service-oriented architectures have been studied extensively and an emphasis has been placed on the work done to deal with the scalability in dynamic service composition process which is our main research subject.

As the main contribution, this work proposed a dynamic service composition approach capable to deal with the difficulties related to scalability. We can mention specifically:

8.4.1 Analysis of the impact of current approach on the known 3 causes of scalability problem in dynamic service composition

The state of art has allowed us to make a global study of the solutions that the researchers brought to the scalability issue which slows down the adoption of the dynamic service composition in the real world. We evaluated the impact of each solution on the three known causes of scalability in the dynamic service composition issue. We observed that these solutions consist in performing each stage of the composition process but do not impact all the three known causes of scalability. The researches are focused in great nomber of service and the complexity of client requests. Indeed, the number of requests which is one

of the sources of scalability in dynamic service composition is not taken into account in the different approaches.

Indeed, the analysis shows that the stages of the dynamic service composition life cycle are oriented towards the two main sources of scalability namely the number of services and the complexity of the requests. The third cause, which is the number of requests, has no directly linked with the activities of performing each stage of the process. So it cannot therefore be solved by performing the stages of the dynamic service composition process. This may explain why it is not taken into account like the other two causes of scalability.

It is on this basis that we have proposed a distributed approach which to deal with the scalability by taking into account the number of requests aspect.

8.4.2 Distributed approach to deal with scalability problem in dynamic service composition

One of the known causes of scalability is not taken into account in the current solutions which are performing each stage of a dynamic service composition process. This is the number of customer requests. This important factor can explain the persistent scalability problems in the dynamic service composition despite the major solutions already existed.

To take this factor into account, we propose a distributed system involving the infrastructure of the end user systems in this process. We have proposed an architecture that can support this vision. We have demonstrated theoretically and then by experimentation that this approach is optimal compared to the non-distributed approach. The main components of our system were presented, their interactions were structured and a monitoring middleware for the synchronization between the client systems and the remote server was detailed.

We have first formalized the objective of our idea of performing the process to make our approach understandable as well as the solution. Our approach makes each system semi-autonomous over time. This approach aims to decentralize the dynamic composition operations of services in all end-user systems of the composition server. This approach decongests the composition server, which finds itself managing only the new requests that arrive in the system.

Concretely, we consider a composition plan as a file desired by a end users. We therefore propose to publish this and store this information at the client level and at composition server level. This would make it possible to locally build a registry of composite services that can locally provide responses to redundant requests submitted in a client systems. This approach also makes it possible to cluster the services in a personalised orientation. Also, this

approach seems to be adapted to the environments of the developing countries because it requires very little in time, a strong collaboration between the principal entities which are the customer, the global services registry and the remote composition engine.

8.5 Limits and futurs works

Our analysis presented a limit of the solutions aiming to solve the problem of scalability in the dynamic service composition. We have proposed for this a new distributed approach which uses end user infrastructures participation to solve this scalability problem. In view of the proposed architecture and the collaboration mechanisms between its various components, there are several aspects which are identified as possible directions for future research activities and some limits.

8.5.1 Develop a dynamic service composition tool

The limit of our study is basically linked in the fact that we have not found a framework of dynamic service composition on which to evaluate our approach and make comparisons. This is why in the future works, we will develop this tool.

8.5.2 Monitoring optimization

Our architecture requires a middleware that ensures consistency between the information available at the composition server level and the composition client systems. For optimal operation and correct synchronization between the various players, it is necessary to define a less expensive communication policy so as not to overload the network. We can foresee an offline communication by detecting the inactivity of an actor and offering him updates for his system. Moving forward, we plan to also integrate the load balancing process to only offer effective composition plan using services with low point solicitation.

8.5.3 Grid of dynamic service composition service

It is also possible to think about setting up a dynamic service composition grid. This space of mutualization which would make it possible to put to use several end-users to generate composition plans in parallel. In fact, the decentralization of dynamic service composition lead to a distributed environment

and it is possible to think about the pooling of resources. Indeed, it is possible to each end-user system to build autonomously composition plans able to be shared with other end-users. This would perform the process but will require a monitoring and a certification mechanism to secure and harmonise the approach.

Bibliography

Abadi, M. and Manna, Z. (1986). Modal theorem proving. In *International Conference on Automated Deduction*, pages 172–189. Springer.

Akkiraju, R., Farrell, J., Miller, J. A., Nagarajan, M., Sheth, A. P., and Verma, K. (2005). Web service semantics-wsdl-s.

Alghamdi, I. A., Goodwin, R., and Rampersad, G. (2011). E-government readiness assessment for government organizations in developing countries. *Computer and Information Science*, 4(3):3.

Alwasouf, A. A. and Kumar, D. (2019). Research challenges of web service composition. *Software Engineering*, pages 681–689.

ANTIC (2017). Stratégie des télécommunications et tic.

Ardagna, D., Comuzzi, M., Mussi, E., Pernici, B., and Plebani, P. (2007). Paws: A framework for executing adaptive web-service processes. *IEEE software*, (6):39–46.

Arkin, A., Askary, S., Fordin, S., and Jekel et al., W. (2002). Web Service Choreography Interface (WSCI) 1.0. Standards proposal by BEA Systems, Intalio, SAP, and Sun Microsystems.

Atsa Etoundi, R. and Ndjodo, M. F. (2005). Human resource constraints driven virtual workflow specification. In *SITIS*, pages 176–182.

Barrett, R., Patcas, L. M., Pahl, C., and Murphy, J. (2006). Model driven distribution pattern design for dynamic web service compositions. In *Proceedings of the 6th international conference on Web engineering*, pages 129–136. ACM.

Baryannis, G. and Plexousakis, D. (2010). Automated web service composition: State of the art and research challenges. *ICS-FORTH, Tech. Rep*, 409.

Belouadha, F.-Z., Omrana, H., and Roudies, O. (2012). A mda approach for defining ws-policy semantic non-functional properties. *arXiv preprint arXiv:1201.1481*.

Benatallah, B., Hacid, M., Léger, A., Rey, C., and Toumani, F. (2005). On automating web services discovery. *VLDB J.*, 14(1):84–96.

Benatallah, B., Hacid, M., Rey, C., and Toumani, F. (2003a). Request rewriting-based web service discovery. In *The Semantic Web - ISWC 2003, Second International Semantic Web Conference, Sanibel Island, FL, USA, October 20-23, 2003, Proceedings*, pages 242–257.

Benatallah, B., Hacid, M.-S., Rey, C., and Toumani, F. (2003b). Request rewriting-based web service discovery. In *International semantic web conference*, pages 242–257. Springer.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. *Scientific american*, 284(5):34–43.

Bernstein, P. A. (1996). Middleware: a model for distributed system services. *Communications of the ACM*, 39(2):86–98.

Bertot, J. C., .Jaeger, P. T., and Grimes, J. M. (2010). Using icts to create a culture of transparency: E-government and social media as openness and anti-corruption tools for societies. *Government Information Quarterly 27*.

Bevilacqua, L., Furno, A., Di Carlo, V. S., and Zimeo, E. (2011). A tool for automatic generation of ws-bpel compositions from owl-s described services. In *2011 5th International Conference on Software, Knowledge Information, Industrial Management and Applications (SKIMA) Proceedings*, pages 1–8. IEEE.

Blake, M. B., Tan, W., and Rosenberg, F. (2010). Composition as a service [web-scale workflow]. *IEEE Internet Computing*, 14(1):78–82.

Bosca, A., Ferrato, A., Corno, F., Congiu, I., and Valetto, G. (2005). Composing web services on the basis of natural language requests. In *Web Services, 2005. ICWS 2005. Proceedings. 2005 IEEE International Conference on*. IEEE.

Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., and Winer, D. (2000). Simple object access protocol (soap) 1.1.

Bucchiarone, A. and Gnesi, S. (2006). A survey on services composition languages and models. *International Workshop on Web Services Modeling and Testing*.

Carman, M., Serafini, L., and Traverso, P. (2003). Web service composition as planning. In *ICAPS 2003 workshop on planning for web services*, pages 1636–1642.

Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., and Shan, M.-C. (2000a). Adaptive and dynamic service composition in eflow. In *International Conference on Advanced Information Systems Engineering*, pages 13–31. Springer.

Casati, F., Ilnicki, S., Jin, L.-J., Krishnamoorthy, V., and Shan, M.-C. (2000b). eflow: a platform for developing and managing composite e-services. In *Proceedings Academia/Industry Working Conference on Research Challenges' 00. Next Generation Enterprises: Virtual Organizations and Mobile/Pervasive Technologies. AIWORC'00.(Cat. No. PR00628)*, pages 341–348. IEEE.

Castillo, P. A., Bernier, J. L., Arenas, M. G., Merelo, J., and Garcia-Sanchez, P. (2011). Soap vs rest: Comparing a master-slave ga implementation. *arXiv preprint arXiv:1105.4978*.

Ceri, S., Comai, S., Damiani, E., Fraternali, P., Paraboschi, S., and Tanca, L. (1999). Xml-gl: a graphical language for querying and restructuring xml documents1the work presented in the paper has been supported by esprit project nr. 28771 'w3i3', and murst project 'interdata'.1. *Computer Networks*, 31(11):1171 – 1187.

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., et al. (2001). Web services description language (wsdl) 1.1.

Clement, L., Hately, A., von Riegen, C., Rogers, T., et al. (2004). Uddi version 3.0. 2, uddi spec technical committee draft. *OASIS UDDI Spec TC*.

Cremene, M., Tigli, J.-Y., Lavirotte, S., Pop, F.-C., Riveill, M., and Rey, G. (2009). Service composition based on natural language requests. In *Services Computing, 2009. SCC'09. IEEE International Conference on*, pages 486–489. IEEE.

Crotty, M. (1998). *The foundations of social research: Meaning and perspective in the research process*. Sage.

Das, R., Patra, M., and Misro, A. (2010). Open source soa for e-governance. In *7th International Conference on E-Governance ICEG*.

de Bruijn, J., Lausen, H., Polleres, A., and Fensel, D. (2006). The web service modeling language wsml: An overview. In Sure, Y. and Domingue, J., editors, *The Semantic Web: Research and Applications*, pages 590–604, Berlin, Heidelberg. Springer Berlin Heidelberg.

De Vaus, D. (2001). *Research design in social research*. Sage.

Demissie, M. G., Phithakkitnukoon, S., Sukhvibul, T., Antunes, F., Gomes, R., and Bento, C. (2016). Inferring passenger travel demand to improve urban mobility in developing countries using cell phone data: a case study of senegal. *IEEE Transactions on intelligent transportation systems*, 17(9):2466–2478.

D.Fensel, C.Bussler, A. (2002). Semantic web enabled web services. *ACM*, 31.

Domingue, J., Roman, D., and Stollberg, M. (2005). Web service modeling ontology (wsmo)-an ontology for semantic web services.

Ehrig, M., Koschmider, A., and Oberweis, A. (2007). Measuring similarity between semantic business process models. In *Proceedings of the fourth Asia-Pacific conference on Comceptual modelling-Volume 67*, pages 71–80. Australian Computer Society, Inc.

Endong, F. (2020). *Prospects and Challenges of E-Government in Black Africa: A Comparative Study of Nigeria and Cameroon*, pages 662–677.

Erl, T. (2005). *Service-Oriented Architecture: Concepts, Technology, and Design*.

Erl, T., Carlyle, B., Pautasso, C., and Balasubramanian, R. (2012). *Soa with rest: Principles, patterns &constraints for building enterprise solutions with rest*. Prentice Hall Press.

Eugster, P. T., Felber, P. A., Guerraoui, R., and Kermarrec, A.-M. (2003). The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131.

Falou, M. E., Bouzid, M., Mouaddib, A.-I., and Vidal, T. (2009). Automated web service composition: a decentralised multi-agent approach. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web*

Intelligence and Intelligent Agent Technology-Volume 01, pages 387–394. IEEE Computer Society.

Fan, G., Yu, H., Chen, L., and Liu, D. (2013). Petri net based techniques for constructing reliable service composition. *Journal of Systems and Software*, 86(4):1089–1106.

Fikes, R., Hayes, P., and Horrocks, I. (2004). Owl-ql: A language for deductive query answering on the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(1).

Flannelly, K. J. and Jankowski, K. R. (2014). Research designs and making causal inferences from health care studies. *Journal of health care chaplaincy*, 20(1):25–38.

Fonou Dombeu, J. V. and Rannyai, N. (2014). African e-government research landscape. *The African Journal of Information Systems*, 6(3):2.

Foster, H., Uchitel, S., Magee, J., and Kramer, J. (2003). Model-based verification of web service compositions. In *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on*, pages 152–161. IEEE.

Garriga, M., Mateos, C., Flores, A., Cechich, A., and Zunino, A. (2016). Restful service composition at a glance: A survey. *Journal of Network and Computer Applications*, 60:32–53.

Gekas, J. and Fasli, M. (2005). Automatic web service composition based on graph network analysis metrics. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1571–1587. Springer.

Grimm, R., Davis, J., Lemar, E., Macbeth, A., Swanson, S., Anderson, T., Bershad, B., Borriello, G., Gribble, S., and Wetherall, D. (2004). System support for pervasive applications. *ACM Transactions on Computer Systems (TOCS)*, 22(4):421–486.

Guan, H.-J., Meng, F.-R., Sun, J.-F., and Du, P. (2008). Web service discovery based on the cooperation of uddi and df. In *Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on*, pages 1–4. IEEE.

Hammami, R., Bellaaj, H., and Kacem, A. H. (2018). Semantic web services discovery: A survey and research challenges. *Int. J. Semantic Web Inf. Syst.*, 14(4):57–72.

Hashemian, S. V. and Mavaddat, F. (2005). A graph-based approach to web services composition. In *The 2005 Symposium on Applications and the Internet*, pages 183–189. IEEE.

Hassina Nacer Talantikite, Djamil Aissani, N. B. (2009). Semantic annotations for web services discovery and composition. *Computer Standards and Interfaces*, 31.

Hatzi, O., Vrakas, D., Bassiliades, N., Anagnostopoulos, D., and Vlahavas, I. (2013). The porsce ii framework: Using ai planning for automated semantic web service composition. *The Knowledge Engineering Review*, 28(2):137–156.

Heeks, R. and Bailur, S. (2007). Analyzing e-government research: Perspectives, philosophies, theories, methods, and practice. *Government information quarterly*, 24(2):243–265.

Hossain, N., Yokota, F., Sultana, N., and Ahmed, A. (2019). Factors influencing rural end-users' acceptance of e-health in developing countries: a study on portable health clinic in bangladesh. *Telemedicine and e-Health*, 25(3):221–229.

Huang, C., Xu, L. D., Cai, H., Li, G., Du, J., and Jiang, L. (2019). A context-based service matching approach towards functional reliability for industrial systems. *Enterprise IS*, 13(2):196–218.

Jaeger, M. C., Rojec-Goldmann, G., Liebetruth, C., Mühl, G., and Geihs, K. (2005). Ranked matching for service descriptions using owl-s. In *Kommunikation in Verteilten Systemen (KiVS)*, pages 91–102. Springer.

Jiang, W., Zhang, C., Huang, Z., Chen, M., Hu, S., and Liu, Z. (2010). Qsynth: A tool for qos-aware automatic service composition. In *2010 IEEE International Conference on Web Services*, pages 42–49. IEEE.

Johnston, A. (2014). Rigour in research: theory in the research approach. *European Business Review*, 26(3):206–217.

Jula, A., Sundararajan, E., and Othman, Z. (2014). Cloud computing service composition: A systematic literature review. *Expert systems with applications*, 41(8):3809–3824.

Karakoc, E. and Senkul, P. (2009). Composing semantic web services under constraints. *Expert Systems with Applications*, 36(8):11021 – 11029.

Kavantzas, N., Burdett, D., Ritzinger, G., Fletcher, T., Lafon, Y., and Barreto, C. (2005). Web services choreography description language version 1.0. World Wide Web Consortium, Candidate Recommendation CR-ws-cdl-10-20051109.

Keller, U., Lara, R., Lausen, H., Polleres, A., and Fensel, D. (2005). Automatic location of services. In *The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings*, pages 1–16.

Klusch, M., Fries, B., and Sycara, K. (2006). Automated semantic web service discovery with owls-mx. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 915–922. ACM.

Klusch, M., Gerber, A., and Schmidt, M. (2005). Semantic web service composition planning with owls-xplan. In *Proceedings of the 1st Int. AAAI Fall Symposium on Agents and the Semantic Web*, pages 55–62. sn.

Kopeckỳ, J., Vitvar, T., Bournez, C., and Farrell, J. (2007). Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, (6):60–67.

Kreger, H. et al. (2001). Web services conceptual architecture (wsca 1.0). *IBM software group*, 5(1):6–7.

Layne, K. and Lee, J. (2001). Developing fully functional e-government: A four stage model. *Government information quarterly*, 18(2):122–136.

Lazovik, A., Aiello, M., and Gennari, R. (2005). Encoding requests to web service compositions as constraints. pages 782–786.

Lécué, F., Delteil, A., and Léger, A. (2008a). Optimizing causal link based web service composition. In *ECAI*, pages 45–49.

Lecue, F. and Mehandjiev, N. (2009). Towards scalability of quality driven semantic web service composition. In *2009 IEEE International Conference on Web Services*, pages 469–476. IEEE.

Lécué, F., Silva, E., and Pires, L. F. (2008b). A framework for dynamic web services composition. In *Emerging Web Services Technology, Volume II*, pages 59–75. Springer.

Leite, L., Moreira, C. E., Cordeiro, D., Gerosa, M. A., and Kon, F. (2014). Deploying large-scale service compositions on the cloud with the choreos enactment engine. In *2014 IEEE 13th international symposium on network computing and applications*, pages 121–128. IEEE.

Li, J., Chen, S., Li, Y., and Zhang, Q. (2011). Semantic web service automatic composition based on service parameter relationship graph. In *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, TrustCom 2011, Changsha, China, 16-18 November, 2011*, pages 1773–1778.

Liao, Z., Teng, Z., Zhang, J., izhi Liu, Xiao, H., and Yi, A. (2019). A semantic concast service for data discovery, aggregation and processing on NDN. *J. Network and Computer Applications*, 125:168–178.

Lin, N., Kuter, U., and Sirin, E. (2008). Web service composition with user preferences. In *European Semantic Web Conference*, pages 629–643. Springer.

Lofstedt, U. (2012a). E-government-assesment of current research and some proposals for future directions. *International journal of public information systems*, 1(1).

Lofstedt, U. (2012b). E-government-assesment of current research and some proposals for future directions. *International journal of public information systems*, 1(1).

Lu, Y. and Xu, X. (2017). A semantic web-based framework for service composition in a cloud manufacturing environment. *Journal of manufacturing systems*, 42:69–81.

Majithia, S., Walker, D. W., and Gray, W. (2004). A framework for automated service composition in service-oriented architectures. In *European Semantic Web Symposium*, pages 269–283. Springer.

Manna, Z. and Waldinger, R. (1983). Deductive synthesis of the unification algorithm. In *Computer Program synthesis methodologies*, pages 251–307. Springer.

Marchese, M. (2003). Service oriented architectures for supporting environments in egovernment applications. In *2003 Symposium on Applications and the Internet Workshops (SAINT 2003), 27-31 January 2003 - Orlando, FL, USA, Proceedings*, pages 106–110.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al. (2004). Owl-s: Semantic markup for web services. *W3C member submission*, 22(4).

McGregor, S. L. and Murnane, J. A. (2010). Paradigm, methodology and method: Intellectual integrity in consumer scholarship. *International journal of consumer studies*, 34(4):419–427.

McIlraith, S. and Son, T. C. (2002). Adapting golog for composition of semantic web services. *KR*, 2:482–493.

McIlraith, S. A., Son, T. C., and Zeng, H. (2001). Semantic web services. *IEEE intelligent systems*, 16(2):46–53.

Medjahed, B. and Atif, Y. (2007). Context-based matching for web service composition. *Distributed and Parallel Databases*, 21(1):5–37.

Medjahed, B. and Bouguettaya, A. (2005). A multilevel composability model for semantic web services. *IEEE transactions on knowledge and data engineering*, 17(7):954–968.

Merrick, P., Allen, S., and Lapp, J. (2006). Xml remote procedure call (xml-rpc). US Patent 7,028,312.

Meshkova, E., Riihijärvi, J., Petrova, M., and Mähönen, P. (2008). A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. *Computer networks*, 52(11):2097–2128.

MINPOSTEL (2020). Stratégie nationale de développement des tic.

Motahari-Nezhad, H., Shekofteh, M., and Kazerani, M. (2018). E-readiness assessment of academic libraries: a case study in iran. *Electron. Libr.*, 36(2):193–207.

Mulugetta, Y., Hagan, E. B., and Kammen, D. (2019). Energy access for sustainable development. *Environmental Research Letters*, 14(2):020201.

Nanda, M. G., Chandra, S., and Sarkar, V. (2004). Decentralizing execution of composite web services. In *ACM Sigplan Notices*, volume 39, pages 170–187. ACM.

Narayanan, S. and McIlraith, S. A. (2002). Simulation, verification and automated composition of web services. In *Proceedings of the 11th international conference on World Wide Web*, pages 77–88. ACM.

Ndou, V. (2004). E-government for developing countries: opportunities and challenges. *The electronic journal of information systems in developing countries*, 18(1):1–24.

Ngeminang, A. (2012). e-government and the cameroon cybersecurity legislation 2010: Opportunities and challenges. *South African Journal of Information and Communication*.

of Science, T. P. O. and Technology (2006). ICT in developing countries. *Postnote*, 261.

Omer, A. M. and Schill, A. (2009). Dependency based automatic service composition using directed graph. In *2009 Fifth International Conference on Next Generation Web Services Practices*, pages 76–81. IEEE.

Omrana, H., Belouadha, F., and Roudiès, O. (2010). A mda approach for describing web services policies. In *Proceedings of the 3rd IEEE International Conference on Web and Information Technologies (ICWIT'10)*, pages 449–460.

Omrana, H., Belouadha, F.-Z., and Roudiès, O. (2012). A composability model for efficient web service's connectivity. In *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems*, pages 483–484. IEEE.

ord Neuman, B. C. (1994). Scale in distributed systems. *ISI/USC*.

Osterwalder, A. (2003). ICT in developing countries. *Lausanne, Switzerland, University of Lausanne*, pages 1–13.

Palvia, S. C. J. and Sharma, S. S. (2007). E-government and e-governance: definitions/domain framework and status around the world. In *International Conference on E-governance*, pages 1–12.

Paolucci, M., Kawamura, T., Payne, T. R., and Sycara, K. (2002). Semantic matching of web services capabilities. In *International semantic web conference*, pages 333–347. Springer.

Papaioannou, I. V., Tsesmetzis, D. T., Roussaki, I. G., and Anagnostou, M. E. (2006). A qos ontology language for web-services. In *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, volume 1, pages 6–pp. IEEE.

Papazoglou, M. P. (2003). Service-oriented computing: Concepts, characteristics and directions. In *4th International Conference on Web Information Systems Engineering, WISE 2003, Rome, Italy, December 10-12, 2003*, pages 3–12.

Patil, A. A., Oundhakar, S. A., Sheth, A. P., and Verma, K. (2004). Meteor-s web service annotation framework. In *Proceedings of the 13th international conference on World Wide Web*, pages 553–562. ACM.

Paulraj, D., Swamynathan, S., Chandran, D., Balasubadra, K., and Prem, M. V. (2016). Service composition and execution plan generation of composite semantic web services using abductive event calculus. *Computational Intelligence*, 32(4):711–737.

Peltz, C. (2003). Web services orchestration and choreography. *Computer*, (10):46–52.

Peña-López, I. et al. (2012). Un e-government survey 2012. e-government for the people.

Peristeras, V., Tarabanis, K., and Goudos, S. K. (2009). Model-driven egovernment interoperability: A review of the state of the art. *Computer Standards & Interfaces*, 31(4):613–628.

Phan, M. and Hattori, F. (2006). Automatic web service composition using congolog. In *26th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'06)*, pages 17–17. IEEE.

Platzer, C., Rosenberg, F., and Dustdar, S. (2009). Web service clustering using multidimensional angles as proximity measures. *ACM Transactions on Internet Technology (TOIT)*, 9(3):11.

Ponnekanti, S. R. and Fox, A. (2002). Sword: A developer toolkit for web service composition. In *Proc. of the Eleventh International World Wide Web Conference, Honolulu, HI*, volume 45.

Pulparambil, S. and Baghdadi, Y. (2019). Service oriented architecture maturity models: A systematic literature review. *Computer Standards & Interfaces*, 61:65–76.

Quinlan, C. (2011). Business research methods. cengage learning emea. *Hampshire*.

Rao, J., Kungas, P., and Matskin, M. (2004). Logic-based web services composition: From service description to process model. In *Proceedings. IEEE International Conference on Web Services, 2004.*, pages 446–453. IEEE.

Rao, J., Küngas, P., and Matskin, M. (2006). Composition of semantic web services using linear logic theorem proving. *Information Systems*, 31(4-5):340–360.

Rao, J. and Su, X. (2005). A survey of automated web service composition methods. *Springer Berlin Heidelberg*, 3387.

Rapti, E., Karageorgos, A., and Gerogiannis, V. C. (2015). Decentralised service composition using potential fields in internet of things applications. *Procedia Computer Science*, 52:700–706.

Raymond, K. (1995). Reference model of open distributed processing (rm-odp): Introduction. In *Open distributed processing*, pages 3–14. Springer.

Relyea, H. C. (2002). E-gov: Introduction and overview. *Government information quarterly*, 1(19):9–35.

Richter, M. M. (1993). Classification and learning of similarity measures. In *Information and Classification*, pages 323–334. Springer.

Rodriguez-Mier, P., Mucientes, M., and Lama, M. (2011). Automatic web service composition with a heuristic-based search algorithm. In *2011 IEEE International Conference on Web Services*, pages 81–88. IEEE.

Rompothong, P. and Senivongse, T. (2003). A query federation of uddi registries. In *Proceedings of the 1st international symposium on Information and communication technologies*, pages 561–566. Trinity College Dublin.

Rostami, N. H., Kheirkhah, E., and Jalali, M. (2013). Web services composition methods and techniques: A review. *International Journal of Computer Science, Engineering and Information Technology*, 3.

Rostami, N. H., Kheirkhah, E., and Jalali, M. (2014). An optimized semantic web service composition method based on clustering and ant colony algorithm. *ACM Transactions on Internet Technology (TOIT)*.

Sabucedo, L. Á. and Anido-Rifón, L. E. (2006). Semantic service oriented architectures for egovernment platforms. In *Semantic Web Meets eGovernment, Papers from the 2006 AAAI Spring Symposium, Technical Report SS-06-06, Stanford, California, USA, March 27-29, 2006*, pages 111–113.

Sadiq, W. and Racca, F. (2003). *Business services orchestration: The hypertier of information technology*. Cambridge University Press.

Saleh, Z. I., Obeidat, R. A., and Khamayseh, Y. (2013). A framework for an e-government based on service oriented architecture for jordan. *International Journal of Information Engineering & Electronic Business*, 5(3).

Saunders, M. N. (2011). *Research methods for business students, 5/e*. Pearson Education India.

Schmidt, C. and Parashar, M. (2004). A peer-to-peer approach to web service discovery. *World Wide Web*, 7(2):211–229.

Scholl, H. (2004). Introduction to the electronic government cluster of minitracks. In *37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the*, pages 114–114. IEEE.

Scotland, J. (2012). Exploring the philosophical underpinnings of research: Relating ontology and epistemology to the methodology and methods of the scientific, interpretive, and critical research paradigms. *English language teaching*, 5(9):9–16.

Sharma, S. K. (2006). E-government services framework. In *Encyclopedia of E-Commerce, E-Government, and Mobile Commerce*, pages 373–378. IGI Global.

Sheng, Q. Z., Qiao, X., Vasilakos, A. V., Szabo, C., Bourne, S., and Xu, X. (2014). Web services composition: A decade's overview. *Information Sciences*, 280:218–238.

Sivashanmugam, K., Verma, K., and Sheth, A. (2004). Discovery of web services in a federated registry environment. In *Web Services, 2004. Proceedings. IEEE International Conference on*, pages 270–278. IEEE.

Song, H., Cheng, D., Messer, A., and Kalasapur, S. (2007). Web service discovery using general-purpose search engines. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 265–271. IEEE.

Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., and Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Communication Review*, 31(4):149–160.

Tanenbaum, A. S. and Van Steen, M. (2007). *Distributed systems: principles and paradigms*. Prentice-Hall.

Thang, H. Q., Thi, Q. P., and Hoang, D. B. (2010). A method of verifying web service composition. In *Proceedings of the 2010 Symposium on Information and Communication Technology, SoICT 2010, Hanoi, Viet Nam, August 27-28, 2010*, pages 155–162.

Tiwari, A. and Mishra, V. K. (2018). Colored petri net based techniques for constructing reliable web service composition. *International Journal*, 6(1):6–8.

Toma, I., Iqbal, K., Moran, M., Roman, D., Strang, T., and Fensel, D. (2005a). An evaluation of discovery approaches in grid and web services environments. In *NODe 2005, GSEM 2005, Erfurt, Germany, September 20-22, 2005 (Net.ObjectDays)*, pages 233–247.

Toma, I., Roman, D., Iqbal, K., Fensel, D., Decker, S., and Hofer, J. (2005b). Towards semanticweb services in grid environments. In *2005 International Conference on Semantics, Knowledge and Grid (SKG 2005), 27-29 November 2005, Beijing, China*, page 107.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. (2008). A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55.

Vedamuthu, A. S., Hirsch, F., Hondo, N. M., and Yalçinalp, Ü. (2007). Web services policy 1.5-primer. *W3C (June 2007)*.

Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S., and Miller, J. (2005). Meteor-s wsdi: A scalable p2p infrastructure of registries for semantic publication and discovery of web services. *Information Technology and Management*, 6(1):17–39.

Wahid, F. (2012). The current state of research on egovernment in developing countries: A literature review. In *International Conference on Electronic Government*, pages 1–12. Springer.

Waldinger, R. J. and Stickel, M. E. (1992). Proving properties of rule-based systems. *International Journal of Software Engineering and Knowledge Engineering*, 2(01):121–144.

Wang, Y. and Stroulia, E. (2003). Flexible interface matching for web-service discovery. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.*, pages 147–156. IEEE.

West, D. M. (2004). E-government and the transformation of service delivery and citizen attitudes. *Public administration review*, 64(1):15–27.

Wieringa, R. and Jonge, W. d. (1995). Object identifiers, keys, and surrogates: object identifiers revisited. *TAPOS*, 1(2):101–114.

working group, E. W. et al. (2004). Web service modeling ontology.

Wu, D., Parsia, B., Sirin, E., Hendler, J., and Nau, D. (2003). Automating daml-s web services composition using shop2. In *International Semantic Web Conference*, pages 195–210. Springer.

Yan, P. and Guo, J. (2010). Researching and designing the architecture of e-government based on soa. In *2010 International Conference on E-Business and E-Government*, pages 512–515. IEEE.

Ying, L. (2010). A method of automatic web services composition based on directed graph. In *2010 International Conference on Communications and Mobile Computing*, volume 1, pages 527–531. IEEE.

Yu, J., Benatallah, B., Casati, F., and Daniel, F. (2008). Understanding mashup development. *IEEE Internet computing*, 12(5):44–52.

Zhang, N., Wang, J., Ma, Y., He, K., Li, Z., and Liu, X. F. (2018). Web service discovery based on goal-oriented query expansion. *Journal of Systems and Software*, 142:73–91.

Zhou, B. and Huang, T. (2008). Semantic web service discovery search with ontology learning. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 5, pages 1048–1051. IEEE.

Zhou, J., Athukorala, K., Gilman, E., Riekki, J., and Ylianttila, M. (2012). Cloud architecture for dynamic service composition. *International Journal of Grid and High Performance Computing (IJGHPC)*, 4(2):17–31.

Zimeo, E., Troisi, A., Papadakis, H., Fragopoulou, P., Forestiero, A., and Mastroianni, C. (2008). Cooperative self-composition and discovery of grid services in p2p networks. *Parallel Processing Letters*, 18(03):329–346.

Zou, G., Chen, Y., Yang, Y., Huang, R., and Xu, Y. (2010). Ai planning and combinatorial optimization for web service composition in cloud computing. In *Proccedding of the International Conference on Cloud Computing and Virtualization*, pages 1–8.