Democratic and Popular Republic of Algeria

Ministry of Higher Education and Scientific Research

LARBI Ben M'HIDI University
-Oum El Bouaghi-

Faculty of Exact Sciences and Sciences of Nature and Life

Department of Mathematics and Computer Science

N° d'ordre **:** .....

Série **:** .....

Dynamic reconfiguration of composite web services

Thesis presented by:

MESSIAID Abdessalam

To obtain the degree of

Doctor in Computer Science

(Option: **Networks and Distributed Systems**)

Presented publically in:____/___/_____

***Before the jury composed of*:**

President	Dr. MARIR Toufik	*MCA*	University of Oum El Bouaghi
Supervisor	Pr. MOKHATI Farid	*Professor*	University of Oum El Bouaghi
Co-Supervisor	Dr. BENABOUD Rohallah	*MCA*	University of Oum El Bouaghi
Examiner	Dr. NESSAH Djamel	*MCA*	University of Khenchela
Examiner	Dr. HEMAM Mounir	*MCA*	University of Khenchela

2021-2022

Acknowledgments

First and foremost, praises and thanks to Allah the almighty.

I would first like to express my deepest appreciation to my supervisor, ***Prof. Mokhati Farid***, for not only his enthusiastic guidance and insightful feedback to sharpen the right direction in the research ideas but also his sincere encouragement when I felt pressure due to the obstacles.

A special appreciation and gratitude to ***Dr. Rohallah Benaboud***, my thesis co-supervisor for his professional engagement, his insightful feedback, and the advice he provided to me throughout this thesis.

I believe that I am one of the luckiest Ph.D. students when I have supervisors like them.

Besides my supervisors, I would like to thank the rest of my thesis committee:

A warm thanks to ***Dr. MARIR Toufik*** at Oum El Bouaghi University, for having done him the honor of being the president of my jury.

Sincere thanks to ***Dr. NESSAH Djamel*** at Khenchela University, for agreeing to be part of my thesis jury.

Sincere thanks to ***Dr. HEMAM Mounir*** at Khenchela University, for agreeing to be part of my thesis jury.

In particular, I am grateful to ***Dr. Hadjer Salem*** at Audensiel R&D Department France, who enriched this work with her knowledge and experience. Also, her modesty, generosity, and personality are such an inspiration.

my appreciation to ***Pr. kAFI Mohammed Redouane*** Head of Center for Computer Systems and Networks and Tele-teaching and Videoconferencing at Ouargla University for lending me a helping hand whenever I need him.

I cannot finish without expressing my appreciation to ***Dr. Karima Boussaha*** at Oum El Bouaghi University and ***Dr. Nour El Houda El Dehimi*** at Oum El Bouaghi University for their help during the period of preparation for my Ph.D.

Dedication

In memory of my mother

To my dear father

To my sisters and brother

To my little family

الملخص

توفر البنية الموجهة نحو الخدمات (SOA) القدرة على دمج العديد من خدمات الويب من أجل تحقيق متطلبات خاصة بالمستخدم. في البيئات الديناميكية ، يمكن أن يؤدي ظهور العديد من الأحداث غير المتوقعة إلى زعزعة استقرار خدمة الويب المركبة والتأثير على جودتها. تعتبر إعادة التكوين الديناميكي لخدمة الويب المركبة ضرورية للتعامل مع مثل هذه التحديات. إعادة التكوين الديناميكي هي القدرة على تغيير بنية خدمة الويب المركبة أو سلوكها في وقت التشغيل دون إيقافها. يأتي البحث المقدم في هذه الأطروحة في سياق توفير إعادة تكوين ديناميكي محسّن لخدمات الويب المركبة. على الرغم من أهميتها ، لم يتم بعد دراسة هذا المجال بشكل كافٍ. في هذه الأطروحة ، نقترح نهجًا لإعادة التكوين الديناميكي لخدمات الويب المركبة بناءً على التنبؤ بالتدهور الوشيك في جودة الخدمة لخدمات الويب المرشحة. تم اقتراح نهج آخر لاستبدال الخدمات الفاشلة مع الحفاظ على خصائص جودة الخدمة الأصلية. يبحث النهج الثاني في صعوبة اختيار خدمة ويب بديلة مثالية ، خاصة إذا كان هناك العديد من الأخطاء في نفس الوقت ، فقد تعترض خدمة إعادة التكوين الديناميكي لعدة عقبات. تُظهر التقييمات التجريبية التي أجريناها على مجموعات بيانات واقعية أن مناهجنا المقترحة تتفوق على العديد من الأساليب الحديثة من حيث الدقة في النهج الأول والكفاءة في النهج الثاني.

الكلمات المفتاحية : إعادة تكوين, خدمات الويب, جودة الخدمة, التنبؤ بالتدهور الوشيك

Résumé

L'architecture orientée services offre la possibilité d'intégrer plusieurs services Web afin de répondre à une exigence spécifique à l'utilisateur. Dans les environnements dynamiques, l'apparition de plusieurs événements imprévus peut déstabiliser le service Web composite et affecter sa qualité. La reconfiguration dynamique du service Web composite est essentielle pour faire face à de tels défis. La reconfiguration dynamique est la possibilité de modifier la structure du service Web composite ou son comportement au moment de l'exécution sans l'arrêter. La recherche présentée dans cette thèse s'inscrit dans le contexte du développement d'une reconfiguration dynamique améliorée aux services Web composites. Malgré son importance, ce domaine n'a pas encore été suffisamment étudié. Dans cette thèse, nous proposons une approche de reconfiguration dynamique de services Web composites basée sur la prédiction de la dégradation imminente de la QoS des services Web candidats. Une autre approche est proposée pour remplacer les services défaillants et maintenir les contraintes de bout en bout d'origine. La deuxième approche étudie la difficulté de sélectionner un service web de substitution optimal, notamment s'il y a plusieurs erreurs en même temps, le service de reconfiguration dynamique peut présenter plusieurs obstacles. Les évaluations expérimentales que nous avons menées sur des ensembles de données réalistes montrent que nos approches proposées surpassent plusieurs méthodes de pointe en termes de précision dans la première approche et d'efficacité dans la seconde approche.

Mots-clés: service web, reconfiguration dynamique, prédiction QoS, apprentissage par renforcement.

Abstract

Service-oriented architecture provides the ability to incorporate several web services in order to achieve a user-specific requirement. In dynamic environments, the appearance of multiple unforeseen events can destabilize the composite web service and affect its quality. The dynamic reconfiguration of the composite web service is essential to dealing with such challenges. Dynamic reconfiguration is the capacity to modify the composite web service structure or behavior while it is operating. The research presented in this thesis takes place in the context of providing enhanced dynamic reconfiguration to composite web services. Despite its importance, this area has not yet been adequately studied. In this thesis, we propose an approach for dynamic reconfiguration of composite web services based on predicting the imminent degradation in QoS of the candidate or partner web services. Another approach is proposed to replace failed services and maintain the original end-to-end constraints. The second approach investigates the difficulty of selecting an optimal substitute web service, especially if there are many errors at the same time, the dynamic reconfiguration service may present several obstacles. The experimental evaluations we conducted on realistic datasets show that our proposed approaches outperform several state-of-the-art methods in terms of accuracy in the first approach and efficiency in the second approach.

Keywords: web service, dynamic reconfiguration, QoS prediction, reinforcement learning.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

SOA Service-oriented architecture

QoS Quality of Service

SOAP Simple Object Access Protocol

WSDL Web Services Description Language

UDDI Universal Description Discovery and Integration

REST Representational state transfer

CWS Composite Web Service

SaaS Software as a Service

HMM Hidden Markov Model

ACO Ant Colony Optimization

PSO Particle Swarm Optimization

SFLA Shuffled Frog Leaping Algorithm

SRL Swarm Reinforcement Learning

General introduction

General Context and Problem Statement

Service-Oriented Architecture (SOA) enables the construction of distributed systems with Web services that can be published, invoked, and composed dynamically at runtime. With the emergence of cloud and Software as a Service (SaaS), many companies and organizations have relied on web services for their needs and business [1]. Responding to a user request often requires composing multiple web services, resulting in a composite web service.

The composition process enables integrating and interconnecting web services to form a composite web service, which seeks to meet not only functional needs but also non-functional requirements, such as QoS restrictions. The context in which service composition is performed is characterized by dynamism and evolving, which makes web services involved in the composition vulnerable to disrupting or affecting the quality of the services that characterize it; therefore, providing reliable and efficient services is a big challenge. In the literature, several research works have been carried out on the composition (static or dynamic) of Web services. However, several other aspects relating to this research area are still at the preliminary stage and have not yet reached a high level of maturity. One of the most interesting challenges in this research domain is the dynamic reconfiguration of composite Web services.

The dynamic reconfiguration of web services is a crucial task in order to ensure consistency and robustness to distributed systems based on web services. According to [2, 3], a dynamic reconfiguration is the ability of the software system to change the sub-part of the system while it is running without interrupting the service. There are several motivations why dynamic reconfiguration should be supported in service-oriented applications. Some of these motivations are: adaptability, high availability, scalability, maintenance and performance.

With the emergence of dynamic reconfiguration it becomes possible to replace the failed services, modify them or delete them dynamically at runtime, which finding replacements for failed services, allows the composite web service to continue running without completely reconfiguring the composition. However, the increasing number of candidate web services with the same functionality but the different quality of service may present the dynamic reconfiguration process to several obstacles, including the difficulty of selecting an optimal candidate web service, especially in the case of multiple failures at the same time. Several authors studied the selection of candidate services in the dynamic reconfiguration [1, 4, 5, 6, 7], but this research remains insufficient and lacks a guarantee of high efficiency in the light of the increasing number of candidate services [8].

Another important research gap relates to when the reconfiguration action is triggered. Much research supports reconfiguration in either a proactive or reactive manner [9]. In the reactive approach the reconfiguration action triggered after an event which causes the need for the reconfiguration. Contrary to the reactive approach, the proactive approach is capable to predict the necessity of the reconfiguration earlier than the problem happens. In the literature, researchers have proposed several methods to improve the efficiency of the reactive approaches to dynamic reconfiguration in composite web services. However, a reactive method for reconfiguration and service continuity takes a long time [10] and affects the performance [11], from selecting the best service to reconfiguring the entire process. A proactive or predictive method for impending problems is one of the most critical suggested solutions in the dynamic reconfiguration problem.

Contributions

The first contribution of this research is to present a comparative study among several dynamic reconfiguration of composite web services approaches by examining the characteristics of these approaches and classifying them with respect to these characteristics. This study has been shown in [12].

The second contribution proposes a swarm reinforcement learning approach to replace multiple failed services and maintain the original end-to-end constraints. This approach combines multi-agent and reinforcement learning technologies to dynamic reconfiguration of composite web services in the presence of failures. ACO (Ant Colony Optimization) is used to improve the exchange of information between agents based on the Pheromone-Q values, inspired by real ants' behavior.

The third contribution of this research is to propose a dynamic reconfiguration method based on HMM (Hidden Markov Model) to predict the imminent degradation in QoS and prevent the invocation of partner web services with degraded QoS values. PSO (Particle Swarm Optimization) and SFLA (Shuffled Frog Leaping Algorithm) are used to improve the prediction efficiency of HMM. The QoS prediction problem is reformulated into a QoS state prediction problem, where we consider QoS states rather than QoS values and consider that a QoS state is composed of several attributes. The proposed approach can predict the QoS according to each user's preferences and can predict the state of similar services without the need for their data (QoS). This study has been shown in [13].

Thesis Plan

This thesis is organized into four chapters. The first chapter introduces the context in which this thesis is placed. The second chapter shows a state of the art of our research problem. The third chapter presents the proposed approach for dynamic reconfiguration of composite web services. The fourth chapter presents a proactive approach for dynamic reconfiguration of composite web services.

- **Chapter 1:** In this chapter, we will introduce the principles of web services technology by defining the SOA paradigm, as well as the protocols, languages and models that relate to the concept of web services. Afterward, we will present the problem of dynamic reconfiguration of web services as well as all the related concepts.

- **Chapter 2:** In this chapter, we will look at some of the approaches that have been suggested in the literature. Then, we will compare and categorize these approaches based on a set of criteria to describe the distinctions between them.

- **Chapter 3:** In this chapter, we will present an approach to dynamic reconfiguration of composite web services, by combining multi-agent and reinforcement learning technologies. We will use ACO (Ant Colony Optimization) to improve the exchange of information between agents based on the Pheromone-Q values, inspired by real ants' behaviour. The results of the proposed approach will be presented, discussed and compared.

- **Chapter 4:** In this chapter, we will present a proactive approach to dynamic reconfiguration of composite web services, by using Hidden Markov Model to predict the imminent degradation in QoS and prevent the invocation of partner web services with degraded QoS values. We will use PSO (Particle Swarm Optimization) and SFLA (Shuffled Frog Leaping Algorithm) to enhance the prediction efficiency of HMM. Finally, the results of the proposed approach will be presented, discussed and compared.

Chapter 1

Service-oriented Architecture SOA

1.1 Introduction

The role of Service Oriented Architecture (SOA) is becoming more prominent as economic competition between companies and research institutions developing software increases. As a result of this competitiveness, increasingly complicated software development activities have emerged, which are frequently prone to failure and delay. SOA addresses these shortcomings by offering efficient modeling that can be quickly changed to changing market demands.
In this chapter, we will introduce the essential concepts of SOA, starting with the definition of SOA, their actors and their characteristics. Next, we will move on to the definition of web service, their standards and quality of service attributes, and then we will discuss web service composition, including its definition, its life cycle of a web service composition, orchestration, service choreography and static and dynamic composition. Finally, we highlight dynamic reconfiguration of composite web services by defining dynamic reconfiguration, presenting its objectives, classification and properties.

1.2 Basics of service-oriented Architectures

Service Oriented Architecture (SOA) is a paradigm in which software is designed and built into elements that provide well-defined services on demand. According to Oasis Group's SOA, it is defined as: "a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains. It provides a uniform means to offer, discover, interact with and use capabilities to produce desired effects consistent with measurable preconditions and expectations". Another definition from [14] considers "SOA as component-based software modules that provide services to other modules". According to these definitions, a service-oriented architecture is based on the principle of considering the basic components of building software applications as services. These services have the following properties [15, 16]:

- a uniform mechanism for data representation and exchange.
- a standard means of communication.
- an interface for the service contract.
- easy to locate and invoke dynamically.
- a standard meta language for describing service offerings.

1.2.1 Characteristics of SOA

In [16, 17] the service-oriented architecture paradigm has principles that must be respected when generating an SOA service:
Loose coupling: helps services meet scalability, flexibility, and fault tolerance requirements by minimizing dependencies and the impact of changes between them and client services.
Interoperability: refers to the ability of services to run on different platforms and to collaborate with each other regardless of their technical specifications.
Reusability: refers to a service's ability to participate in various service compositions.
Discoverability: involves that services be released in a form that allows them to be easily found for subsequent usage.
Autonomy: allows the service to execute the self-management of all its treatments on the logic that it encapsulates.
Stateless: allows the services to change its status after the processing of the application.
Composability: requiring that services be designed so that they can participate as members of other service compositions if necessary.

1.2.2 The actors of SOA

The service-oriented architecture is based on three basic communication primitives represented by publication, discovery and invocation, which define a model of interaction

between three types of actors: service provider, service client, and service registry [18]. From the service provider's point of view, services are created and deployed (publication) in a directory (service register). From the client's point of view, the client sends requests to the service register to discover a service that meets their needs (discovery). Once the service has been chosen, the client invokes the provider to request the use of the service (invocation). Figure 1.1 below illustrates the interaction between these actors.

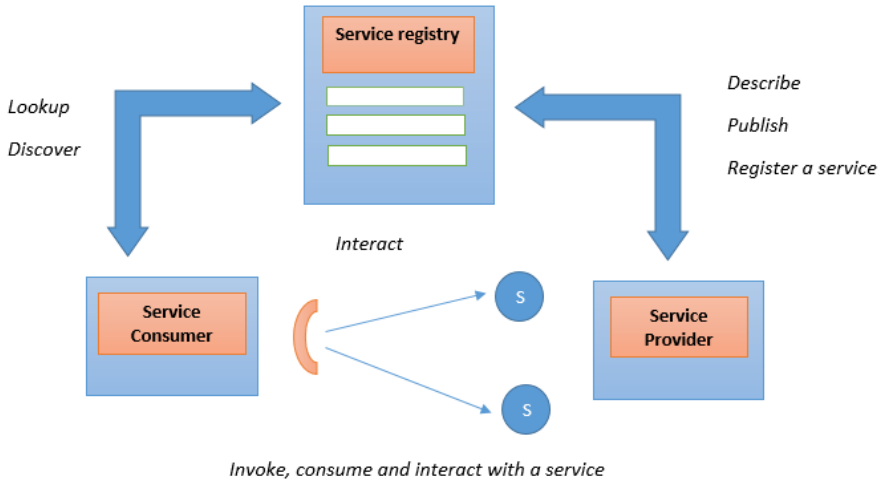


Figure 1.1: SOA function [19]

1.2.3 Architectural stack of SOA

According to [18], the SOA architectural stack consists of two parts: a functional part which includes the transport layer, the service communication protocol, service description, the service layer, business process and service registry, and a part related to quality of service which includes policy, security, transaction and management. Figure 1.2 shows the components of both parts. These components are detailed as follows::

1. **The functional part :** this part includes:

- **The Transport layer:** is the mechanism that sends service requests from the consumer to the provider, as well as answers from the provider to the consumer.
- **The Service Communication Protocol:** it is defined as a means agreed upon by the service provider and the service consumer to indicate what is required and what will be received.
- **The Service Description layer:** is an agreed-upon structure for specifying what the service is, how it is called, and what needs are required to contact the service successfully.
- **The Service layer:** specifies an adequate service that is made accessible for usage.
- **The Business Process layer:** is a set of services, described in a special order with an appropriate set of rules to satisfy particular requirements. It should be noted that a business process may be seen as a single service, which leads us to believe that business processes can be composed of services of different granularity.
- **Service Registry:** is a repository of service descriptions and data that can be used by service providers to edit their services, and by service consumers to discover or find available services.

2. **Service quality aspects:** this part includes:

- **A Policy:** a collection of regulations that govern how a service provider makes a service available to customers. Policies might be functional or non-functional properties.
- **Security:** a collection of rules that may be used to identify, authorize, and control service customers' access.
- **Transaction:** a set of properties that may be applied to a collection of services to get a consistent outcome. For example, if a collection of three services is to be utilized to accomplish a given function, all of them must run.

- **Management:** a collection of characteristics that may be used to manage the services provided or acquired.

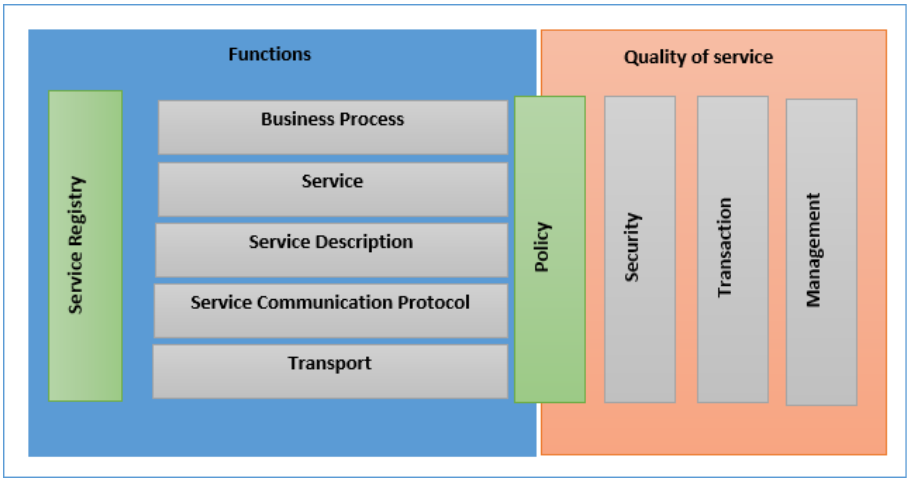


Figure 1.2: The different elements of a service oriented architecture

1.2.4 Roles in service-oriented architecture

The different roles in a service-oriented architecture are the following:

1. The service consumer is an application, software module or other service that requires a service. It posts a request to the service directory, communicates with the service through the transport layer, and then performs the service function. The service consumer executes the service according to the predefined interface contract.

2. The service provider is a network-accessible entity that accepts and executes requests from consumers. It publishes its services and interface contracts to the service directory so that the service consumer can discover and access the service.

3. A service directory is an essential element for service discovery. It contains a directory of available services and allows consultation of service interfaces by interested consumers.

Each component in a service-oriented architecture may perform one (or more) of the three responsibilities outlined above.
The service-oriented architecture has the possible operations described as follows:

- Edit/Publish: for a service description to be accessible, it must be announced so that it may be discovered and invoked by a service consumer.
- Discovery: a service requester locates a service via a query to the service directory based on the criteria of the desired service.
- Binding/Invocation: after searching for the service description, the consumer invokes the service based on the information in the service description.

The key objects in a service-oriented architecture are :

- Services: when a service is made accessible by the publishing of its interface definition, a service consumer can invoke it.
- Service description: a service description describes is how the service consumer communicates with the service provider. It defines the format of both the service request and the answer. This description can indicate a set of pre-conditions, post-conditions, and/or quality of service (QoS).

1.3 Web services

1.3.1 Definition and concepts of Web service

According to [20], 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). 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." It can be said that web services technology is the most promising choice to meet the objectives of service-oriented architecture. Web services can be described, published, localized or invoked over the network to perform certain tasks using standardized XML. Today, two types of Web services are widely used: SOAP-based Web services and RESTful Web services [21]. The former is based on an interface described in an easily exploitable description called WSDL (Web Services Description Language), a distributed directory of services allowing their publication and discovery called UDDI (Universal Description Discovery and Integration), and an information exchange protocol called SOAP (Simple Object Access Protocol), while the latter conforms to REST architectural principles [22].

1.3.2 Architecture of Web services

Web services have the ability to work together using common standards by fulfilling the requirement of interoperability. In this section we highlight two types of architectures. The first is the so-called reference architecture, traditionally used for isolated web services. The second architecture is more complete and is frequently used when creating Web services. This is called extended or stack architecture.

1. **Reference architecture**

   According to [23], the reference architecture has three important objectives: the first, is identification of the functional components, the second is the definition of the relations between these components and the third is the establishment of a set of constraints on each component in order to guarantee the properties of the overall architecture. The reference architecture for web services (Figure 1.3) is based on the following three roles:

   - **The service provider:** corresponds to the owner of the service. It is made up of the service reception platform.

- **The client:** corresponds to the service requester. It is made up of the application that will search for and invoke a service.
- **The Service registry:** is a registry of service descriptions providing facilities for posting services to providers as well as facilities for finding services for customers.

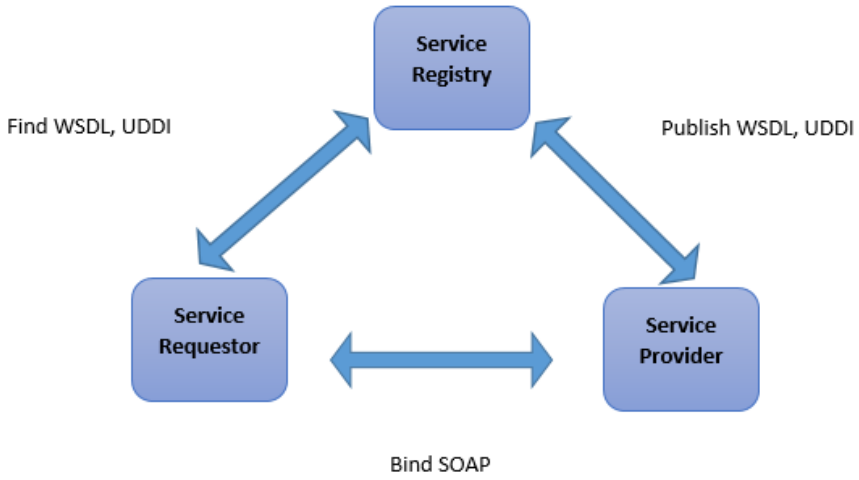


Figure 1.3: The Web Services Model [23]

2. **Extended architecture**

The extended architecture consists of several layers overlapping each other. Figure 1.4 describes an example of such an architecture.
We provide an explanation of the extended architecture layers below.

(a) **Basic infrastructure (Discovery, Description, Exchange):** these are the technical foundations established by the reference architecture. We distinguish between exchanges of messages established by SOAP, the service description by

WSDL and the search for web services that organizations want to use through the UDDI registry;

(b) **Transversal layers (Security, Transactions, Administration, QoS):** These are the layers that make the effective use of web services viable in the industrial world;

(c) **The Business Process layer (Business Process):** This upper layer allows the integration of Web services, it establishes the representation of a "Business Process" as a set of Web services. In addition, the description of the use of different services that make up this service is available through this layer.

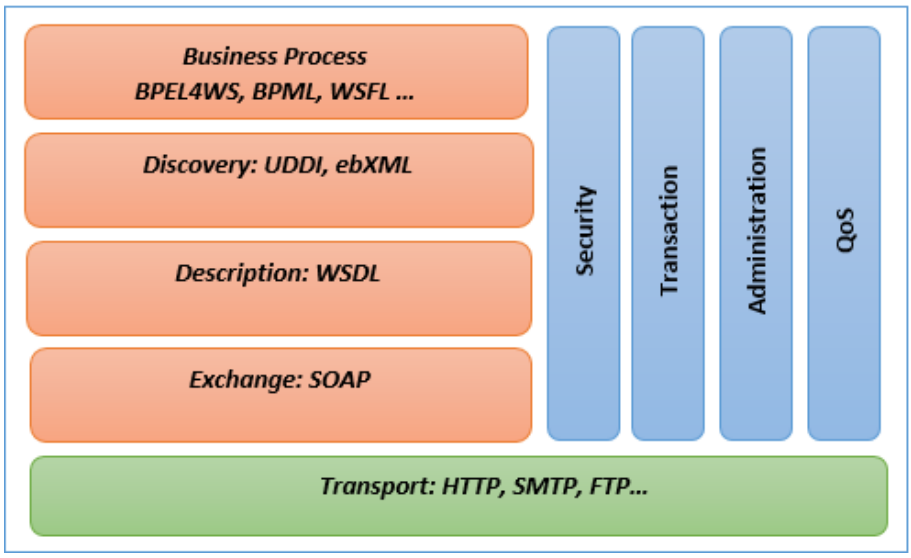


Figure 1.4: Web services stack Architecture [23]

1.3.3 SOAP Web service

SOAP web services are software entities that are interoperable with other distributed systems using a protocol such as SOAP, usually transmitted via the HTTP protocol with XML

serialization [20]. What follows is a presentation of the use of web services technologies by provider and requester actors illustrated in Figure 1.5 [18] :

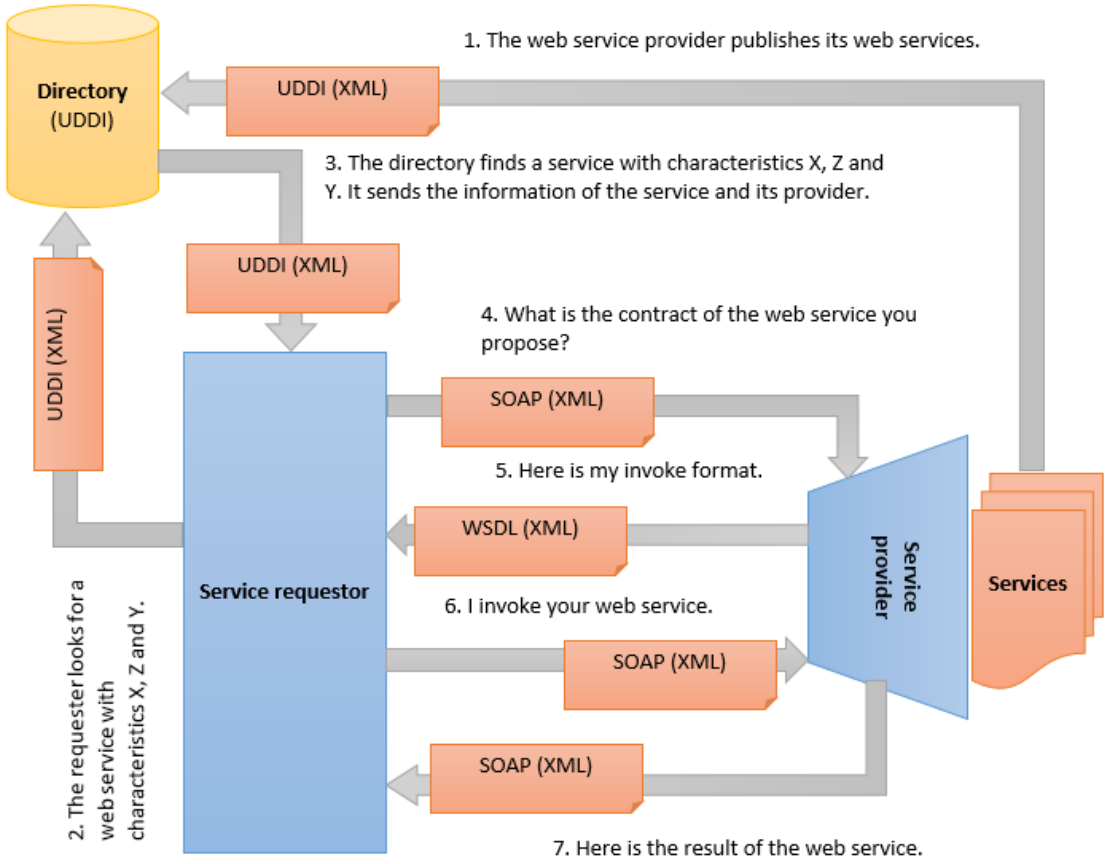


Figure 1.5: The use of web services technologies by the provider and requester actors.[18]

The mechanism of execution of web services based on three standards such as UDDI, WSDL and SOAP.

1. **SOAP (Simple Object Access Protocol)**

   SOAP is an XML-based messaging protocol for exchanging information over the Internet [24]. It is a platform and language independent protocol. SOAP supports remote procedure invokes transported over HTTP. A SOAP message is an XML document consisting of [14] (see Figure 1.6):

   - **SOAP Envelope** that identifies the XML document as a SOAP message.
   - **SOAP Header** which contains header information.
   - **SOAP Body** It allows the transmission of requests and responses between the systems. It is composed of one or more sub-elements, which are: Fault element and Message element. The former is used to indicate the transmission failures of SOAP messages; the second contains the data to be transmitted via the SOAP protocol.

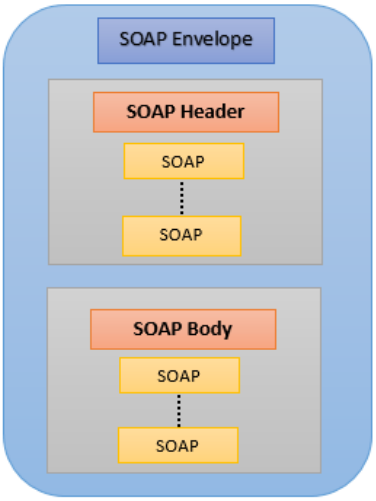


Figure 1.6: The SOAP Envelope.[14]

2. **UDDI (Universal Description, Discovery and Integration)**

The UDDI protocol is one of the critical building blocks necessary for effective Web services. UDDI develops a standard interoperable platform that enables organizations and applications to locate and use Web services over the Internet in a rapid, easy, and dynamic manner [9]. In a UDDI registry, each publication is an XML document that consists of the following main elements Business Entity (about the provider), Business Service (about the service), Binding Template (about the access to the service) and tModel (type of service), the following Figure 1.7 shows the main elements of a publication in UDDI.

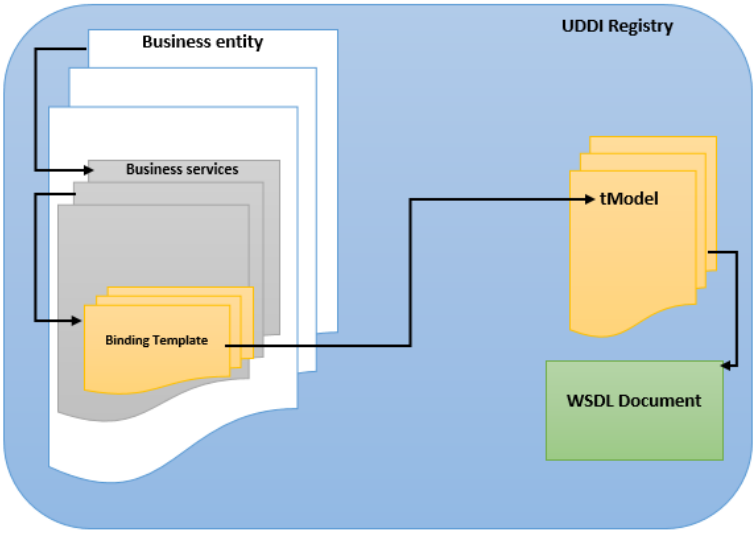


Figure 1.7: Entities composing a UDDI directory.[18]

To facilitate the operation of searching for information such as the providers of a given service or the means of invoking it, UDDI organizes all the information it contains into three pages, specified in XML. These pages are the following.

- **White Paper:** they provide the customer with all the information about the supplier that he has entered in the Business Entity element (such as company name, physical address, company description, etc).

- **Yellow Paper:** This is a non-technical description of the services by the providers that describes the category of the company, the industry in which the company operates, the type of services and the quality of service.
- **Green Paper:** Based on the WSDL description of the web services, precise technical information about the services provided is included with an indication of how to access them.

WSDL (Web Services Description Language)

According to [25] "Web Services Description Language (WSDL) describes a web service in XML format. WSDL represents a contract between the service provider and the user of the service", in other words, a WSDL document describes the interface of a web service and provides users with a point of contact. A complete WSDL service document provides two-level description: **abstract description** and **concrete description**.

3. (a) **The abstract level:** The abstract level describes operations, data types and messages in detail. In other words, the abstract level answers the following questions: What operations are available? What are the type definitions of the input/output messages? And with which input/output messages do the operations work?
    - **Data types:** is the element that defines the types of data used in the messages exchanged by the web service.
    - **Message:** specifies the types of operations supported by the web service. It allows you to incorporate a sequence of correlated messages without having to specify the characteristics of the data flow.
    - **Port Type:** is a logical grouping or collection of operations supported by one or more transport protocols. It is analogous to a definition of an object containing a set of methods.

   (b) **the Concrete level:** describes how a client invokes the service? What protocol

to use? and Where is the service? It has two main components: Binding Information about the protocol used and the location address of the service.

- **Bindings:** Describes how a type of port is implemented for a particular protocol (eg HTTP), and an invocation mode (e.g. SOAP). This description is made by a given set of abstract operations.
- **Port:** Specifies a URL address that corresponds to the implementation of the web service by a provider and identifies one or more transport protocol bindings for a given Port Type.
- **Service:** Specifies the full address of the web service, and allows a remote application access point to choose to expose multiple categories of operations for various types of interactions.

The following figure 1.8 illustrates the WSDL document structure.

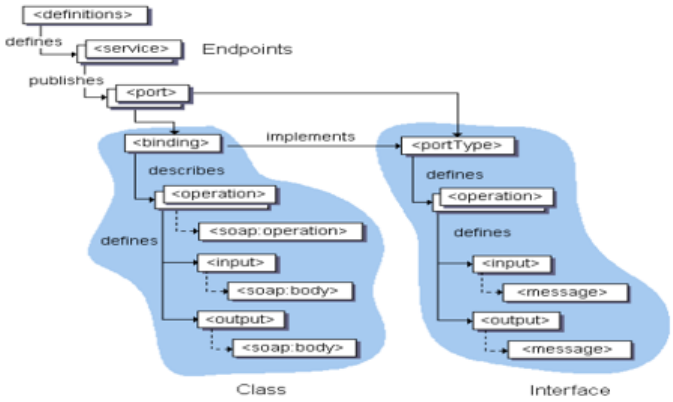


Figure 1.8: The WSDL document structure [26]

1.3.4 REST Web service

REST is an abbreviation for Representational State Transfer. It was proposed by [22] in his thesis as an abstraction of the architectural elements within a distributed hypermedia system. REST is an architectural style based on a set of restrictions that, when applied to architectural components, allow for the optimization of certain criteria particular to the requirements of the system to be developed. RESTful application are encouraged to be simple, lightweight, and quick due to the following principles [22]:

- **Resource Identifier:** A resource is identified by a URI (Uniform Resource Identifier), which provides a global address space for the discovery of resources and services.
- **Resource representation:** The resource can be represented in different formats, but the most popular ones are XML and JSON.
- **Operation on the resource:** The resources are manipulated by transferring the representations through a uniform interface addressed by the resource identifier.

1.4 Web services composition

1.4.1 Definition

With the increasing demand for more complex web services, it has become important for the designer to combine the functionality of a group of services. This process is named web service composition [27]. In the literature, several definitions of service composition have been defined. For example, in [28], web service composition is the process of combining many services into a single service in order to execute more complicated operations. The services invoked during the service composition are named component services. From a set of services available in a registry, we can build a composite service [29]. Figure 1.9 shows the principle of service composition.

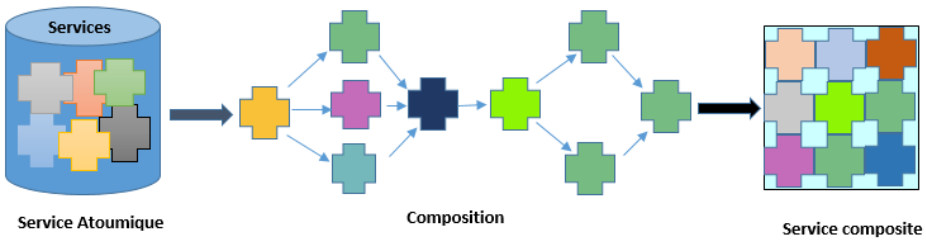


Figure 1.9: The principle of service composition.

1.4.2 Life cycle of a web services composition

Many authors such as [27] defined the life cycle of a web service composition as six activities: wrapping services, setting outsourcing agreements, assembling composite services, executing services, monitoring services, and evolving services.These are shown in figure 1.10

1. **Wrapping services:** This task guarantees that any service may be accessed throughout the composition (regardless of its data model, its interaction protocol, etc.).

2. **Setting outsourcing agreements:** This task entails negotiating and establishing contractual responsibilities among the partner services.

3. **Assembling composite services:** This task allows an abstract description of the services to be constructed. This composition starts with the identification of the services to be used and ends with the specification of inter-service interactions based on the shared protocol.

4. **Executing services:** This task allows you to carry out composition specifications while adhering to certain practical constraints (availability, reliability, etc).

5. **Monitoring services:** This step provides for the verification of a variety of features and attributes, such as controlling access to services, monitoring status changes, and

message exchanges. This enables the identification of errors (contractual violations), the measurement of service performance, and the prediction of exceptions.

6. **Evolving services:** This phase guarantees that the composition evolves (i.e. the modification of the services invoked, use of new services, consideration of feedback from the control activity).

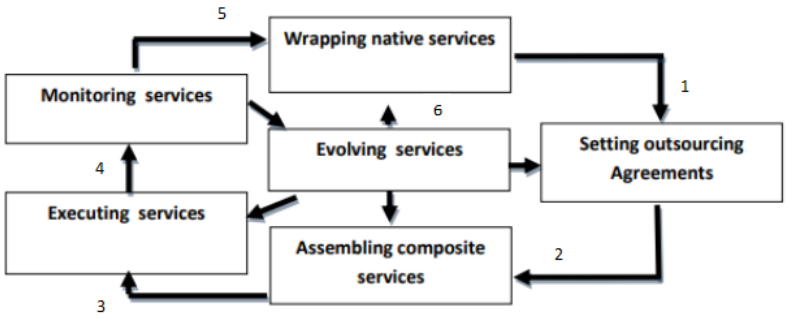


Figure 1.10: Life cycle of a service composition [27]

1.4.3 Service Orchestration and Service Choreography

Web service composition may be characterized in two ways: orchestration and choreography.

- **Orchestration**

  According to [30] the orchestration of Web services enables the order and sequencing of these services to be established in accordance with a well-defined framework. It defines how the services can interact with one another, as well as the order in which the various interactions are carried out.The principle of orchestration is illustrated in figure 1.11.

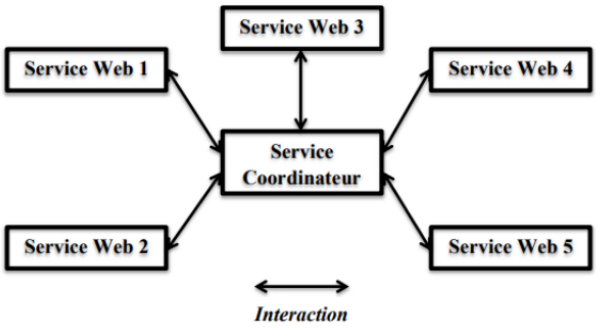


Figure 1.11: Principle of web services orchestration. [30]

- **Choreography**

  In [30], the author mentioned, that choreography enables the tracking of the sequence of messages exchanged in the context of web service composition. It is typically related to the description of existing conversations between multiple parties, including customers, providers and partners.The principle of choreography is illustrated in Figure 1.12

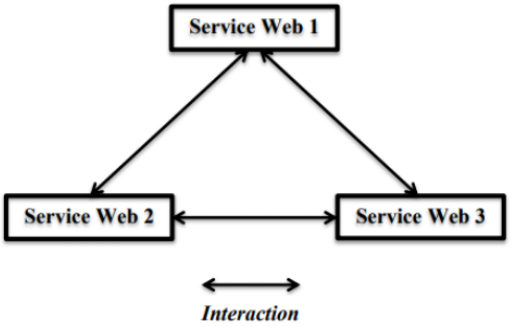


Figure 1.12: principle of web services choreography [30]

1.4.4 Static vs Dynamic Composition

Static composition occurs when an application is designed before deployment. The components in question are selected, connected, and assembled [31]. This type of composition is appropriate for confined situations where the components do not change frequently. Static composition groups the orchestration methods and the choreography methods for the web service composition mentioned above. Unlike static composition, dynamic composition allows users to discover, select and dynamically combine Web services due to a user request. dynamic composition responds to the needs of flexibility and adaptation, besides having several advantages compared to static composition.

1.5 Dynamic Reconfiguration

1.5.1 Definition of Dynamic Reconfiguration

The malfunction of a software component or the appearance of different functionalities of this component may be the major reason for the dynamic reconfiguration of a software application. There are a variety of definition suggestions for dynamic reconfiguration. [32] defined dynamic reconfiguration as "changing part of an application while it operates. Dynamic reconfiguration are expressed only at the configuration level, i.e. as changes to the program's connectivity structure.". In the case of web services, we have selected the one that we believed would be most suitable: it represents the ability of the software system to change the sub-part of the system while it is running without interrupting the service [2, 3].

1.5.2 Objectives of dynamic reconfiguration

The task of dynamic reconfiguration relies on a set of objectives, among which we mention the following [33].

1. **Correctional reconfiguration:** Some applications may experience a change in their behavior during runtime, exposing them to performance defects that require corrective reconfiguration. The corrective reconfiguration starts by identifying the application components responsible for the faulty behavior. After that, it replaces these components with new ones that supposed work more adequately. The new components must provide the same functionality as the old components.

2. **Evolutionary reconfiguration:** clients change their needs over time, requiring additions and modifications to the services provided to meet these changing needs. Evolutionary reconfiguration responds to these needs: either by adding new components to provide the new required functionality or by replacing existing components with others offering more functionality. In this case, it must be guaranteed that the replacement does not compromise the integrity of the application.

3. **Adaptative reconfiguration:** An application is generally intended to be deployed and executed in a particular context. This context, which can be reflected in the environment in which the application should run, can change over time. Certain changes in the context can make the behavior of the application inconsistent. For this reason, the application must be reconfigured to take into account the new parameters of its execution context.

4. **Perfective reconfiguration:** The goal of this type of reconfiguration is to improve the performance of an application, even if its behavior is consistent.

1.5.3 Classification of dynamic reconfiguration

According to [34] and [35] dynamic reconfiguration can be divided into two categories: **1) Programmed reconfiguration, 2) Unpredictable reconfiguration.**

- **Programmed reconfiguration** Programmed reconfiguration are those that have been foreseen by the developer and that have therefore been taken into account during

the design of the application. When specific criteria are satisfied, the application can switch from one configuration to another at runtime without should to change any of its components. However, this approach also has some limitations. First of all, it is ad-hoc, i.e. it is linked to a specific application. Second, the number of configurations or modes is already fixed and this decreases the power of adaptability when unpredictable events appear. Finally, adding new modes or configurations requires redesigning the entire system and recompiling it.

- **Unpredictable reconfiguration** Unpredictable reconfiguration are those changes that were unforeseen or unanticipated at design time but become necessary during the life of the application. The reconfiguration manager must examine the modification request and ensure that it does not contradict system coherence or system invariant. A set of invariant in this category of reconfiguration provides a basis for preserving system coherence.

1.5.4 Dynamic reconfiguration properties

In [36] thesis, Besseron identified several properties of the reconfiguration process. We mention the following

- **The Consistency** means that a reconfiguration must leave the application in a correct state.
- **The Generalization** indicates that the reconfiguration process must be able to support all types of reconfiguration on all types of components.
- **The Scalability** Indicates that the reconfiguration should be applicable to the whole application or only to a small part of it. If only a part of the application is affected by the reconfiguration, the rest of the application must be able to continue working during the reconfiguration.

- **The Efficiency** means that the reconfiguration time should be as small as possible in order to reduce application interruption.

1.5.5 Dynamic Reconfiguration of web services

The dynamic reconfiguration of web services composed in the life of its execution, is a crucial task in order to ensure consistency and robustness to distributed systems based on web services. According to [33] a service is generally represented by the ISC specifications (interfaces, scenario and its constraints) see figure 1.13.

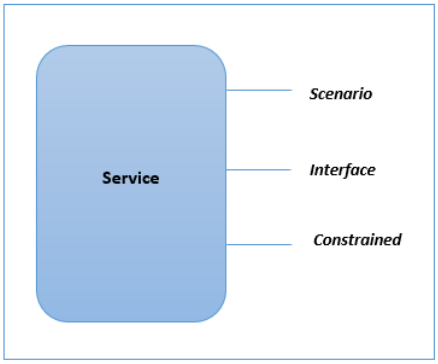


Figure 1.13: graphical representation of a web service [33].

This representation offers the possibility of having a service composed of several sub-services as shown in figure 1.14, in which case the composed (resulting) service will have its own ISC specifications. Thus this representation offers flexibility for dynamic reconfiguration operations such as adding new services, removing existing services (temporarily or permanently) and connecting and disconnecting services and connectors.

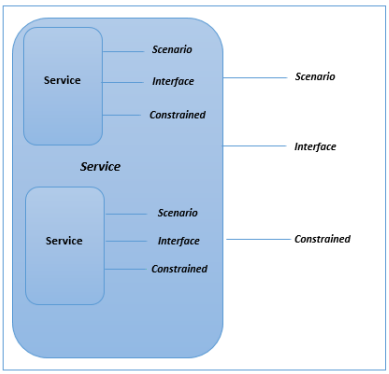


Figure 1.14: graphical representation of a web service composed [33].

1.6 Conclusion

This chapter has been devoted to the presentation of some key concepts constituting the scientific context in which this thesis took place. We first introduced service-oriented architecture, through the presentation of the basics of service-oriented architecture, a brief overview of web services and the composition of web services. We then presented dynamic reconfiguration, with particular emphasis on the definition and objectives of dynamic reconfiguration as well as the classification of dynamic reconfiguration and the properties that a reconfiguration process must have. Finally, we illustrated the notion of dynamic reconfiguration on web services by presenting the ISC specifications (interfaces, scenario and its constraints). In the next chapter, we present some works that deal with the dynamic reconfiguration of composite web services by introducing the aspect of dynamic reconfiguration of composite web services and the classification of the approaches of dynamic reconfiguration of composite web services.

Chapter 2

Survey on dynamic reconfiguration of composite web services

2.1 Introduction

Web services, as software applications exposed on the Web, are invoked by users to meet their requirements. Responding to a user request often requires composing multiple web services, resulting in a composite service. In the literature, several research works have been carried out on the composition (static or dynamic) of Web services. However, several other aspects relating to this research area are at the preliminary stage and have not yet reached a high level of maturity. One of the most interesting challenges in this research domain is the dynamic reconfiguration of composite Web services. In fact, invoking a pre-composed web service is susceptible to various unforeseen changes that can occur in the user's preferences and the environment in which the application operates. Therefore, a dynamic reconfiguration of the composite web service is essential so that it adapts to the user's invocation. During the last decade, many research works [5, 6, 37, 38, 39, 40, 41, 42, 43] have been done in this field. These works have brought very important elements of answers to some problems relating to dynamic reconfiguration. In this chapter, we present a survey of the most important works, which deal with the dynamic reconfiguration of web service composition. We also describe each work and present its strengths and weaknesses through a comparative study.

2.2 Aspects of dynamic reconfiguration of composite web services

The importance of dynamic reconfiguration of service-oriented applications has been imposed by various aspects: adaptability, high availability, scalability, maintenance and performance, all focused on the quality of service [37, 44] and user preferences.

- **An adaptive system** is a system that can adapt their behavior to unpredictable environmental changes. For example, mobile and ubiquitous systems also require dynamic adaptation under fluctuating resource availability conditions in the form of system updates when they are active [38].

- **High-availability systems** are those that must ensure operational continuity for long periods and must adapt to changes during these periods without the declining quality of service. The motivation for high availability could be economical, as with e-commerce and banking systems or security as an air defense system [34].

- **A scalable system** is a system that can acquire new functions over its lifetime to adjust to unforeseen changes. Dynamic reconfiguration is highly useful for providing scalability functionality to these systems [45].

- **Maintenance** is the correction of problems that can occur during the life of a software system, such as programming errors, bugs, failures, software failures, and security vulnerabilities. Dynamic reconfiguration may resolve the maintenance problem by permitting corrective evolution [34].

- **Performance** refers to how well a system behavior is done. Making systems more efficient and faster can be seen as a priority today. Optimizing performance in dynamical environments necessitates structural modifications such as replacing one component with another or adding another one. So, dynamic reconfiguration allows enabling these systems to make the desired structural changes [34].

2.3 Synthesis on dynamic reconfiguration of composite web services

In recent years, many researchers have focused on dynamic reconfiguration of web service composition. In this study, we will investigate some recent research in this field and conduct a comparative study about them.

(C. Mohamed et al, 2014) [37] proposed an approach to select the most desirable configuration from a group of substitutes in order to optimize end-user satisfaction. The suggested approach is based on using a list of components' non-functional characteristics. The approach begins by proposing a model using Acme (an Architecture Description

Language (ADL)) to represent the non-functional characteristics of the components. The authors assume that they have an autonomous reconfigurable component-based system that includes four layers (figure 2.1): A component pool, a reconfiguration management unit, an analysis and decision-making unit, a monitoring unit and an end-user requirement. The authors offer five steps for selecting the best configuration from a variety of choices. First, generating the potential configurations, when the user asks the system for a defined functionality, the analysis and decision-making layer analyzes the user requirements and produces all the potential configurations that provide the necessary functionality. Second, in the filtering step, check each potential configuration to guarantee that the system's consistency is always saved. Third, the normalization step is to standardize the non-functional characteristics values. Fourth, evaluate the alternative step, evaluate the candidates and nominate the most preferred configuration in the candidate list. The authors used the weighted sum model (WSM) to evaluate each component according to its list of non-functional characteristics. Finally, by applying the selected configuration step, the reconfiguration management layer is responsible for reconfiguring the components and applying the selected configuration to choose the best configuration based on the user's preferences.

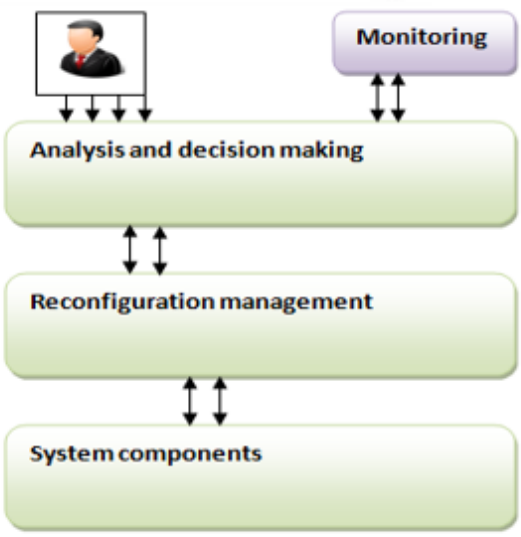


Figure 2.1: The architecture of an autonomous reconfigurable component-based system [37]

(J.Hillman and I.Warren, 2004) [38] proposed OpenRec an open framework including a reconfiguration management unit for managing dynamic reconfiguration, and a thoughtful component model. (figure 2.2). OpenRec has three main characteristics. First, it supports an extensible set of algorithms for reconfiguration. The reason is that one algorithm is not suitable for all applications. Second, it supports measuring the cost of reconfiguration. This enables developers to create a comparative analysis of multiple reconfiguration algorithms. Third, the framework is itself reconfigurable. Particularly, algorithms for reconfiguration can be replaced at runtime. The authors used OpenRec ML for scripts of reconfiguration. OpenRec ML is a language based on XML that is used to describe configurations of components and changes to existing configurations. In terms of scalability, the framework for large systems is not scalable [39]. Later in [39], the authors proposed an approach to preserve the integrity of the application during the runtime change period. The proposed approach uses ALLOY [46] to describe the system constraints. To verify limitations automatically at reconfiguration time, the authors complete OpenRec

with ALLOY Analyzer using a service-oriented architecture.

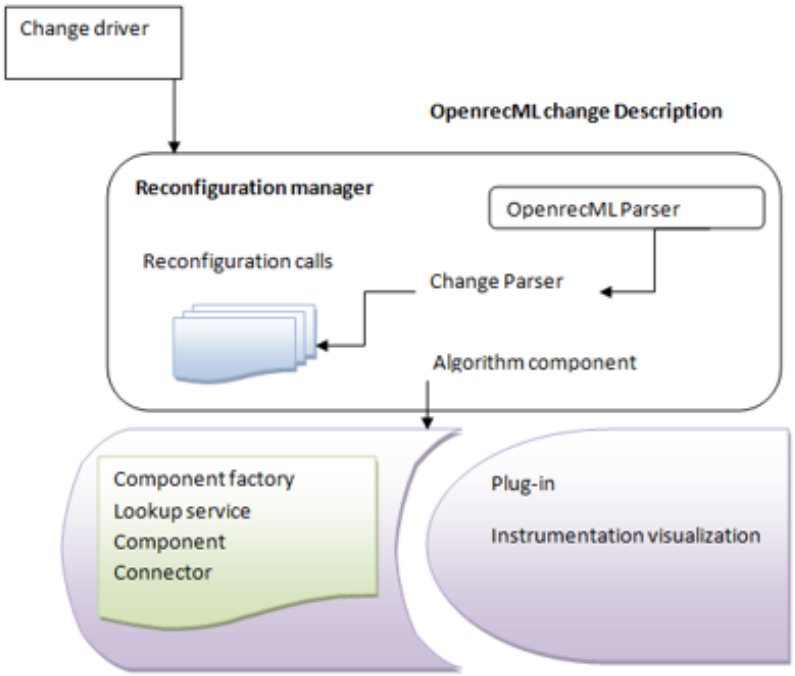


Figure 2.2: OpenRec Architecture [38]

In [40], (Y. Li et al, 2011) presented an approach to configure an SOA-based application by replacing component services for satisfying new QoS constraints. The authors introduced a factor named "global significance value" to show the contribution of the QoS of each component service to the overall QoS of the application which contains these component services. They used the notion of partial derivative to calculate the significance of services. Each service has four attribute significance values and one global significance value. This approach tries to replace each component service according to the global significance value. If the global significance value of one service is bigger, its contribution to the improvement of the QoS of the application is greater and it should be replaced firstly. If all attempts failed in the first phase, it replaces multiple component services by other service. Finally, the algorithm halts until any satisfactory solution is found. Four QoS attributes of

Web services are considered: Response Time (Ts), Cost (Cs), Reliability (Rs), Availability (As) and four service composition structures: Sequence structure, Parallel structure, Selection structure, and Loop structure.

(K. lin et al, 2010) [41] proposed an approach to maintain the original end-to-end QoS constraints by replacing the region of fault service. In the first step, the suggested approach attempts to discover substitutes for failed services one by one, then, for all failures, it expands these failing services with their neighbors and attempts to find replacements. In all failures, the extended procedure with adjacent services continues until requirements are satisfied or the number of integrated services surpasses a preset threshold factor c ($0 < c < 1$). The reconfiguration algorithm will be stopped and the whole service process will be recomposed if all repair regions together contain too many services (more than c*s, s is the process size). The service process is represented by a directed acyclic graph (DAG) and is specified in a language called BPELQ [41] BPELQ defines both the structural flow and QoS information for the service process. The authors implemented the Adaptation Manager in the Llama ESB middleware. The service process reconfiguration in the Llama system (figure 2.3) includes four main components: Accountability Authority (AA), Adaptation Manager, QBroker and Llama ESB. The first component is responsible for identifying services that cause a performance problem in a service process. The second component is in charge of reconfiguration region identification and reconfiguration execution decisions. The third component provides process planning, service selection as well as BPEL generation for a service process. Finally, the Llama enterprise service bus (ESB) has built-in support for dynamic service replacement by re-routing service messages to new services.

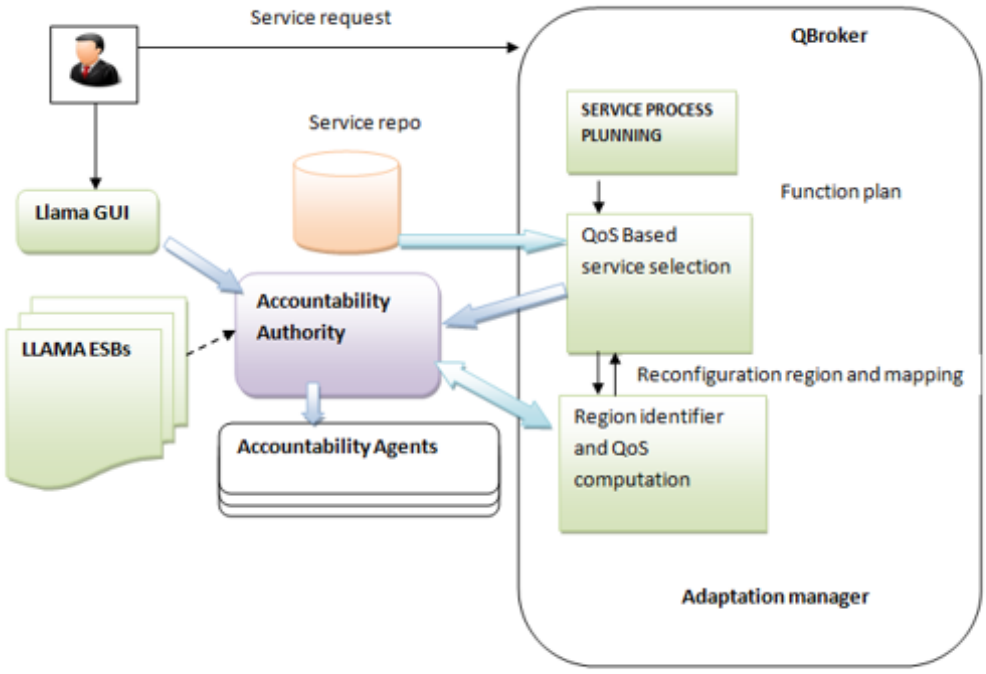


Figure 2.3: Reconfiguration in Llama [41]

(C. Lv et al, 2015) [42] proposed an event-driven continuous query algorithm to cope with different types of change, and thus enable evolution of service composition (figure 2.4). The authors present the composition results in the form of DAG (Directed Acyclic Graphs), and each service Wi in the graph is represented as Wi= IWi, OWi , selfQoS, allQoS, count, Enabled, where IWi / OWi is the input/output of the web service, selfQoS is its own QoS value of Wi, allQoS records the optimal overall QoS from start to current service, Count is assigned to the size of IWi initially, Wi is enabled if its count goes to zero, ie all IWi are available. Enabled value set TRUE when count is 0. The authors use the algorithm of Sim-Dijkstra in their previous work [26] to find the optimal result of the composition of the service in order to store the graph in two inverted index tables and for each change they will affect the status of other services, the system conducts a forward search starting from service changed and only updates its successors that become affected services. This

process continues until all current successors are unaffected. Otherwise, the system needs to update the graph structure and service affected status. The system generates a new service composition if the final affected service is presented in the original service composition. The author used n-m substitution instead of 1-1 substitution. The algorithm is not very practical in case of service providers violate their service-level agreement.

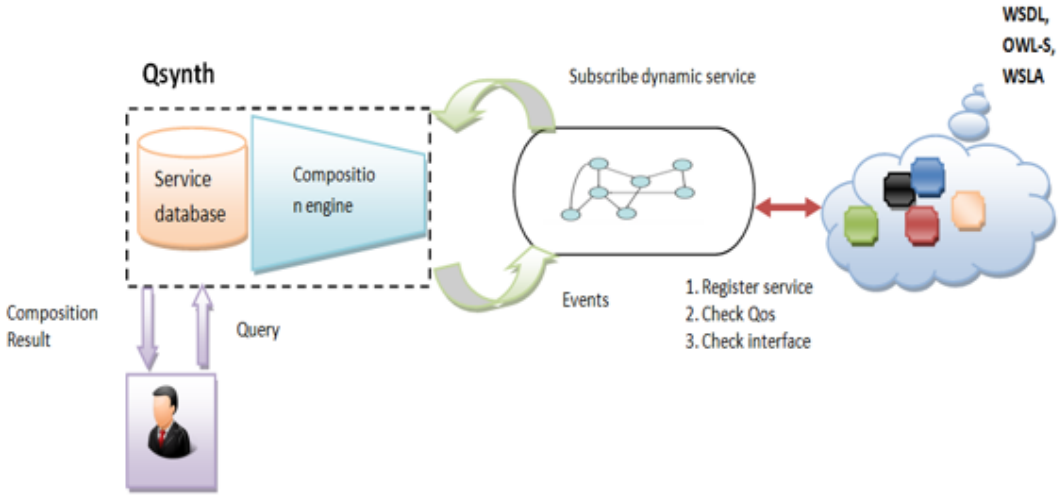


Figure 2.4: Architecture of event-driven continuous query algorithm [42]

(H. Gao et al, 2018) [6] proposed a service selection method based on the consistency of the interface operation and the non-functional requirement of quality of service workflow (QoW) for reconfiguration process (figure 2.5). This work aims to provide a convenient and efficient way of ensuring the correctness and reliability of a service workflow in the event of a service failure; the authors proposed an architecture of the service workflow reconfiguration including four modules: the function-matching process, the non-functional computing process, the reconfiguration pattern, and the workflow execution and monitoring. The function-matching process is accountable for monitoring the functional behaviors at the interface operation level. The nonfunctional computing process is responsible for calculating each reconfiguration's QoW value, which is based on the QoS of the service

composition, and calculating the QoW value based on their branch structures. The reconfigurations pattern is responsible for assigning service to an activity node of the workflow. The authors design four types of reconfigurations pattern, one-to-one, many-to-one, one-to-many and many- to-many mode. The basic service repository is accountable for storing services in such a manner that they are categorized into separate repositories that are distributed according to their domain and application.

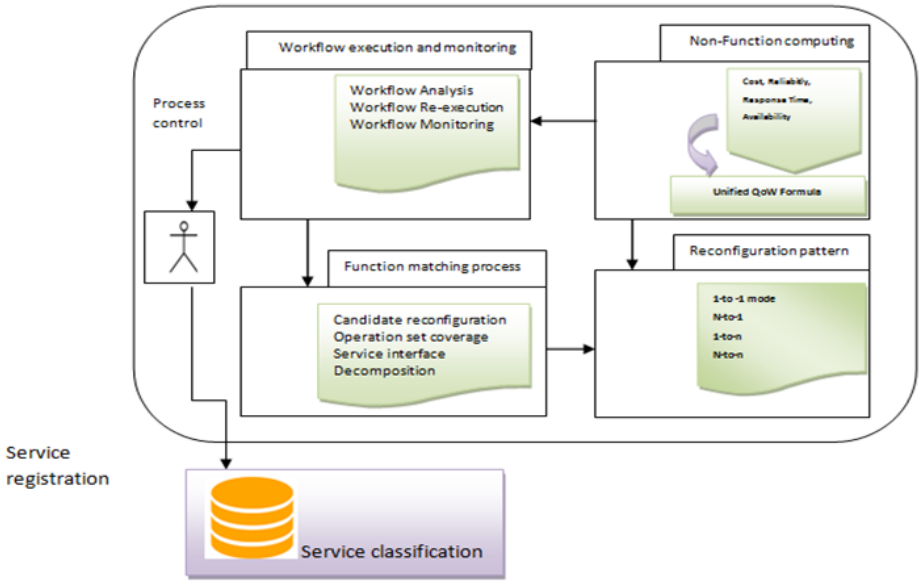


Figure 2.5: Architecture of the Service Workflow Reconfiguration [6]

(S. Saurabh et al, 2015) [43] studied the problem of dynamic reconfiguration according to two approaches. The first one produces an optional path of services from its predecessor services to the fulfillment of the service process. This makes it possible to switch to a new optional service path without the faulty service. The second approach offers an alternative service path from the beginning to the end of a service process. This approach is based on the reality that service set expansion is permitted until the defective region receives an optional path that is efficient QoS for continuity of the system. Authors used Directed Acyclic Graph (DAG) (figure 2.6) to find out the execution path of service composition. This approach is inappropriate when several faults occur at the same time.

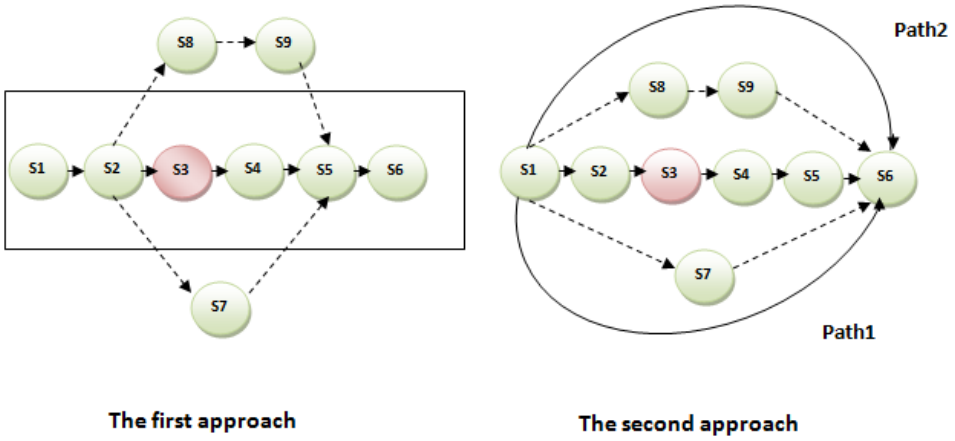


Figure 2.6: A DAG of Service Composition [43]

(F. Boudries et al, 2019) [5] proposed the using of a meta-heuristic to replace multiple failed services and maintain the original end-to-end constraints. This metaheuristic is a hybridization of two meta-heuristics particle swarm optimization and Shuffled frog-leaping algorithm. PSO is known for its rapid convergence, whereas SFLA is known for its global research capability. The Authors present a service composition as a triple $CWS = (AS, O, QoScws)$ where: AS is a subset of registered abstract services, O is a set of composition operators such as sequential operator, loop operator, parallel operator, conditional operator. $QoScws = QoScws(q), q = 1...nbq$, where $QoScws(q)$ represents the qth quality attribute value of CWS, while nbq is the number of quality attributes. The Authors used the function $F(CWS)$ for finding the best reconfiguration, where $F(CWS)$ is a weighted sum of the differences QoS values of the composition. In this work, a composite web service is considered as an initial solution, and some services associated with this initial solution are considered failed. Furthermore, this approach is used to discover replacements for these services while enhancing global QoS requirements. The approaches study the case one-to-one substitution and did not use the reconfiguration pattern.

Kahlon et al. [10] proposed a hybrid approach distributed between the service client and the service provider to manage a situation when QoS values degrade. The approach is based on the publish/subscribe mechanism. On the provider side, the provider uses the Service Monitor Agent (SMA) to monitor web service data to all the clients. The SMA analyses the log in real time, and generates alerts (or notifies) when it detects anomalous events. On the client-side, a service client dispatches mobile agents to partner service providers. The Mobile Client Agent (CMA) subscribes to the SMA and waits for notifications. when the CMA gets notified of the QoS degradation, it informs the client. Afterwards the customer decides to substitute an alternative to the service. This prototype is also based on the exponentially weighted moving average (EWMA) to predict the degradation of QoS value. And it is based on three services: the service manager, service provider and service repair,these services are controlled by the workflow manager (Figure 2.7). However, this paper lacks a discussion of the prediction accuracy problem.

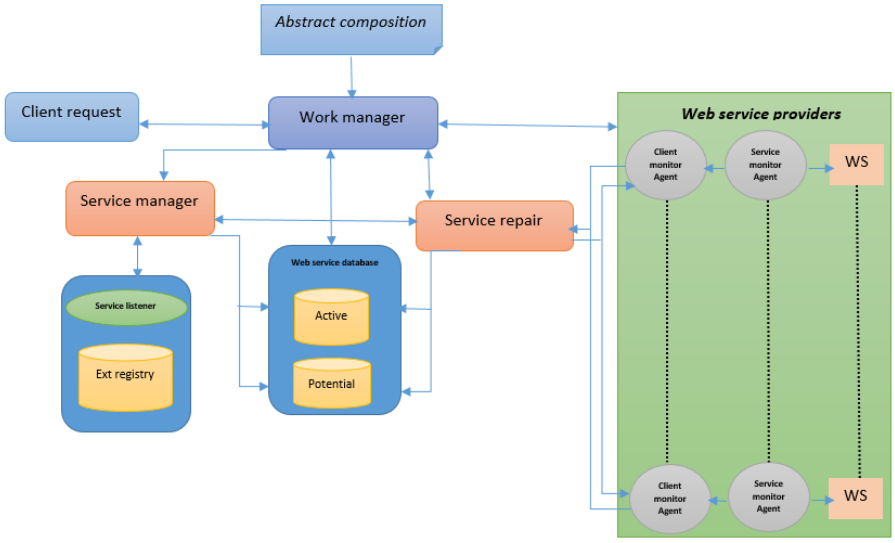


Figure 2.7: Architecture of the framework [10]

Mostafa et al [47], proposed a proactive approach to the web services composition (WSC), where used Q-learning as a Reinforcement Learning technique to adjust to dy-

namic change in WSC. the Authors represented the WSC process based on the Markov Decision Process (MDP). By investigating the web service execution log's historical data the approach can observe the WSC adaptation change proactively.

Zadah et al (2011) [48], Proposed an approach based on failure-prediction to achieve a self-healing architecture to reduce failures in web services, it used three agents Figure 2.8: Monitoring agent, for measure quality parameters in communication level and predict the value of QoS by used Time series forecasting (TSF) and neural network (NN), diagnosing agent, for diagnosing the faults in service executions by current and future values of quality parameters, and repair agent, for carrying out the possible solution failures occur used selection agent.

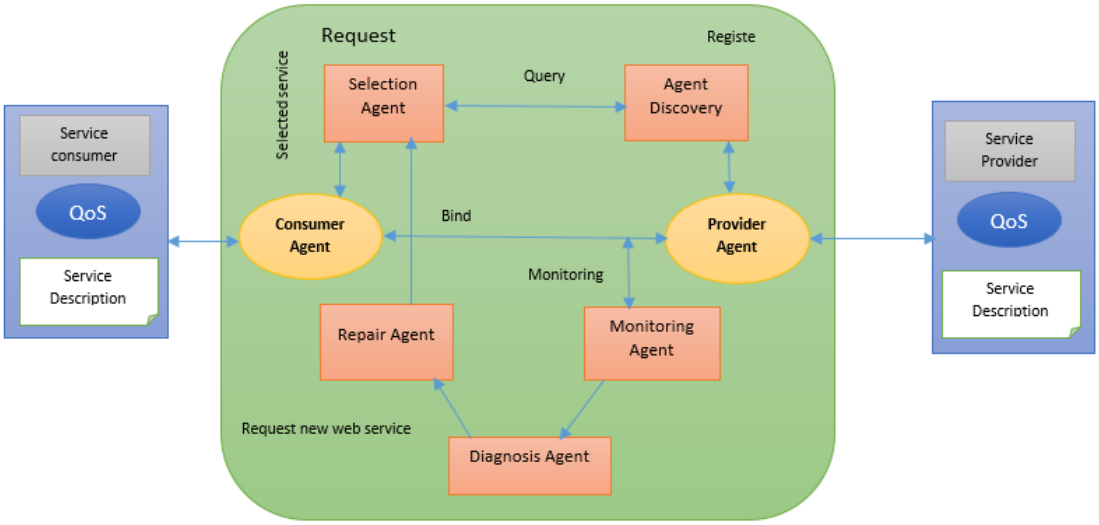


Figure 2.8: A self-Healing Architecture for web service based on Failure Prediction and a MAS [48]

Dia et al [49], proposed a self-healing approach based on performance prediction. They used the semi-Markov model for performance prediction. also a service reliability model has been proposed to find a replacement composite service. This framework (Figure 2.9)

begin by monitoring each component service in the composite one by try to predict whether the data transmission speed of the service is a deviation from the estimated one during the invocation. If the QoS at invocation time of a certain service is predicted to be a large deviation, and the corresponding substitute composite service is not available, the re-selection process triggers. This latter relies on QoS reliability for a new available alternative. This approach assumed that the data transfer speed, failure rate, and service request cost should be constant, which does not exist in a dynamic environment.

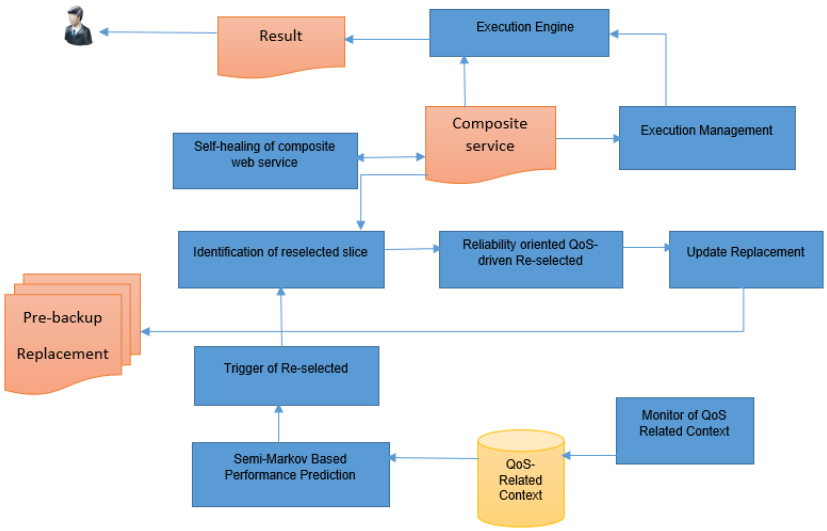


Figure 2.9: performance prediction-based QoS-driven self-healing web service composition [49]

(Guillaume Babin et al, 2016) [50] proposed, a correct-by-construction formal approach based on refinement using the Event-B method, which defines a compensation mechanism to repair failed services at runtime. They defined the Compensation as "a suspension of the currently running process or activity and a transfer of the execution to a compensating process or activity". The authors focused on checking the correctness of compensation via the invariants preservation using refinement method. Meanwhile, they defined a formal model for equivalent, degraded, and upgraded service compensations. but this paper does

not address the non functional aspects.

(Mingkun and Xiaohui, 2016) [51] proposed an empirical approach to accelerate QoS-aware runtime adaptation. They built decision maker to estimate the probabilities that candidate services will be used for upcoming adaptation scenarios, then candidate services are pruned based on these probability estimates to reduce the search space. By exploring historical records they used the Support Vector Machines (SVMs) to implement the empirical decision maker (Figure 2.10), However, this approach has some limitations such as the assumption of users' preferences and QoS attributes are unchanged for continuous runtime adaptations.

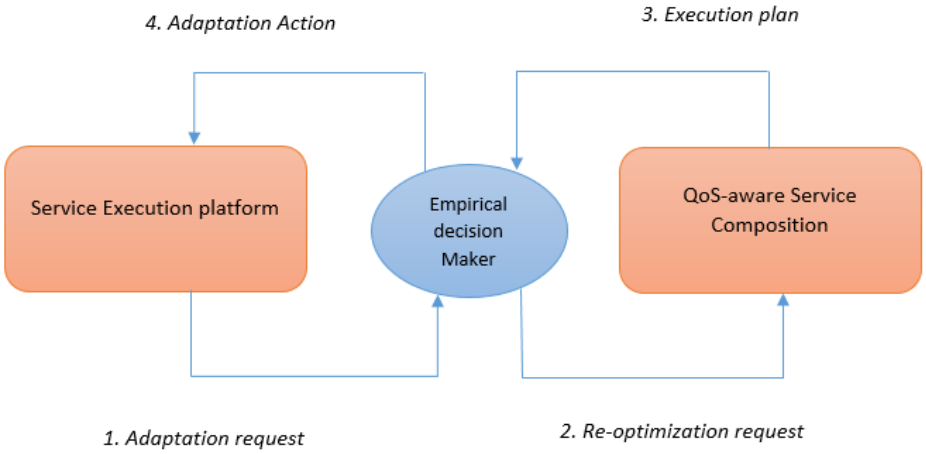


Figure 2.10: The framework of runtime adaptation system [51]

(Huan Zhou et al, 2011) [52], used the Physarum model for dynamic service composition and reconfiguration of Internet-ware systems, this model inspired from the formation and behavior of such biological adaptive networks. The organism Physarum polycephalum which has the single cell explores the available space and connects food sources. In the absence of central control mechanisms this organism can find the shortest path and also

have an efficient reconfiguration capacity in case of a traffic accident. Based on a mathematical model, the authors used the mechanism of Physarum protoplasm flow changes to find the shortest path that can be used to solve the dynamic service composition and reconfiguration problem. The authors focused on the sequential service composition model.

(Zhu et al, 2017) [53], proposed a collaborative QoS prediction approach based on the adaptive matrix factorization to perform online QoS prediction for candidate services. They used a reactive adaptation when encountering a change. the authors formulated the problem of QoS prediction to leverage historical QoS data observed from different users to accurately estimate QoS values of candidate services, and to adapt to QoS fluctuations over time (Figure 2.11). This approach has two contributions: the first is the algorithm for online learning and the second is the time evolution.

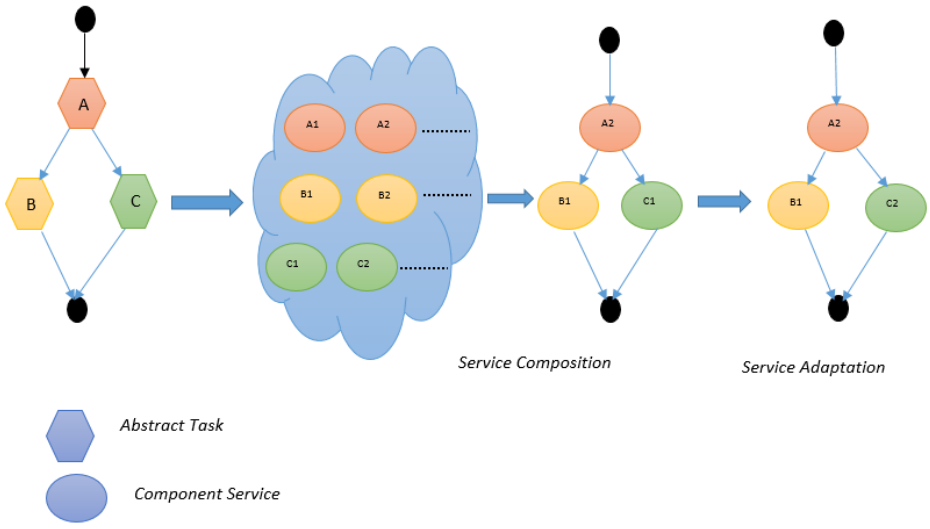


Figure 2.11: An example for runtime service adaptation [53]

2.4 Classification of approaches to dynamic reconfiguration of composite web services

In this section, we offer a related summary of the surveyed approaches. As we see during the survey, dynamic reconfiguration is a part of the process of restoring the service to its normal state or improving it (see Figure 2.12), which has begun by the monitoring and ending by the reconfiguration and validation.

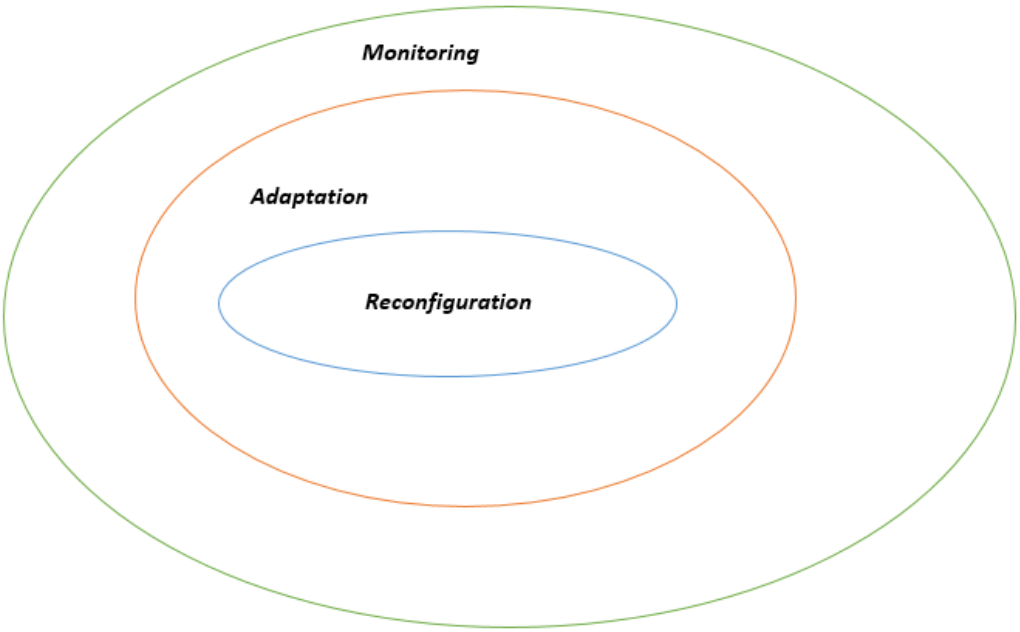


Figure 2.12: classification of monitoring process, adaptation and reconfiguration.

To identify the eventual shortcomings in the field of dynamic reconfiguration, we present many criteria of comparison which we believe to help to clarify this research field. We will overview the main techniques used by the researchers to adapt to environmental changes to evaluate the similarity of those approaches. As shown in Figure 2.13. We present five

of the main regards of dynamic reconfiguration. The first regard is the strategy used, which means the reconfiguration action can trigger in a proactive manner or reactive way. Some approaches such as [37] and [43] used the reactive strategy, while [53] and [54] used the proactive strategy. The second regard is the reconfiguration goal, which specifies the reconfiguration aim based on QoS requirements. Some approaches, such as [55], focus on single QoS optimization, whereas [56, 57, 58] focus on multi QoS criteria. The third regard is reconfiguration action, that used to address a reconfiguration problem. The reconfiguration action may include a single substitution, a region substitution, or a structural change. The fourth regard is the model used, which is the model that may be used to carry out a reconfiguration action such as mathematical model, graph model, and model-driven. The fifth regard is approach-based, which refers to the approaches that might be used to carry out a reconfiguration action, such as prediction-based, bio-inspired-based, agent-based, and hybrid-based.

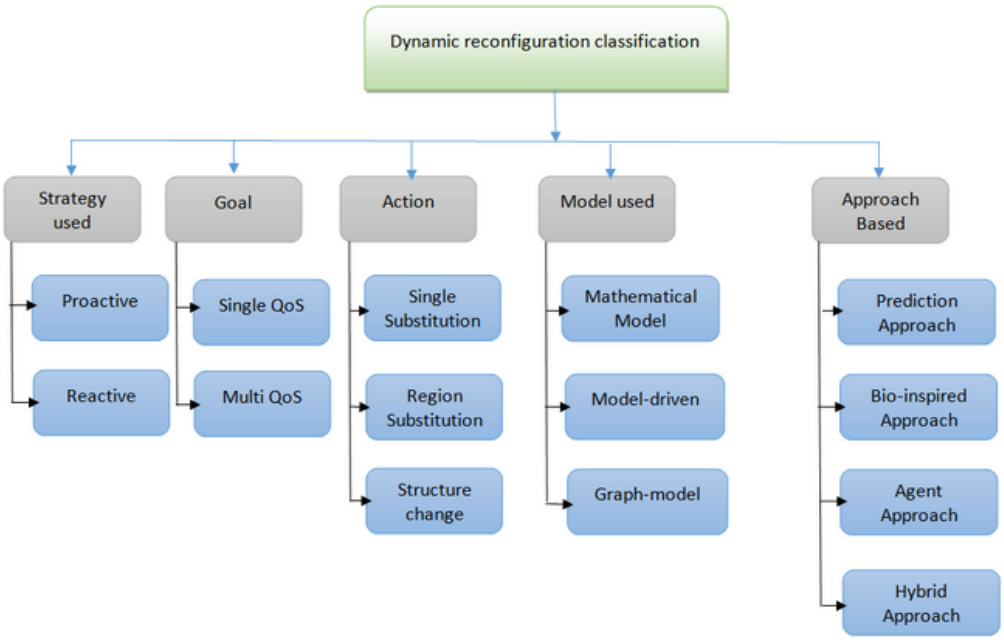


Figure 2.13: The main aspects of dynamic reconfiguration in CWS

There are some criteria that are not mentioned or addressed in the classification of dynamic reconfiguration research, in the following we present the most prominent of them:

- **Correctness proof:** Preserve the correct and reliable execution during and after the system reconfiguration.
- **Configured Faults:** The process of reconfiguration can be affected by the number of faults that appear at the same time, some authors used one fault and others used multi-faults.
- **Independency:** This attribute specifies if the approach is independent of the application context or was developed for a particular type of software application.
- **Scalability:** It means a scale up of the dynamic reconfiguration approach once the

system grows.

- **Transparency:**Means that the approach is transparent from the application developer and does not require any effort from the developer to create a reconfigurable system.
- **Type of change:** It is the operation used in the dynamic reconfiguration process such as add, remove, link, unlink and replace one-by-one or n-to-m.
- **User preference:** Means an exploiting of the non-functional properties to maximize the user satisfaction.

The next tables (2.1 and 2.2) illustrate the analytical study performed on these selected researches and based on the criteria of comparison quoted above.

Table 2.1: Comparative criteria

	Strategy used	Goal	Action	Model used	Approach Based
[37]	reactive	Multi QoS	reconfiguration replaced	Acem(ADL)	Algorithm +Wsm
[38]	reactive	/	reconfiguration replaced	Framework	OpenRec(xml)
[40]	reactive	Multi QoS	Single replaced	/	Algorithm
[41]	reactive	Multi QoS	Single+Region	LLma ESB	Algorithm
[42]	reactive	Singl QoS	Single + Region(n-m)	Graph Model	QsythTools+Algorithm
[6]	reactive	Multi QoS	Single + Region(n-m)	Framework	Workflow reconfiguration
[43]	reactive	Singl QoS	Structure change(SP)	Graph Model	Algorithm
[5]	reactive	Multi QoS	Single + Region(n-m)	Mathematical Model	Bio-inspired Approach
[10]	proactive	Multi QoS	Single replaced	Framework	Predictive+pub /sub mechanism
[59]	proactive	Multi QoS	Single replaced	Model driven	Hybrid approach (MDP+Qlearning)
[48]	proactive	Multi QoS	Single replaced	Framework	Hybrid approach (TSFP+NN)
[49]	proactive	Multi QoS	Single replaced	Model driven	Predictive approach
[50]	reactive	/	Single replaced	Formel Model	Event-B refinement
[51]	proactive	/	/	SVM	Algorithm
[52]	reactive	/	Switch path	Mathematical model	Bio-inspired
[53]	reactive	Multi QoS	Single replaced	Mathematical model	Algorithm

Table 2.2: Comparative criteria

	Correctness proof	Configured Faults	Independency	Scalability	Transparency	User preference
[37]	/	One Fault	I	S	T	yes
[38]	/	/	I	/	T	/
[40]	yes	Multi Fault	/	/	/	/
[41]	/	One Fault	I	S	T	/
[42]	/	Multi Fault	I	S	T	/
[6]	/	Multi Fault	I	S	T	/
[43]	/	One Fault	I	/	T	/
[5]	/	One Fault	/	/	/	/
[10]	/	Multi Fault	I	S	T	yes
[59]	/	Multi Fault	I	S	T	yes
[48]	/	Multi Fault	I	S	T	yes
[49]	/	Multi Fault	I	/	T	/
[50]	yes	/	/	S	T	/
[51]	/	Multi Fault	I	S	T	/
[52]	/	Multi Fault	I	S	T	yes
[53]	/	Multi Fault	I	S	T	yes

Despite the existence of several works that focus on the most important methods in the context of dynamic reconfiguration of composite web services, we think that this field needs more investigations to fix their shortcomings. The main shortcomings identified by our comparative study consisted of:

- Guarantee the correctness of dynamic reconfiguration.
- Use of the graphic model.
- Mechanisms to guarantee the consistency of the architecture before and after the dynamic reconfiguration.
- Strategies to combine the structural and behavioral change in dynamic reconfiguration.
- Use of all types of changes add/remove, link/unlink, update.
- Use of the Architecture Description Languages (ADLs) that can allow rigorously specifying the global architecture of a system, which can be analyzed by automated tools. Finally, suggest further researches on proactive approaches with a focus on prediction methods.

2.5 Conclusion

Dynamic reconfiguration is a pivotal point in SOA. The objective of this chapter is to analyze and compare various dynamic reconfiguration of composite web services approaches to deal with the unforeseen changes. The studied approaches have some limitations. One of the limitations is that they cannot treat the correctness during and after the reconfiguration. Another disadvantage is that the majority of these techniques are incapable of dealing with structural and behavioral change at the same time and lack the graphic model to represent the architecture of dynamic reconfiguration. In the next chapter, we will present the second approach to the dynamic reconfiguration of composite web services,

by combining multi-agent and reinforcement learning technologies. We will use ACO (Ant Colony Optimization) to improve the exchange of information between agents based on the Pheromone-Q values.

Chapter 3

Dynamic reconfiguration of composite web services using SRL-ACO

3.1 Introduction

Service-Oriented Architecture (SOA) enables the construction of distributed systems with Web services that can be published, invoked and composed dynamically at runtime. With the emergence of cloud and SaaS, many companies and organizations have relied on web services for their personal and business needs [1]. In response to user's demands and requirements, a single web service can not satisfy the required functionality, so the composition of web services is a significant step in meeting their needs [44]. The composition process enables integrating and interconnecting web services to create a composite web service, which seeks to meet its functional needs and non-functional requirements, such as QoS restrictions. However, there are a large number of candidate services with the same functionality but different quality of service. The challenge is to build an efficient composite web service, which is aware the quality of service in order to find the best web service that respects all the quality of service constraints, such as response time, price and availability [60]. Up to date, several approaches have been adopted to solve the QoS-aware automatic service composition problem. The environment in which service composition is executed is characterized by dynamism and evolving, which makes a web services involved in the composition vulnerable to disrupting or affecting the quality of the services that characterize it, therefore, providing reliable and efficient services are a big challenge, and with the emergence of dynamic reconfiguration it becomes possible to replace the failed services, modify it or delete it dynamically at runtime, which finding replacements for failed service allows the composite service to continue running without completely reconfiguring the composition.

However, the increasing number of candidate web services may present the dynamic reconfiguration service to several obstacles, including the difficulty of selecting an optimal alternative web service, especially in the case of several errors at the same time, the problem becoming an optimization problem belonging to the category NP-hard [8]. To address the above challenges, combining multi-agent and reinforcement learning technologies (swarm reinforcement learning : SRL)[61] can assure the efficiency of dynamic reconfiguration of service-oriented services when failed service take place. SRL is a kind of Machine Learning

(ML) concerned with how an agent learns to achieve a goal through many episodes in a dynamic environment, where an optimal policy (the best composition) could be obtained rapidly by using multi-agent algorithms in which multiple agents learn through combine two learning strategies. First, every agent learns on their own using one of the standard reinforcement learning algorithms. In this contribution, we use the Q-learning method [62]. Second, to accelerate the learning rate, the agents regularly communicate information among themselves while learning individually based on the shared information. The efficient techniques of swarm reinforcement learning is extremely dependent on a technique of sharing information that should be adequately prepared. In this chapter, we have used an ant colony optimization (ACO) method [63] for the information communication method. This technique is based on real-life ant behavior utilizing trail pheromones. The synthetic pheromone acts as a communication medium between ants. Combining standard Q-learning techniques with a synthetic pheromone allows speeding up the learning process of cooperating agents, in order to replace multiple failed services and maintain the original end-to-end constraints.

3.2 Overview of Proposed Approaches

Recently, dynamic reconfiguration of composite service has received considerable attention, especially when faced with large-scale problems. Many solutions have been developed by leveraging techniques, such as integer programming, meta-heuristic, graph planning and machine learning. Ardagna and Pernici [64] proposed an approach based on integer programming to optimize the composition of Web services, and adaptive optimization at runtime. The authors formalized the service composition problem as a mixed integer linear programming problem to select web services that satisfy the predefined QoS constraints. In [65, 66] the authors also used integer programming to eliminate useless candidate Web services which cannot be part of the optimal solution in order to reduce the search space by considering the global QoS constraints and used a skyline operator. But this kind of methods are only applicable to small-scale problems, and lack self-adaptability. So, they cannot solve the problem of service composition in a large-scale and dynamic environment.

Meanwhile it was suggested in [5] an approach based on meta-heuristic to replace multiple failed services and maintain the original end-to-end constraints. This meta-heuristic is a combination of two other meta-heuristics: particle swarm optimization and Shuffled frog-leaping algorithm. Liu et al. [67] proposed an approach based on Ant Colony Optimization (ACO) and GA to solve the QoS-aware service composition problem, where ACO is used to generate an initial population in the GA algorithm. Wang et al. [68] used ant colony optimization (ACO) to choose a composition near to optimal; they have analyzed the performance by varying several parameters (the rate of evaporation of the pheromone, the initial population ...). Yan et al. [69] formalised service composition as a partially effective planning graph, and replaced the undesired services using a greedy technique. Some researches [70, 71, 72] have applied Multi-agent techniques in dynamic service composition problems, but they did not consider the adaptation and efficiency in large-scale and complex scenarios. Many researchers [73, 74, 75] focused on machine learning, especially, on reinforcement learning methods to propose an adaptive service composition approaches in dynamic environment.

Despite the fact that traditional reinforcement learning is effective in achieving adaptability, these methods cannot ensure high efficiency in a large-scale and complex scenario. Wang et al. [76] integrate multi-agent and reinforcement learning to face the shortcomings in the previous works. The authors used Markov Decision Process (MDP) to model web service composition. This proposition lacks communication technique with agents and may trap in local optimum. Moreover Wang et al. [77] proposed an adaptive service composition approach based on QoS prediction and reinforcement learning. The authors used a time series prediction method based on LSTM to help estimate the QoS in a dynamic environment, and reinforcement learning to make dynamic service selection. They are based on the Markov decision process to model the web service composition in a dynamic environment.

3.3 Background

In this section, we have mainly discussed some background information on the composition of QoS based web service as well as swarm reinforcement learning and ant colony optimization.

3.3.1 QoS of Web Service

The web service is characterized by many non-functional attributes such as quality of service, which are essential in selecting the optimal web service in light of the significant increase in web services with the similar functionality, for this purpose we define a web service's QoS, as follow: It is a set of attributes such as $WS = \{QS_1, QS_2, QS_3, QS_4\}$ where QS_1,QS_2,QS_3 and $QS4$ represent response time, throughput, reliability, and availability respectively. The QoS parameters of a web service can be classified either positively or negatively [1]. If a QoS parameter is positive (e.g. throughput, availability) the larger values will show better quality. Whereas negative one (e.g. response time), smaller values means better quality. So, they are normalized as shown by Equation (3.1) for negative parameters and positive ones,

$$\text{Norm(QoSi)} = \begin{cases} \frac{\text{Qmax}-(\text{Qi})}{\text{Qmax}-\text{Qmin}} & \text{if Qmax} - \text{Qmin} \neq 0 \quad \text{for positive QoS attributes} \\ \frac{\text{Qi}-\text{Q min}}{\text{Qmax}-\text{Qmin}} & \text{if Qmax} - \text{Qmin} \neq 0 \quad \text{for negative QoS attributes} \\ 1 & \text{if Qmax} - \text{Qmin} = 0 \end{cases} \tag{3.1}$$

Where Q_i is the i_{th} QoS parameter of web service and the $Qmax$ and $Qmin$ are the maximum and minimum values of i_{th} QoS parameters respectively.

3.3.2 QoS of Web Service Composite

Many complex tasks require compositing of many web services among themselves to perform these tasks efficiently and rapidly, as web services are composed as a sequential,

parallel or loop structure, and due to the rapid growth in the web services with similar functionality. Finding an efficient web service to create a new service that meets user's requirements is a challenging task. We deal in this approach with the sequential structure that is the basis of the other structures [78]. We define the composite web service as a group of web services linked between them according to a specific structure that results in the best service quality according to the user's needs. The following Figure 3.1, shows a set of tasks linked on the sequence, whereas for each task, we select the most appropriate web service from service candidate sets, so finding the optimal web service out of all services candidates is about finding pathways that meet the user's needs. The challenge is how to find the best way among the available paths, so that the overall quality of service is optimal, while all the QoS constraints are satisfied. We have defined the quality of service for a composite of web services as follows: QoS vector of web service composition WSC is represented as $Q(wsc) = \{Qrt(wsc), Qthr(wsc), Qavail(wsc), Qre(wsc)\}$, where $Qi(wsc)$ is the ith global attribute of the web service composition, $i = \{Responsetime, Throughput, Reliability, Availability\}$.

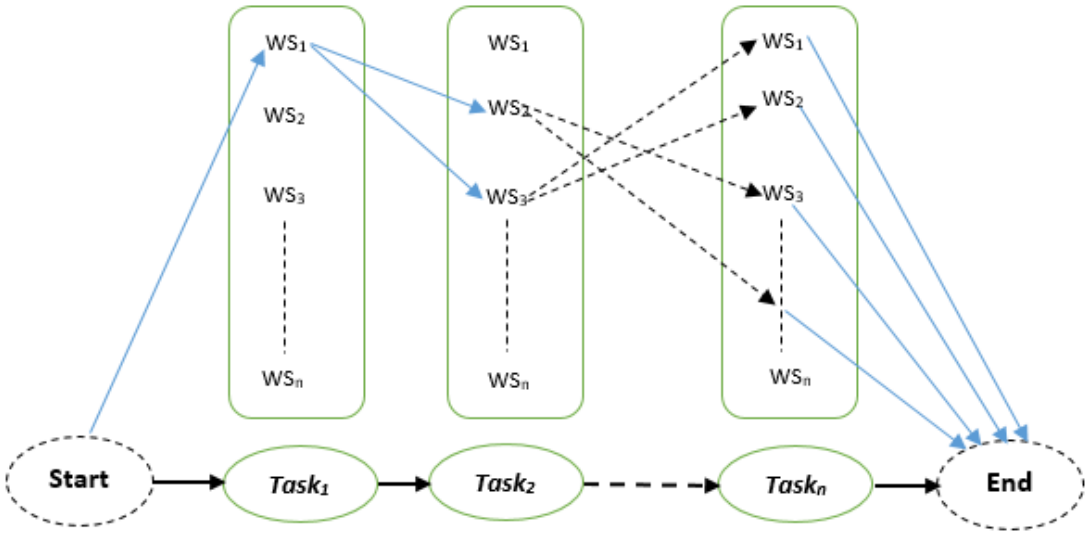


Figure 3.1: Web service composition structure.

$Qi(wsc)$ is the aggregate from the values of the ith QoS attributes of web services that participate in the composition of the web services. Most commonly met QoS aggregation functions can be represented by additive, multiplicative and Min-operator attributes. For example, availability and reliability can be aggregated as a product, response time can be aggregated as a sum and throughput can be aggregated as a Min-operator. Table 3.1 shows that QoS aggregation functions for the four aforementioned QoS attributes.

Table 3.1: the QoS aggregation functions

QoS Attributes	Aggregation functions
Response Time	$\sum_{s \in WSC} QoS_{rt}$
Throughput	$\min_{s \in WSC} = \{\boldsymbol{qos}_{thr}(\boldsymbol{S}_1), \ldots, \boldsymbol{qos}_{thr}(\boldsymbol{S}_i)\}$
Reliability	$\prod_{s \in WSC} \boldsymbol{qos}_{rel}(s)$
Availability	$\prod_{s \in WSC} \boldsymbol{qos}_{\text{aval}}(\boldsymbol{s})$

3.3.3 Fitness Function

Selecting the optimum reconfiguration from all potential combinations is viewed as an optimization issue that must be solved by optimizing a fitness function. In our case, the fitness function $f(wsc)$ is a sum of all QoS attributes with regard to the user's preferences, which is calculated as follows:

$$f(wsc) = \sum_{i=1}^{4} QoS_i(wsc) * w_i \tag{3.2}$$

where $QoS_i(wsc)$ determines the value of the ith quality attribute of WSC and w_i presents the user's preference weight of the QoS_i ,The weights are assigned in such a way that: $w = \sum_{i=1}^{4} w_i$

3.3.4 Swarm Reinforcement learning method (SRL)

Reinforcement Learning (RL) is an unsupervised learning approach where an agent learns how to reach a goal by trial-and-error interacting with a dynamic environment. The main elements of reinforcement learning (shown in Figure 3.2) can be identified as follows:

- **Agent** is the learner and the decision maker.
- **Environment** is where agent learns and decides what actions to perform.
- **Action** is a set of actions which the agent can perform.
- **State** is the state of the agent in the environment.
- **Policy** refers to the function of $\pi : S \times A \rightarrow R+$ which defines how an agent acts from a specific state, and is such that $\pi(S \mid A)$ represents the probability of choosing a in s. It is the vital part of an RL agent in the sense that it alone is sufficient to determine the agent behaviour.

- **Reward** it's the recompense provided by the environment when the agent selected an action.

The objective of agent or learner is to act in a way that maximizes the reward obtained from the environment after performing an action [79]. Specifically, an agent executes an action a_t. After execution, the reward value r_{t+1} is obtained from the environment, and then the agent transfers it to the new state s_{t+1}. The agent will adjust the action selection according to the reward, and finally form its own action selection strategy, which obtains the maximum reward value. Formally, we can present reinforcement learning as Markov Decision Process (MDP) as a tuple (S, A, P, R):

1. a discrete set of environment states S;
2. a discrete set of agent actions A;
3. $p : S \times A \times S \mapsto [0, 1]$, is the transition probability from s to s' given that action a is taken, $P(s, s', a) = P(s' \mid s, a)$;
4. $R : S \times A \times S \to R$ is the reward received from action a when going from s to s', A reward function $R(s, s', a)$ is a measure for immediate action and a value function is for the long run.

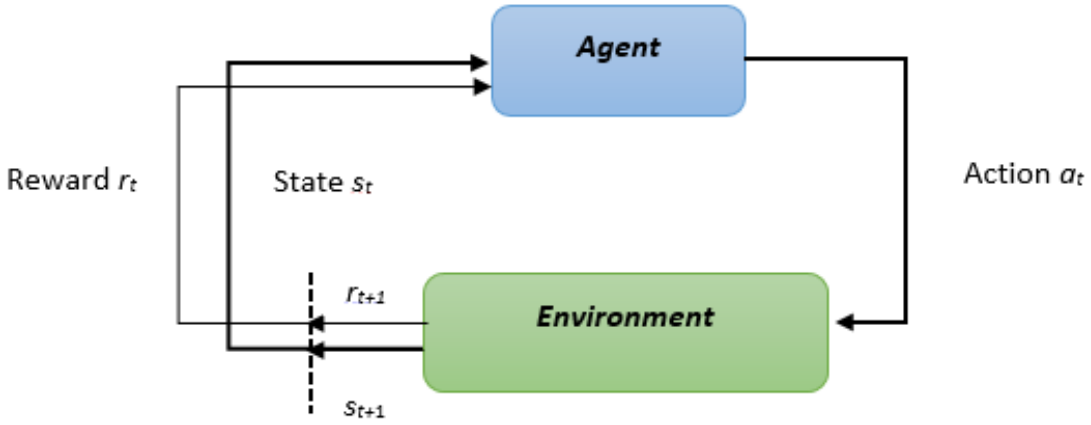


Figure 3.2: framework of reinforcement learning

3.3.5 Q-learning

The quality learning (Q-learning) [80] is the result of the development of an off-policy Temporal Difference (TD) control algorithm [81]. The Q-learning algorithm learns how to estimate the quality of an action a in a given state s based on the predicted cumulative reward (Q-value)[82], the agent updates its estimation Q-value $Q(s, a)$ as follows:

$$Q\left(s_t, a_t\right) \leftarrow Q\left(s_t, a_t\right) + \alpha\left[r_{t+1} + \gamma \max Q\left(s_{t+1}, a\right) - Q\left(s_t, a_t\right)\right] \tag{3.3}$$

in formula 3.3, s_t be the state of the agent at time t. and a_t be the action executed by agent at time t. the s_{t+1} is the resulting state after executing the action a_t in the state s_t and r_{t+1} is the awaiting reward after agent be in the new state s_{t+1}. α is the learning rate its range is from 0 to 1 $(0 < \alpha < 1)$, and γ is the discount rate is ranged between $(0 < \gamma < 1)$.

The objective is to optimize the action-value function for each state-action pair in order

to deduce an optimal policy that follows the following formula:

$$\pi^*(s) = \text{argmax}_a \, Q(s,a), (\forall a \in A(s)) \tag{3.4}$$

With the aforementioned objective in mind 3.4, it will be necessary to properly evaluate after each Q-Learning process the agent's Q-values. The evaluated value $Eval$ is based on rewards where the agent in each learning episode benefits during the previous one episode, and is defined by:

$$Eval = \sum_{k=1}^{N} d^{N-k} r_k \tag{3.5}$$

Where rk is the compensation for the kth action, $(d < 1)$ denotes the discount parameter, and N is the number of actions in the latest episode.

3.3.6 Ant colony optimization algorithm (ACO)

The ACO meta-heuristic was inspired, essentially, by studies on the behavior of real ants carried out by Deneubourg et al.[83]. One of the problems studied was to understand how insects, such as ants, can find the shortest route from the nest to the food source and the way back. It has been found that the means used to communicate between ants which search for paths, is the deposit of traces of pheromone, i.e., a chemical substance that ants can detect.

Given a graph above mentioned in Figure 3.1. An instance of the dynamic web service composition problem is given by a graph (C, E), where C is the set of service candidate sets and E is the set of edges that connect the selected services from each service candidate set, to create a composition that meets the user's requirements. With the starting edge and the edge of the target, this problem can be transformed into the problem of finding the shortest path between the two edges. Using ants' behavior finding the path between the food source and the colony, this algorithm is used to find an ideal path.

In the ACO algorithm proposed by Marco Dorigo [84], at every iteration t $(1 \leq t \leq t_{\max})$, and every ant k $(k = 1, ..., M)$ moves into the graph and creates a total path in n steps. The

path connecting a candidate service i and a candidate service j for each ant is determined by the following factors:

- The list of the candidate services previously visited, which represents the potential moves at each step, while the ant k is on the service i: J_i^k.
- The inverse of the distance among the services: $\eta_{ij} = \frac{1}{d_{ij}}$, called visibility. This distance is used to guide the selection of ants towards nearby services, and to prevent services that are too distant or that do not satisfy the objective function in our case;
- The amount of pheromone deposited on the edge connecting the two services, called the intensity of the track. This parameter defines the attractiveness of part of the overall path and changes with each passage of an ant. It is, in a way, a global memory of the system, which evolves by learning.

The ACO Contributes how good each edge through exchanging information between Multiple ants to solve the shortest path problem, where each ant places pheromones on the edges of the path it travels through, after that the ants takes the pathway that has a strong trail of pheromones as a shortest path [61].
The transition probability[85] is as follows:

$$p_{ij}^k(t) = \begin{cases} \frac{(\tau_{ij}(t))^\alpha \cdot (\eta_{ij})^\beta}{\sum_{l \in J_i{}^k} (\tau_{il}(t))^\alpha \cdot (\eta_{ij})^\beta} & \text{si } j \in J_i^k \\ 0 & \text{si } j \notin J_i^k \end{cases} \tag{3.6}$$

Where $\tau_{ij}(t)$ is the pathway intensity and η_{ij} is the visibility. α and β are two parameters of controle, with $\alpha = 0$, only the visibility of the service is taken into account; the service that satisfactory our objective function is therefore chosen at each step. On the contrary, with $\beta = 0$, only the pheromone tracks are playing. After a complete turn, each ant leaves a certain amount of pheromone $\Delta\tau_{ij}(t)$ throughout its path, an amount which depends on

the quality of the solution found.

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^{k}(t)} & \text{si}(i,j) \in T^{k}(t) \\ 0 & \text{si}(i,j) \notin T^{k}(t) \end{cases} \tag{3.7}$$

Where $L^k(t)$ is the distance of the pathway, $T^k(t)$ is the path created by the ant k at iteration t, and Q is a fixed parameter.
The algorithm will be deficient without the process of evaporating the pheromone traces. Indeed, to avoid being trapped in sub-optimal solutions, it is necessary to allow the system to "forget" the wrong solutions. We therefore counterbalance the additivity of the tracks by a constant reduction in the values of the edges at each iteration. The rule for updating tracks is given.

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \tag{3.8}$$

Where m is the number of ants and ρ the rate of evaporation.

3.4 Modeling dynamic reconfiguration problem of composite web services using SRL-ACO

To apply the SRL-ACO algorithm into process of dynamic reconfiguration when failed services take place, we must firstly define the reward function. The reward function is the cumulative reward of all the services executed for each sub-composition that potentially substitute the failing services [86]. In this contribution, we used the fitness function mentioned in Formula 3.2 to measure the reward function after each executed service.

The idea of the pheromone in the ACO can be applied to the reinforcement learning method by considering a state-action (s, a) as an edge e, and each agent updates its own Q-values $Q(s, a)$ by using the Pheromone-Q values $Qp(s, a)$, where the Pheromone-Q values play the same role as the pheromone in ACO. Thus, we can provide the updating equation

for the Pheromone-Q value $Qp(s,a)$ according to equation (3.8) in ACO, as Follows:

$$Q_{\mathrm{p}}(s,a) \leftarrow (1-\rho)Q_{\mathrm{p}}(s,a) + \sum_{i=1}^{I} \frac{Eval_i}{\sum_{r=1}^{I} Eval_r} Q_i(s,a) \tag{3.9}$$

Where I is the number of the agents, and $(p < 1)$ is the pheromone evaporation rate. The evaluated value $Eval$ is based on rewards, which calculated as follow:

$$\mathrm{Eval} = \begin{cases} \frac{1}{\frac{\mathrm{Qmax}-(\mathrm{Qi})}{\mathrm{Qmax}-\mathrm{Qmin}}} & \text{if Qmax} - \text{Qmin} \neq 0 \quad \text{for positive QoS attributes} \\ \frac{1}{\frac{\mathrm{Qi}-\mathrm{Q\,min}}{\mathrm{Qmax}-\mathrm{Qmin}}} & \text{if Qmax} - \text{Qmin} \neq 0 \quad \text{for negative QoS attributes} \\ 1 & \text{if Qmax} - \text{Qmin} = 0 \end{cases} \tag{3.10}$$

The suggested swarm reinforcement learning approach works as follows:

Algorithm 3.1 SRL-ACO algorithm for dynamic reconfiguration of composite web service

begin

S : set of states / services in composition;

A : set of actions / candidates services for each failed service;

nb_FS : number of failed services in composition;

$I \leftarrow nb\_FS$: number of agents;

T: number of episodes;

$i(s, a)$: state-action value of agent i for action a in state s;

Y: number of episodes for which Q-learning is performed in the inner loop;

Step1 : Initialization

Initial value of $Q_i(s, a)$ and set $t \leftarrow 0$;

Step2: Update $Q_i(s, a)$ for each agent i;

$Q_i(s, a) \leftarrow Q_i(s, a) + \alpha \left\{ r + \gamma \max_{a^* \in A(s_n)} Q_i\left(s_n, a^*\right) - Q_i(s, a) \right\}$

Step3: Calculate the evaluated value *Eval* for Q-values for every agent and every episode using eq (3.10);

Step4: Update $Q_p(s, a)$ by the interaction among the agents using eq (3.9);

Step5: $Q_i(s, a) \leftarrow \mathrm{Q}_p(s, a)$;

$t \leftarrow t + IY$;

If $t > T$, terminate this algorithm.

Otherwise, return to **Step2**;

end

3.5 Results and Discussions

In this section, we performed a simulation study to evaluate the performance of our proposed algorithm SRL-ACO in two aspects. With a real-world case study, the first aspect demonstrates the contribution of our method to dynamic reconfiguration when the failed services take place. The second aspect illustrates the scalability of our approach in the face of various failures at the same time when compared with the PSO algorithm. Python language has been used to implement the algorithms on a Lenovo Core i3 2.5 GHz, 4 GB RAM, and Windows 10 operating system.

3.5.1 Case Study

Figure 3.3 describes a prototype application for family village holiday ordering with a simplified workflow consisting of four abstract tasks. Particularly, the customer begins the search for a holiday family village (s1) by sending a query to family space owners and recreation service providers. After selecting the destination, the customer contacts the flight suppliers for the available flight information. Then the customer can book a flight (s2). After that, the customer will book the taxi towards the hotel (S3). Following confirmation, an online payment services transaction be initiated (eg Paypal) (S4).

Figure 3.3: A prototype application for holiday family village planning.

In this experiment, we exploit the dataset#2 proposed by Zheng [87]. We have randomly chosen 5 candidates of web services for every abstract task from this dataset. Each candidate has two characteristics: $QoS1$ and $QoS2$, which relate to response time and throughput, respectively. the values of QoS for every candidate service are shown in Table 3.2. In order to evaluate our approach, we vary the number of failure from one failure, tow failures, and three failures. The algorithm is performed for each case to select replacement services for failed services that fulfill global QoS criteria.

Table 3.2: QoS values from WSdream Dataset [87]

Abstract service	Candidate service	Response time	Throughput
	S_{11}	0.391	0.5671875
	S_{12}	0.644	0.5636646
S_1	S_{13}	0.68	0.53382355
	S_{14}	0.394	0.9213198
	S_{15}	0.389	0.93316
	S_{21}	0.394	0.9213198
	S_{22}	0.401	0.9052369
S_2	S_{23}	0.406	0.8940887
	S_{24}	0.419	0.86634845
	S_{25}	0.447	0.81208056
	S_{31}	0.485	0.7484536
	S_{32}	0.491	0.7393075
S_3	S_{33}	0.483	0.7515528
	S_{34}	0.535	0.67850465
	S_{35}	0.483	0.7515528
	S_{41}	0.484	0.75
	S_{42}	0.485	0.7484536
S_4	S_{43}	0.336	0.92862684
	S_{44}	0.341	1.3139408
	S_{45}	0.353	0.18197021

Figure 3.4 illustrates the results obtained by executing the SRL-ACO algorithm in the various failure scenarios. We have calculated that the average value of fitness after each failure in different scenarios, and in the same way we calculated the value of fitness after doing the substitutes suggested by the algorithm, based on the results shown in Figure 3.4. We notice that the proposed approach not only provides replacements for failed services but also improves the Quality of service constraints. For example, in the case of two services that failed, the fitness value after execution of our algorithm in the process of dynamic reconfiguration is improved by 25%, and in the case of three services that failed, the algorithm also provided a better fitness value of 12%.

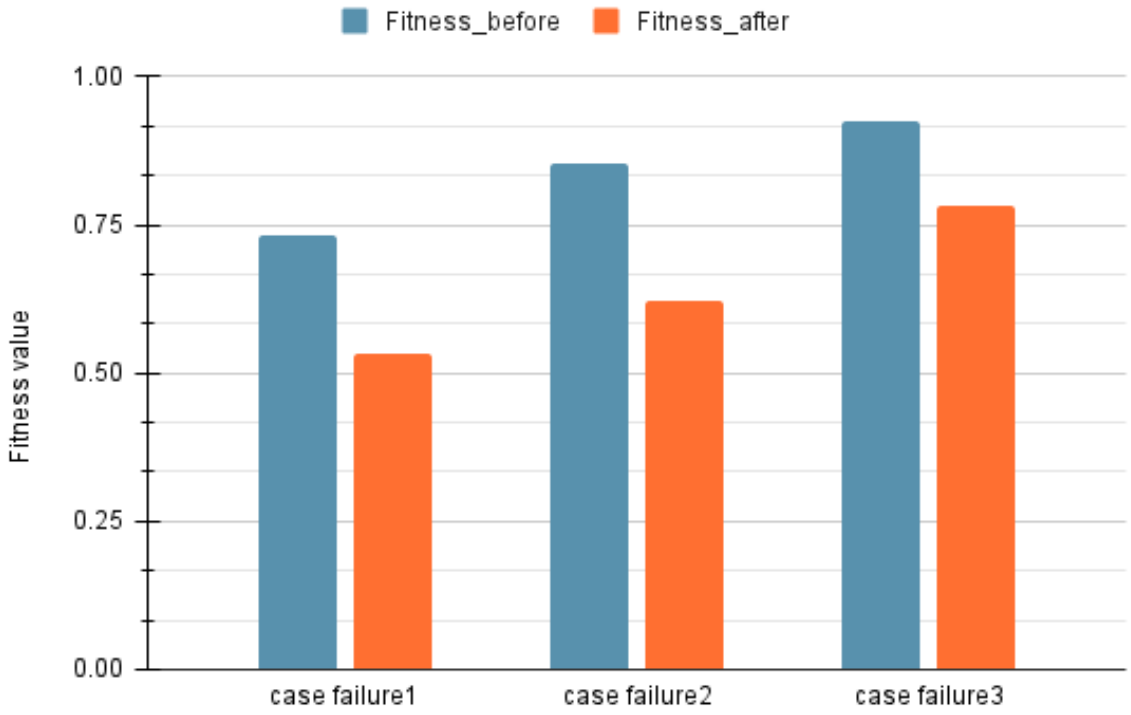


Figure 3.4: The results obtained by executing the SRL-ACO algorithm in the various failure scenarios.

3.5.2 Simulation of the proposed approach

In this part, we present a simulation for each failed service using a large-scale of candidates services. This simulation highlights the contribution of our approach to dynamic reconfiguration and its scalability in the case of several failures. Based on the fitness solutions, we present a comparative study on various simulation scenarios regarding the PSO algorithm. We consider the different simulation parameters of the algorithm in following table 3.3:

Table 3.3: Simulation parameters

Simulation parameters	
Sizes of composite services.	{5, 10, 20, 40, 50}
Set of candidates service in each failing service	{50, 100, 200, 400, 500}
Rat of failure	{10%, 20%, 50%, 80%}
QoS parameters	Response time and Throughput
Dataset	WSDream[87]

To study the effect of failure increase rate and candidate services on the dynamic reconfiguration process, we calculated the fitness values of the obtained solution with and without reconfiguration using SLR-ACO and PSO.

Impact of increasing the failure rate on dynamic reconfiguration process

Figure 3.5 illustrates a comparison of the fitness of the SLR-ACO and PSO algorithms. We have carried out experiments on a composite service of 10 abstract services and 50 candidate services for each failing service, with varied failure rate from 10% to 80%. We have observed for the two algorithms, that, increasing the failure rate affects the fitness result obtained after using reconfiguration service when compared to the fitness result before using reconfiguration service. We conclude that, the higher the failure rate, the higher the fitness result with a better result for the SLR-ACO.

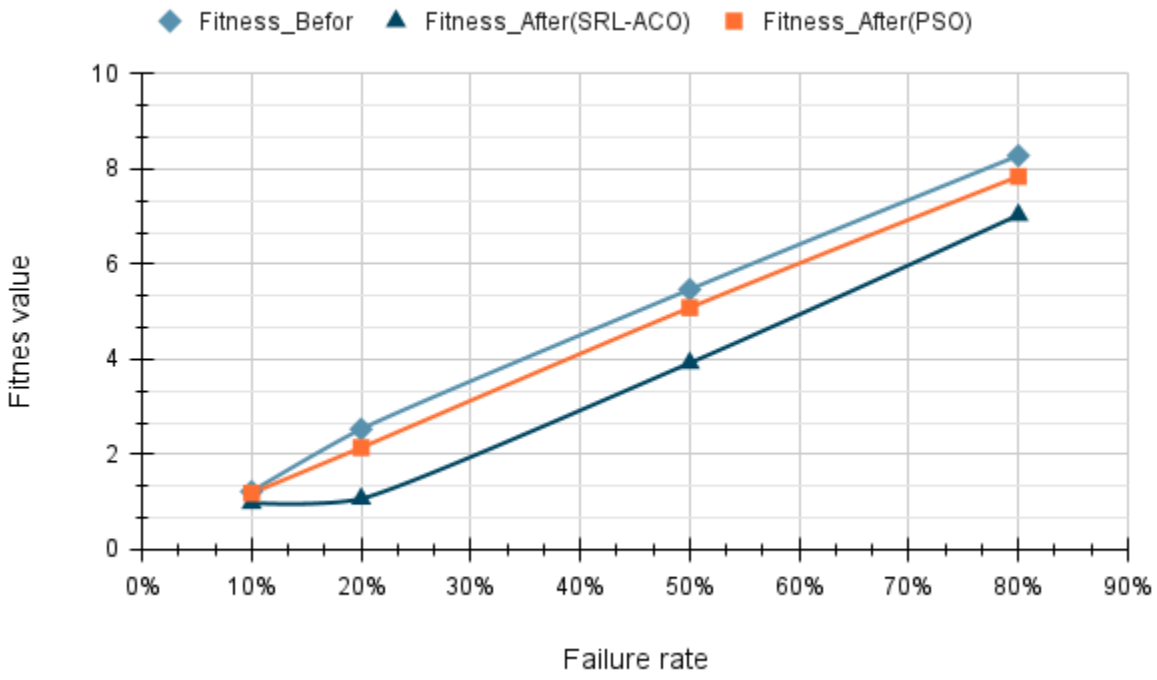


Figure 3.5: Impact of increasing the failure rate on dynamic reconfiguration process.

Impact of increasing the number of candidate services on dynamic reconfiguration process

To study the effect of the increasing number of candidate services on dynamic reconfiguration, we carried out experiments on a composite service of 10 abstract services and a failure rate of 50% with a varies of the number of candidate services from 50 to 500.
Figure 3.6 shows the comparison of the fitness value of the composition before reconfiguration, and after reconfiguration. We can see that the fitness value for both algorithms is optimized, with the SRL-ACO having a higher value. We conclude that an increase in the number of candidate services does not affect the fitness result of both algorithms.

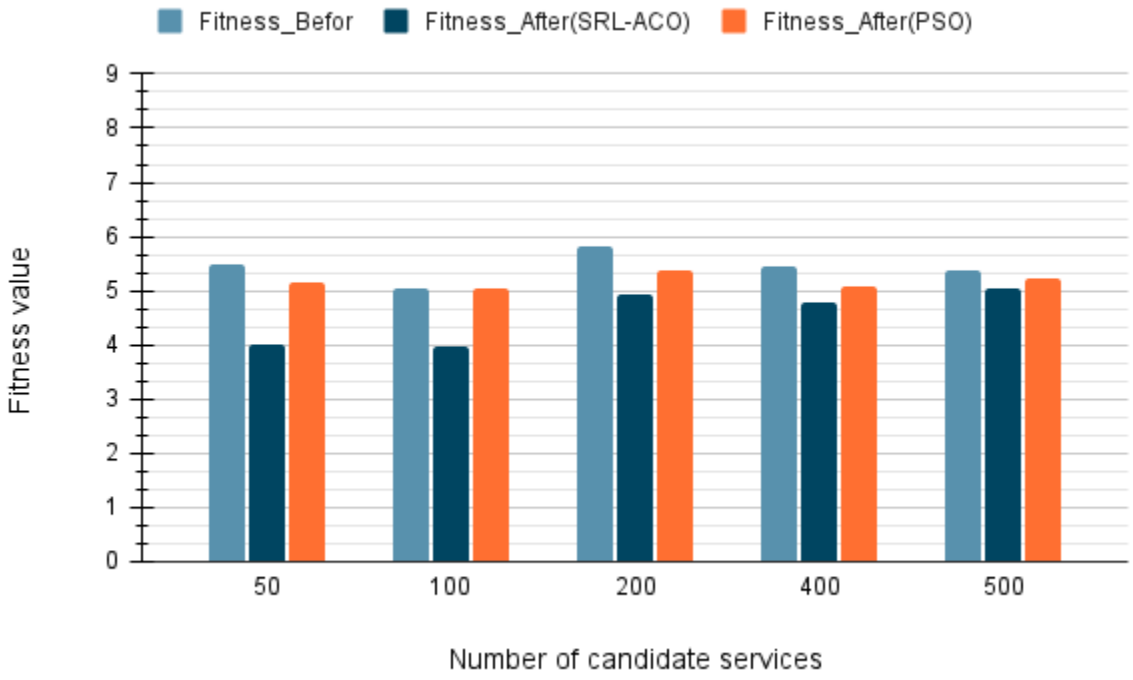


Figure 3.6: Impact of increasing the number of candidate services on dynamic reconfiguration process.

3.6 Conclusion

we have presented, in this chapter, a novel approach to dynamic reconfiguration in the presence of failed services, using an ACO-based swarm reinforcement learning method. Multiple agents, in swarm reinforcement learning techniques, learn within two methods of learning. In the first one, each agent learns independently using the traditional reinforcement learning method (Q-learning). In the second one, agents exchange information among them using population-based methods. In learning through exchanging information, a Q-value update method must be efficient to get an appropriate strategy in a shorter learning time.

ACO is widely utilized in many areas, due to its positive feedback, parallelism, and easy combination of other algorithms. The suggested approach involves the regular exchange of information between all agents. These latter use pheromone-Q values for updating their values. Know that pheromone-Q values have been inspired by the reality existing in the logic of ants.

We have proposed and demonstrated an ant colony optimization-based swarm reinforcement learning method for the dynamic reconfiguration problem in composite web services. The experimental findings show that the proposed approach that uses Pheromone-Q values is efficient in the dynamic reconfiguration problem.

Chapter 4

Management of dynamic reconfiguration of web services using QoS prediction

4.1 Introduction

Web services, as software applications exposed on the web, are invoked by users to meet their needs. Responding to a user request often requires the composition of several web services, resulting in a composite web service (CWS). The invocation of a pre-composed web service is subject to various unexpected changes that can affect the quality of service (QoS), besides the functionality and the environment in which the application operates. Therefore, a dynamic reconfiguration of the composite web service is essential to ensure that it can adapt to various unanticipated QoS changes. However, using a reactive method for reconfiguration and service continuity takes a long time [10] and affects the performance [11], from selecting the best service to reconfiguring the entire process. Hence, a proactive or predictive method for impending problems in the QoS is one of the most critical suggested solutions in the dynamic reconfiguration. In the proactive approach, preventing the invocation of web services that are subject to imminent degradation in QoS is a major challenge; this is evident, firstly, in the case of several degraded web services during the workflow process, where it becomes difficult to apply the dynamic reconfiguration service, starting from selecting the appropriate service until the completion of the workflow. Secondly, after performing the dynamic reconfiguration of the degraded web services and replacing them with other services similar to those in the functionality, the latter may experience an imminent degradation in QoS, which requires the request of the dynamic reconfiguration service again or several times. One of these situations may impede the proper functioning of the CWS process and consequently result in a dissatisfied user. Several efforts have been made to cover some of the stated challenges. However, enhancing the dynamic reconfiguration of a composite web service in a dynamic environment remains a major challenge. In this regard, and to address the above challenges, a prediction of imminent degradation in QoS to prevent partner web services' invocation with degraded QoS values is proposed in this chapter. We propose an improved HMM-based agent to predict the imminent degradation of QoS of web services. The prediction accuracy of HMM is improved by using a hybrid meta-heuristic algorithm (SFLA–PSO). We propose to recognize the potential states that web services could be in throughout their mission using QoS his-

torical data and the k-means clustering method. To the best of our knowledge, this is the first time that the QoS state has been used for QoS prediction. The contributions of this chapter are fourfold, as follows: 1. An improved dynamic reconfiguration method which avoids invoking degraded web services by predicting QoS for the candidate web service; 2. A new hybrid algorithm of the SFLA–PSO and Baum–Welch algorithms to improve the HMM initial emission matrix; 3. The QoS prediction problem is reformulated into a QoS state prediction problem, where we consider QoS states rather than QoS values and consider that a QoS state is composed of several attributes; 4. The proposed approach can predict the QoS according to each user's preferences and can predict the state of similar services without the need for their data (QoS).

4.2 Overview of proposed approaches

State-of-the-art web service QoS prediction approaches have been widely studied in the literature. Generally, the proposed methods can be classified into two main categories: approaches for recommendation and selection services [7, 53, 88, 89, 90, 91, 92], and approaches for the monitoring and dynamic adaptation of web services [10, 49, 53, 93]. The first category includes many studies based on the collaborative filtering (CF) algorithm to enable the recommendation and selection of the optimal web service by predicting the unknown QoS values based on historical user data [54, 94, 95, 96]. The filtering algorithm can be divided into two kinds [97]: memory-based CF and model-based CF. A personalized online performance prediction framework based on the past usage experience from different users is proposed [54], using collaborative filtering approaches to enable the optimal web service selection. A collaborative QoS prediction approach based on the adaptive matrix factorization is proposed [53] to perform online QoS prediction for candidate services. They used a reactive adaptation when encountering a change. Song et al. [98] proposed a method for the personalized QoS prediction of dynamic web services to predict the QoS value of the requested service, where it finds the users and services similar to the target user and target service, respectively. Xiong et al. [94] proposed a deep hybrid collaborative filtering based matrix factorization approach to improve web service recommendation.

Waseem et al. [99] proposed an approach for web service selection based on response time, in which the web service is predicted using the HMM model. The author restricts his work to predicting the web service's behaviour in terms of response time.

Our contribution belongs to the second category in which several approaches have been proposed. Kahlon et al. [10] proposed a hybrid approach distributed between the service client and the service provider to manage a situation when QoS values degrade. The approach is based on the publish/subscribe mechanism, on the provider side, to communicate the monitoring data to all the clients, and on the client-side, to inform the client of the QoS degradation. It is also based on the exponentially weighted moving average (EWMA) to predict the degradation of QoS value. However, this work lacks a discussion of the prediction accuracy problem. Aschoff et al. [100] proposed a ParProAdapt framework that detects the need for changes in the web service composition based on the function approximation and the failure of spatial correlation techniques. In this work, the expected service operation of QoS values was modeled using (EWMA). The authors did not consider the users' context when adapting instances of the same composition. Gao et al. [101] proposed the cloud-edge based dynamic reconfiguration to service workflow. The authors considered the value and cost attributes of the service to evaluate the stability and service invocation, respectively. They used the long short-term memory (LSTM) neural network to predict the stability of services. This study is limited to the mobile environment, which is distinguished by limited resource storage and users that move often. Wang et al. [77] proposed a service composition approach based on QoS prediction and reinforcement learning. Specifically, they used a recurrent neural network to predict the QoS. Chen et al. [102] designed a framework to evaluate the survivability of SOA-based applications using the HMM model. The main idea of the proposed framework evolves around monitoring activities based on service logs or run time statistics provided by the service provider. However, their approach is contingent on the service provider; also, the author did not discuss the various hidden states or probabilistic insight of remote web services. Moustafa et al. [59] proposed a proactive approach to web services composition (WSC), where used Q-learning as a Reinforcement Learning technique to adjust to dynamic change in WSC. the Authors represented the WSC process based on the Markov Decision Process (MDP).

By investigating the web service execution log's historical data the approach can observe the WSC adaptation change proactively. Nonetheless, according to [9, 103], the proactive approach can anticipate reconfiguration before the problem occurs. Several methods have been proposed in the literature that used a proactive prediction-based approach [4, 48, 54] for web service composition. Most of these methods predict events that may never occur, which increases the lost time and reduces service reliability [104]. So the use of statistical methods capable of predicting the imminent change in the QoS is crucial. Among these methods is the (HMM) Hidden Markov Model.

HMM has been used in many fields, including industrial applications; a few of the studies on (SOA) Service Oriented Architecture have touched upon this kind of prediction; perhaps the most important of them is proposed in [99, 105]. The authors of [99] used the HMM to predict the web service's behavior in terms of response time and selected the optimal web service at run time. The authors of [105] have proposed a model for predicting the QoS-satisfied capability of composited components based on HMM. HMM showed promising results in QoS prediction and should be investigated more deeply. The framework we propose in this chapter differs from the methods mentioned above in that it supports prediction based on user preferences. It also supports predicting the state of similar services without requiring their data. More precisely, the state of the candidate services can be predicted based on the active service's QoS. It also allows the study of the gradient in the degradation of the service state.

4.3 The Proposed Framework Architecture

The main role of the proposed framework (see Figure 4.1) is to manage the dynamic reconfiguration of composite web services. This is accomplished by improving the efficiency of dynamic reconfiguration and predicting the imminent degradation of the QoS of web services. A new service predictor that prevents the potential unsatisfied change in non-functional properties is proposed. It avoids invoking web services with degraded QoS values.

For the sake of simplicity, let us consider the following definitions:

- A web service is the main component of the workflow process; it is defined by its domain and QoS attributes;
- An Agent HMM is the agent responsible for the analysis;
- QoS attributes are: response time, throughput, reliability and availability .

The framework is comprised of four components: a workflow manager, a service predictor, service improvement and service reconfiguration detailed as follows:

4.3.1 Workflow Manager

The Workflow Manager is the main component of the proposed framework; it receives the addresses of the services used in the composition and then transfers them to the predictor service to predict the imminent degradation of QoS. If there is an undesirable change in the QoS of the invoked web service, the workflow manager invokes the improvement service to find similar services that improve the user's preferences and then inform the reconfiguration service to replace the degraded service.

4.3.2 Service Predictor

The service predictor is the module responsible for predicting impending QoS changes. The proposed processing is as follows: First, the predictor assigns an agent to each web service and the agent monitors the web service by collecting QoS information; second, the QoS information is processed using Box–Cox transformation and normalisation; third, it is clustered according to its current HMM state (which reflects its QoS). Finally, if an agent detects degradation in a web service QoS, it notifies the predictor service, which informs the workflow manager to prepare a substitution of a similar web service with the help of the service reconfiguration.

4.3.3 Service Improvement

The service improvement module fills the internal database from an external service register using a recommendation algorithm. It selects the service that satisfies the user's preferences and saves a set of service candidates corresponding to the concrete service of each composition. We discuss the recommendation algorithm in future work.

4.3.4 Service Reconfiguration

The service reconfiguration module aims to optimize the user's preferences and avoid the imminent QoS change. The workflow manager may receive a notification from the predictive module or the improvement module. The notification is received from the predictive module in the case of an imminent change in QoS and is received from the improvement module when a service that better meets users' needs is available. In both situations, the workflow manager sends a message to the reconfiguration module to perform an appropriate substitution using its internal database.

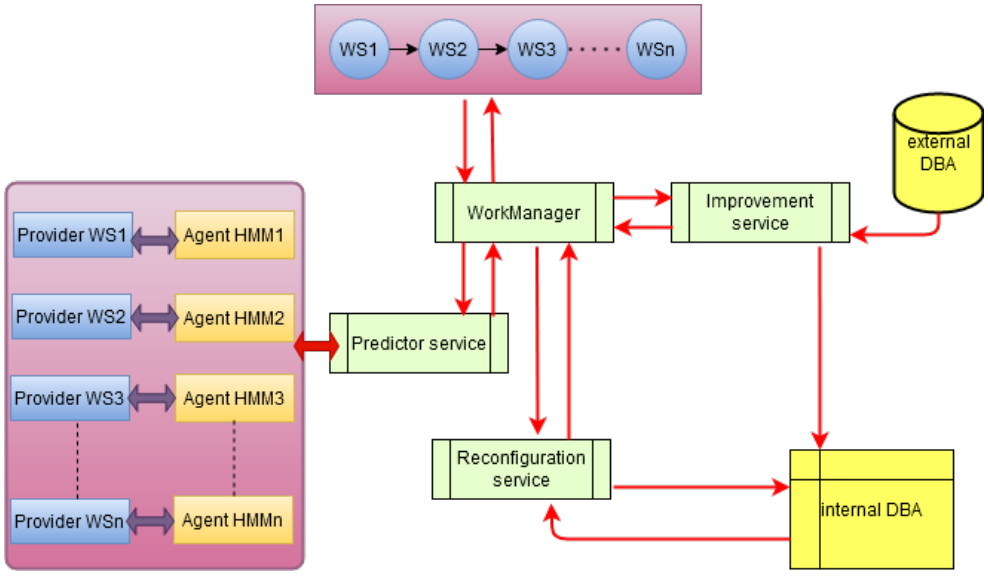


Figure 4.1: The proposed CWS framework architecture [13].

In Figure 4.2, using a UML sequence diagram, we illustrate an overview of our proposed approach to predict impending changes to the suggested web services and dynamic reconfiguration of composite web services. First, once the client invokes a composite web service to perform a task, the work manager informs the predictor of the selected web services' addresses. Second, the predictor contacts each web service provider through agents to obtain historical QoS data (e.g., response time, throughput, reliability and availability) for them. The agents are responsible for analyzing and processing historical QoS data by applying our proposed approach to predict the impending changes to the suggested web services. Thirdly, historical data for each web service are grouped into clusters to obtain all possible states that may happen to the web service. Fourthly, suppose the agent predicts the existence of a change in the state of a web service; in that case, the predictor is notified and, in turn, reports to the work manager. The latter requests the service reconfiguration to replace the web service the QoS of which is degraded with another one that has a better QoS. We represent the N states $(s_1, \ldots, s_n)$ for each web service (WS), where s_1 represents the normal web service state and s_n represents the faulty state; the states between the two previous states represent the progression in degradation from best state to faulty state.

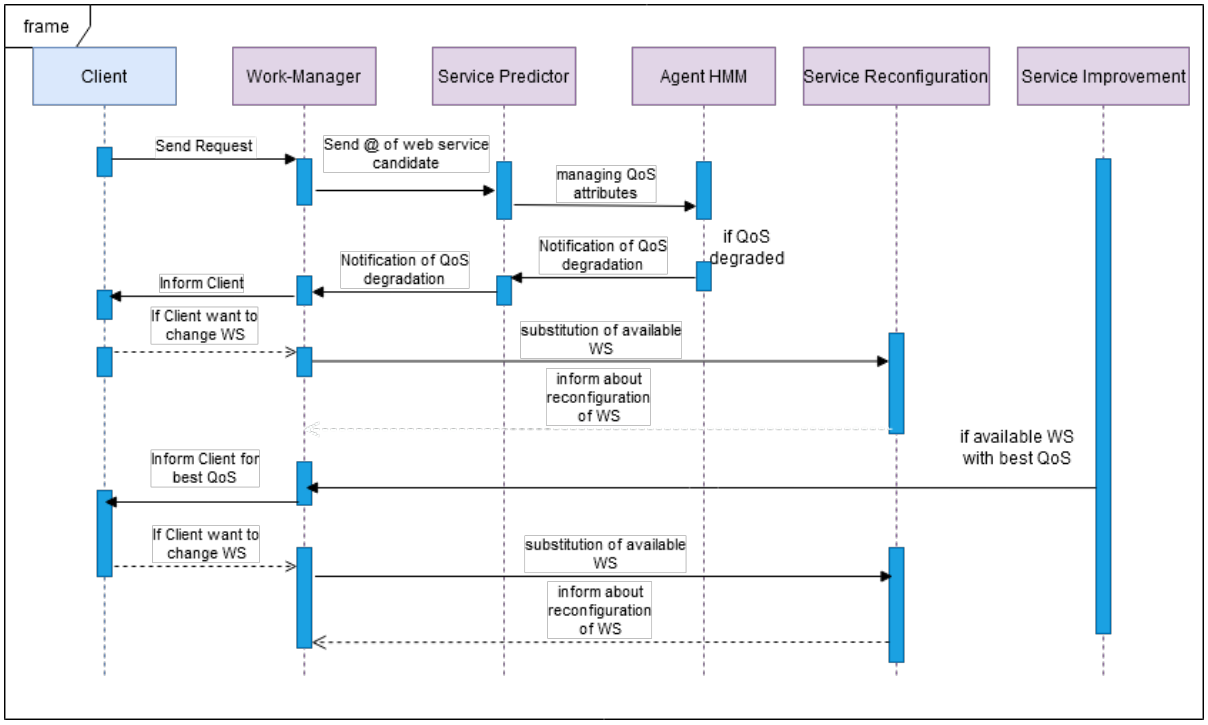


Figure 4.2: Framework execution process [13].

In all of the following, we focus on the predictor service, which contains the main contribution of our research.

4.4 A New Dynamic Reconfiguration

A new dynamic reconfiguration process is proposed. It is based on three main steps: First, pre-processing the QoS data to be vectorized and normalized; second, the QoS clustering service regroups the similar historical QoS data for each participating web service in the composition, to identify the possible N states that a web service could be in throughout their mission; third, an improved Hidden Markov Model is proposed to assure QoS prediction accuracy.

4.4.1 Pre-Processing QoS

The QoS attributes, such as response time and throughput, have different ranges; one attribute might overpower another, which negatively affects the efficiency of the clustering algorithm. Moreover, a real-world QoS dataset takes on highly skewed distributions with large variations [53, 106]. Therefore, it is necessary to use a method to stabilize the data variance and make the data more normally distributed while setting it in the same range, for example [0, 1]. In our research, we used a combination of the Box–Cox transform method [107] and a normalization method [108] for QoS data. The former was chosen for its suitability for distributions with outliers and extreme skews, and the latter was chosen for mapping a dataset at the same scale, which is presented as follows:

$$Norm(Qi) = \begin{cases} \frac{(\text{Box cox}(Q\text{i})) - \text{Qmin}}{\text{Qmax} - \text{Qmin}} & \text{if } \text{Qmax} - \text{Qmin} \neq 0 \\ 1 & \text{if } \text{Qmax} - \text{Qmin} = 0 \end{cases} \quad for\ a\ positive\ QoS\ attributes$$

$$Norm(Qi) = \begin{cases} \frac{\text{Qmax} - (\text{Box cox}(Q\text{i}))}{\text{Qmax} - \text{Qmin}} & \text{if } \text{Qmax} - \text{Qmin} \neq 0 \\ 1 & \text{if } \text{Qmax} - \text{Qmin} = 0 \end{cases} \quad for\ a\ negative\ QoS\ attribute, \tag{4.1}$$

where Q_{min} and Q_{max} are the minimal and the maximal QoS values respectively, and

$$Box\,\text{Cox}\,(Q_i) = \begin{cases} \frac{\left({Q_i}^{\theta} - 1\right)}{\theta} & \text{if } \theta \neq 0 \\ \log(Qi) & \text{if } \theta = 0 \end{cases} \tag{4.2}$$

In the relationship, (4.2) θ is the transformation parameter.

4.4.2 QoS K-Means Clustering

The main idea behind this step is to identify the possible states (N states) in which web services could be throughout their mission. Similar non-functional properties (QoS) are

grouped together for each web service participating in the composition, and N states are considered in this work to be discrete states inferred from the values of observed QoS, such as the response time, reliability, availability, and throughput. The K-means clustering algorithm is applied using a Euclidean distance scale to measure the similarity in clustering.

4.4.3 Improved Hidden Markov Model with SFLA-PSO Hybridization

The Hidden Markov Model is a time series model that is able to infer a sequence of hidden states from an observation sequence. It is defined by the set of parameters $\lambda = (A, B, \pi)$, where A is the transition probability matrix , B represents the observation model distributions and π is the set of initial probabilities. In a formal way, an HMM is a stochastic process which has the following elements:

1. A set of N-hidden states $S = \{S_1, S_2, \ldots, S_N\}$. Note that a state of the model at time t is denoted by $q_t \in S, 0 \leq t \leq T$, where T is the length of the observation sequence and q_t denotes the current state.

2. A transition probability matrix $A = \{a_{ij}\}$. a_{ij} is the probability of transition from a state S_i at time $t-1$ to another state S_j at time t

$$a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i), 1 \leq i, j \leq N. \tag{4.3}$$

3. An observation probability matrix denoted by $B = \{b_j(x_t)\}$ defined by the probability of emitting an observation x_t at time t given S_j

$$b_j(x_t) = P(x_t \mid q_t = S_j). \tag{4.4}$$

4. An initial state vector $\pi = \{\pi i\}$ is defined by the probability of beginning in the state S_i $(t = 0)$;

$$\pi_i = P(q_0 = S_i), 1 \leq i \leq N. \tag{4.5}$$

In this work, we redefine the QoS prediction model as follows:

Definition 4.1 (**QoS state**). *We consider the QoS state at time t as the hidden state of an HMM denoted by s_t and a random variable taking values in $\{S_1, \ldots, S_n\}$ (i.e., good ,medium, ..., and bad). It represents the QoS state of web service at time instant t. The probability of transition from a state S_i at time $t-1$ to another state S_j at time t is*

$$a_{ij} = P(q_t = S_j \mid q_{t-1} = S_i), 1 \leq i, j \leq N, \tag{4.6}$$

where $S \in \{$good, medium, ..., and bad$\}$.

Definition 4.2 (**QoS observations**). *A sequence of observations $X = \{x_0, x_1, \ldots, x_t, \ldots, x_T\}$, where x_t is the QoS information observed at time t is defined by the f-attributes, $x_i = \{f_1, f_2, \ldots, f_m, \ldots, f_M\}$. For instance, for $M = 4$, x_i may be in the set $\{$response time, reliability, availability and throughput$\}$. The probability of emitting an observation x_t at time t given S_j is*

$$b_j(x_t) = P(x_t \mid q_t = S_j). \tag{4.7}$$

Definition 4.3 (**QoS initial state vector**). *We assume that the initial state vector for each web service in the CWS begins in the appropriate QoS.*

Definition 4.4 (**The HMM decoding problem for QoS prediction**). *Our approach predicts the next state q_{t+1} for each web service in the CWS, given the model λ^* re-estimated at time (t) corresponding to the sequence of observations $X = \{x_0, x_1, \ldots, x_t\}$, where x_t is the QoS information observed at time t. The Baum–Welch algorithm [109]—based on the forward (α) and backward (β) iterative algorithm—is used to re-estimate the model of HMM, where λ^* estimate the model most representative of the sequence of observations X at time t. The Baum–Welch algorithm is not discussed in this chapter, for further information refer to [109]. We predict the state q_{t+1} as follows:*

$$q_{t+1} = \underset{1 \leq j \leq N}{\arg\max} \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right), \quad 1 \leq i, j \leq N, \tag{4.8}$$

with $\alpha_0(i) = \pi_i b_i(x_0), 0 \leq t \leq T$ *and* N *is the number of degradation states in which a service can appear.*

Baum–Welch Limits for HMM Parameter Learning Applied to QoS Prediction

The Baum–Welch (BW) algorithm is used for estimating the parameters of the HMM. However, the most critical limitation is that the algorithm efficiency of BW depends on the initial value of the HMM parameters, in particular, the initial value of the emission matrix [110] and the possibility of its convergence to the local optima [111]. Thus, proposing a method that improves the initial value of the emission matrix is necessary.

The hybridization of methods such as the Shuffled Frog Leaping Algorithm (SFLA) [112] and Particle Swarm Optimization (PSO) [113] with the Baum–Welch algorithm is a good alternative for achieving this goal. We chose the SFLA because it has a robust global optimization capacity [5, 114], and the PSO for its rapid convergence and robustness [113, 115].

A New Method for Selecting the Initial Emission Matrix

Finding the best emission matrix is an optimization issue that requires optimizing a fitness function. We consider the fitness function as the sum of the product of the forward and backward variables of the BW algorithm, for which the probability of generating the sequence of QoS information $(x_0, x_1, x_2, \ldots, x_t)$ should be maximized. The fitness function is formulated in Equation (4.9):

$$f(\lambda) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(l), \tag{4.9}$$

where $\alpha(i)$ and $\beta(i)$ are the forward and backward variables, respectively. $f(\lambda^*)$ are the optimal HMM parameters for the successive QoS information given at time t.

In our hybridization of SFLA and PSO methods, we assume that the initial population of the emission matrix B_0 (initial emission matrix) is created with a random function, and

it is sorted in descending order according to the value of the fitness function. Then the population is divided into K sub-populations after achieving the best global solution, and each worst solution B_0 in each sub-population learns from the best solution of their sub-population using the PSO algorithm. After that, all sub-populations are combined and split into new sub-populations. The process terminates when the final conditions are satisfied. The proposed algorithm of SFLA–PSO for an improved Baum-Welch initialization is detailed in Algorithm 4.1 as follows:

Algorithm 4.1 SFLA–PSO algorithm for an enhanced Baum–Welch initialization [13]

Input : initial transition probability matrix: A_0;
initial state vector π_0
Initialize the coefficients c_1, c_2, r_1, r_2;
Initialize the $Nb - iteration$

Output: estimate model $\lambda = (A, B, \pi)$;

begin
  Generate random population of P solution of initial emission matrix B_0 (b-solution);
  Calculate the fitness function of each B_0 in P;
  Sort the population descending order of their fitness function values;
  Get the *gbest_solution* of population P;
  while *Nb-iteration* **do**
    divide the population into k sub-populations;
    for *each sub-population* **do**
      get the *best_solution* and the *worst_solution* of sub-population;
      Update the learning rate z using
      $z^{i+1} = c_1 r_1^i z^i + c_2 r_2^i \left(best\_solution^i - worst\_solution^i\right)$
      Update the *worst_solution* of each B_0 using
      $worst\_solution^{i+1} = worst\_solution^i + z^{i+1}$
    end
    Shuffle the sub-populations;
    calculate the fitness function of each B_0 in P using baum-welch algorithm;
    Sort the population in descending order of their fitness function values;
    Update the global best solution *gbest_solution*;
  end
  return $\lambda^* = (A, B, \pi)$.
end

The PSO adaptation for the Baum–Welch algorithm considers the following concepts: B_0 is considered as a particle in the swarm, and the PSO global best particle is selected according to the SFLA–PSO global best solution. The *best_solution* and the *worst_solution* are considered respectively as the best solution and the worst solution in each sub-population.

The proposed algorithm pursues the following steps: First, an initial value is given to the transition matrix A_0 and initial state vector π_0, as well as the initial values of the learning parameters $c1$ and $c2$ associated to the components' learning rate and the random values $r1$ and $r2$, which are in the range [0,1] (input). Second, the population P associated with the emission matrix B_0 is generated randomly. Then, the fitness function is calculated for each B_0 in P using the Baum–Welch algorithm. Next, B_0 is sorted in descending order according to their fitness function values and the global best value is gated (lines 1 to 5). In the next step, the emission matrix population is divided into k sub-populations and, for each sub-population, the worst solution is optimized by using the worst and best solutions in the sub-population (lines 6 to 14). Then, in lines 15 to 18 the grouping is performed again, calculating the fitness function for each B_0 in P, sorting B_0 in descending order according to their fitness function values and updating the global best value. By repeating this process for a defined number of iterations, the algorithm converges to the best B_0 solution.

4.5 Experiments and Evaluation

4.5.1 Dataset

Experiments were performed using data from the WSDream dataset #3 because it is suitable for large experiments where it counts a total number of 4500 web services. These web services were invoked by 142 users in 22 different regions of the world in 64 different time slots with a 15-minute interval, in which each real web service was invoked 142×64 times.

4.5.2 Experiments Settings

The proposed approach was simulated using both JADE (Java Agent DEvelopment framework) for implementation and Python for data pre-processing on a 64-bit windows 10 with core intel(R) i5-3450 processor and 4.00 GB of memory.

Blocked cross-validation was used to evaluate the prediction accuracy. We used data from 80 different users over 64 different time slots and mapped into five subsets of data arranged in chronological order. Each of these subsets of data contains a total of 1024 measurable pairwise values of response time and throughput. These five sub datasets are again divided into two parts, one containing 80% of the data as a training dataset and the other containing 20% of the data as a test dataset. To the best of our knowledge, the blocked cross-validation has never been used to analyze the WSdream dataset in the field of QoS prediction for web services.

Table 4.1 shows the parameters used for SFLA–PSO and Box–Cox transformation.

Table 4.1: Table of parameters

SFLA-PSO		BoxCox transformation
c1=c2=2.0	acceleration coefficients	$\alpha = (0.007-, 0.05-)$
P=200	population size	
m=10	number of sub-population	
n=20	population size in sub-population	
Maxiter=200	terminated condition	

4.5.3 Evaluation Metrics

Experiments were conducted in terms of prediction accuracy to prove the effectiveness of our proposed approach. To this extent, we used two different metrics to evaluate the clustering algorithm and the QoS prediction method.

Clustering Algorithm Evaluation Metric

To evaluate the clustering algorithm, we used the Silhouette score [106]; this score is commonly used as a synthetic indicator to evaluate the general quality of clustering, as well as to determine the appropriate number of clusters. This index is based on the so-called silhouette coefficient, which can be expressed according to Equation (4.10).

$$S(\mathbf{x}) = \frac{b(\mathbf{x}) - a(\mathbf{x})}{\max(a(\mathbf{x}), b(\mathbf{x}))}, \tag{4.10}$$

where $a(x)$ is the average dissimilarity of the xth point to all other points in the same cluster and $b(x)$ is the average dissimilarity of the xth point with all objects in the closest cluster. This measure has a range of $[-1, 1]$. In our approach, we selected the number of clusters that returns the highest average; a score of 1 means the clustering algorithm is appropriate.

QoS Prediction Evaluation Metric

We evaluated the prediction accuracy of our approach using two metrics, including MAES (Mean Absolute Error State) and RMSES (Root Mean Square Error State). A smaller MAES means a higher prediction accuracy. MAES is computed according to formula (4.11).

$$\text{MAES} = \frac{1}{T} \sum_{i \in N} (1 - \text{ Probstate }_i), \tag{4.11}$$

where Probstate_i is the probability that a web service is predicted to be in state i, even if it is originally in this state, T is the total number of test cases, and N is the number of states. RMSES represents the Root Mean Square Error State, and is computed according to formula (4.12).

$$RMSES = \sqrt{\frac{\sum_{i \in N} (1 - \text{ Probstate }_i)^2}{T}}. \tag{4.12}$$

4.5.4 The Clustering Algorithm Evaluation

To evaluate the k-means clustering algorithm on the WSDream3, we ran this algorithm for a different number of clusters (k) and then plotted the average silhouette score (ten independent runs). The clustering result for the WSDream dataset is shown in Figure 4.3. As observed in this figure, the number of clusters that maximizes the silhouette coefficient is $K = 3$.

Figure 4.3: The number of clusters.

4.5.5 Evaluation of the QoS Prediction Algorithm

We conducted three kinds of experiments: the prediction algorithm evaluation, the success prediction ratio and the failure prediction ratio.

Accuracy Evaluation of the Prediction Algorithm

We selected 40 web services randomly and computed the average of the prediction algorithm for 10 independent runs. Results are depicted in Figure 4.4. As can be seen, even for highly changing web services, such as web service 6 in Figure 4.4(a), the algorithm achieves an accuracy greater than 75%. For less changing web services, such as web service 17 in Figure 4.4(b) , the algorithm achieves an accuracy equal to 90%. Moreover, our formulation of the QoS prediction problem as a QoS state prediction problem further enhances the algorithm's prediction accuracy. From this, we conclude that the dynamic change in the QoS attributes of web services has no effect on the performance of our approach.

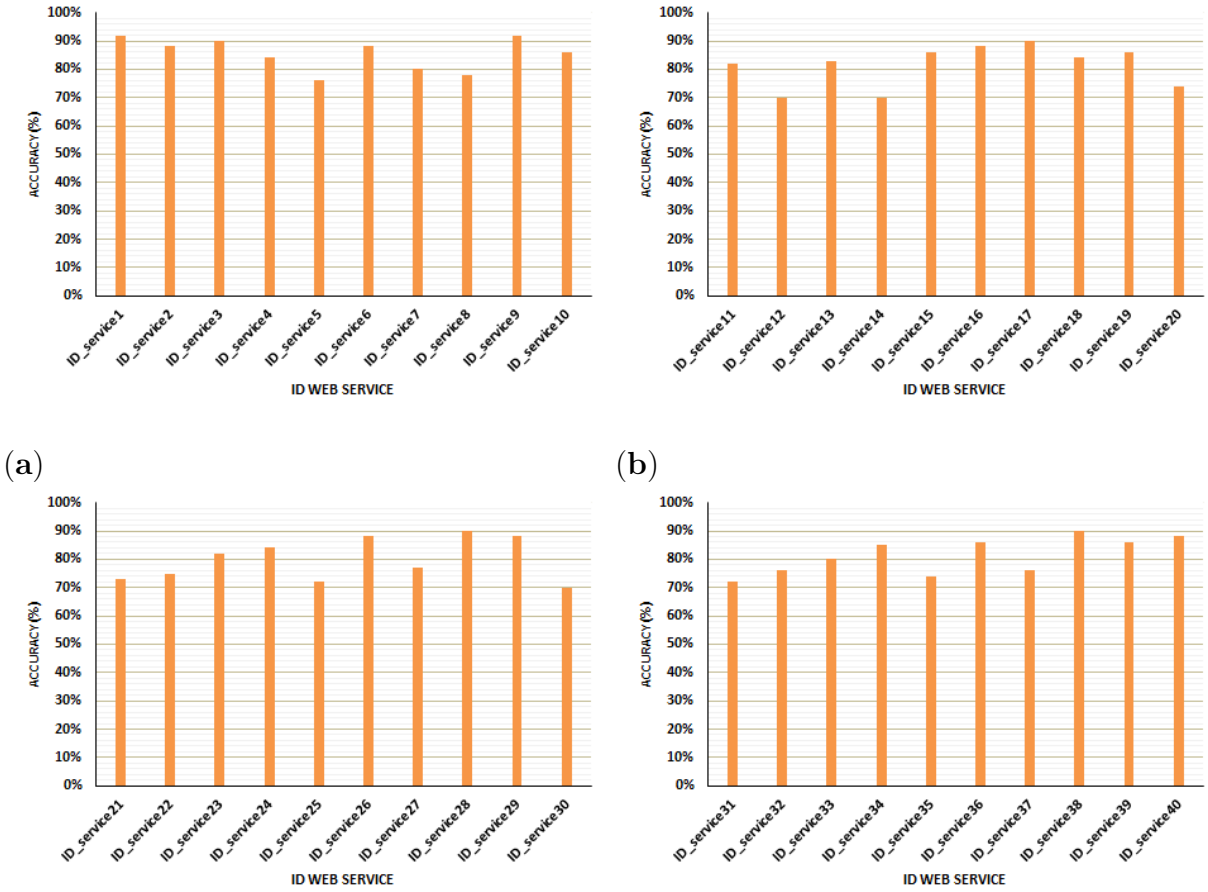


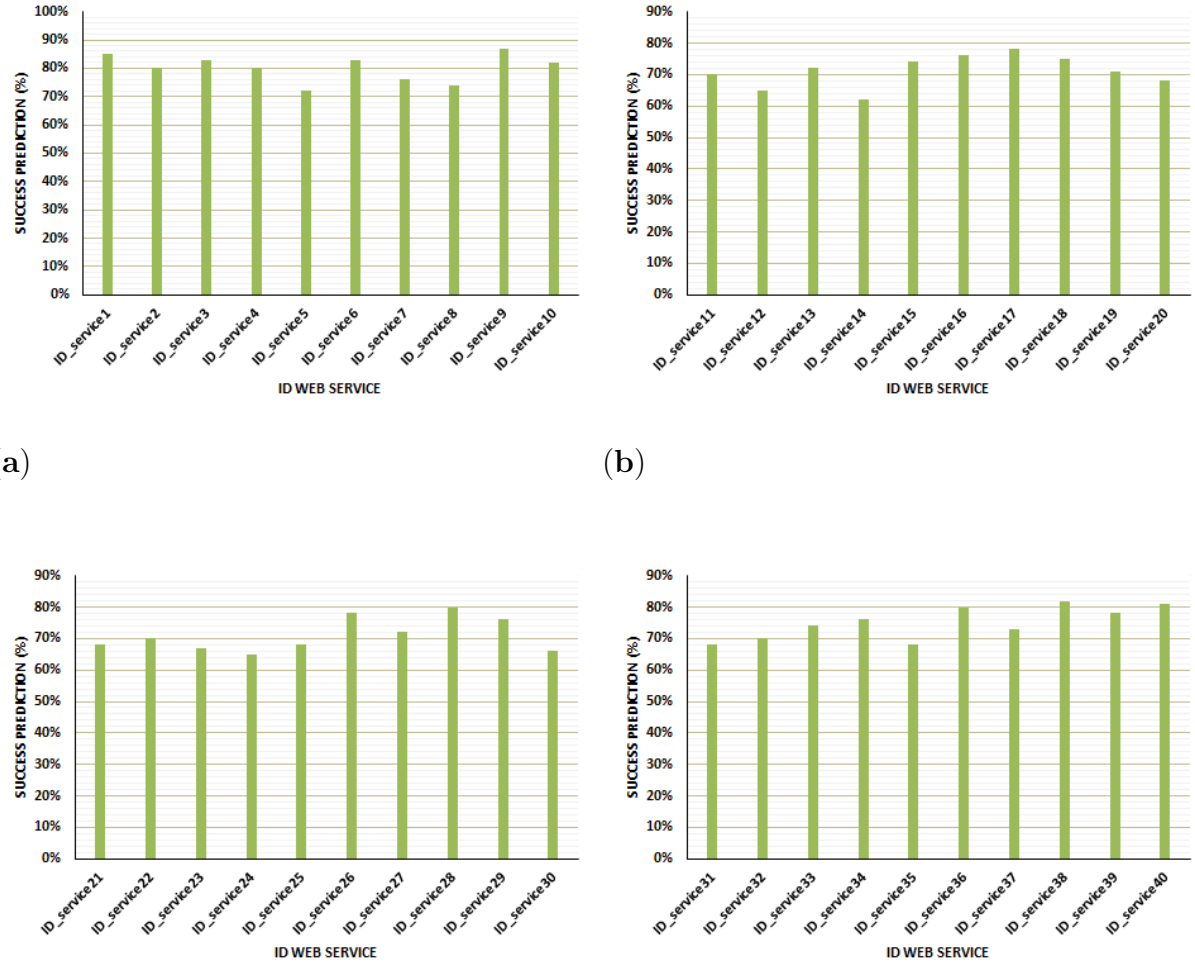
Figure 4.4: The prediction algorithm evaluation results in terms of accuracy [13].

Success Prediction Ratio

The success prediction ratio [116] computes the number of a successful state prediction divided by the total number of performed predictions as formulated in Equation (4.13). Let us remember that a state represents a web service's quality at a time instant t. We computed this ratio for forty different web services, and the obtained results are depicted in Figure 4.4. On average, a successful prediction ratio more than 75% was attained as

shown in Figures4.4(a), 4.4(b) and 4.4(d)

$$SuccessRatio = \frac{Number\ of\ successful\ predictions}{Number\ of\ predictions}. \tag{4.13}$$



(**a**) (**b**)

(**c**) (**d**)

Figure 4.4: The prediction algorithm evaluation results in terms of success prediction ratio [13].

Failure Prediction Ratio

The failure prediction ratio [116] computes the number of incorrect state predictions divided by the total number of performed predictions as formulated in Equation (4.14). We computed this ratio for forty different web services, and the obtained results are depicted in Figure 4.5. On average, a failure prediction ratio of less than 25% was attained as shown in Figures4.5(a) and 4.5(d)

$$FailureRatio = \frac{Number\ of\ failed\ predictions}{Number\ of\ predictions}. \tag{4.14}$$

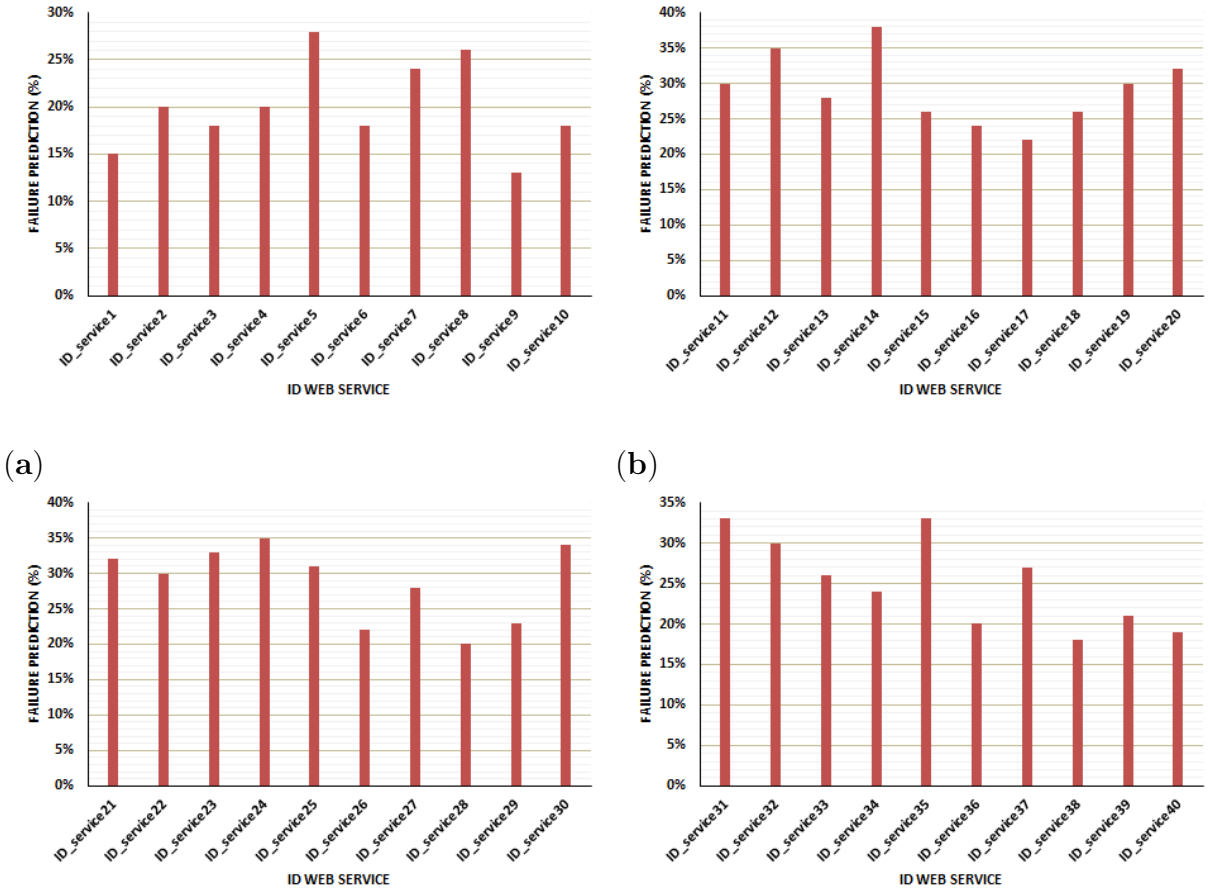


Figure 4.5: The prediction algorithm evaluation results in terms of failure prediction ratio [13].

4.5.6 Accuracy Evaluation

To investigate the impact of the proposed PSO–SFLA hybridization for the Baum–Welch algorithm on the prediction accuracy, we compared two variants of our proposed approach. The first variant is called the 'Improved HMM based Agent' and denotes the method proposed in Section 4.4.3. The second variant uses traditional HMM using the Baum–

Welch algorithm and is denoted by 'HMM based Agent'. Table 4.2 shows the comparative results of the two discussed variants in terms of MAES and RMSES. The obtained results show that our proposed PSO–SFLA hybridization method for Baum–Welch improved the prediction accuracy on average by 7% according to MAES and by 8.8% according to RMSE. Considering the dynamics of the web service environment in the WSDream dataset, we can say that our approach produced remarkable results compared to the state-of-the-art relying on local data.

Table 4.2: HMM state prediction accuracy evaluation

	Improved HMM Based Agent		HMM Based Agent	
	MAES	RMSES	MAES	RMSES
data-set1	0.565	1.387	0.661	1.453
data-set2	0.507	1.262	0.620	1.434
data-set3	0.456	1.161	0.554	1.371
data-set4	0.416	1.123	0.435	1.131
data-set5	0.371	1.101	0.404	1.104
Avrage	**0.463**	**1.206**	0.534	1.298

4.5.7 Case Study

In this section, we evaluate the effectiveness of our prediction approach through a case study. As our interest is in QoS evaluation, we will ignore the potential issue of functional mismatch, as we randomly selected ten candidate web services for each abstract task from our dataset. Figure 4.6 depicts a prototype application for a COVID-19 test, ordered with a simplified workflow composed of four abstract tasks. Specifically, people who believe they have symptoms related to the COVID-19 virus start to request an online testing service that allows them to test for COVID-19 (questionnaire) and receive the results (S1). The patient can then book an appointment at one of the nearby COVID-19 (S2) testing centres, based on the patient locator service (S3). After confirmation, the ambulance or taxi service for the patient (S4) will be booked.

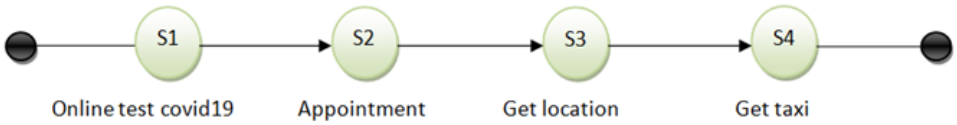


Figure 4.6: A prototype application for online test COVID-19.

We consider three application execution settings:

1. Ideal case: in this case, we assume that all the web services are chosen with the best QoS parameters to handle the client request. There is no degradation of QoS during the work process in all selecting web services. This is the optimal case; it acts as a benchmark.

2. Case of normal reactive approach: When the degradation occurs in a particular web service already chosen in the work process, the execution process interrupts, selecting the best candidate for the web service to be replaced with in the work process.

3. Case of our approach: We apply the IHMM-based Agent to QoS prediction. For each pre-selected web service (candidate ws), we use the prediction results in dataset 5. That is, each task dynamically chooses the best service according to current QoS predictions.

Assuming the maximum acceptable execution time is 1 s, where the execution time is defined as the time difference between a service user sending a request and receiving the corresponding response [87]. Figure 4.7 demonstrates the representative outcome of application execution time under the execution conditions quoted above. The execution in the ideal situation (ideal case), without any degradation of the quality of service, requires an average execution time of 0.28 s, and this is unlikely to happen due to the dynamic nature of the web services. For the second case (reactive solution), the results show an increase in the implementation time by 0.87 s, and this is due to the dynamic environment of the web service, where when a degradation in the quality of service occurs, it is replaced by another service that may also be subjected to degradation, which makes the execution

time more likely to increase with the repeated invocation of dynamic reconfiguration. In the case of our approach (third case), the results show a very noticeable improvement estimated to be 54% from 0.87 s to 0.4 s in execution time, and this refers to the optimal use of dynamic reconfiguration during and before execution.

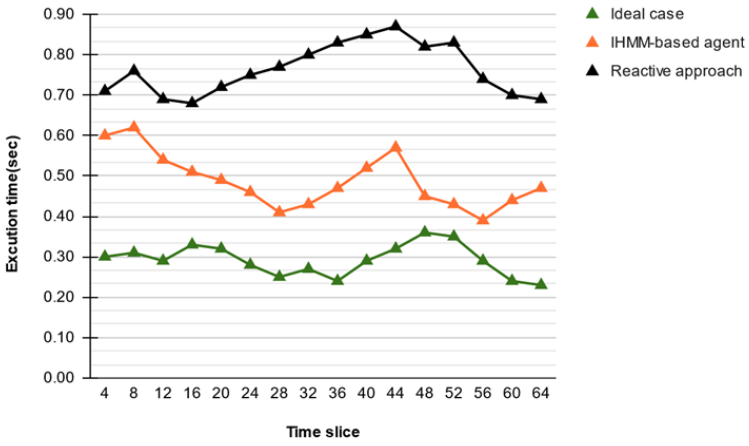


Figure 4.7: Comparison of three cases execution [13].

We compare the proposed framework and the works that are close to our approach, such as [10, 53], where we repeat experiments 50 times with our proposed approach, taking into account the results found in comparative research, as shown in Figure 4.8. We note that our approach reduced the use of dynamic reconfiguration (adaptation) by 9% more than the results obtained in [10] despite our use of a real dataset. Our approach also outperformed that reported in [53] by 5%.

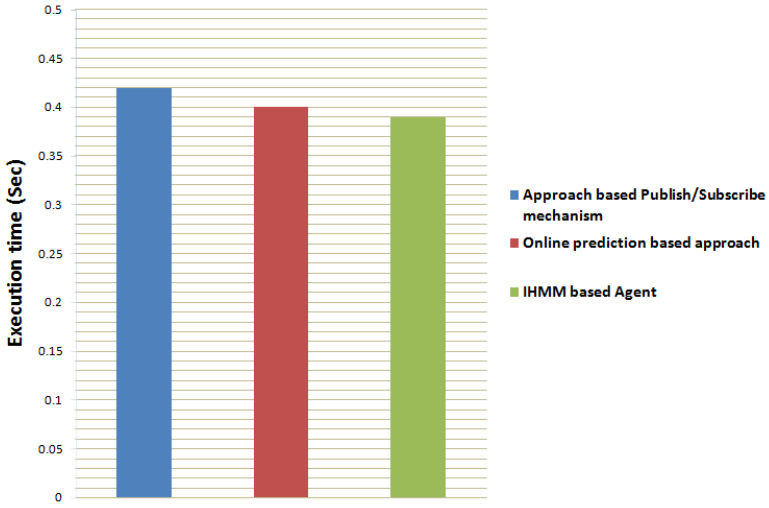


Figure 4.8: Execution time comparison [13].

4.6 Conclusions

In a service-oriented architecture environment, service reconfiguration provides the ability to substitute or adapt a disrupted web service or degraded QoS values during the workflow process. As the dynamism of the web services environment increases, the workflow process may be subject to multiple interruptions from the service reconfiguration due to fluctuations in the QoS values of web services. In this chapter, a method for predicting the QoS status of web services is proposed. This method uses the HMM to predict the QoS status, where the prediction accuracy of HMM is improved using a hybrid meta-heuristic algorithm (SFLA-PSO). The Box–Cox transform method is also used to stabilize the data variance and make the data more normally distributed. Moreover, the K-means clustering algorithm is used to obtain the possible states of QoS. The proposed method is evaluated using block cross-validation on a real dataset and, with different evaluation metrics, the proposed method achieves better prediction results compared to state-of-the-art methods. For future work, we plan to improve the clustering algorithm with further QoS attributes (such as reliability and availability). A further study will compare the accuracy and effi-

ciency of QoS status prediction and QoS values prediction.

Conclusion and Perspectives

Conclusion and Perspectives

Service-oriented architecture offers the ability to interconnect several web services to fulfill a user specific-requirement. In dynamic environments, the emergence of several unforeseen events can destabilize the composite web service and affect its quality. To deal with such an issue, the composite web service must be dynamically reconfigured. The core goal of this work discussed in this thesis is to offer a proactive approach to a dynamic reconfiguration of composite web service. Upon that, the research has three principal contributions:

Conducting a related study of dynamic reconfiguration approaches. The study begins by reviewing approximately twenty different approaches supporting the dynamic reconfiguration of composite web services. Then, exploring the characteristics of these approaches and classifying them according to this characteristics.

Proposing a new approach for dynamic reconfiguration when failed services take place, using an ACO-based swarm reinforcement learning method. The suggested approach involves the regular exchange of information between all agents. These latter use pheromone-Q values for updating their values. Know that pheromone-Q values have been inspired by the reality existing in the logic of ants.

Proposing a proactive approach to a dynamic reconfiguration of composite web services. This approach uses the HMM (Hidden Markov Model) to predict the QoS status of web services, where the prediction accuracy of HMM is improved using a hybrid meta-heuristic algorithm PSO (Particle Swarm Optimization) and SFLA (Shuffled Frog Leaping Algorithm).

Future Work

The work performed in this thesis covers one aspect of dynamic reconfiguration in a

service-oriented architecture. The proposed methods represent solutions focusing on substituting failed services or preventing the impending degradation of web services. Possible directions for future work are:

In chapter 3, more research will be tacked to enhance our second contribution by:

1. Making a comparative study for the learning through exchanging information methods supporting reinforcement learning to deal with the problem of dynamic reconfiguration of composite web services.
2. Studying the strategies to combine the structural and behavioral change in a dynamic reconfiguration of composite web services.
3. Providing mechanisms to ensure the composition's consistency before and after dynamic reconfiguration take place.

In chapter 4, we plan to:

1. Improve the clustering algorithm with further QoS attributes (such as reliability and availability).
2. Compare the accuracy and efficiency of QoS status prediction and QoS values prediction.

Bibliography

[1] Doaa H Elsayed, Eman S Nasr, M Alaa El Din, and Mervat H Gheith. A new hybrid approach using genetic algorithm and q-learning for qos-aware web service composition. In *International Conference on Advanced Intelligent Systems and Informatics*, pages 537–546. Springer, 2017.

[2] Jeff Kramer and Jeff Magee. Dynamic configuration for distributed systems. *IEEE Transactions on Software Engineering*, (4):424–436, 1985.

[3] Juraj Polakovic. *Architecture logicielle et outils pour systèmes d'exploitation reconfigurables*. PhD thesis, 2008.

[4] Rafael Aschoff and Andrea Zisman. Qos-driven proactive adaptation of service composition. In *International Conference on Service-Oriented Computing*, pages 421–435. Springer, 2011.

[5] Fouzia Boudries, Samia Sadouki, and Abdelkamel Tari. A bio-inspired algorithm for dynamic reconfiguration with end-to-end constraints in web services composition. *Service Oriented Computing and Applications*, 13(3):251–260, 2019.

[6] Honghao Gao, Wanqiu Huang, Xiaoxian Yang, Yucong Duan, and Yuyu Yin. Toward service selection for workflow reconfiguration: An interface-based computing solution. *Future Generation Computer Systems*, 87:298–311, 2018.

[7] Wei Lo, Jianwei Yin, Shuiguang Deng, Ying Li, and Zhaohui Wu. An extended matrix factorization approach for qos prediction in service selection. In *2012 IEEE Ninth International Conference on Services Computing*, pages 162–169. IEEE, 2012.

[8] Jun Li, Xiao-Lin Zheng, Song-Tao Chen, William-Wei Song, and De-ren Chen. An efficient and reliable approach for quality-of-service-aware service composition. *Information Sciences*, 269:238–254, 2014.

[9] Christian Krupitzer, Felix Maximilian Roth, Sebastian VanSyckel, Gregor Schiele, and Christian Becker. A survey on engineering approaches for self-adaptive systems. *Pervasive and Mobile Computing*, 17:184–206, 2015.

[10] Navinderjit Kaur Kahlon, Kuljit Kaur Chahal, and Sukhleen Bindra Narang. Managing qos degradation of component web services in a dynamic environment. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 14(2):162–190, 2018.

[11] Tao Yu, Yue Zhang, and Kwei-Jay Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 1(1):6–es, 2007.

[12] Abdessalam Messiaid, Farid Mokhati, Rohallah Benaboud, and Salem. Survey on dynamic reconfiguration of composite web services. In *International Conference on Advanced Intelligent Systems for Sustainable Development*, pages 225–235. Springer, 2018.

[13] Abdessalam Messiaid, Farid Mokhati, Rohallah Benaboud, and Hajer Salem. Towards dynamic reconfiguration of a composite web service: An approach based on qos prediction. *Electronics*, 10(13):1597, 2021.

[14] James McGovern, Oliver Sims, Ashish Jain, and Mark Little. *Enterprise service oriented architectures: concepts, challenges, recommendations*. Springer Science & Business Media, 2006.

[15] Sayed Hashimi. Service-oriented architecture explained, 2003.

[16] Kamala Manasa Dhara, Madhuri Dharmala, and Chamma Krishan Sharma. A survey paper on service oriented architecture approach and modern web services, 2015.

[17] Rahamatullah Khondoker, Abbas Siddiqui, Bernd Reuther, and Paul Müller. Service orientation paradigm in future network architectures. In *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 346–351. IEEE, 2012.

[18] Amina BEKKOUCHE. *Vers une composition automatique des services web sémantiques*. PhD thesis, 2018.

[19] Fatiha Houacine. *Service-Oriented Architecture for the Mobile Cloud Computing*. PhD thesis, 2016.

[20] David Booth. Web, Services Architecture, 2004.

[21] Cesare Pautasso, Olaf Zimmermann, and Frank Leymann. Restful web services vs." big"'web services: making the right architectural decision. In *Proceedings of the 17th international conference on World Wide Web*, pages 805–814, 2008.

[22] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[23] Heather Kreger et al. Web services conceptual architecture (wsca 1.0). *IBM software group*, 5(1):6–7, 2001.

[24] Marc Hadley, Noah Mendelsohn, J Moreau, H Nielsen, and M Gudgin. Soap version 1.2 part 1: Messaging framework. *W3C REC REC-soap12-part1-20030624, June*, pages 240–8491, 2003.

[25] Shivanand Kini. Understanding wsdl and how the wsdl editor in netbeans enterprise pack simplifies wsdl development. on line, 2006. Accessed: 2006.

[26] Shivanand Kini. Understanding wsdl and how the wsdl editor in netbeans enterprise pack simplifies wsdl development. on line, 2004. Accessed: 2006.

[27] Boualem Benatallah, Remco M Dijkman, Marlon Dumas, and Zakaria Maamar. Service-composition: concepts, techniques, tools and trends. In *Service-Oriented Software System Engineering: Challenges and Practices*, pages 48–67. IGI Global, 2005.

[28] Hye-Young Paik, Angel Lagares Lemos, Moshe Chai Barukh, Boualem Benatallah, and Aarthi Natarajan. *Web service implementation and composition techniques*, volume 256. Springer, 2017.

[29] Stéphanie Chollet. *Orchestration de services hétérogènes et sécurisés*. PhD thesis, 2009.

[30] Sonia Jamal. *Environnement de procédé extensible pour l'orchestration-Application aux services web*. PhD thesis, 2005.

[31] Schahram Dustdar and Wolfgang Schreiner. A survey on web services composition. *International journal of web and grid services*, 1(1):1–30, 2005.

[32] Markus Endler and Jiawang Wei. Programming generic dynamic reconfigurations for distributed applications. In *1992 International Workshop on Configurable Distributed Systems*, pages 68–79. IET, 1992.

[33] Ghouthi ABDELLAOUI. *La reconfiguration dynamique des systèmes pervasifs à base des systèmes multi agents*. PhD thesis, 2018.

[34] Mohammad Charaf Eddin. *Contribution to dynamic reconfiguration in component-based systems: consistency and non-functional properties specification*. PhD thesis, 2015.

[35] Kaveh Moazami-Goudarzi. *Consistency preserving dynamic reconfiguration of distributed systems*. PhD thesis, 1999.

[36] Xavier Besseron. *Tolérance aux fautes et reconfiguration dynamique pour les applications distribuées à grande échelle*. PhD thesis, 2010.

[37] Mohammad Charaf Eddin and Zoubir Mammeri. Non-functional properties aware configuration selection in component-based systems. In *15th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 1–7. IEEE, 2014.

[38] Jamie Hillman and Ian Warren. An open framework for dynamic reconfiguration. In *Proceedings. 26th International Conference on Software Engineering*, pages 594–603. IEEE, 2004.

[39] Ian Warren, Jing Sun, Sanjev Krishnamohan, and Thiranjith Weerasinghe. An automated formal approach to managing dynamic reconfiguration. In *21st IEEE/ACM International Conference on Automated Software Engineering (ASE'06)*, pages 37–46. IEEE, 2006.

[40] Ying Li, Xiaorong Zhang, YuYu Yin, and Yuanlei Lu. Towards functional dynamic reconfiguration for service-based applications. In *2011 IEEE World Congress on Services*, pages 467–473. IEEE, 2011.

[41] Kwei-Jay Lin, Jing Zhang, Yanlong Zhai, and Bin Xu. The design and implementation of service process reconfiguration with end-to-end qos constraints in soa. *Service oriented computing and applications*, 4(3):157–168, 2010.

[42] Chen Lv, Wei Jiang, Songlin Hu, Jiye Wang, Guoliang Lu, and Zhiyong Liu. Efficient dynamic evolution of service composition. *IEEE Transactions on Services Computing*, 11(4):630–643, 2015.

[43] Saurabh Shrivastava and Ashish Sharma. An approach for qos based fault reconfiguration in service oriented architecture. In *2013 International Conference on Information Systems and Computer Networks*, pages 180–184. IEEE, 2013.

[44] Wei Li. Qos assurance for dynamic reconfiguration of component-based software systems. *IEEE Transactions on Software Engineering*, 38(3):658–676, 2011.

[45] Mohammad Charaf Eddin. Towards a taxonomy of dynamic reconfiguration approaches. *Journal of Software*, 8(9):2202–2207, 2013.

[46] Daniel Jackson, Ian Schechter, and Hya Shlyahter. Alcoa: the alloy constraint analyzer. In *Proceedings of the 22nd international conference on Software engineering*, pages 730–733, 2000.

[47] Ahmed Moustafa and Minjie Zhang. Learning efficient compositions for qos-aware service provisioning. In *2014 IEEE International Conference on Web Services*, pages 185–192. IEEE, 2014.

[48] Mahmoud Hossein Zadeh and Mir Ali Seyyedi. A self-healing architecture for web services based on failure prediction and a multi agent system. In *Fourth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2011)*, pages 48–52. IEEE, 2011.

[49] Yu Dai, Lei Yang, and Bin Zhang. Qos-driven self-healing web service composition based on performance prediction. *Journal of Computer Science and Technology*, 24(2):250–261, 2009.

[50] Guillaume Babin, Yamine Ait-Ameur, and Marc Pantel. Web service compensation at runtime: formal modeling and verification using the event-b refinement and proof based formal method. *IEEE Transactions on Services Computing*, 10(1):107–120, 2016.

[51] Mingkun Yang and Xiaohui Hu. Svm-based efficient qos-aware runtime adaptation for service oriented systems. In *2016 IEEE International Conference on Web Services (ICWS)*, pages 396–403. IEEE, 2016.

[52] Huan Zhou, Zili Zhang, Yuheng Wu, and Tao Qian. Bio-inspired dynamic composition and reconfiguration of service-oriented internetware systems. In *International Conference in Swarm Intelligence*, pages 364–373. Springer, 2011.

[53] Jieming Zhu, Pinjia He, Zibin Zheng, and Michael R Lyu. Online qos prediction for runtime service adaptation via adaptive matrix factorization. *IEEE Transactions on Parallel and Distributed Systems*, 28(10):2911–2924, 2017.

[54] Yilei Zhang, Zibin Zheng, and Michael R Lyu. An online performance prediction framework for service-oriented systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 44(9):1169–1181, 2014.

[55] Shuiguang Deng, Longtao Huang, Wei Tan, and Zhaohui Wu. Top-k automatic service composition: A parallel method for large-scale service sets. *IEEE Transactions on Automation Science and Engineering*, 11(3):891–905, 2014.

[56] Fang Qiqing, Peng Xiaoming, Liu Qinghua, and Hu Yahui. A global qos optimizing web services selection algorithm based on moaco for dynamic web service composition. In *2009 International forum on information technology and applications*, volume 1, pages 37–42. IEEE, 2009.

[57] Zhao Shanshan, Wang Lei, Ma Lin, and Wen Zepeng. An improved ant colony optimization algorithm for qos-aware dynamic web service composition. In *2012 International conference on industrial control and electronics engineering*, pages 1998–2001. IEEE, 2012.

[58] Jian Liu, Kaipei Liu, Wen Wei, et al. A hybrid genetic and particle swarm algorithm for service composition. In *Sixth International Conference on Advanced Language Processing and Web Information Technology (ALPIT 2007)*, pages 564–567. IEEE, 2007.

[59] Ahmed Moustafa and Minjie Zhang. Towards proactive web service adaptation. In *International Conference on Advanced Information Systems Engineering*, pages 473–485. Springer, 2012.

[60] Amina Bekkouche, Sidi Mohammed Benslimane, Marianne Huchard, Chouki Tibermacine, Fethallah Hadjila, and Mohammed Merzoug. Qos-aware optimal and automated semantic web service composition with user's constraints. *Service Oriented Computing and Applications*, 11(2):183–201, 2017.

[61] Hitoshi Iima, Yasuaki Kuroe, and Shoko Matsuda. Swarm reinforcement learning method based on ant colony optimization. In *2010 IEEE international conference on systems, man and cybernetics*, pages 1726–1733. IEEE, 2010.

[62] CJCH WatkinsandP. Dayan,"q-learning,". *Machine Learning*, 8:279–292, 1992.

[63] Marco Dorigo, Mauro Birattari, and Thomas Stutzle. Ant colony optimization. *IEEE computational intelligence magazine*, 1(4):28–39, 2006.

[64] Danilo Ardagna and Barbara Pernici. Adaptive service composition in flexible processes. *IEEE Transactions on software engineering*, 33(6):369–384, 2007.

[65] Mohammad Alrifai, Dimitrios Skoutas, and Thomas Risse. Selecting skyline services for qos-based web service composition. In *Proceedings of the 19th international conference on World wide web*, pages 11–20, 2010.

[66] Vynska Amalia Permadi and Bagus Jati Santoso. Efficient skyline-based web service composition with qos-awareness and budget constraint. In *2018 International Conference on Information and Communications Technology (ICOIACT)*, pages 855–860. IEEE, 2018.

[67] Huan Liu, Farong Zhong, Bang Ouyang, and Jiajie Wu. An approach for qos-aware web service composition based on improved genetic algorithm. In *2010 International conference on web information systems and mining*, volume 1, pages 123–128. IEEE, 2010.

[68] Rongxi Wang, Li Ma, and Yanping Chen. The application of ant colony algorithm in web service selection. In *2010 International Conference on Computational Intelligence and Software Engineering*, pages 1–4. IEEE, 2010.

[69] Yuhong Yan, Pascal Poizat, and Ludeng Zhao. Self-adaptive service composition through graphplan repair. In *2010 IEEE International Conference on Web Services*, pages 624–627. IEEE, 2010.

[70] Hongxia Tong, Jian Cao, Shensheng Zhang, and Minglu Li. A distributed algorithm for web service composition based on service agent model. *IEEE Transactions on Parallel and Distributed Systems*, 22(12):2008–2021, 2011.

[71] Vadim Ermolayev, Natalya Keberle, Sergey Plaksin, Oleksandr Kononenko, and Vagan Terziyan. Towards a framework for agent-enabled semantic web service composition. *International Journal of Web Services Research (IJWSR)*, 1(3):63–87, 2004.

[72] Zakaria Maamar, Soraya Kouadri Mostefaoui, and Hamdi Yahyaoui. Toward an agent-based and context-oriented approach for web services composition. *IEEE transactions on knowledge and data engineering*, 17(5):686–697, 2005.

[73] Ahmed Moustafa and Minjie Zhang. Multi-objective service composition using reinforcement learning. In *International Conference on Service-Oriented Computing*, pages 298–312. Springer, 2013.

[74] Hongbing Wang, Xuan Zhou, Xiang Zhou, Weihong Liu, Wenya Li, and Athman Bouguettaya. Adaptive service composition based on reinforcement learning. In *International conference on service-oriented computing*, pages 92–107. Springer, 2010.

[75] Ivan J Jureta, Stephane Faulkner, Youssef Achbany, and Marco Saerens. Dynamic web service composition within a service-oriented architecture. In *IEEE International Conference on Web Services (ICWS 2007)*, pages 304–311. IEEE, 2007.

[76] Hongbing Wang, Xiaojun Wang, Xingzhi Zhang, Qi Yu, and Xingguo Hu. Effective service composition using multi-agent reinforcement learning. *Knowledge-Based Systems*, 92:151–168, 2016.

[77] Hongbing Wang, Jiajie Li, Qi Yu, Tianjing Hong, Jia Yan, and Wei Zhao. Integrating recurrent neural networks and reinforcement learning for dynamic service composition. *Future Generation Computer Systems*, 107:551–563, 2020.

[78] Ping Wang. Qos-aware web services selection with intuitionistic fuzzy set under consumer's vague perception. *Expert Systems with Applications*, 36(3):4460–4466, 2009.

[79] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996.

[80] Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, 1989.

[81] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[82] Anton Orell Wiehe, Nil Stolt Ansó, Madalina M Drugan, and Marco A Wiering. Sampled policy gradient for learning to play the game agar. io. *CoRR*, 2018.

[83] Jean-Louis Deneubourg, Jacques M Pasteels, and Jean-Claude Verhaeghe. Probabilistic behaviour in ants: a strategy of errors? *Journal of theoretical Biology*, 105(2):259–271, 1983.

[84] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477. IEEE, 1999.

[85] Eric Bonabeau, Marco Dorigo, and Guy Théraulaz. From natural to artificial swarm intelligence, 1999.

[86] Hongbing Wang, Xiaojun Wang, Xingguo Hu, Xingzhi Zhang, and Mingzhu Gu. A multi-agent reinforcement learning approach to dynamic service composition. *Information Sciences*, 363:96–119, 2016.

[87] Zibin Zheng, Yilei Zhang, and Michael R Lyu. Investigating qos of real-world web services. *IEEE transactions on services computing*, 7(1):32–39, 2012.

[88] Duksan Ryu, Kwangkyu Lee, and Jongmoon Baik. Location-based web service qos prediction via preference propagation to address cold start problem. *IEEE Transactions on Services Computing*, 2018.

[89] Kai Su, Bin Xiao, Baoping Liu, Huaiqiang Zhang, and Zongsheng Zhang. Tap: A personalized trust-aware qos prediction approach for web service recommendation. *Knowledge-Based Systems*, 115:55–65, 2017.

[90] Xiaoke Zhu, Xiao-Yuan Jing, Di Wu, Zhenyu He, Jicheng Cao, Dong Yue, and Lina Wang. Similarity-maintaining privacy preservation and location-aware low-rank matrix factorization for qos prediction based web service recommendation. *IEEE Transactions on Services Computing*, 2018.

[91] Amin Keshavarzi, Abolfazl Toroghi Haghighat, and Mahdi Bohlouli. Adaptive resource management and provisioning in the cloud computing: A survey of definitions, standards and research roadmaps. *KSII Transactions on Internet & Information Systems*, 11(9), 2017.

[92] Xi Chen, Zibin Zheng, Qi Yu, and Michael R Lyu. Web service recommendation via exploiting location and qos information. *IEEE Transactions on Parallel and distributed systems*, 25(7):1913–1924, 2013.

[93] Philipp Leitner, Anton Michlmayr, Florian Rosenberg, and Schahram Dustdar. Monitoring, prediction and prevention of sla violations in composite services. In *2010 IEEE International Conference on Web Services*, pages 369–376. IEEE, 2010.

[94] Ruibin Xiong, Jian Wang, Neng Zhang, and Yutao Ma. Deep hybrid collaborative filtering for web service recommendation. *Expert systems with Applications*, 110:191–205, 2018.

[95] Lingshuang Shao, Jing Zhang, Yong Wei, Junfeng Zhao, Bing Xie, and Hong Mei. Personalized qos prediction forweb services via collaborative filtering. In *Ieee international conference on web services (icws 2007)*, pages 439–446. IEEE, 2007.

[96] Jieming Zhu, Pinjia He, Zibin Zheng, and Michael R Lyu. A privacy-preserving qos prediction framework for web service recommendation. In *2015 IEEE International Conference on Web Services*, pages 241–248. IEEE, 2015.

[97] Zhen Chen, Limin Shen, Feng Li, Dianlong You, and Jean Pepe Buanga Mapetu. Web service qos prediction: when collaborative filtering meets data fluctuating in big-range. *World Wide Web*, pages 1–26, 2020.

[98] Yiguang Song, Li Hu, and Ming Yu. A novel qos-aware prediction approach for dynamic web services. *Plos one*, 13(8):e0202669, 2018.

[99] Waseem Ahmed, Yongwei Wu, and Weimin Zheng. Response time based optimal web service selection. *IEEE Transactions on Parallel and distributed systems*, 26(2):551–561, 2013.

[100] Rafael Roque Aschoff, Andrea Zisman, and Pedro Alexandre. Parallel adaptation of multiple service composition instances. In *Engineering Adaptive Software Systems*, pages 115–134. Springer, 2019.

[101] Honghao Gao, Wanqiu Huang, and Yucong Duan. The cloud-edge-based dynamic reconfiguration to service workflow for mobile ecommerce environments: A qos prediction perspective. *ACM Transactions on Internet Technology (TOIT)*, 21(1):1–23, 2021.

[102] Leilei Chen, Qing Wang, Wei Xu, and Liang Zhang. Evaluating the survivability of soa systems based on hmm. In *2010 IEEE International Conference on Web Services*, pages 673–675. IEEE, 2010.

[103] Andreas Metzger, Chi-Hung Chi, Yagil Engel, and Annapaola Marconi. Research challenges on online service quality prediction for proactive adaptation. In *2012 First International Workshop on European Software Services and Systems Research-Results and Challenges (S-Cube)*, pages 51–57. IEEE, 2012.

[104] Abdenour Soualhi, Guy Clerc, Hubert Razik, François Guillet, et al. Hidden markov models for the prediction of impending faults. *IEEE Transactions on Industrial Electronics*, 63(5):3271–3281, 2016.

[105] Qingtao Wu, Mingchuan Zhang, Ruijuan Zheng, Ying Lou, and Wangyang Wei. A qos-satisfied prediction model for cloud-service composition based on a hidden markov model. *Mathematical Problems in Engineering*, 2013, 2013.

[106] Amin Keshavarzi, Abolfazl Toroghi Haghighat, and Mahdi Bohlouli. Enhanced time-aware qos prediction in multi-cloud: a hybrid k-medoids and lazy learning approach (qopc). *Computing*, 102(4):923–949, 2020.

[107] Jason Osborne. Improving your data transformations: Applying the box-cox transformation. *Practical Assessment, Research, and Evaluation*, 15(1):12, 2010.

[108] S Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *CoRR*, 2015.

[109] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[110] Hong Zheng, Ruoyin Wang, Wencheng Xu, Yifan Wang, and Wen Zhu. Combining a hmm with a genetic algorithm for the fault diagnosis of photovoltaic inverters. *Journal of Power Electronics*, 17(4):1014–1026, 2017.

[111] Fanny Yang, Sivaraman Balakrishnan, and Martin J Wainwright. Statistical and computational guarantees for the baum-welch algorithm. *The Journal of Machine Learning Research*, 18(1):4528–4580, 2017.

[112] Muzaffar Eusuff, Kevin Lansey, and Fayzul Pasha. Shuffled frog-leaping algorithm: a memetic meta-heuristic for discrete optimization. *Engineering optimization*, 38(2):129–154, 2006.

[113] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. Ieee, 1995.

[114] Godar J Ibrahim, Tarik A Rashid, and Mobayode O Akinsolu. An energy efficient service composition mechanism using a hybrid meta-heuristic algorithm in a mobile cloud environment. *Journal of Parallel and Distributed Computing*, 143:77–87, 2020.

[115] Shufen Qin, Chaoli Sun, Guochen Zhang, Xiaojuan He, and Ying Tan. A modified particle swarm optimization based on decomposition with different ideal points for many-objective optimization problems. *Complex & Intelligent Systems*, pages 1–12, 2020.

[116] Hanane Amirat, Nasreddine Lagraa, Philippe Fournier-Viger, and Youcef Ouinten. Myroute: a graph-dependency based model for real-time route prediction. *J Commun*, 12:668–676, 2017.