PSL Research University
Université Paris-Dauphine

Mémoire présenté en vue de l'obtention
de l'Habilitation à Diriger des Recherches

Transactional and QoS-aware Internet Service Management

Maude Manouvrier

Jury :

Bernd Amann,	Professeur, Université Pierre and Marie Curie	*rapporteur*
Salima Benbernou,	Professeur, Université Paris Descartes,	*rapportrice*
Omar Boucelma,	Professeur, Université d'Aix-Marseille,	*rapporteur*
Dario Colazzo,	Professeur, PSL Research University, Université Paris-Dauphine	*président de jury*
Lynda Mokdad,	Professeur, Université Paris-Est, Créteil	*examinatrice*
Marta Rukoz,	Professeur, PSL Research University, Université Paris-Dauphine et Université Paris-Ouest Nanterre La Défense	*coordinatrice*

Date de soutenance : 12 décembre 2016

A ma petite DCB, mes grands-parents et J. de Romilly.

Remerciements

Je remercie tout particulièrement :

Marta Rukoz, Professeur à l'Université Paris Ouest-Nanterre La Défense, et chercheuse au LAMSADE de PSL Research University l'Université Paris-Dauphine, avec qui je collabore depuis ma thèse de Doctorat. Son enthousiasme imperturbable, sa très grande curiosité scientifique, sa patience et son infinie gentillesse ont été et restent pour moi un moteur et un exemple. Je la remercie de tout mon cœur d'avoir accepté de travailler avec moi, de m'avoir tant appris et de m'avoir guidée depuis tant d'années. Merci infiniment d'avoir accepté de coordonner mon habilitation et de me faire l'honneur de participer à ce jury.

Salima Benbernou, Professeur à l'Université Paris Descartes, pour avoir chaleureusement accepté de rapporter sur mon travail et pour me faire l'honneur de participer à ce jury.

Bernd Amann, Professeur à l'Université Pierre and Marie Curie, pour ses conseils et sa gentillesse transmis à chacune de nos rencontres. Merci d'avoir accepté de rapporter sur mon travail et de me faire l'honneur de participer à ce jury.

Omar Boucelma, Professeur à l'Université d'Aix-Marseille, pour avoir travaillé avec nous à l'élaboration de plusieurs dossiers de candidature ANR et pour sa sympathie. Merci d'avoir accepté de rapporter sur mon travail et de me faire l'honneur de participer à ce jury.

Lynda Mokdad, Professeur à l'Université Paris-Est, Créteil, pour avoir travaillé avec nous à l'élaboration de plusieurs dossiers de candidature ANR, pour sa joie de vivre communicative et pour sa bienveillance. Merci de me faire l'honneur de participer à ce jury.

Dario Colazzo, Professeur de PSL Research University - Université Paris-Dauphine, pour ses encouragements et pour sa gentillesse. Merci de me faire l'honneur de participer à ce jury.

Yudith Cardinale, Professeur à l'Université Simón Bolívar au Venezuela, pour son amitié, son enthousiasme et tout le travail que nous avons fait ensemble. Merci infiniment, Yudith, d'avoir accepté de relire mon mémoire et pour tous tes conseils et remarques.

Cécile Murat et Virginie Gabrel, Maîtres de Conférences à l'Université Paris-Dauphine, pour m'avoir initiée à la RO, pour toutes nos chouettes séances de travail et pour avoir été présentes à chacun de mes moments de doute.

Joyce El Haddad, Maîtres de Conférences à l'Université Paris-Dauphine, pour toutes ces super années de collaboration.

Daniela Grigori, Professeur à l'Université Paris-Dauphine, pour m'avoir permis d'encadrer la thèse d'Amine avec elle et pour ces encouragements.

Geneviève Jomier, Professeur à l'Université Paris-Dauphine, qui a été ma directrice de thèse et qui m'a initiée à la recherche. Sans elle, je ne serai pas là aujourd'hui. Merci pour son soutien depuis toutes ces années.

Rafael Angarita, N. Vu Hoàng (alias Rin), Mohammed Lamine Mouhoub (alias Amine), pour m'avoir permis de co-diriger leur thèse et pour tout le travail que nous avons réalisé ensemble.

Tous les membres du LAMSADE, Marie-Hélène (partie vers d'autres horizons mais toujours là pour moi), Juliette, Hawa, Mireille, Olivier, Alexis (pour m'avoir mis la "bonne" pression), Khalid, Elsa, Michel, et tous celles et ceux que je ne peux ajouter par manque de place et qui font que je suis très heureuse de travailler dans ce laboratoire.

Résumé

La gestion d'un grand nombre de services dans l'Internet crée de nombreux problèmes de recherche ouverts allant de la découverte à la composition et l'exécution fiable de services. La découverte correspond à l'identification des services capable d'atteindre un objectif donné. La composition consiste à regrouper ou agréger de services existants, en vue d'en créer un nouveau composite. Une exécution est fiable lorsqu'elle est tolérante aux panne ou auto-corrective (*self-healing*). Les travaux de recherche, présentés dans ce document, décrivent les approches, modèles et analyse, que nous avons proposés pour la gestion des services Internet. Ce document contient cinq chapitres. Un premier chapitre présente les définitions des concepts liés à notre travail. Les chapitres 3 et 4 présentent les approches que nous avons proposées pour la sélection de services transactionnels basée sur la qualité de service. Le chapitre 5 concerne les travaux, que nous avons réalisés dans le cadre d'encadrement de doctorants, sur l'exécution fiable de services et la découverte de services. Finalement, le chapitre 6 présente quelques perspectives de recherche.

Mots-Clés : *Services transactionnels, Qualité de Service, séléction de services, Exécution fiable de services, Recherche de service*

Abstract

The management of ultra large number of services in the global Internet creates many open problems from discovering to composing services and offering a reliable service execution. Discover services consists in identifying the services that are able to accomplish a given goal. Composing them correspond to grouping or aggregating existing services to create a new composite one. A service execution is reliable when it is fault-tolerant or self-healing. Research works presented in this document describes the approaches, models and analysis, we have proposed for the management of Internet services. This document is decomposed into five chapters. A first chapter introduces the definitions of the concepts related to our work. Chapters 3 and 4 present the transactional and QoS-aware selection approaches we have proposed. Chapter 5 concerns the works, we have carried out with PhD students, about the reliable service execution and the service discovery. Finally, chapter 6 outlines some research perspectives.

Keywords : *Transactional services, Quality of Service, Service selection, Reliable service execution, Service search*

Contents

Chapter 1

Introduction

1.1 Context

A substantial part of the digital content is exposed via Web services (WS), i.e., internet-based programmable application components accessible to other applications over the Web [42, 226]. In 2010, `seekda.com`, for example, provided an index of more than 28,000 web services [188]. Nowadays, many web sites (e.g., Facebook[1], Twitter[2] or Flickr[3]) offer access to a part of their data through Web API (Application Programming Interface). `Programmableweb.com`, for example, is a directory containing more than 15,000 Web APIs of different types of services. Therefore, we are seeing a proliferation of data offered in the Web and particularly a proliferation of services [211], called Internet services in [188], allowing to access these data and a proliferation of their uses.

As explained in surveys [64, 88, 122, 215], the management of ultra large number of services in the global Internet creates many open problems from discovering to composing services. Service discovery is "the identification of services that are capable of accomplishing a given objective" [210]. Service composition "encompasses all the processes that create added-value services called composite or aggregated services, from existing services" [143].

This document presents a synthesis of the research work we have undertaken on these topics, since 2006.

1.2 Manuscript's organization

This manuscript contains 6 chapters. Chapter 2 introduces the definitions of the concepts related to our work. Our contributions are presented in chapters 3, 4 and 5, the last one concerning particularly the works carried out with PhD students. Chapter 6 outlines some research perspectives.

In the following, references corresponding to our articles are underlined (see pages 47 to 49) while references corresponding to the related work are in normal text (see references starting from page 50).

[1]`https://developers.facebook.com/docs`
[2]`https://dev.twitter.com/overview/documentation`
[3]`https://www.flickr.com/services/api/`

1.3 Contributions synthesis

From our point of view, a service management framework must: (1) be able to discover/identify services achieving a specific goal, (2) compose them when no single service can reach the needed goal, and (3) establish mechanisms to ensure a reliable execution of the resulting composite service. The execution of a composite service is reliable if, in case of component service failure, the negative impacts are negligible for the user. Figure 1.1 represents such a service management framework. Our research works address all the three aforementioned steps, as shown our references listed in each box representing a specific step.

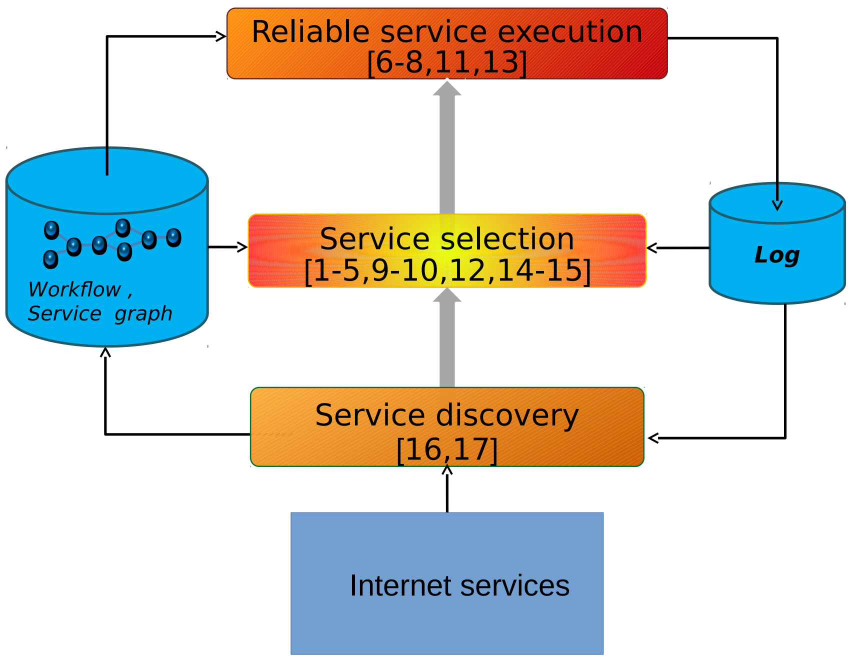
Reliable service execution [6-8,11,13]
Workflow , Service graph
Service selection [1-5,9-10,12,14-15]
Log
Service discovery [16,17]
Internet services

Figure 1.1: Our vision of a service management framework (cited references correspond to our publications).

1.3.1 Service composition

A big part of our research work mainly deals with service composition. More precisely, we have defined approaches to select service components of a composite service, in such way the user preferences are optimized. The user preferences considered deal with Quality of Service (QoS), for example service execution time, and with transactional properties. Transactional properties are a set of rules that ensures a reliable (fault-tolerant) execution of services. In [18], we have proposed a survey comparing the QoS and transactional-aware service composition approaches. We considered two kind of composite service representations, by workflow (see Chapter 3), representing the composite service as a combination of functionalities, and by graph (see Chapter 4), representing the input/output dependencies between services.

When the composite service is represented by a workflow (equivalent to a series-parallel directed graph [229]), component services are generally grouped by functionalities (e.g., services providing image compression algorithms can be grouped together, services allowing to reduce image noises can also constitute a service class). Therefore, the composition problem consists in selecting a component service for each functionality of the workflow. In [12], and in its extending version [3], we have proposed an heuristic algorithm to locally optimize the QoS (measured by a weighted aggregated function of QoS criteria) of a transactional composite service. In [3], we also have formalized the transactional properties of a composite or non-composite service. In [14], we have proposed a 0-1 linear program for the QoS and transactional-aware service composition, when the composition is specified by a workflow. This approach globally optimizes the QoS of the resulting composite service. In [4], we have presented a complexity analysis of the workflow-based service composition problem, in regard to the structural properties of the workflow and the QoS models used.

When services are linked by their input/output dependencies, no workflow is specified, and the service composition consists in selecting services by matching their parameters (I/O) so that the resulting composite service can produce a set of outputs from a set of inputs provided by a user. This problem, generally called QoS-aware automatic syntactic service composition problem, is tackled by the standard benchmark Web Service Challenge (WSC) (see in [50] for an overview of this benchmark). In [2], [9], and [10], we have modelled the service composition problem by Coloured Petri Net [125]. Our approach in [10] only deals with transactional service composition (without considering QoS). The heuristic proposed in [2] is an extension of our approach [10], locally optimizing the QoS. In [9], our algorithm is an A*-based heuristic allowing a global QoS optimization approach. Then, in [14], we have modelled the QoS and transactional-aware service composition by a linear program globally optimizing a weighted sum of QoS criteria. Our model had been extended to minmax-type criterion (e.g.„ service execution time), with or without considering transactional properties, in [15], and has been experimented using the WSC data sets. In [20], we present a complexity analysis of the QoS-syntactic service composition problem, and propose an original formulation in terms of scheduling problem with AND/OR constraints, using a directed graph structure.

1.3.2 Service search and service execution

Our work also deals with the steps processed before and after the service composition: the service search or discovery and the service execution. In [8], as part of a PhD student work, we have defined a model to support self-healing composite service executions while maintaining the QoS requirements, even in presence of failures. In [5], we have proposed a framework for QoS and transactional service selection based on crowdsourcing. Concerning transactional composite service execution, we have proposed, in [13], a transactional execution model of composite service, exploiting the transactional properties of its components. In [16], as part of a PhD student work, we had proposed a framework for searching semantic data and services using SPARQL. A demonstration of this framework is presented in [17].

1.3.3 Content-based image retrieval

Before working on service composition, we had worked, from 1996 to 2011, on Content-Based Image Retrieval (CBIR). During my PhD thesis [34], we have proposed a set of structures for managing similar large objects, and particularly a structure, called *Generic Quad-Tree*, presented in [22, 28, 30], to store similar images. Two images are defined similar if, for both images, the quad-trees (recursively cutting the images in four quadrants), that encode them, are similar i.e., differ only for a relative small number of nodes (see our survey [26] on quad-tree-based image representation and retrieval, for more details). The Generic Quad-Tree optimizes the memory space of similar images and allows image processing operations, like images comparison, comparison of the same area in different images, or simultaneous updates applied in different images. After my PhD thesis, we also have defined an index structure, in [24, 29], for content-based image retrieval when images are organized by quad-trees, and a generalized distance between image quad-trees, in [23, 32].

Content-based image retrieval can be improved by integrating the spatial layout of objects in the image. Several approaches have been proposed for integrating spatial relationships into the image description (see our study [27] about the spatial relationship representation on symbolic images). Therefore, our research interest have been lied on the spatial information embedded into image content description for scene retrieval. Particularly, within the PhD work of Nguyen Vu Hoang [21][178], I had supervised with Marta Rukoz and Valérie Gouet-Brunet, we have proposed an image content representation describing the spatial layout with triangular relationships of visual entities, which can be symbolic objects or low-level visual features.

Due to page limitation, this manuscript does not tackle our CBIR research works but only presents the research works we have done about service management. Nevertheless, image management is an application domain of services. Thus, several examples, used in this document, deal with service providing image processing or content-based image retrieval algorithms.

Chapter 2

Preliminaries

Internet services are "internet-based programmable application components that are published using standard interface decription language and that are universally available via standard communication protocols" [226]. Drawing our inspiration from [236], we define an internet service description ontology in Figure 2.1. The following sections describe in details this figure. Section 2.1 deals with the technological aspects of services. The syntactic description of services is defined in Section 2.2. Differences between non-composite and composite service are presented in Section 2.3. Section 2.4 characterises the service registry or repository. Quality of Service (QoS) is tackled in Section 2.5. Section 2.6 addresses the service transactional properties and Section 2.7 defines the life-cycle of service composition. Finally, Section 2.8 positions our work in relation to the different provided definitions of this chapter.

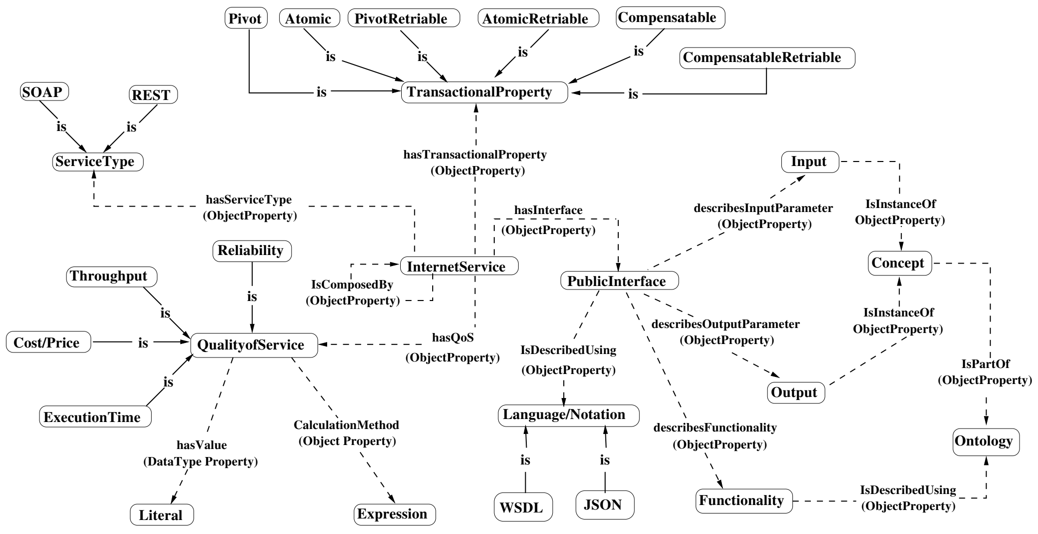
Pivot
Atomic
PivotRetriable
AtomicRetriable
Compensatable
CompensatableRetriable
is
TransactionalProperty
SOAP
REST
ServiceType
hasTransactionalProperty (ObjectProperty)
hasServiceType (ObjectProperty)
hasInterface (ObjectProperty)
Input
describesInputParameter (ObjectProperty)
IsInstanceOf ObjectProperty)
Reliability
Throughput
InternetService
IsComposedBy (ObjectProperty)
PublicInterface
Concept
describesOutputParameter (ObjectProperty)
Cost/Price
QualityofService
hasQoS (ObjectProperty)
IsDescribedUsing (ObjectProperty)
Output
IsPartOf (ObjectProperty)
ExecutionTime
Language/Notation
hasValue (DataType Property)
CalculationMethod (Object Property)
describesFunctionality (ObjectProperty)
Ontology
WSDL
JSON
Functionality
Literal
Expression

Figure 2.1: Our internet service description ontology (inspired from [236]).

2.1 Service-oriented architectural style

As defined in [215], a Web Service (WS) "is often seen as an application accessible to other over the Web". According to the World Wide Web Consortium (W3C) [72] and recent surveys [180, 235]: "a web service is a software system identified by a Uniform Resource Identifier[1] (URI), whose public interfaces and bindings are defined and described using eXtensible Markup Language[2] (XML) or JavaScript Object Notation[3] (JSON)". The Web Services Description Language (WSDL) [63] is an XML-based interface definition language generally used for describing the functionality offered by a web service. JSON-WSP (JavaScript Object Notation Web-Service Protocol) is a web-service protocol that uses JSON for service description, requests and responses [79]. "Web service definition can be discovered by other software systems. These systems may then interact with the web service in a manner prescribed by its definition, using messages conveyed by internet protocols" [72].

Web services, commonly just called "service" [180], are considered as the "best" implementation of the Service-Oriented Architecture (SOA) [257]. They are often categorized as two types of implementations [38, 187, 215]: Simple Object Access Protocol (SOAP)-based [81] ones, also called 'Big" or WS* services, and Representational State Transfer (RESTful) ones [201]. According to [215, 198, 236], SOAP-based services (described using WSDL [63]) are more related to complex business/enterprise applications (using business process modelling). On the other hand, RESTful services are light-weight data/resource-centric services, prevalent in Web 2.0 [176] applications, due to their flexibility and simplicity. They are generally preferred by Government department or public data providers and are well suited for "tactical ad hoc integration over the Web". Comparison of both technologies can be found in [38, 113, 126, 187]. Recent works have been done to combine SOAP and RESTful services [141, 140, 198] or propose to convert SOAP-based services to RESTful ones [228].

Remark 2.1.1 *In our work, we do not consider technological aspects of services and only take into account the syntactic description of services, and their non-functional QoS properties, as defined in the next subsections.*

2.2 Syntactic description of service

An internet service can be seen as a function provided through a web-accessible endpoint. As the syntactic description of functions in programming language, an internet service can then be described by its input and output parameters: a service needs or consumes one or several inputs to be invoked and executed, in order to produce one or several outputs. As explained in [76], services in this case are regarded as black-boxes that accept inputs to produce outputs.

Definition 2.2.1 Syntactic description of service.
Let be s a service. A service s is syntactically described by its set of inputs, $in(s)$, and its set of outputs, $out(s)$.

For example, authors in [129] had implemented several image processing services taking an image as input and producing an output image (e.g. reducing the noise in the input image or transforming a multi-channel input image into a single-channel one) – see Table 2.1. Web Service Challenge (WSC) proposes an incremental synthetic data benchmark providing syntactic service description since 2005: a WSDL standard file [63], called `services.wsdl`, describes input and output parameters of all services available in the benchmark test sets (see in [238] for a description of WSC). Author in [141] had extracted the syntactic description of several SOAP and REST APIs of `ProgrammableWeb.com`.

[1]`https://en.wikipedia.org/wiki/Uniform_Resource_Identifier`
[2]`http://www.w3.org/XML/`
[3]`http://www.json.org/`

Table 2.1: Examples of image processing services, from [129]

Service	**Inputs**	**Outputs**	**Functionality**
s_1	A multi-channel image	A single-channel (gray-scale) image	Converts input image to a gray-scale one
s_2	A single-channel image and a threshold	A transformed single-channel image	Applies a binary thresholding method to input image
s_3	An image and a kernel size	A transformed image	Applies a blurring function for reducing input image noise using Gaussian filter
s_4	An image and a kernel size	A transformed image	Applies a blurring function for reducing input image noise using Median filter

Syntactic description is used in many approaches. For example, authors in [242] propose a multi-level index, whose first levels are constructed according to the inputs and the outputs of services. In [139], services inputs and outputs are stored in a relational database.

Each input and output parameter can be instance (i.e. a leaf) of a concept defined in an ontology (i.e. a hierarchy of concepts or specification of conceptualization). For example, a real image is an instance of concepts *multi-channel image* or *single-channel image*, both being sub-concepts of concept *image*; authors in [105] present another example of a concept hierarchy about snapshot (image) collected by camera sensor. Web Service Challenge had integrated an OWL [56] ontology in 2007, and structured data types in 2008 [51]: each input/output parameter is defined as an instance of the ontology (i.e. as a leaf in a concept hierarchy) stored in the file called `taxonomy.owl`. Enriching the syntactic description of services with semantic information allows to determine semantic services. Linking input and output parameters to an ontology is a first step to semantic description of services. Authors of [257] propose for example an automatic annotation of inputs and outputs of web services, each parameter being an instance of an OWL class in the DBpedia[4] ontology (see in [142] for a description of DBpedia).

A service can also be described by its functionality, i.e. a semantic (generally textual) description of what the service does (see for example last column of Table 2.1). Two services with the same input and output sets can be differentiated by their functionality. For example, two service, one applying a Gaussian filter and one applying a Median filter, as the one implemented in [129], take both an image as input to produce a burred output image from the input one. Service functionality can also be defined using an ontology.

Remark 2.2.1 *In our work, we suppose that all services are syntactically described by their inputs/outputs or by functionality. The mean to extract such description is out of the scope of our work.*

[4]`http://dbpedia.org/ontology/`

2.3 Single vs. composite service

A service that works individually or in a stand-alone manner, i.e. without explicitly relying on another service, to achieve a desired requirement, is called single in [186, 235] or elementary[5] in [214].

A service can be composite, i.e. composed by several other services, in order to achieve a complex goal that cannot be fulfilled by a single individual service [186, 235]. The process of developing a composite service is called service composition [89].

Definition 2.3.1 Composite service/component service.
A service s is a composite service if it is composed by several other services, called component services. Let be $S_c(s) = \{sc1, ...sc_n\}$, the set of n component services of a composite service s, with sc_i the ith component of s.

A composite service can be represented either by a workflow, a graph or a Petri-Net. When a composite service is represented by a worklfow (i.e. a structured and ordered control flow of functionalities or tasks), as we did in [3, 4, 12, 13, 14], it can be designated as: an abstract workflow representing a set of functionalities and a concrete one, where each functionality is associated with a component service.

Definition 2.3.2 Workflow.
An abstract workflow is a set of activities (also called tasks) combined by patterns, representing the temporal dependency between activities, each activity representing a functionality. A workflow becomes concrete when activities have been associated with (performed by) services (one service per activity).

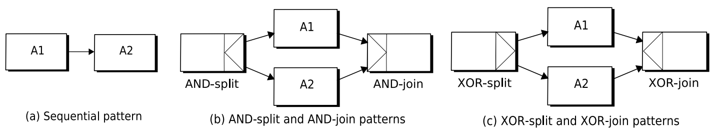
A1
A2
(a) Sequential pattern
A1
A2
AND-split
AND-join
(b) AND-split and AND-join patterns
A1
A2
XOR-split
XOR-join
(c) XOR-split and XOR-join patterns

Figure 2.2: Three workflow patterns (from [3]).

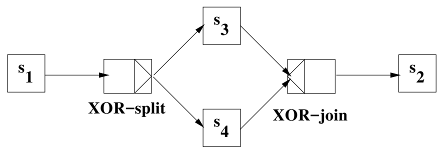

Figure 2.3: A composite service (composed by services of Table 2.1) represented by a concrete workflow.

Activities in a workflow are organized using patterns (see in [232] for a description of the workflow patterns). Figure 2.2 represents three workflow patterns, based on the YAWL model [231], we used in our work. The sequential pattern of Figure 2.2.(a) represents the fact that activity A_1 must be executed before activity A_2. AND and

[5]Several authors call it also an atomic service (see for example [120, 215, 260]). However we do not use this term which has another meaning in transactional properties (see Section 2.6).

XOR patterns start with a split and finish with a join, and are composed of two or more branches. In Figures 2.2.(b) and 2.2.(c), the first branch contains activity A_1 and the second branch contains activity A_2. An AND pattern represents parallel execution of activities (for all branches). For XOR pattern, activities of only one branch have to be executed. Figure 2.3 presents an example of an image processing composite service, whose components are described in Table 2.1.

Note that a workflow with interlaced patterns sequence, AND and XOR, is a Serie-Parallel directed multigraph [87], whose nodes represent activities and patterns, as we have explained in [4].

A composite service can also be represented by a graph, $G = (V, U)$. When V contains vertices representing both services and inputs and outputs, as we did in [15, 20], the graph is called a ServiceData graph:

Definition 2.3.3 ServiceData graph.
Let be S the set of vertices associated with services and D the set of vertices associated with the inputs and the outputs of services. To simplify, let us consider that a vertex $i \in S$ is associated with a service $s_i \in S$. A ServiceData graphs is a graph $G = (V, U)$, such that:

- $V = S \cup D$, with $S \cap D = \emptyset$,
- $\forall i \in S$, i represents a service s_i,
- $\forall j \in D$, j represents an input or an output of a service,
- An arc $u = (i, j) \in U$, with $i \in S$ and $j \in D$, represents the fact service s_i produces output j ($j \in out(s_i)$) and an arc $u = (j, i) \in U$ represents the fact service s_i needs input j to be executed ($j \in in(s_i)$).

G is a bipartite graph: there does not exist any directed arc linking two vertices belonging to S or two vertices belonging to D.

Figure 2.4 gives an example of a composite service represented by a ServiceData graph. In this figure, vertices representing services are drawn with squares, while vertices representing inputs and outputs are drawn with circles.

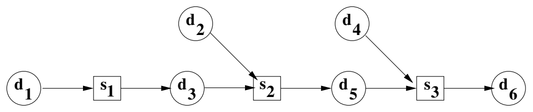

Figure 2.4: An composite service represented by a ServiceData graph.

Note that when an ontology is associated with the syntactic description of services (as in Web Service Challenge for example – see details in Section 2.2), each input i of a service s is represented by one vertex $i \in D$, and each output o of a service s is represented by several vertices in D, as many as the concept of o has ancestors in the ontology (i.e., as many super-concepts from the ontology concept root to concept o).

A composite service can also be represented by a Petri Net (PN) [116], as we did in [2, 9, 10] (see in [1] for a survey of service approaches using PN). A Petri net [190] is a directed bipartite graph whose nodes represent transitions (i.e. events that may occur) and places (i.e. conditions to fire transitions). Compared with a ServiceData graph, a Petri Net is dynamic: marking and tokens allow to model the dynamic behaviour of a process.

Definition 2.3.4 Service Petri Net. A Service Petri Net is 3-tuple (P, T, F) where: P is a finite non-empty set of places, corresponding to the inputs and outputs services, T is a finite set of transitions corresponding to the set of services, and $F : (P \times T) \cap (T \times P) \longrightarrow \{0, 1\}$ is a flow relation indicating the presence (1) or the absence (0) of arcs between places and transitions defined as follows: $\forall pl \in P, (\exists t \in T | F(pl, t) = 1)$ means that pl corresponds to an input of the service associated with transition t and $(\exists pl \in P | F(t, pl) = 1)$ means that pl corresponds to an output of the service associated with transition t.

Figure 2.5 gives an example of a composite service represented by a Service Petri Net. Places are drawn with circles. Transitions are drawn with black rectangles. Both are labelled by either inputs or outputs identifier or service number.

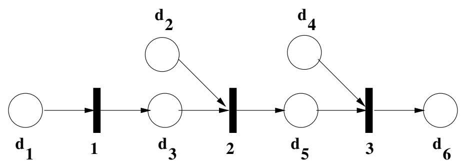

Figure 2.5: A composite service represented by a Service Petri Net.

When the graph vertices only represent services ($V = S$), the graph is called a Service Dependency graph. An arc between two vertices i and j exists if the corresponding service s_i produces at least an output which is an input of service s_j. Such graph is used for example in [105, 127].

Definition 2.3.5 Service Dependency graph.
A Service Dependency graph is a graph $G = (V, U)$, where $\forall i \in V$, i represents a service $s_i \in S$ and where $\forall u = (i, j) \in U$, we have $out(s_i) \cap in(s_j) \neq \emptyset$.

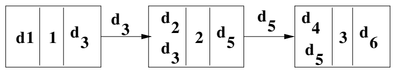

Figure 2.6: A composite service represented by a Service Dependency Graph.

Figure 2.6 gives an example of a composite service represented by a Service Dependency graph. In this figure, vertices representing services are drawn with squares divided in three parts representing the inputs, the service number and the outputs and arcs are labelled by inputs and outputs.

Remark 2.3.1 *In our work, we represent the composite services by workflow, Service-Data graph or Petri-Net.*

2.4 Service registry/repository

A service registry or repository is a searchable directory providing service descriptions [252]. For example, the *Jena Geography Dataset*[6] groups the descriptions of almost 200 geography services that have been gathered from different web sites. The *Bio-Catalogue*[7] provides a curated catalogue of 1189 web services devoted to life science.

[6]`http://fusion.cs.uni-jena.de/professur/jgd/`

[7]`https://www.biocatalogue.org/`

ProgrammableWeb[8] directory is described as the "the world's largest API repository", listing more than 14,000 APIs for various services. The OASIS standard Universal Description, Discovery, and Integration (UDDI[9]) directory is a registry of web service descriptions. Service descriptions can be stored in a relational database, as in [139].

Authors in [120] make a distinction between service registry and service repository. For them, a service repository contains the human or machine understandable service descriptions that can be utilized in software development. These descriptions are usually static. On the other hand, the service registry contains a machine-interpretable run-time (or dynamic) descriptions of all the services that are currently available. Authors in [251] consider that the service registry is a component of the repository that stores service meta-data. In our work we do not make any distinction between registry or repository.

Definition 2.4.1 Service registry/repository.
A service registry, also called repository, is a set of available services.

When composite services are represented by a workflow (see Def. 2.3.2), services having the same functionality, but differing in their non-functional properties, are gathered into a group/cluster, also called a community in [162, 235, 260]. A registry is therefore a set of service communities (one community per functionality).

When composite services are represented by a graph (a ServiceData graph, see Def. 2.3.3, a Petri Net, see Def. 2.3.4, or a Dependency graph, see Def. 2.3.5), the entire registry is generally represented by a graph and composite services represent a sub-graph of the entire graph corresponding to the registry. As we explained in [20], a ServiceData graph can be, for example, easily built from the files provided by WSC-09 synthetic benchmark[10] [135]. This benchmark provides 5 test sets, each one containing one service registry described by: `services.wsdl` a file describing services of the registry by their input/output and `taxonomy.owl` containing the ontology. The first file contains all the services' identifier and their inputs and outputs, each one being defined as an instance of the ontology (i.e. as a leaf in a concept hierarchy) stored in the second file. Each service is then represented in the ServiceData graph by a vertex s. Each input i of a service s is represented by vertex i. Each output o of a service s is represented by several vertices, as many as the concept of o has ancestors (i.e., super-concepts) in the ontology. As previously said, author in [141] had built a Dependency graph from the syntactic description of several SOAP and REST APIs of `ProgrammableWeb.com`. In [20] we mentioned that, for building a Dependency graph, one needs to read the service registry and, for each service, one has to match its inputs with the outputs of all the other services, inducing an $O(|S|)$ complexity for each service (cf. [179]). Consequently, the building step of a Dependency graph is $O(|S|^2)$.

Remark 2.4.1 *In our work, we suppose that the registry is available. When using workflow, we suppose that services of the registry are grouped by functionality. When we use a graph structure, we either build the graph from provided files (as the ones of the WSC-09 synthetic benchmark) or we consider that the registry graph exists, considering that the graph building is out of the scope of our work. This is the same, when we use Petri Net.*

[8] `http://www.programmableweb.com/apis/directory`

[9] `http://uddi.xml.org/`

[10] Available at `http://www.it-weise.de/documents/files/wsc05-09.zip`.

2.5 Quality of Service (QoS)

The International Standards Organisation (ISO) defines quality as "the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs" [121]. Quality of Service (QoS) is employed for describing non-functional characteristics of services [262]. A discussion on QoS aspects for service-oriented systems is presented in [138].

QoS encompasses several criteria [119] such as: execution time, throughput, cost or reliability. Some criteria are positive or negative [264]: positive means that the higher the value, the better the quality (e.g. throughput or reliability), while negative means that the higher the value, the lower the quality (e.g. execution time or price). In the following, we only define the criteria used in our work. Definitions of the other criteria can be found in [255], for example.

QoS criteria apply both to single services and to composite ones [255]. When composite services are represented by a workflow, the QoS computation of a composite service depends on the patterns used. Authors of [255] and [119] present several aggregation functions (minimum, summation or multiplication). To simplify, in the following, we represent a composite service by a graph and present the aggregate functions we recalled in [20]. Let $\mathcal{P}_s$ be the set of paths in the graph representing a composite service s and let $S_c(s) = \{sc_1, ..sc_n\}$ be the set of service components composing s.

Definition 2.5.1 Execution time.
The execution time (also called response time) of a service s is the time $e(s) \geq 0$ (generally in seconds or milliseconds) needed by s to produce its outputs $out(s)$ from its inputs $in(s)$. When a service s is a composite one, its execution time is computed by summing the execution time of the components of the maximal path: $e(s) = \max\limits_{\mu \in \mathcal{P}_s} \left(\sum\limits_{sc_i \in \mu} e(sc_i) \right)$.

Definition 2.5.2 Throughput.
The throughput of a service s is the average rate $t(s) \geq 0$ of successful service executions for a given period of time. It is generally expressed by a number of successful executions per second. When a service s is a composite one, its throughput is the minimum throughput of its components:
$t(s) = \min\limits_{sc_i \in S_c(s)} t(sc_i) = \min\limits_{\mu \in \mathcal{P}_s} \left(\min\limits_{sc_i \in \mu} t(sc_i) \right)$.

Since 2009, the Web Service Challenge [135] has extended the syntactic service description with the both aforementioned QoS criteria, a WSLA [158] file storing the QoS (response time and throughput) criteria values of each service.

Definition 2.5.3 Cost.
The cost (also called price) of a service s is the fee a user has to pay to invoke the service. When a service s is a composite one, its cost is computed by summing of its component cost: $c(s) = \sum\limits_{sc_i \in S_c(s)} c(sc_i)$.

Definition 2.5.4 Reliability.
The reliability (also called successful execution rate) of a service s, denoted $r(s)$, with $0 \leq r(s) \leq 1$, is the probability of its successful execution. When a service s is a composite one, its reliability is computed by: $r(s) = \prod\limits_{sc_i \in S_c(s)} r(sc_i)$.

For real values of such criteria, the readers can refer to the QWS Dataset[11] [39], a set of QoS values extracted from 5,000 real web services, collected until 2008 or, to the WS-DREAM[12] [262], an evaluation of user-observed QoS of more than 30 million real-word web services from distributed locations, from 2009 to 2012. Authors of [157] had implemented a QoS registry collecting the QoS of all available services. Surveys can be found in [136, 247] for example.

QoS criteria can be considered one by one or can be aggregated. In [44, 255] and [3, 14], for example, a weight is assigned to each QoS criterion and a simple additive weighting technique [250] is used to assign a quality score to each service as follows:

Definition 2.5.5 QoS score.
Let $\{q_1(s)...q_k(s)\}$ be k QoS criteria associated with a service s, let $V_j(s)$ be the value of criterion $q_j(s)$ for service s and let $w_j \in [0,1]$ be the weight assigned to criterion $q_j(s)$ such that $\sum_{j=1}^{k} wj = 1$. Let Vj^{max} and V_j^{min} be respectively the maximal and the minimal value of criterion q_j and, Q_j^+ and Q_j^- be respectively the sets of positive and negative criteria. The QoS score of service s, denoted $Score(s)$, is computed by the following formula (see for example in [107, 250, 255]):

$$Score(s) = \sum_{Q_j^+} w_j * \frac{V_j(s) - V_j^{min}}{Vj^{max} - V_j^{min}} + \sum_{Q_j^-} w_j * \frac{V_j^{max} - V_j(s)}{Vj^{max} - V_j^{min}}$$

The higher the value of score $Score(s)$, the better the quality of service s.

Weight assigned to the QoS criteria can be defined by the user or assigned using different approaches. For example, the Best-worst multi-criteria decision-making method [200] is used in [213] to compute and normalize the QoS criteria weights.

Remark 2.5.1 *In our work, we suppose that the QoS values of services are available. How these values are extracted and stored is out of the scope of our work.*

2.6 Transactional properties

As explained by [49]: "To make service-oriented applications more robust, web services must be examined from a transactional perspectives". Surveys on transaction management in service-oriented applications can be found in [99, 220] and in our book chapter [18]. Authors in [70] propose a generic framework for testing web services transactions. As explained in [221], transaction management was initially proposed in the context of database in order to ensure consistency and reliability of data centric applications in case of failure (see in [110, 111]), and traditionally a transaction is a set of operations locally constituting an atomic execution unit. Transactions are one of the most fundamental concepts to deliver reliable applications [147]. The Two-Phase Commit (2PC) protocol [91] guarantees the ACID properties (Atomicity, Consistency, Isolation, Durability), defined in [110, 111, 115]. Several advanced transaction models (e.g. SAGA model [101]) have been proposed to extend these properties to distributed applications or long-running transactions – see in [78] or [80] for a synthesis.

Service-based transaction differ from traditional ones [149] and are not traditional ACID one due to the loosely-coupled, autonomous, and heterogeneous nature of the execution environment [99]. Several transaction models, protocols and standards for services have been proposed to relax some of the ACID properties [221]: we can mention

[11] `http://www.uoguelph.ca/~qmahmoud/qws/`

[12] `https://github.com/wsdream/wsdream-dataset`

OASIS Business Transactions Protocol (BTP) [71], Web Service Transaction [68] that includes Web Service Business Activity (WS-BA) [96] and Web Service Atomic Transaction (WS-AT) [151] or Web Services Transaction Management (WS-TXM) [67] for example – see [150] for a survey and in [70] for a comparison. Different transactional properties have also been associated to services (see in [18] for a survey), generally extending the classical ACID properties of database transactions [110, 111, 115] to services, by particularly relaxing the atomicity (i.e., all or nothing) property.

Before defining these transactional properties, we introduce several recovery techniques, we had defined in [18]. As Liu et al. [152], we consider, in [18], that the lifecycle of a service contains two phases: an active phase corresponding to the service execution (see blue lines in Figure 2.7) and a completed phase beginning after the end of the service execution. A failure can occur during the active phase. Depending on which life-cycle phase the service is in, different recovery techniques can be applied to preserve the relax atomicity (see Figure 2.7):

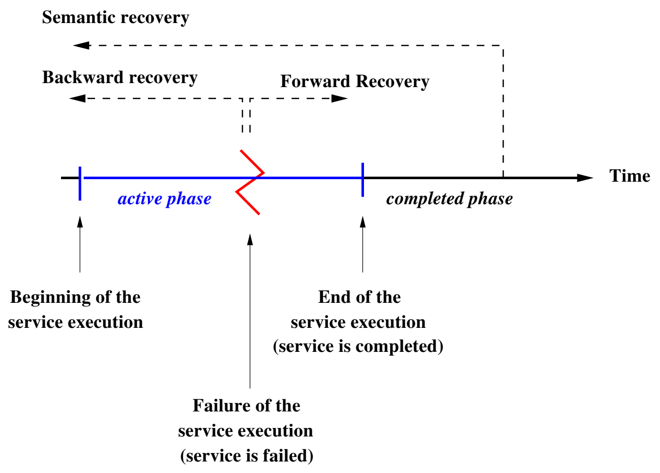
Semantic recovery
Backward recovery
Forward Recovery
Time
active phase
completed phase
Beginning of the service execution
End of the service execution (service is completed)
Failure of the service execution (service is failed)

Figure 2.7: The service lifecycle and the different possible recovery techniques.

Definition 2.6.1 Backward recovery.
After a failure has occurred during a service execution, a backward recovery consists in restoring the state that the system had at the beginning of the service execution or in reaching a state semantically closed to the state that the system had at the beginning of the service execution. A service supports backward recovery if, in case of failure during its execution, all the effects produced by the service before the failure can be rolled back or semantically undone.

Definition 2.6.2 Forward recovery.
After a failure has occurred during a service execution, a forward recovery consists in repairing the failure to allow the failed service to continue its execution.

In [80], authors said that "backward recovery backtracks to an earlier and correct state of the system before proceeding, while forward recovery attempts to correct the error before proceeding". Moreover, they said that some processes cannot simply undone and forgotten but require to execute further counteractions, also knows as compensations. Compensation, also called post facto process in [110], is generally associated with semantic recovery:

Definition 2.6.3 Semantic recovery.
After the end of a service execution, a semantic recovery consists in reaching a state, which is semantically closed to the state the system had before the service execution.

To support these recovery techniques, several transactional properties have been defined. We present below the main used ones, that are based on the multi-database system transaction model of [165]. These properties have been analysed, formalized or used, even recently, in different service approaches [61, 70, 97, 98, 147, 173, 196, 197] and, in our work, in [2, 3, 5, 15]. They can also be integrated in the WSDL interface or in the OWL ontology associated with the services [61, 173].

Definition 2.6.4 Pivot service.
A (non-composite) service s is pivot, denoted p, if its effects remain forever and cannot be semantically undone once it has completed successfully. If it fails, then it has no effect at all. Therefore a pivot service supports backward recovery. A completed pivot service cannot be rolled back and cannot be semantically undone.

As an example, let consider that to obtain an improved and cost effective handling of medical diagnostic imaging processes and infrastructures (e.g., PACS, teleradiology) involving large datasets, as explained in [216], doctors have to use irreversible image compression technologies and cannot store all original compressed images. A service providing an irreversible image compression technology (and deleting the original image after the compression) is pivot. Indeed, it can fails (compression could be stopped due to technical problem before the end) and it cannot be semantically undone (compression being irreversible, it is impossible to recover the original image which has been deleted after the compression process).

Definition 2.6.5 Compensatable service.
A (non-composite) service s is said to be compensatable, denoted c, if it exists another service s', or compensation policies, which can semantically undo the execution of s. Service s supports backward recovery and can be semantically undone while service s' has to guarantee a successfully termination.

As an example, a service providing reversible image compression technique (see for example [148]) is compensatable.

Definition 2.6.6 Pivot/Compensatable retriable service.
A (non-composite) service s is said to be retriable, denoted r, if it guarantees a successfully termination after a finite number of invocations. Therefore a retriable service supports forward recovery. The retriable property is never used alone but is combined with properties p and c defining pivot retriable service, denoted pr and supporting forward recovery, and compensatable retriable service, denoted cr and supporting both forward recovery and semantic recovery.

As an example, a service providing reversible image compression technique (see for example [148]) could be compensatable retriable.

Figure 2.8 presents the state diagrams of these transactional properties. Final states of the diagrams are represented by dotted lines.

A transactional (non-composite) service has the following definition:

Definition 2.6.7 Transactional (non-composite) service.
A (non composite) service s is transactional if it supports the aforementioned transactional properties p, c, pr or cr for its correct usage. Let $TP(s)$ be the transactional property of a non-composite service s: $TP(s) \in \{p, pr, c, cr\}$.

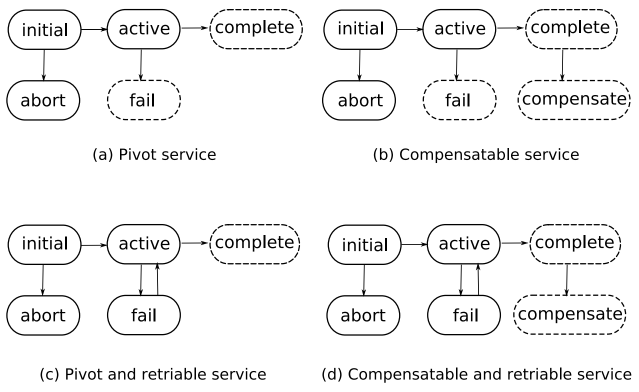

Figure 2.8: The state diagrams of transactional properties p, c, pr and cr.

In [3], we have extended the pivot, compensatable and retriable properties to composite services, defining atomic, compensatable, retriable, and transactional composite service:

Definition 2.6.8 Atomic (composite) service.
A composite service s is said to be atomic, denoted a, if once all its component services complete successfully, their effect remains forever and cannot be semantically undone. On the other hand, if one component service does not complete successfully, then all component services, that are successfully executed previously, have to be compensated. Therefore, an atomic composite service supports the same criteria than a pivot non-composite one, except that the backward recovery of an atomic composite service allows to reach a state of the system semantically closed to the state that the system had at the beginning of the service execution.
A similar definition is used by [154]: "A composite service is atomic if it can be treated as a unit of work. That is, it can use compensation mechanism to ensure that either all of its component services complete successfully or none of them do".

Definition 2.6.9 Compensatable (composite) service.
A composite service s is said to be compensatable, denoted c, if all its component services are compensatable.

Definition 2.6.10 Atomic/compensatable retriable (composite) service.
A composite service s is said to be retriable, denoted r, if all its component services are retriable. The retriable property is combined with properties a and c defining atomic retriable (ar) composite service (supporting the "same" criteria than a pivot retriable non-composite service) and compensatable retriable (cr) composite service (supporting the same criteria than a non-composite compensatable retriable service).

Using the aforementioned definitions, we can deduce that the characteristics of transactional property p or a (i.e., no effect or effects forever) are included in the characteristics of the other transactional properties. The characteristics of transactional property pr or ar (i.e. forward recovery) are included in the characteristics of property cr, and the characteristics of transactional property c (i.e. semantic recovery) are included in the characteristics of property cr.

Definition 2.6.11 Transactional composite service.
A composite service s is transactional if its transactional property $TP(s)$ is in $\{a, ar, c, cr\}$.

The transactional property (TP) of a composite service depends on the transactional property of its components and on the execution order of these components. In [3], we define some rules to define the transactional property of a composite service, recalled in Table 2.2. For example, a pivot or an atomic service s_1 cannot be composed with any atomic, pivot or compensatable service s_2, as rule 1 of Table 2.2 dictates (s_1 being executed before s_2 or in parallel with s_2). Indeed, if s_2 fails then s_1, being pivot (resp. atomic), cannot be compensated, making therefore the composite service, composed by s_1 and s_2, non transactional. Moreover, if s_1 and s_2 are executed in parallel and if s_1 is atomic or pivot, s_2 cannot be neither atomic retriable nor pivot retriable. Indeed, s_1 can failed requiring therefore a compensation of s_2 to preserve the transactional property of the composite service.

Table 2.2: Transactional rules of [3]

Transactional property of a service	**Sequential incompatibility**	**Parallel incompatibility**
a, p	a, c, p (rule 1)	a, p, c, ar, pr (rule 2)
pr, ar	a, p, c (rule 3)	a, p, c (rule 4)
c		a, p, ar, pr (rule 5)
cr		

These rules have been also represented by a automaton (see Figure 2.9) containing five states. State I is the initial one, representing an empty service (with no component). The final states c, cr, a, and ar correspond to the transactional properties of a composite service. When one of the final states is reached, a transactional composite service is obtained. The alphabet of the language accepted by the automaton is $\{'p', 'c', 'cr', 'pr', ';p', '//p', ';pr', '//pr', ';c', '//c', ';cr', '//cr', ';a', ';ar', '//ar'\}$. This alphabet represents the transactional properties of component services (which can be either elementary services or composite ones) executed in sequence (;) or in parallel (//). Thanks to this automaton and Table 2.2, we can notice that a composite service sc can only contain one pivot non-composite service or one atomic composite one.

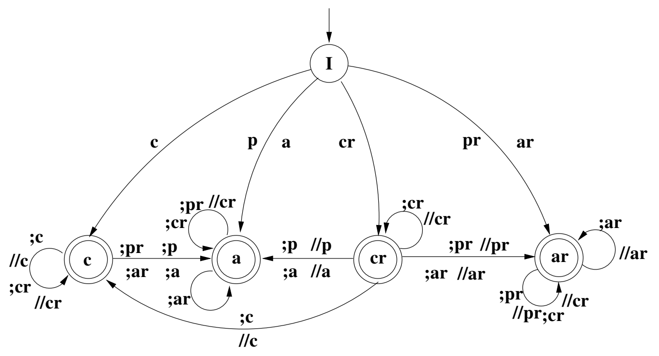

Figure 2.9: An automaton modelling the resulting TP of a composite service depending on its components' TP.

Several authors had extended the aforementioned transactional property rules by considering *switch* and *loop* workflow patterns in [241], *if* and *while* in [93] and mutually exclusive choice in [85]. Authors in [196] have defined another property: cancelable. A cancelable service being a service that can be cancelled by an external entity during its execution. This property is associated with properties p, a , c and r. Authors in [93] consider property r alone, without associating it with p or c.

Remark 2.6.1 *In our work, we have only considered transactional properties a, p, ar, pr, c and cr. We consider that these properties characterize the services we used in our approaches.*

2.7 Life-cycle of service composition

"A service composition is a process that combines a set of logically related services to achieve a given goal" [228]. This process can be divided in different phases or steps [120, 215]. In [3, 18], we decomposed the composition process into a three-steps process: query specification (also called definition phase in [215]), the component selection step and the execution phase.

The first step consists in specifying a composition query, request or composition requirement, which describes the expected resulting composite service. This composition query depends on the service registry structure (see Section 2.4) and on the service description (syntactic, with or without QoS and transactional properties). In the Web Service Challenge [50, 135], for example, where services are syntactically described (see Section 2.2) and are associated with two QoS criteria, the query is a tuple containing the set of known concept instances, a set of concepts which should be found and a registry.

Definition 2.7.1 Composition query.
Let be $\mathcal{Q}$ a query specifying the composite service to reach. When services of the registry are grouped by functionality, $\mathcal{Q} = (WF_\mathcal{Q}, W_\mathcal{Q}, TP_\mathcal{Q})$, with: $WF_\mathcal{Q}$ an abstract workflow (see Def. 2.3.2), $W_\mathcal{Q}$ a set of weights associated to the QoS criteria considered for the services (see Def. 2.5.5), and $TP_\mathcal{Q}$ the transactional property (see Section 2.6) of the expected resulting composite service. When the service registry is organized by a graph (see Def. 2.3.3 to 2.3.4), $\mathcal{Q} = (I_\mathcal{Q}, O_\mathcal{Q}, TP_\mathcal{Q})$, with: $I_\mathcal{Q}$ a set of known service inputs, $O_\mathcal{Q}$ a set of expected outputs and $TP_\mathcal{Q}$ the transactional properties (see Section 2.6) of the expected resulting composite service (when services are described by transactional properties).

The second step of the service composition process is the component selection. It consists in searching/discovering/finding, in the registry, the services that can be combined to compute the expected composite service described by the composition query. The composite service obtained after this step should match the composition query $\mathcal{Q}$. The component selection can be either top-down or bottom-up [54]. In top-down selection approach, the composition logic is created on a high level and the abstract model is refined to a service composition by selecting the appropriate candidate services (e.g. one per activity of the abstract workflow) [54]. In bottom-up selection, candidate services fulfilling the query goal are selected and the composition logic (or control flow) is deduced from the selected component services [18].

Definition 2.7.2 Top-down component service selection.
When $\mathcal{Q} = (WF_\mathcal{Q}, W_\mathcal{Q}, TP_\mathcal{Q})$ and when services are organized into functionality communities (see Def. 2.4.1), the component service selection is denoted as top-down and consists in choosing, from the registry, one service s_i per activity A_i of workflow $WF_\mathcal{Q}$. The set $S_c(s)$ of component services of resulting composite service s is such that: $S_c(s) = \{s_1, ..., s_n\}$, s_i performing activity A_i of workflow $WF_\mathcal{Q}$, $\forall i \in \{1, ...n\}$.

Definition 2.7.3 Bottom-up component service selection.
When $\mathcal{Q} = (I_\mathcal{Q}, O_\mathcal{Q}, TP_\mathcal{Q})$ and when services are organized into a graph (see Def.

2.3.3 to 2.3.5), the component service selection is denoted as bottom-up and consists in defining the sub-graph G_c that allows to obtain outputs $I_{\mathcal{Q}}$ from inputs $I_{\mathcal{Q}}$. The resulting composite service s, represented by sub-graph G_c, is such that: $in(s) \subseteq I_{\mathcal{Q}}$ and $out(s) \supseteq O_{\mathcal{Q}}$.

When several candidate services meet the requirements, the best matched services need to be selected [215]. When services are described by QoS criteria, the best matched services are the ones that optimize the QoS of the resulting composite service. The composition problem is then called QoS-aware service composition.

Definition 2.7.4 QoS-aware service selection.
Depending on the considered QoS criteria (see Section 2.5), the QoS-aware service selection consists in selecting the component services that either minimize the considered QoS criterion (e.g. execution time or cost) or maximize it (e.g. reliability, throughput or QoS score).
The Web Service Challenge [50, 135] provides test sets to experiment QoS-aware service selection approaches.

When services are described by transactional properties, the best matched services are the one whose transactional properties allow to reach the expected transactional property of the resulting composite service. In this case the composition problem is transactional-aware.

Definition 2.7.5 Transactional-aware service selection.
The transactional-aware service selection consists in selecting service components in order to obtain a composite service s which transactional property $TP(s)$ is such that [5]:

if $TP_{\mathcal{Q}} = a$ then $TP(s) \in \{a, ar, c, cr\}$,
if $TP_{\mathcal{Q}} = ar$ then $TP(s) \in \{ar, cr\}$,
if $TP_{\mathcal{Q}} = c$ then $TP(s) \in \{c, cr\}$,
if $TP_{\mathcal{Q}} = cr$ then $TP(s) = cr$.

Indeed, the resulting composite service s must provide at least the transactional support required by query $\mathcal{Q}$ (see Section 2.6 for the characteristics of transactional properties).

Definition 2.7.6 QoS and Transactional-aware service selection.
The QoS and transactional-aware service selection consists in simultaneously considering QoS optimization and transactional property support.

The last step of the service composition process is the service execution. During this phase, an execution engine (either centralized or distributed) is responsible to invoke all selected component services [215]. During the composite service execution, component service can failed or the QoS of a component service can change (in such way that the resulting QoS of the composite service becomes no more optimal). In this case, recovery techniques should be used to retrying the failed component service, compensating it or executing an alternative service [8]. A service execution is fault-tolerant when it allows one or several of the aforementioned recovery technique to ensure a reliable execution of a composite service. When the choice of the strategy is automatically done, the service execution is denoted dynamic or self-healing.

Definition 2.7.7 Self-healing fault-tolerant composite service execution.
A fault-tolerant composite execution is self-healing when in case of a component service failure or in case of QoS change (even during failure free execution), the most appropriate recovery technique (retrying the failed component service, compensating it or executing an alternative service) is automatically chosen.

2.8 Overview of our work

The research work presented in this manuscript deals with service management. Table 2.3 summaries our contributions. A big part of our work concerns top-down or bottom-up component service selection. Table 2.4 positions our different component service selection approaches and surveys, according to the concepts defined in this chapter. These approaches are more detailed in Chapters 3 and 4. We have also proposed several approaches allowing service execution or service search. Table 2.5 positions these approaches, complementary to our service selection approaches and presented in Chapter 5, according to the concepts defined in this chapter.

Table 2.3: Summary of our contributions

Ref.	**Description**
[12] and its extended version [3]	Heuristic algorithm for Top-down Transactional and QoS-aware service composition with formal definitions and proofs of the transactional properties.
[14]	Linear program for Top-down Transactional and QoS-aware service composition.
[4]	Complexity analysis and models comparison of Top-Down QoS-aware service composition.
[10]	Heuristic algorithm for Bottom-up Transactional-aware service composition.
[2, 9]	Heuristic algorithms for Bottom-up Transactional and QoS-aware service composition.
[15]	Linear program for Bottom-up Transactional and QoS-aware service composition.
[1]	Survey and comparison of service composition approaches (component service selection and execution steps) based on Petri Net.
[20]	Complexity analysis and models comparison of Bottom-up service composition based on ServiceData and Dependency graph.
[18]	Survey and comparison of transactional and transactional-QoS (Top-down or Bottom-up) service composition approaches (component service selection and execution steps).
[13]	Transactional model for composite service execution.
[5]	Framework for QoS-transactional service selection based on crowdsourcing.
[6, 8]	Fault-tolerant self-healing composite execution model.
[16, 17]	Framework for semantic service search.
[11]	Fuzzy atomicity measure for composite service execution.

Table 2.4: Characteristics of our component service selection approaches

Ref.	**QoS-aware**	**Transactional-aware**	**Transactional and QoS-aware**	**Selection type**	**Considered QoS**
[3, 12, 14]			X	Top-down based on workflow	QoS Score
[4]	X			Top-down based on workflow	Cost, execution time and reliability
[9]			X	Bottom-up based on Colored Petri Net	Execution Time
[10]		X		Bottom-up based on Petri Net	
[1, 2]			X	Bottom-up based on Colored Petri Net	QoS Score
[15]	X		X	Bottom-up based on ServiceData graph	QoS Score and execution time and throughput
[20]	X			Bottom-up based on ServiceData graph	Cost, execution time throughput and reliability

Table 2.5: Characteristics of our complementary approaches

Ref.	**Concepts used**
[13]	Workflow and composite service execution
[5]	Transactional properties, QoS and service selection
[8]	Petri Net, transactional properties, QoS and composite service execution.
[16, 17]	Syntactic service description and service discovery.
[11]	Transactional properties and composite service execution.

Chapter 3

Top-down Transactional and QoS-aware service selection

As explained in [102], a composite service (see Def. 2.3.1) is similar to a workflow (see Def. 2.3.2 p. 8). There is a long history between web services issues and workflow community [265]. Industry and business solutions are mainly workflow-based because of the WSDL service interface definition, but several RESTful composition approaches (35% of the approaches analysed in survey [102]) have also adopted a workflow view. Workflows are particularly adapted for several application domains as e-Science [84] and image processing [109, 199] for example.

This chapter addresses the QoS and Transactional-aware service selection based on workflow (see Def. 2.7.2) and presents our approaches and surveys published in [3, 12] and in [4, 14]. The chapter is organized as follows. Section 3.1 defines the component service selection problem when workflows are used. Section 3.2 describes the heuristic approach we have proposed in [3, 12]. Section 3.3 concerns the approach we had proposed in [4, 14] using linear programming. Section 3.4 presents our complexity analysis of the problem. Finally, Section 3.5 deals with the limits of our work and highlights some future work.

3.1 Problem definition

As explained in Def. 2.7.2 (p. 18), the query $\mathcal{Q}$ for a Top-down Transactional and QoS-aware service selection is composed by an abstract workflow, denoted $WF_{\mathcal{Q}}$.

In our work, we only consider the three more frequent patterns: sequence, parallel (AND) and exclusive choice (XOR) – see Figure 2.2. These patterns can be recursively concatenated and interlaced.

As an example, let consider the input abstract workflow represented in Figure 3.1. This workflow contains 7 activities for content-based image retrieval i.e., for retrieving the images which are considered to be similar to a query image. After a first activity of image analysis (activity A_1), the process consists in three exclusive choices (using pattern XOR). The first branch of the exclusive choice pattern contains three activities: A_2 to decompose the image into a quad-tree – see for example Figure 3.2.(c) – and two parallel activities (using pattern AND), A_3 and A_4, to compute a similarity based on the level 1 and level 2 sub-images resulting from the quad-tree decomposition, as the ones we proposed in [23, 25]. The second branch contains only one activity A_5 which computes a similarity based on the image color histogram, as the ones presented in

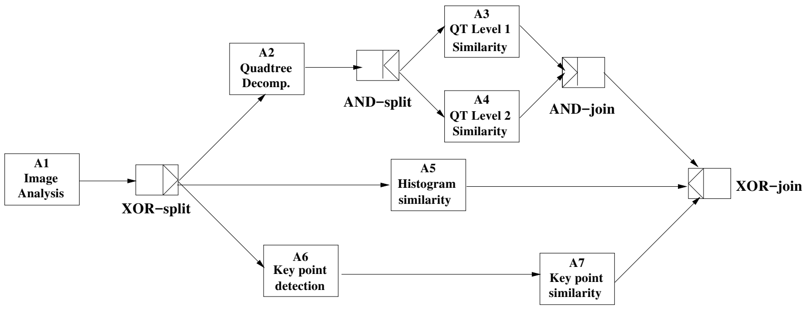

Figure 3.1: An abstract content-based image retrieval workflow.

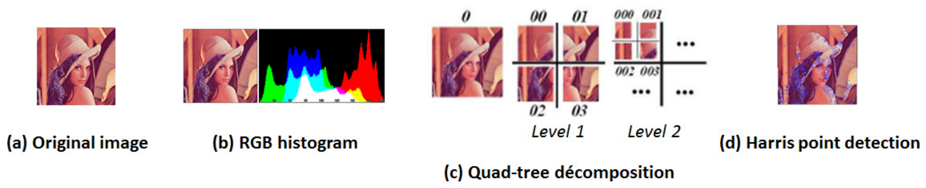

Figure 3.2: Several examples of image feature extraction.

[185] – see Figure 3.2.(b)[1] for a color histogram example. The last branch contains two activities: A_6 which detects image key points – see for example Figure 3.2.(d) [2] – and A_7 which computes a similarity based on the spatial relationships between the detected salient points of the image is computed, as we did in [21].

The input of the Top-down QoS and Transactional-aware service selection process contains query $\mathcal{Q}$ and the service registry, where services are organized into functionality communities (see Def. 2.4.1). Figure 3.3.(a) represents the same example of the input abstract workflow of Figure 3.1. Figure 3.3.(b) represents a part of the service registry, each service being clustered in one functionality class (functionalities are represented by color in the Figure), each workflow activity being associated with one class. The output of the selection process is a composite service represented by a concrete workflow (see Def. 2.3.2), activities being performed by services (one service per activity for sequence and AND patterns, and one service per activity for only one branch of the XOR patterns). In our example, several concrete workflows can appear depending on the XOR pattern branch selected. Figure 3.3.(d) represents an example of 3 composite services, each one representing an instantiation of each branch of the XOR pattern.

Definition 3.1.1 Top-down QoS and Transactional-aware composition problem.
The Top-down QoS and Transactional-aware composition problem consists in simultaneously [4]: (i) determining an end-to-end route (represented by a concrete workflow) and (ii) selecting one service per activity of the selected end-to-end route, such that the resulting composite service is transactional (see Def. 2.6.11) and the QoS selected criteria (see Section 2.5) are optimized. The end-to-end route depends on workflow patterns. It is a path from the first vertex of the workflow to the last one, containing

[1]From `http://opensource.graphics/tag/histogram/`

[2]From `https://mvassilevich-test.googlecode.com/hg/src/python/scikits.image/DOC/html/auto_examples/plot_harris.html`

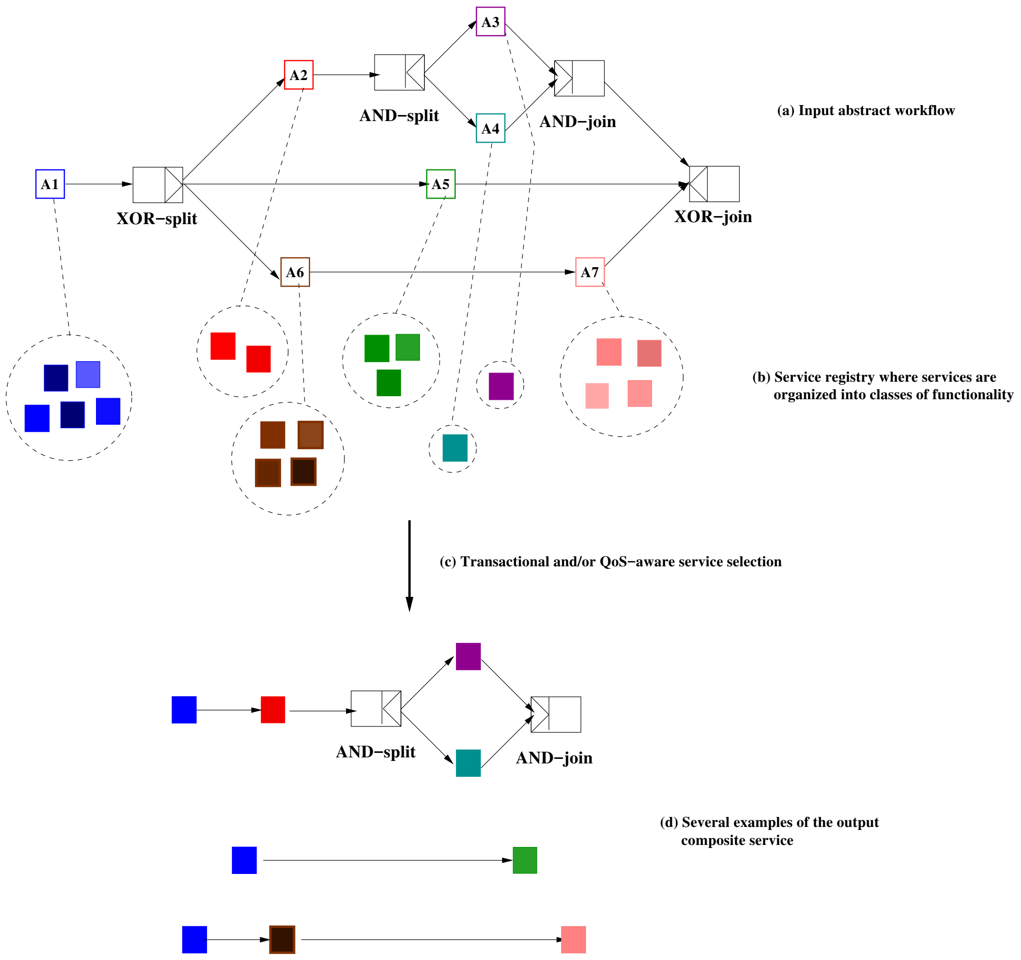
A1
A2
A3
A4
A5
A6
A7
XOR−split
AND−split
AND−join
XOR−join
(a) Input abstract workflow
(b) Service registry where services are organized into classes of functionality
(c) Transactional and/or QoS−aware service selection
AND−split
AND−join
(d) Several examples of the output composite service

Figure 3.3: The top-down service selection process.

all branches of each AND pattern belonging to the path, and containing exactly one branch of each XOR pattern belonging to the path.

We tackle this problem using different approaches: an heuristic one, presented in Section 3.2 and an approach based on linear programming, presented in Section 3.3. We also analysed the complexity of the problem, as recalled in Section 3.4.

3.2 Heuristic approach

In [12], and in its extended version [3], we had proposed an heuristic approach. It consists in selecting component services of a transactional composite service represented by a workflow, where each selected component service locally optimizes the QoS, computed by a score (see Def. 2.5.5). This algorithm is detailed in the next subsection.

3.2.1 Description

The inputs of our heuristic approach are: an abstract workflow and a service registry, where each service is described by its functionality, its transactional property and a

QoS score. In our algorithm, the input abstract workflow is analysed vertex per vertex, incrementally building a composite service. At each step, the transactional property (TP) of the current composite service (composed by the components selected at the previous steps and representing a sub-part of the final composite service) is computed. At the beginning of the algorithm, the transactional property of the current composite service (which has no component) is I, the root of the automaton of Figure 2.9 (p. 17). This transactional property is updated after each component selection. Indeed, the transactional property of a component determines the possible transactional properties that services, selected during the next steps (called candidate services), could have. For example, as soon as a pivot or an atomic component service is selected, only retriable services can be selected for the next activities of the workflow (see Automaton of Figure 2.9).

When the current vertex is an activity A_i, the algorithm selects the service, that: (i) can performed activity A_i (i.e., which is member of the corresponding functionality class associated with A_i), (ii) have the best QoS (in our case the maximum QoS score among the services of the class) and (iii) whose transactional property respects the rules recalled in Automaton of Figure 2.9.

When the current vertex of the analysed workflow is an AND-Split pattern, the algorithm recursively selects services that locally optimize the QoS for all the activity of all branches of the pattern, each branch being considered as a sub-workflow i.e., a sub-composite service. The transactional property of the composite service represented by the first branch of the pattern, after the service selection of its activities, determines the authorized transactional properties of the composite services represented by the other branches of the pattern.

When the current vertex of the analysed workflow is a XOR-Split pattern, the activities of all branches of the pattern are associated with services and the algorithm selects the branch whose corresponding composite service has the best QoS.

3.2.2 Related work

As far as we know, when our articles have been published, our approach was the first one that not only fulfilled the global transactional property of the composite service but also guaranteed to have locally the best QoS component services. A recent survey [93] seems to show that it was always the case in 2013 (see Table 3 in [93]). When our articles have been published, most approaches only dealt with QoS-aware selection [43, 123, 134, 239, 255, 256] (see e.g., surveys in [114, 218]), but did not consider transactional properties. Other approaches [61, 62, 147, 161, 173, 174, 192] took into account transactional properties but were generally based on a model, the Acceptable Termination States (ATS), that implies to explore all the possible end-to-end route making the approaches neither simple or scalable. Few approaches had proposed QoS and transactional-aware service selection [155, 240] but by only considering the compensatable property. A comparison of these approaches is presented in our survey [18].

In [223], authors propose an extension of our approach introducing two new patterns (if and while). They define a genetic algorithm to determine a near-optimal transactional composite service. However, they conduct experiments on only small size instance (6 activities and 72 services), while our approach has been experimented on much more bigger data sets (more than 40 activities and 3000 services) to prove our approach scalability. More recently, authors of [241] propose an approach based on ant colony optimization for transactional and QoS-aware selection. They compare their approach to our and extend our transactional rules to switch and loop patterns. Their experiments show a better optimality than our approach (because the globally optimized the QoS while our heuristic locally optimizes it) but with lower computational time results. Authors of [196] extend our transactional properties, adding a cancellable one, and proposed a dynamic service selection which only considers transactional properties without considering QoS. Authors in [196] propose a genetic algorithm for Top-down Transactional and QoS-aware service selection. They extended the automaton we had proposed in [3], by adding a mutually exclusive patterns and by considering the retriable property without associating it with the pivot or compensatable one. Their approach allows a global QoS optimization.

3.3 Linear program

Our approach presented in [3, 12] locally optimizes the QoS. In order to globally optimize the QoS, we had proposed new linear programs for Top-down QoS and Transactional-aware service composition in [14], optimizing a classical weighted sum of QoS criteria, and for Top-down QoS-aware service composition in [20], where cost is minimized while execution time and reliability constraints are satisfied. This section describes these linear programs and compares it with the related works.

3.3.1 Description

As explained in [191]: "When practical problems are formulated as combinatorial optimisation models one must often include logical implications between decisions. It is useful to express these implications as linear constraints involving binary variables, since linear constraints offer the possibility of using linear programming and branch and bound as an initial solution method." We therefore decide to represent the service composition problem by a 0-1 linear program.

As we remarked in [14, 20], when some global QoS constraints or global transactional constraint have to be satisfied, services cannot be chosen independently for each activity of the workflow. Thus, it is not possible to decompose the composition problem into two stages which consist in, firstly, in selecting an end-to-end route, and secondly selecting the best service for each activity of this end-to-end route. The choice of the end-to-end route and services is linked. In our models, decision variables are defined for both activities and services selection.

A workflow can be represented by a graph $G = (V, E)$ whose vertex are activities and patterns. The inputs of our linear program-based models contain the graph G, representing the workflow and the lists of services which can performed the different activities of the workflow. In [14], services are described by their QoS score (see Section 2.5) and their transactional property. In [20], services are described by a set of vectors containing the values of three QoS criteria (cost, execution time and reliability). Using this inputs, we define different kinds of constraints: constraints induced by the workflow (describing the workflow structure), those induced by the QoS constraints, those resulting from the transactional properties and finally those modelling the selection constraints (e.g., a selected activity must be realized by exactly one service).

The QoS and transactional-aware model of [14] has been compared with our heuristic approach of [3, 12]. In our experiments, the running time of our linear program (using CPLEX solver[3]) was always better than the one of our heuristic approach whereas our linear program always determines the exact solution. Our QoS-aware model of [20] also offers promising computational experiments, showing that composition problem with sum-type and a single QoS criterion to optimize is tractable, even for large size instances (more than 200 activities with 100 services per activities). However, the same problem with an additional QoS constraint is more difficult to solve for large size instances. Graph reduction has to be conducted (since a workflow is a series-parallel

[3]`http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`

graph [87] as we explained in [20]) in order to simplify the linear formulation of the problem.

3.3.2 Related work

The workflow-based QoS-aware service composition problem has received a lot of interest (see e.g., surveys in [114, 124, 168, 218]). As shown in [20], concerning the workflow structure, two classes of workflow are usually considered.

In the first one, the workflow induces only one end-to-end route (this is the case when the workflow contains SEQ and AND patterns only) then the problem is to select one WS for performing each activity of this end-to-end route. In the large majority of the studies considering this class of workflow, the QoS-aware service composition problem is modelled as a Multidimensional Multi-choice Knapsack Problem (MMKP) (see e.g., [253, 254]). An heuristic is proposed to solve this problem in [60].

In the second class, there exist several possible end-to-end routes (this is the case when the workflow contains XOR patterns) then the composition problem simultaneously induces the selection of one end-to-end route, and the selection of one WS for each activity belonging to the selected end-to-end route. Authors in [255] claim that all end-to-end routes of the workflow must be generated in order to find the best composite WS. Therefore, they propose to decompose the problem into two steps: (step 1) an end-to-end route is chosen, and (step 2) they propose a MIP formulation to select one WS per activity belonging to the chosen route in order to maximize QoS criteria. With such a decomposition, the obtained solution is not necessarily optimal. Authors in [264] for example have always found better results than [255], using SAT-based cost planning solver. In [254], the proposed model contains as many constraints as end-to-end paths in the workflow. Therefore, as the number of end-to-end paths grows exponentially with the workflow size, some large-size instances may not be formulated and even less solved. Thus, authors propose to find near-optimal solutions in polynomial time with an approximate algorithm. The real challenge is then to include into the same optimization problem the selection of activities and the selection of WS for performing each activity structured by a complex workflow. That is what we do in our approaches.

Compared with graph approach [156, 254], we do not have to enumerate all the potential end-to-end routes in the workflow and the number of variables and constraints of our proposed model is polynomial in function of the number of activities and WS per activity.

3.4 Complexity analysis

To the best of our knowledge, before our article [4], the theoretical complexity of the top-down service composition problem had not been extensively studied. Only the connection between this problem and the MMKP, which theoretical complexity is well known, had been shown [48]. More recently in [11], Goldman and Ngoko propose a polynomial-time algorithm to determine the composite WS with minimal average execution time. Their algorithm is based on a recursive reduction of the workflow.

As we show in [4], the theoretical complexity can be analysed more precisely in regard to the structural property of the considered workflow. More precisely, we have proved that the composition problem can be solved in polynomial time when optimizing a sum-type QoS criterion (for cost or QoS score for example) or a product-type QoS criterion (for reliability for example) on a workflow (with interlaced patterns SEQ, AND and XOR). When a min/max-type QoS criterion is used (for execution time for example), the problem can be solved in $O(m)$, with m the number of leaves in the decomposition tree associated with the serie-parallel graph representing the workflow (based on the decomposition algorithm of [229]). We also have remarked that if the workflow only contains SEQ and AND patterns (without exclusive choice of activities) the problem of determining the composite WS with minimal execution time is simplified and can be formulated as a longest path problem between the source and the sink of serie-parallel graph associated with the workflow. Thus, considering only one QoS criterion to optimize, the top-down service composition problem is easy to solve (even for more complex criterion like execution time). The computational difficulty comes from QoS constraints (e.g., the execution time has to be less than 5 s). Indeed, let us recall that, even for workflow with SEQ patterns only, Yu and Lin show in [253] that the composition problem with one QoS constraint is exactly the Multiple-choice Knapsack Problem which is known to be NP-hard. Recently, authors in [37] also show the NP-hardness of the problem and also establish that it is solvable in polynomial time for one criterion for complex workflow (including OR-pattern).

3.5 Conclusion and prespectives

To the best of our knowledge, the heuristic we had proposed in [12] and in [3], and the linear program we had defined in [14], are both the first approaches allowing Transactional and QoS-aware top-down service composition. The model we had proposed in [20] for QoS-aware top-down service composition is very interesting since it avoids the enumeration of all end-to-end routes and since the number of variables and constraints is a linear function of the workflow size. Moreover, in [20] we have more precisely analysed the theoretical complexity in regard to the structural property of the considered workflow.

However, in our work, we only consider three patterns (sequence, parallel and exclusive choice). A first perspective concerns the extension of our approach to others patterns (conditional pattern, loop ...).

As shown by our experiment results in [20], the computation time of global QoS-aware top-down service composition is more important when several QoS constraints are considered. A second perspective concerns a study on graph reduction (since the graph associated to the workflow is series-parallel) in order to simplify the mixed integer linear formulation. On the other hand, it would also be useful to propose approximate algorithms to find solutions very rapidly and to compute the distance between the optimal solution (that can be found by our global approach) and the approximate one and thus to obtain an experimental approximation ratio.

In [3, 12], we assume that it always exists one service of each transactional property for each task of the workflow, which can be a strong hypothesis as said in [97]. This limit does not exists any more in [14, 20] since approach is global. Authors in [97] and [98] also argue that our approach of [3, 12] is limited to stateless (non conversational) services. As explained in [172], terms such "conversational" and "stateless" are widely used in the taxonomy of web services. A stateless service is considered as black box. At the opposite, a conversational service is composed of a set of operations over which a workflow is specified [74]. Authors in [97, 98] considered that only [48] and [171] offer Top-down QoS-aware conversational (also called statefull) service composition, and that they are the first ones considering both QoS and transactional properties. However, if as in [74], we consider each operation of a conversational service as an elementary service, therefore as in [244] we can treat stateful/conversational and stateless services in a uniform way. Moreover, authors [172] argue that "it is possible to have a stateful implementation of a stateless service specification."

Finally, our work concerns what authors in [215] define as static composition. Indeed, the composition of service is done at design time. As explained in [215]: "Static

Web services composition works well if the business partners involved in the process are relatively fixed, and service functionalities or composition requirements do not, or rarely, change." A perspective concerns that adaptation of our work to runtime composition. In [19], a preliminary version of context-based transactional top-down service composition is presented. In this paper, we are addressing a service composition problem in a context-aware setting when user and service context changes (e.g., services becoming busy or unavailable) while the business process is executing.

In Top-down approach, a service is selected for each activity of the workflow. Links between two consecutive services in the workflow are not considered or verified. At the opposite, in Bottom-up approaches, services are linked by their inputs and outputs. The next chapter presents our bottom-up approaches, where instead of predefining an abstract workflow for the selection of component services, our approaches consider the service registry as a big graph where services are syntactically described (see Def.2.2.1 - pp. 6) and deduce the composition logic (or control flow) from the selected component services.

Chapter 4

Bottom-up Transactional and QoS-aware service selection

When the Transactional and QoS-aware service selection is Bottom-up, instead of providing a workflow, the query contains a set of inputs and outputs (a syntactical service description - see Section 2.2), and a composite service is automatically designed, from the available services (represented by a given graph), to produce the outputs from the provided inputs. As explained in [76], in this case, the automatic service composition is considered at operational level and services are regarded as black-box that accept inputs to produce outputs. This kind of service composition problem has spawned a synthetic benchmark, the Web Service Challenge (WSC) [50, 135, 238], to allow an evaluation of service composition approaches. There is also a growing interest of this kind of problem in the field of image processing (see for example [129, 166]).

This chapter deals with our approaches published in [2, 9, 10, 15, 20]. The chapter is organized as follows. Section 4.1 defines the Bottom-up service composition problem. Section 4.2 describes the heuristic approaches we had proposed in [2, 9, 10]. Section 4.3 concerns the approach we had proposed in [15] using linear programming. Section 4.4 presents our complexity analysis of the problem and the formulation, we proposed in [20], in terms of scheduling problem with AND/OR constraints. Finally, Section 4.5 deals with the limits of our work and highlights some future work.

4.1 Problem definition

As explained in Def. 2.7.3 (p. 18), in Bottom-up service composition, the query $\mathcal{Q}$ contains two sets of parameters, one for inputs and one for outputs ($I_\mathcal{Q}$ and $O_\mathcal{Q}$ respectively), and the service registry is generally represented by a graph G (see Section 2.4). Therefore:

Definition 4.1.1 Bottom-up QoS service composition problem.
As we define in [20], the Bottom-up QoS and Transactional-aware composition consists in defining a composite service s such that:

1. (Feasibility part of the problem) $in(s) \subseteq I_\mathcal{Q}$ and $out(s) \supseteq O_\mathcal{Q}$, and
2. (Optimality part) the resulting composite service is transactional and it optimizes the QoS (e.g., minimizing the execution time or maximizing the throughput).

The graph representing the service registry is an input the service selection process. When the service registry is represented by a bipartite graph $G = (V, U)$ (see Def. 2.3.3

and 2.3.4 p. 9), with $V = S \cup D$, the feasibility part consists in finding a sub-graph $G^c = (V^c, U^c)$ of G, with $V^c = D^c \cup S^c$, such that:

- Vertices $\in D^c$ with no predecessor correspond to the vertices associated with inputs in I_Q in G and vertices $\in D^c$ with no successor correspond to the vertices associated with outputs in O_Q.
- If a vertex $i \in (D \setminus I_Q)$, and $i \in V^c$, then at least one arc $(j, i) \in U$, with $j \in S$, belongs to U^c (it is due to the fact that each data $i \notin I_Q$ has to be produced by at least one service).
- If a vertex $i \in S$, and $i \in V^c$, then all arcs $(j, i) \in U$, with $j \in D$, belong to U^c (it is due to the fact that each service i can be executed if and only if all its inputs data are available).
- G^c does not contain any directed cycle.

To better understand the problem, let us consider the simple example of service registry we used in [20], represented in Table 4.1, which contains 9 services and 9 data.

s	1	2	3	4	5	6	7	8	9
$in(s)$	j	k, i	l, m	m	m	i	j	n, i	l
$out(s)$	m	j, m	n	i	i, q	r, q	l	t	n, i

Table 4.1: An example of a simple service registry

The bottom-up service composition problem, using this registry, with $I_Q = \{j, k\}$ and $O_Q = \{q, t\}$, is represented in Figure 4.1. Depending on the transactional properties and on the QoS criteria (not represented in the figure), several resulting composite services are possible. Figure 4.1 only represents the resulting composite services that resolve the feasibility part of the problem.

We tackle this problem using different approaches: heuristic ones, presented in Section 4.2 and an approach based on linear programming, presented in Section 4.3. We also analysed the complexity of the problem and propose an original formulation of the problem, as recalled in Section 4.4.

4.2 Heuristic approach

We had proposed heuristic methods based on Petri Net for bottom-up transactional service composition in [10] and for transactional and QoS-aware service composition in [9, 2]. All these approaches adapt a Petri-Net unfolding algorithm to select component services.

4.2.1 Description

In [2, 10], the Colored Petri Net (CPN) (see in [263] for a description of CPN) formalism is extended to incorporate the transactional properties: the color function associated with transitions and tokens (which describe the selection process progress) represents the transactional properties of the current selected composite service. Our approaches use CPN model because it allows to describe not only a static vision of a system, but also its dynamic behaviour; it is expressive enough to capture the semantics of complex service combinations and their respective interactions. Such structure allows

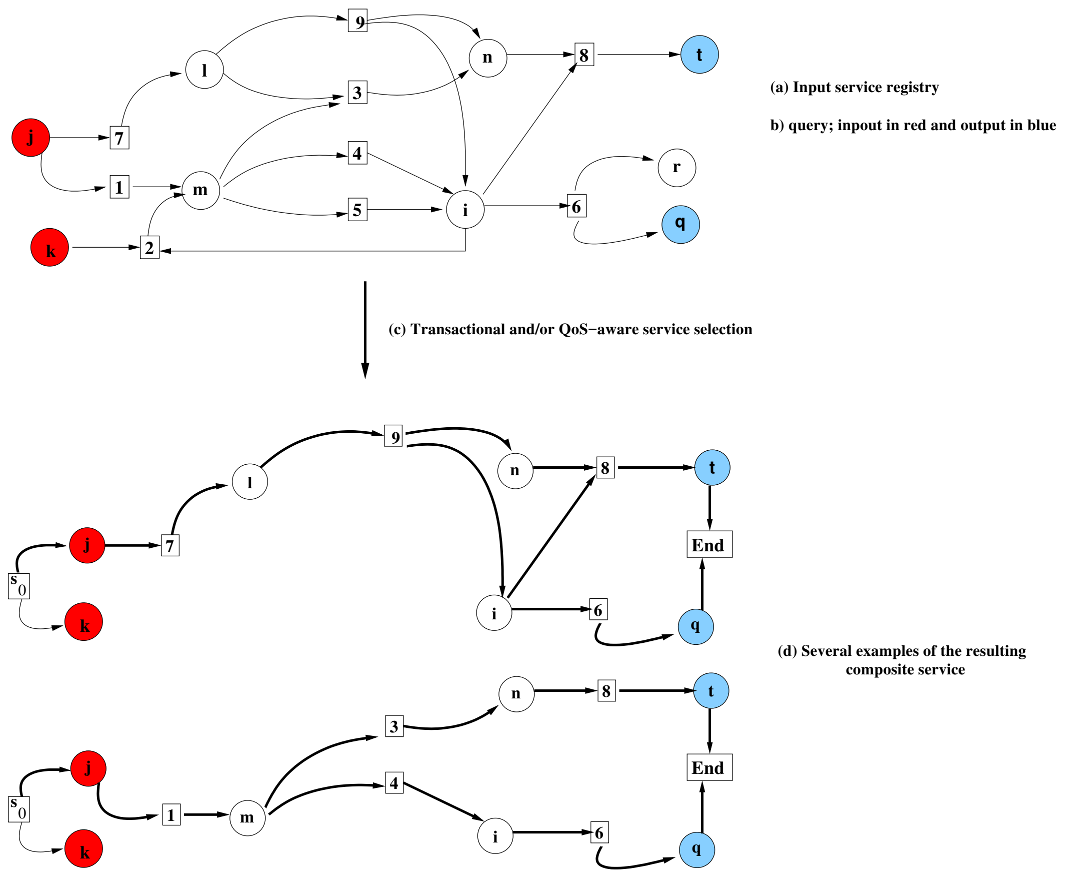
(a) Input service registry
b) query; inpout in red and output in blue
(c) Transactional and/or QoS−aware service selection
(d) Several examples of the resulting composite service
j
k
l
m
n
i
r
t
q
1
2
3
4
5
6
7
8
9
s0
End

Figure 4.1: The bottom-up service selection process using registry of Table 4.1, with $I_{\mathcal{Q}} = \{j, k\}$ and $O_{\mathcal{Q}} = \{q, t\}$.

us to know the transactional property of the current composite service, after each component selection.

In [2, 10], the input of our algorithm is query $\mathcal{Q}$ and the service registry is represented by a Colored Petri Net. The output is a transactional composite service (represented by a CPN) in [10], which moreover locally optimizes the QoS in [2]. Our algorithm adapts a Petri Net unfolding approach to perform a Best-First Search, which stops when a desired marking (i.e., needed outputs are produced), reachable from an initial marking (i.e., from available inputs) in the CPN, is found. The algorithm is composed by four steps. Step 1 verifies the admissibility of $\mathcal{Q}$: it verifies that at least one service can be invoked using inputs of $I_{\mathcal{Q}}$ and that all outputs of $O_{\mathcal{Q}}$ can be produced by services of the registry. Step 2 identifies the services of the registry (i.e., the transitions of the CPN) that may be useful to produce the outputs of $O_{\mathcal{Q}}$ by only considering transactional properties. This step produces a reduced CPN. Steps 1 and 2 are inspired from the yellow colouring step of the Service Aggregation Matchmaking (SAM) algorithm of [65, 66]. Step 3 detects the transitions in the reduced CPN that corresponds to the services component that locally optimize the QoS and that produce outputs of $O_{\mathcal{Q}}$. Step 4 returns a CPN only containing the places and transitions that correspond to the components of the resulting composite service.

Our experimental results show that taking into account the transactional properties additionally with the QoS optimization does not have a real impact on the execution time of the service selection. The execution time depends on the size of the registry, the complexity of our algorithm being $\mathcal{O}(|S|)$. While our algorithm [2] locally optimizes the QoS, our experiments show that it founds the best QoS solution in 83% of evaluated cases, considering different service registry sizes.

The approach proposed in [9] also extends the greedy algorithm SAM [65, 66] but differs from our aforementioned approaches in several aspects. Firstly, transactional properties and QoS are aggregated in a utility function used to select services. Therefore no color is used: services and service registry are represented by standard (not colored) Petri Net. Secondly, the work follows a global optimization approach, implementing an A*-based heuristic [117].

4.2.2 Related work

As far as we know, one of the first approaches using Petri Nets to model WS composition is [116]. This work proposed a Petri Net-based algebra to represent a Web service by a Petri Net, called Service Net. Because this approach did not include the management of time and resources of services, it has been extended, for example in [258], to Colored Petri Net, where color of tokens represents the type of information managed by services. Recently in [76], Fuzzy Predicate Petri Nets have been used to integrate fuzzy semantic into Bottom-up service selection.

Bottom-up QoS-aware automatic syntactic service composition problem has attracted a lot of attention in recent years. They can be classify into five groups: search approaches as in [128, 159], planning graph approaches as in [75, 245], integer linear programming ones as in [182, 249], dependency graph approaches as in [106, 127] and Petri Net based ones as in [65, 66, 144, 195]. However, as far as we know, when our articles have been published, our approaches were the first ones that allow Bottom-up Transactional and QoS-aware service composition. In [9], we had compared our approach with the Service Aggregation Matchmaking (SAM) algorithm of [65, 66]. Our approach outperforms SAM by identifying service compositions that better meet the QoS and that are transactional, while the composition time of both approaches are in the same order of magnitude. Our approach [2] has been extended, in [8],[45, 46, 69], to manage the execution of a transactional composite service represented by a CPN. This extension particularly concerns the PhD work of R. Angarita [8],[45, 46], which is detailed in Chapter 5.

When services are syntactically described by their inputs/outputs and when the service registry is represented by a graph, related work approaches are generally compared using the synthetic Web Service Challenge (WSC) [50, 51, 135, 238]. The following section presents the linear program we had proposed in [15] for Bottom-up Transactional and QoS-aware service selection and our experimental results obtained using the WSC benchmark.

4.3 Linear program

The approach we had proposed in [15] allows a Bottom-up QoS-aware service selection which is adaptable: the considered QoS can be either a QoS score (as defined in Def. 2.5.5 p. 13) or a single QoS criteria to optimize (e.g., execution time – see Def. 2.5.1 p. 12 – or cost, see Def. 2.5.3). Moreover our model can be extended to consider transactional properties.

4.3.1 Description

In [15], the service registry is represented by a ServiceData Graph (see Def. 2.3.3, p. 9). Our 0-1 linear program contains three kinds of decision variables: variables associated with graph nodes representing services (allowing to determine the components of the final service composition), variables associated with edges (allowing to determine the sub-graph representing the final service composition) and temporal variables associated graph nodes (representing the topological order of the components of the final composite service).

We had defined constraints modelling the input/output of each service of the registry, constraints implied by the query (defining the provided inputs and the needed outputs), constraints linking decision variables (particularly decision variables associated with nodes representing services and edges) and constraints to eliminate directed cycles. QoS score can be considered in the objective function or QoS constraints (for execution time or cost for example) can be added to our model. Moreover, our linear program can be extended by adding constraints modelling the transactional properties.

Our experiments (using CPLEX solver) have been done without transactional properties using the well-known WS-Challenge 2009 (WSC-09) benchmark [135], and when considering transactional properties, using the synthetic service repository we had used in [2]. Our model finds the optimal solution for all the 5 test sets of the WS-Challenge in less time than the time-out fixed by the WS-Challenge. Experiments done, when transactional properties have been considered, highlights the important difficulty inducing by transactional requirements.

4.3.2 Related work

When our model integrates the transactional properties, our experimental results show that, when an optimal solution exists, our model can find it generally faster than our approximate approach proposed in [2]. When our model does not take into account the transactional properties (then only considering QoS), our model is comparable, in terms of quality of the solutions, to the recent related approaches experimented with WSC-09 test sets, for example [128, 202, 245]. The approach of [128] won the WSC-09 by proposing a polynomial-time algorithm to solve the composition problem minimizing the execution time or maximizing the throughput. Authors in [202, 245] propose Dijkstra's algorithms. Moreover, our model always finds the optimal solution while the other aforementioned approaches, for example [202], can not always find it for the biggest data set without any cleaning process (that filters all services relevant for the query and discards the rest).

Integer Linear Programming (ILP) model have been proposed for QoS-aware service composition in [182, 249]. In these approaches, a composite service is decomposed into stages: a stage contains one service or several services executed in parallel. The associated ILP represents the problem of selecting one or several services per stage. Thus, the number of variables and constraints can be huge since there are proportional to the number of services and data, times the number of stages. Moreover, the number of stages is not known in advance; only upper bounds can be chosen (the worst one is to set the number of stages equals to the number of services). Our experimental results, without considering transactional properties, show that our model dominates the more recent one proposed in [182].

The next section presents an analysis of the complexity of the QoS-aware problems and a new formulation of the problems.

4.4 Complexity analysis

In [20], we classified the Bottom-up QoS-aware automatic syntactic service composition problem into four categories depending on the QoS criterion type: (i) QoS criterion to be minimized (e.g., for execution time), (ii) QoS criterion to be maximized (e.g., for throughput), (iii) sum-type criterion to be minimized (e.g., cost) and (iv) product-type criterion to be maximized (e.g., for reliability).

Categories (i) and (ii) has been considered by the Web Service Challenge (WSC) 2009 [135]. The service composition problem presented in WSC-08 [51] only tackles the feasibility part of the composition problem (see Def. 4.1.1) without QoS while QoS is considered in WSC-09 [135]. Although the problem presented in WSC-08 is NP-Hard, the problem presented in WSC-09 can be solved with polynomial time algorithms. In [20], we show that the problem of category (i) is exactly a well studied project scheduling problem with AND/OR constraints. In a scheduling problem with AND/OR precedence constraints, two kinds of jobs are considered: jobs with AND precedence constraints which can be executed when all their precedence jobs are finished, and jobs with OR precedence constraints can be performed when at least one of their precedence jobs is terminated. In [169], authors propose a Dijkstra-like algorithm to solve the project scheduling problem with AND/OR constraints. This algorithm is similar to the one proposed in [127] for Bottom-up service composition based on dependency graph. In [20], we propose to adapt the algorithm of [169] to Bottom-up service composition based on a ServiceData graph (without computing a dependency graph). Our adaptation is more efficient on WSC-09 test sets than related algorithms [127, 128, 159].

Some authors, for example in [127] and in [203], claim that their algorithm, initially proposed for categories (i) and (ii), can be applied to category (iv). However, we prove, in [20], that problem of category (iv) is NP-hard, and we exhibit a polynomial case. Problem of category (iii) is proved to be NP-hard in [106]. We also exhibit a polynomial case in [20] for such problem, associated with a particular class of ServiceData graph.

4.5 Conclusion and perspectives

To the best of our knowledge, the heuristic we had proposed in [2, 9, 10] and the linear program we had defined in [15], are the first approaches allowing Bottom-up Transactional and QoS-aware service composition. Our models are extendible since they can only take into account transactional properties without QoS [10], or QoS without transactional properties [15] or both [2, 9, 10, 15]. Moreover, in [15] different kinds of QoS optimization can be done: optimizing a sum-type criterion (e.g., QoS score or cost) or maximizing a QoS criterion (e.g., execution time). Our experimental results show that our approaches outperform the related ones: in [9], we offer a better QoS optimization than in the Service Aggregation Matchmaking (SAM) algorithm of [65, 66]; using the WSC-09 benchmark, our 0-1 linear program dominates the one proposed in [182]; and our scheduling problem with AND/OR constraints formulation [20] is more efficient than the algorithms of [127, 128, 159].

The difficulty inducing by transactional properties being important, our 0-1 linear program, depending on CPLEX solver, is too long to find optimal solutions for big-size test sets. Therefore, a first perspective of our work is to propose specific resolution methods to solve such big problems.

Thanks to the simple graph structure of the ServiceData graph, the models we proposed in [15, 20] are the most efficient ones for exactly solving, in polynomial time, the service composition problem with a maxmin-type criterion (like execution time or throughput criterion). Unfortunately, for sum-type criterion (like cost QoS criterion) or product-type criterion optimization (like reliability), polynomial time approaches can not be applied, since this problem has been shown as NP-hard. As second perspective, approximate approaches should be defined for solving such NP-hard composition problems.

The next chapter presents the approaches we have proposed for transactional composite service execution (how allowing a fault-tolerant service execution) and for service search (how finding services based on their syntactic description).

Chapter 5

Towards a full service management framework

A full service management framework must: (1) be able to discover/identify services achieving a specific goal, (2) compose them when no single service can reach the needed goal, and (3) establish mechanisms to ensure a reliable execution of the resulting composite service. This chapter presents the works we have carried out within two PhD works, Rafael Angarita's PhD [5, 6, 8][45], I had supervised with Marta Rukoz, and Mohamed Lamine Mouhoub's PhD [16, 17], I am supervising with Daniela Grigori. These proposed approaches are complementary to the selection ones, presented in Chapters 3 and 4, since they concern the composite service execution (step following the composite service selection) and the service search (step done before the service selection). Section 5.1 deals with the self-healing transactional composite service and Section 5.2 concerns the service search using SPARQL.

5.1 Self-healing transactional composite service

Once a transactional composite service has been selected, using for example the selection approaches we had proposed in Chapters 3 and 4, its reliable execution should be automatically supported. In [8] (more detailed in [45]), we propose a self-healing (i.e non intrusive dynamic fault-tolerant) approach for composite services supported by knowledge-based agents capable of making decisions at runtime. This section presents this approach and its related work.

5.1.1 Description

As explained by Kephart and Chess [131], the self-healing property of a system is the ability of the system to automatically detect, diagnose, and repair software and hardware problems. Our self-healing transactional composite service approach is a continuation of the work proposed by Cardinale and Rukoz in [69]. It implements three different recovery mechanisms: backward recovery through service compensation, forward recovery through service retry and replacement, and check-pointing as an alternative strategy. As explained in Definition 2.6.1 (p. 14), backward recovery consists in restoring the state that the system had at the beginning of the service execution or in reaching a state semantically closed to the state that the system had at the beginning of the service execution. Forward recovery (see Def. 2.6.2, p. 14) consists in repairing the failure to allow the failed service to continue its execution. Backward recovery can imply long times and lost work due to the compensation of previously successfully executed component services. Forward recovery can also cause long waiting times and deterioration of QoS criteria (e.g., price and reputation) due to component service retry or replacement. Both ensure the *all-or-nothing* transactional property [8], stating that each component service of a composite one must either complete successfully or have almost no effect (i.e., can be semantically undone). Check-pointing [207, 233] is used to relax the atomicity (*all-or-nothing*) transactional property, allowing to obtain partial results and to resume the execution later.

Monitoring QoS, environment and execution state, we have defined a self-healing model, formally described using the Colored Petri net formalism, that allows to dynamically decide the best recovery strategy (backward, forward or check-pointing) in case of a component failure – more detailed can be found in [7] and [45, 47]. We have proposed an agent architecture where service agents are knowledge-based agents, i.e., are self- and context-aware, able to make deductions based on the information they have about the whole composite service, about themselves, and about what is expected and what it is really happening at runtime [45]. We have extended our agent architecture, in [6], to provide self-healing capabilities for Web of Things applications, where agents are the representation of physical objects, web services or humans in the Web. We recently have also proposed a model, in [11], to measure the fuzzy atomicity of composite service execution. This model complements our self-healing one by providing an accurate measure of atomicity. It ensures a fuzzy *all-or-nothing* execution of a composite service and allows to envisage different recovery strategies depending on a the acceptable fuzzy atomicity the user may accept in case of failure.

In [5], we have also proposed a service replacement approach which is complementary to our execution system for fault-tolerant composite services. This work in progress allows to determine the most suitable transactional service from a set of functionally equivalent services, according to the opinions of multiple independent contributors, using crowds of voting services. Each voting service ranks candidate services for a selection or a service replacement, according to their own criteria, and do not have to know nor interact with each other; the only requirement is to rank a list of candidate services. Experimental evaluation has been done using three kinds of voting services: ones based on normalized advertised QoS parameters, ones based on historical QoS values and others based on QoS prediction.

5.1.2 Related work

Works supporting reliable execution of composite services can be classified into two groups [45]: fault tolerant approaches [36, 57, 58, 69, 153, 208, 237, 260], focusing in recovery mechanisms, and self-healing research ones [52, 145, 175, 219, 248, 259], dealing with QoS monitoring and pre/post condition satisfaction. Among fault tolerant approaches, some [58, 69, 153] use transactional properties, as we did. Others [57, 208, 237] use exception handling, and others are based on redundancy and design diversity (replicating component services) [36] or are optimization approaches [260]. Among self-healing approaches, some [52, 219] are build on the top of BPEL (*Business Process Execution Language*) [230], dedicated to BPEL developers, while others [145, 175, 248, 259] are non-BPEL-based, generally based on QoS monitoring.

As far as we know, when our work has been published, none of the related fault tolerant approaches consider dynamism in the execution environment to adapt the decision regarding to which recovery strategy is the most appropriate [47]. Compared with self-healing approaches, our approach is non intrusive (i.e., transparent for users and developers) and selects the most appropriate recovery and preventive strategy using a measure expressed in terms of (expected, current, and remaining) QoS and (expected and produced) outputs [47]. Moreover, while our approach is not built on top of accepted standards (e.g., BPEL), its execution engine has been designed to operate with the least possible human intervention. A recent work [40] is related to our. It presents a self-healing framework for managing bottom-up faults in composite services, in which bottom-up faults correspond to the faults that happen at a component service and whose knowledge is propagated to any composite service that uses the faulty component. In this work, each service is associated with a coordinator, similar to our agents, except that coordinator and service are deployed on the same node, to detect infrastructure faults, while in our approach the execution control is detached from services. As we do (see details in [7] and [45]), authors in [40] use a rule-based approach to apply multiple recovery strategies. However, the selected recovery strategy depends on the detected faults, while in our approach, we do not distinguish among the types of faults. Moreover, authors in [40] do not consider transactional properties nor QoS monitoring. Both approaches can be therefore be complementary.

Concerning the crowdsourcing for candidate service selection, only few approaches [217, 243] exist. Authors in [243] propose a consensus-based service selection approach in which different agents evaluate each candidate service QoS to aggregate it into one accepted QoS assessment. The work presented in [217] presents a first step toward hybrid human-automatic web service discovery. None of them consider transactional properties. Moreover, we propose a framework to gather together different service selection approaches, and to take advantage of their multiple techniques with a minimal effort. In our framework, contributors (services or human experts) have neither to know each other nor to form social networks to reach consensus about QoS of candidate services.

5.1.3 Conclusion and perspectives

Our self-healing composite service approach is based on transactional properties and on knowledge-based agents. The transactional properties are used as a deep-seated notion for fault tolerance, allowing to implement automatic backward and recovery mechanisms that do not need developers or any other kind of human intervention [45]. The transactional approach is extended with knowledge-based agents allowing a decision making process at runtime. Our approach deals with services at a conceptual providing a formal model for composite service executions.

The perspectives we envision are the following ones. Firstly, our agent architecture can be extended to Web of Things (WoT) applications, as we propose in [6]. Formal definition, implementation and experimentation have to be done to achieve this extension. Secondly, we would like to take into account the possible changes that may appear during the composite service execution during the selection process, in order to define reliable composite service selection. We would like to provide new models that offer new service compositions that integrate, as soon as they are defined, the envisaged recovery strategies. The challenge is to investigate a novel approach based on robust optimization, never used for service composition to our knowledge. Performance evaluation should also be done and, as explained in [170], is a challenging issue. Finally, concerning the crowdsourcing-based in progress approach, presented in [5], analysis and evaluation using different realistic scenarios and different voting systems should be done, and management policies for voting agents should be implemented.

To be able to have a composite service to execute or to have a service that can replace a failed one, identification of services, accomplishing a given goal, should be done. This step is generally called service search or service discovery. The next section presents a work in progress allowing to search services using SPARQL [194] queries.

5.2 Service search using SPARQL

Service discovery, also called service search in [133], concerns the identification of suitable existing services that satisfy a specific goal [210]. It has been considered as one of the key challenges for achieving efficient service oriented computing [217]. Service discovery approaches differ in their support of service description language(s), the organization of the search, and means used to select services [132]. Semantic web service [164, 189] enriches services with semantic ontology-based annotations to better support their discovery, composition and execution. As explained in [100], using semantic web query language, especially SPARQL [194], which is the current W3C Recommendation, allows to perform semantic service discovery. From a user perspective, data and services provide complementary view of information source, and users need to perform aggregated searches to identify data and services [183]. In [16, 17], we present a framework for searching semantic data and services using SPARQL. This section presents this framework and its related work.

5.2.1 Description

Authors in [210] consider SPARQL as the most widely used language for knowledge based employing the Resource Description Framework (RDF) and Web Ontology Language (OWL) [55]. RDF[1] [167] is a standard model for data interchange on the Web. It is used to represent information modelled as a "graph" [112, 193], whose nodes are entities (e.g., *students*, *lessons*, *teachers*), and whose edges are relationships (e.g., *TakesAClassAbout*, *Teachs*). A RDF data collection consists of a set of subject-property-object triples [90, 112]. An example of such triple can be: *student*1 (subject) takes a class (property) about *web service* (object). OWL[2] is the language for defining ontologies on the Web [55]. It shares many characteristics with RDF [118]. SPARQL is a graph-based query language [112], which is the W3C standard for querying RDF data [90]. Several OWL/RDF service descriptions exist: for example OWL-S, "OWL for Services", a service description ontology based on OWL [163, 177], or Minimal Service Model (MSM[3]), a RDF vocabulary that can represent the syntax of Web services or Web APIs [189].

The goal of our framework, called LIDSEARCH [17], for Linked Data and Service Search, is to extend a search of linked data with a service discovery to find relevant services that provide complementary data related to a user query on knowledge bases. Given (1) a SPARQL query on linked data, composed of sets of triple patterns and selection variables, and (2) a collection of RDF/OWL service descriptions (using OWL-S [163] or MSM [189] for example), the process is decomposed into several steps. First, it analyses the data query by extracting the output informations needed by user from the input ones provided in the data query. Then, it generates multiple service queries consisting of sets of triples that match the input and output of the data query with the input and output of a service. This matching can be either exact, i.e., resulting services have exactly the inputs/outputs extracted from the data query, or resulting services can return some of the inputs or of the some outputs extracted from the data query, providing additional information or allowing service composition. Generated service queries can be enriched by semantic lookup, such as concept lookup, when no concept has been specify in a triple of the data query, or similarity looked, finding similar concepts of a specific one.

The framework has been implemented using Apache Jena framework[4] to manage

[1]`https://www.w3.org/RDF/`

[2]`https://www.w3.org/2004/OWL/`

[3]`http://iserve.kmi.open.ac.uk/ns/msm/msm-2014-09-03.html`

[4]`https://jena.apache.org/`

SPARQL queries and RDF. A demonstration[5] has been presented in [17]. Experiments are also be done using some queries of the OWL-S-TC benchmark[6].

5.2.2 Related work

The majority of services available on the web lack of explicit and sufficient semantic information [257], and offer many heterogeneous formats [132]. Moreover, paucity of service registries does not allow expressive queries on services [189]. Therefore, many approaches and benchmarks have been proposed to better support semantic service discovery, as shown in surveys [132, 133, 177].

Our framework is inspired by the approach of [183] which aims to look for services that are related to a given query based on keywords comparison between a SQL-like query and a service ontology. This approach relies on semantic annotations generated using WordNet[7] on top of the classic service descriptions in order to expand the search area with the taxonomy provided by semantics [17]. Another work, related to the answer of a user query using services, is the ANGIE system [193]. This system enriches RDF knowledge bases by incorporating data from web services that provide encapsulated functions. This work assumes the existence of a global schema for both data and services. On the other hand, our approach consists in generating different service queries, depending on the provided service descriptions, without any global schema.

Several approaches are complementary to ours: SPARQL query processing (see for example in [181] for a recent survey), SPARQL federation approaches, such as FedX [212] or HiBISCuS [209], both used in LIDSEARCH [17] to optimize the SPARQL query management, and approaches enriching service description with semantic annotations, based for example on DBPedia knowledge base [142] as in [77, 257], or approaches that encode service description as linked data [189, 225], offering semantic service models to generate service query.

5.2.3 Conclusion and perspectives

The SPARQL-driven approach for searching linked data and related relevant services, we propose in [16], offers many perspectives not only in standard service discovery domain but also in emergent domains. For example, authors of [77] propose a mobile service discovery based on semantic annotations from DBpedia knowledge base. The Minimal Service Model proposed for iServe plateform [189] has been extended for advanced semantic discovery of things as services [234]. Therefore, our framework can also be extended in a similar way.

As it is a not yet finish work, several improvements of our approach are needed. Firstly, comparison with related works, and particularly with [100, 206, 210], should be improved. Secondly, more experiments should be done to totally validate the approach. Finally, QoS integration, query optimization, and improvement of search capabilities, can also be added to our framework. No QoS is currently considered to our approach to discover services or rank the services returned by the query. It should be integrated in our approach. Our framework is currently based on SPARQL federation approaches [212, 209]. Cloud-based approaches (see in [130] for a recent survey) or RDF distribution algorithms (see in [82] for a recent comparison of distributed approaches over a cluster computing engine) can also be envisaged to process our SPARQL queries. Concerning the improvement of search capabilities, we could introduce some similarity measures, as the ones proposed for example in [41], for enriching our semantic lookup

[5]`https://sites.google.com/site/lidsearch/`

[6]`http://projects.semwebcentral.org/projects/owls-tc/`

[7]`http://wordnet.princeton.edu/`

and comparing the services. We should also offer mechanisms (e.g. inferences from ontologies) to guarantee the completeness of our SPARQL query result, as it has been recently done for example in [83]. Moreover, when no single service satisfies the user query, a composite service should be found. Therefore, improvement should be done for simultaneously carrying out service search and automatic service composition, considering for example our previous approaches (presented in Chapter 3) or related work (for example [206]).

5.3 Conclusion

Both approaches presented in the chapter represent the first (service search) and the last step (service execution) of a full service management. As far as we know, only authors of [204, 206] have proposed an integrated semantic web service discovery and composition framework. They use two pre-existing components, iServe [189], for service discovery, and CompositIT [205], for service composition. In [206], experiments of this integrated framework have been done using WS-Challenge 2008 [51], that contains syntactic synthetic service descriptions without QoS. The approach presented in [204] extends the one proposed in [206], by integrating QoS optimization and minimizing the number of component services. Experiments have been done using WS-Challenge 2009 [135]. However, neither transactional properties nor the reliable execution of the resulting composite service have not been considered in this integrated framework. Integrating the selection approaches, presented in chapters 3 and 4, with the service search and the reliable service execution approaches, presented in this chapter, may allow us to propose an integrated QoS and transactional-aware semantic web service discovery and composition framework.

Chapter 6

Research Perspectives

In this document, we have summarized the approaches and analyses we have proposed for the Transactional and QoS-aware service selection (see Chapters 3 and 4), and the propositions, we have done under two PhD works, concerning the service execution and the service search (see Chapter 5). We conclude this document by identifying several open research issues we would like to tackle in the future: the combination of Top-down and Bottom-up selection, the robust dynamic service composition, the transactional properties issue, and the challenge of providing a full framework that allows dynamic discovery, composition, and fault-tolerant execution of composite Internet services.

6.1 Combining Top-down and Bottom-up selection

When service selection is Top-down, as we did in [3, 4, 12, 14] (see Chapter 3), a workflow with a fixed number of abstract activities should be defined, and for each activity, a set of concrete services, that can implement the activity, should be known. In such approach, no input/output links between services are considered. As explained in [203], both workflow and candidate services are assumed to be predefined beforehand, inducing a resulting composition with a fixed size. In the other hand, when service selection is Bottom-up, as we did in [2, 9, 10, 14, 20] (see Chapter 4), services are linked by their input/output dependencies in a graph and the selection consists in selecting a sub-graph that satisfies a query corresponding to a pair of inputs and outputs sets. In this case, user can only expresses what he wants (outputs) and what he is able to provide (inputs), without being able to really express the business process he finally wants to execute. Since we deeply study both approaches, we should be able to propose a model that mix both types of selection and analyse if such combination can improve the service selection process.

6.2 Robust dynamic service composition

Until now, we firstly execute a service selection process (by identifying the components of a composite service that satisfies a query), using the approaches we have presented in Chapters 3 and 4, and then execute the resulting composite service, using the self-healing approach we have presented in Section 5.1. The selection is done with an optimistic assumption: characteristics and behaviours of services seldom change. Failures or changes in service characteristics (e.g., QoS degradation) are considered during the execution. A recent approach [160] has been proposed to enable evolution of service composition. Following a static Top-down service composition, presented in [128], the authors propose to update the service dependency graph when selected services change and to rebuild the composition is case of the original one is invalid or suboptimal. Such proposition is really equivalent to an approach that firstly composes services and then manages failures and changes during the execution of the composition, as we do until now. However, in order to prevent undesirable impacts, notably the degradation of the QoS performances of the composition or the component failures, we have to take into account possible changes in the workflow or in the graph, during the optimization process. Based on the recovery strategies we offer in [6, 8], we would like to provide new models that offer new service compositions that integrate, as soon as they are defined, the envisaged recovery strategies.

6.3 Transactional properties issue

In [3], we extended the most used transactional properties of services, pivot, retriable, and compensatable, to composite services, and we defined a formal model of transactional composite service. As indicated in Chapter 2, this model has been extended by several authors (see for example [85, 93, 196, 241]). Transactional properties of a composite service implicitly describe its behaviour in case of failures and are considered to ensure a relaxation of the classical *all-or-nothing* property, as explained in Section 2.6. Even if some recent works analyse how such transactional properties can affect the QoS of a composite service (see for example [86]) or how transactional requirements can be specified (see for example [197]), more work should be done to be able to provide real services implementing such transactional properties. For example, several questions remains unresolved such as: how to really implement the retriable property which is a very strong restriction or how to transform the *all-or-nothing* paradigm to an *all-or-something* based on check-pointing. A first step to answer these questions is presented in our recent approach [11].

6.4 Toward a full transactional service management

As we previously mentioned, we consider that a full service management framework must: (1) be able to discover/identify services achieving a specific goal, (2) compose them when no single service can reach the needed goal, and (3) establish mechanisms to ensure a reliable execution of the resulting composite service. Moreover, we consider that real service benchmarks are needed to evaluate such framework. However, to verify the scalability of the proposed algorithms, selection approaches, even recent ones [146, 160, 204], are generally experimented using the Web Service Challenge [50, 135], i.e., using synthetic randomly generated services or using old QoS data extracted from real services, e.g., the QWS Dataset of [39]. Some authors, e.g., in [141], manually extract the information they need (e.g., inputs/outputs of services to build a dependency graph), from several selected real services. However, as far as we know, no real service registry is available to allow real comparison of the proposed approaches. Building such real service registries is a real challenge. We would like to analyse why such real benchmark does not exist and how it could exist.

Service composition is a very active area of research [143]. While many works have addressed the topic of Internet services from multiple angles (see surveys [143, 215, 218, 224]), and have proposed solutions for service discovery, composition, and evaluation, no framework really ensures and supports reliable service composition and execution. Our research work is a first step of such framework. Indeed, the resulting composite service obtained after one of our transactional service selection approaches can be the input of our self-healing composite service execution. The next challenge consists in integrating these approaches with service discovery, in order to provide a full transactional service management framework.

References

Personal references on services

Papers in international journals

[1] Y. Cardinale, J. El Haddad, M. Manouvrier, and M. Rukoz. Web Service Composition Based on Petri Nets: Review and Contribution. In *Resource Discovery (RED) - LNSC 8194*, pages 83–122, 2013. Volume containing extended papers of the works presented at 5th Int. Workshop on Resource Discovery 2012, through a two-step peer-review process.

[2] Y. Cardinale, J. E. Haddad, M. Manouvrier, and M. Rukoz. CPN-TWS: a coloured petri-net approach for transactional-QoS driven Web Service composition. *Int. Journal of Web and Grid Services (IJWGS - 2011 Impact Factor 1.919)*, 7(1):91–115, 2011.

[3] J. El Haddad, M. Manouvrier, and M. Rukoz. TQoS: Transactional and QoS-Aware Selection Algorithm for Automatic Web Service Composition. *IEEE Trans. on Services Comp.*, 3(1):73–85, 2010. Special Section on Transactional Web Services - Selectivity rate: 21%.

[4] V. Gabrel, M. Manouvrier, and C. Murat. Web services composition: Complexity and models. *Discrete Applied Mathematics*, 196:100–114, 2015.

Papers in international conferences and workshops

[5] R. Angarita, M. Manouvrier, and M. Rukoz. A Framework for Transactional Service Selection Based on Crowdsourcing. In *Int. Conf. on Mobile Web and Intelligent Inf. Sys. (MobiWis) - LNCS 9228*, pages 137–148, 2015.

[6] R. Angarita, M. Manouvrier, and M. Rukoz. An Agent Architecture to Enable Self-healing and Application-context-aware Web of Things Applications. In *Int. Conf. on Internet of Things and Big Data - IoTBD*, 2016. Position Paper accepted as a Short Paper for a 20 minutes Oral Presentation (2016).

[7] R. Angarita, M. Manouvrier, M. Rukoz, and Y. Cardinale. A knowledge-based approach for self-healing service-oriented applications. In *Int. ACM Conf. on Management of Digital EcoSystems (MEDES)*, 2016.

[8] R. Angarita, M. Rukoz, and M. Manouvrier. Dynamic Composite Web Service Execution by Providing Fault-Tolerance and QoS Monitoring. In *PhD Symp. of the Int. Conf. on Service-Oriented Comp. (ICSOC - CORE 2014 A-ranked conf.) - LNCS 8831*, pages 371–377, 2014. PhD Symposium.

[9] E. Blanco, Y. Cardinale, M.-E. Vidal, J. El Haddad, M. Manouvrier, and M. Rukoz. A transactional-qos driven approach for web service composition.

In *Int. Workshop on Resource Discovery (RED) - LNCS 6799*, pages 23–42. Springer, 2012.

[10] Y. Cardinale, J. El Haddad, M. Manouvrier, and M. Rukoz. Web service selection for transactional composition. In *Int. Conf. on Computational Science (ICCS - ERA 2010 A-ranked conf.)*, volume 1(1) of *Procedia CS*, pages 2689–2698, 2010.

[11] Y. Cardinale, J. El Haddad, M. Manouvrier, and M. Rukoz. Measuring Fuzzy Atomicity for Composite Service Execution. In *Int. Conf. on Open and Big Data (OBD)*, 2016. Invited paper.

[12] J. El Haddad, M. Manouvrier, G. Ramirez, and M. Rukoz. Qos-driven selection of web services for transactional composition. In *IEEE Int. Conf. on Web Services (ICWS)*, pages 653–660, 2008. Selectivity rate 16%.

[13] J. El Haddad, M. Manouvrier, and M. Rukoz. A Hierarchical Model for Transactional Web Service Composition in P2P Networks. In *IEEE Int. Conf.on Web Services (ICWS)*, pages 346–353, 2007. Selectivity rate 18%.

[14] V. Gabrel, M. Manouvrier, I. Megdiche, and C. Murat. A new 0-1 linear program for qos and transactional-aware web service composition. In *IEEE Symp. on Comp. and Comm. (ISCC)*, pages 845–850, 2012.

[15] V. Gabrel, M. Manouvrier, and C. Murat. Optimal and Automatic Transactional Web Service Composition with Dependency Graph and 0-1 Linear Programming. In *Int. Conf. on Service-Oriented Comp. (ICSOC - CORE 2014 A-ranked conf.) - LNCS 8831*, pages 108–122, 2014. Selectivity rate 15%.

[16] M. Mouhoub, D. Grigori, and M. Manouvrier. A Framework for Searching Semantic Data and Services with SPARQL. In *Int. Conf. on Service-Oriented Comp. (ICSOC - CORE 2014 A-ranked conf.) - LNCS 8831*, pages 123–138, 2014. Selectivity rate 15%.

[17] M. Mouhoub, D. Grigori, and M. Manouvrier. LIDSEARCH: A SPARQL-Driven Framework for Searching Linked Data and Semantic Web Services. In *The Semantic Web: ESWC Satellite Events (CORE 2014 A-ranked conf.), LNCS 9341*, pages 112–117, 2015. Demo Session.

Chapter in international books

[18] Y. Cardinale, J. El Haddad, M. Manouvrier, and M. Rukoz. *Handbook of Research on Non-Functional Properties for Service-oriented Systems: Future Directions*, chapter Transactional-aware web service composition: A survey, pages 116–141. IGI Global-Advances in Knowledge Management (AKM) Book Series - S. Reiff-Marganiec and M. Tilly Ed(s)., 2011. Chapter accepted after a peer-review process.

Technical reports

[19] J. El Haddad, M. Manouvrier, S. Reiff-Marganiec, and M. Rukoz. Context-based transactional service selection approach for service composition. Note de recherche 49, LAMSADE - Université Paris-Dauphine, 2010. `https://basepub.dauphine.fr/handle/123456789/5251`.

[20] V. Gabrel, M. Manouvrier, C. Murat, and K. Moreau. Qos-aware automatic syntactic service composition problem: complexity and resolution. Technical report, Cahier du Lamsade N°367 - Univeristé Paris-Dauphine, 2015. `http://www.lamsade.dauphine.fr/sites/default/IMG/pdf/cahier_367.pdf`.

Personal references on content-based image retrieval

Papers in international journals

[21] N. V. Hoàng, V. Gouet-Brunet, M. Rukoz, and M. Manouvrier. Embedding spatial information into image content description for scene retrieval. *Pattern Recognition (ERA 2010 A*-ranked J. - 2010 Impact Factor 2.607)*, 43(9):3013–3024, 2010.

[22] M. Manouvrier, M. Rukoz, and G. Jomier. Quadtree representations for storage and manipulation of clusters of images. *Image Vision Comput. (ERA 2010 B-ranked J. - 2002 Impact Factor 1.029)*, 20(7):513–527, 2002.

[23] M. Rukoz, M. Manouvrier, and G. Jomier. Delta-distance: A family of dissimilarity metrics between images represented by multi-level feature vectors. *Inf. Retr. (ERA 2010 B-ranked J. - 2006 Impact Factor 1.744)*, 9(6):633–655, 2006.

Papers in international conferences and workshops

[24] G. Jomier, M. Manouvrier, V. Oria, and M. Rukoz. Multi-level index for global and partial content-based image retrieval. In *IEEE Int. Workshop on Managing Data for Emerging Multimedia Applications (EMMA), in conj. with IEEE Conf.on Data Eng. (ICDE)*, page 1176, 2005.

[25] M. Manouvrier, M. Rukoz, and G. Jomier. A generalized metric distance between hierarchically partitioned images. In *Int. Workshop on Multimedia data mining - mining integrated media and complex data, MDM*, pages 33–41, 2005.

Chapter in international books

[26] M. Manouvrier, M. Rukoz, and G. Jomier. Quadtree-Based Image Representation and Retrieval. In *Spatial Databases: Technologies, Techniques and Trends*, pages 81–106. IDEA Group Publishing, Information Science Publishing and IRM Press, Y. Manolopoulos, A. Papadopoulos and M. Vassilakopoulos Ed.(s), 2005. Chapter accepted after a peer-review process.

Papers in national journals

[27] V. Gouet-Brunet, M. Manouvrier, and M. Rukoz. Synthèse sur les modèles de représentation des relations spatiales dans les images symboliques. *Revue des Nouvelles Technologies de l'Information (RNTI)*, 1(1):19–54, 2008. Numéro spécial "Les relations spatiales : de la modélisation à la mise en œuvre" - Rapport CEDRIC-CNAM No 1600 - Version étendue de 51 pages Rapport technique CEDRIC-CNAM No 1325.

[28] G. Jomier, M. Manouvrier, and M. Rukoz. Storage and management of similar images. *Journal Braz. Comp. Soc.*, 6(3):13–25, 2000.

Papers in national conferences

[29] G. Jomier, M. Manouvrier, V. Oria, and M. Rukoz. Indexation multi-niveaux pour la recherche globale et partielle d'images par le contenu. In *20èmes Journées Bases de Données Avancées, (BDA), Actes (Informal Proceedings).*, pages 177–196, 2004.

[30] G. Jomier, M. Manouvrier, and M. Rukoz. Stockage et gestion d'images par un arbre quaternaire générique. In *Proc. 15èmes Journées Bases de Données Avancées, BDA*, pages 405–424, 1999.

[31] G. Jomier, M. Manouvrier, M. Rukoz, J. Ramirez, and Y. Valero. Mis: Un prototipo de un sistema de manipulación de imágenes similares. In *XXVème Congrès Latino - Américain en Informatique (Panel)*, 1999.

[32] M. Rukoz, M. Manouvrier, and G. Jomier. Distances de similarité d'images basées sur les arbres quaternaires. In *18èmes Journées Bases de Données Avancées, (BDA)*, 2002.

Other personal references

[33] W. Cellary, S. Gançarski, G. Jomier, and M. Manouvrier. Chapitre 8 - Les versions. In *Bases de données et internet Modèles, langages et système, sous la direction de A. Doucet et G. Jomier*, Traité IC2 Information - Commande - Communication, pages 235–255. Edition Lavoisier, 2001.

[34] M. Manouvrier. *Objets Similaires de Grande Taille dans les Bases de Données.* PhD thesis, Univeristé Paris-Dauphine (France), Jan. 2000.

[35] C. B. Medeiros, O. Carles, F. D. Vuyst, G. Hébrail, B. Hugueney, M. Joliveau, G. Jomier, M. Manouvrier, Y. Naïja, G. Scémama, and L. Steffan. Vers un entrepôt de données pour le trafic routier. In *Actes de la 2ème journée francophone sur les Entrepôts de Données et l'Analyse en ligne, EDA*, pages 119–138, 2006.

Related work references

[36] H. Abdeldjelil, N. Faci, Z. Maamar, and D. Benslimane. A diversity-based approach for managing faults in web services. In *IEEE Int. Conf. on Advanced Information Networking and Applications (AINA)*, pages 81–88, 2012.

[37] F. N. Abu-Khzam, C. Bazgan, J. El Haddad, and F. Sikora. On the Complexity of QoS-Aware Service Selection Problem. In *Int. Conf. on Service-Oriented Comp. (ICSOC) - LNCS 9435*, pages 345–352. Springer, 2015.

[38] P. Adamczyk, P. H. Smith, R. E. Johnson, and M. Hafiz. Rest and web services: In theory and in practice. In *REST: from research to practice*, pages 35–57. Springer, 2011.

[39] E. Al-Masri and Q. H. Mahmoud. QoS-based discovery and ranking of web services. In *Int. Conf. on Comp. Communications and Networks (ICCCN)*, pages 529–534. IEEE, 2007.

[40] A. Alhosban, K. Hashmi, Z. Malik, B. Medjahed, and S. Benbernou. Bottom-up fault management in service-based systems. *ACM Tran. on Internet Technology (TOIT)*, 15(2):7, 2015.

[41] S. Aljalbout, O. Boucelma, and S. Sellami. Modeling and Retrieving Linked RESTful APIs: A Graph Database Approach. In *Conf. on the Move to Meaningful Internet Systems (OTM)*, pages 443–450, 2015.

[42] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web services.* Springer, 2004.

[43] M. Alrifai and T. Risse. Combining global optimization with local selection for efficient qos-aware service composition. In *Int. Conf. on World Wide Web (WWW)*, pages 881–890. ACM, 2009.

[44] M. Alrifai, D. Skoutas, and T. Risse. Selecting skyline services for QoS-based web service composition. In *Int. Conf. on World Wide Web (WWW)*, pages 11–20. ACM, 2010.

[45] R. Angarita. *An approach for Self-healing Transactional Composite Services.* PhD thesis, Université Paris-Dauphine (France), Dec. 2015. `https://tel.archives-ouvertes.fr/tel-01281384/document`.

[46] R. Angarita, Y. Cardinale, and M. Rukoz. Reliable composite web services execution: towards a dynamic recovery decision. *Electronic Notes in Theoretical Computer Science*, 302:5–28, 2014.

[47] R. Angarita, M. Rukoz, and Y. Cardinale. Modeling dynamic recovery strategy for composite web services execution. *World Wide Web*, 19(1):89–109, 2016.

[48] D. Ardagna and B. Pernici. Global and local qos constraints guarantee in web service selection. In *IEEE Int. Conf. on Web Services (ICWS)*, 2005.

[49] Y. Badr, D. Benslimane, Z. Maamar, and L. Liu. Guest Editorial: Special Section on Transactional Web Services. *IEEE Trans. on Services Comp.*, 3(1):30–31, 2010.

[50] A. Bansal, S. Bansal, M. B. Blake, S. Bleul, and T. Weise. Overview of the Web Services Challenge (WSC): Discovery and Composition of Semantic Web Services. In *Semantic Web Services*, pages 297–311. Springer, 2012.

[51] A. Bansal, M. Blake, S. Kona, S. Bleul, T. Weise, and M. Jaeger. WSC-08: Continuing the Web Services Challenge. In *IEEE CEC*, pages 351–354, July 2008.

[52] L. Baresi and S. Guinea. Dynamo and self-healing bpel compositions. In *Int. Conf. on Software Engineering*, pages 69–70, 2007.

[53] P. Bartalos and M. Bieliková. Automatic dynamic web service composition: A survey and problem formalization. *Computing and Informatics*, 30(4):793–827, 2012.

[54] G. Baryannis, O. Danylevych, D. Karastoyanova, K. Kritikos, P. Leitner, F. Rosenberg, and B. Wetzstein. Service composition. In *Service research challenges and solutions for the future internet*, pages 55–84. Springer, 2010.

[55] S. Bechhofer. Owl: Web ontology language. In *Encyclopedia of Database Systems*, pages 2008–2009. Springer, 2009.

[56] S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL Web Ontology Language Reference. Technical report, W3C, February 2004. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/` - Ext. on Nov. 2015.

[57] J. Behl, T. Distler, F. Heisig, R. Kapitza, and M. Schunter. Providing fault-tolerant execution of web-service-based workflows within clouds. In *Int. Workshop on Cloud Computing Platforms*, 2012.

[58] N. Ben Lakhal, T. Kobayashi, and H. Yokota. FENECIA: failure endurable nested-transaction based execution of composite Web services with incorporated state analysis. *The VLDB Journal*, 18(1):1–56, 2009.

[59] D. Benslimane, S. Dustdar, and A. Sheth. Services Mashups: The New Generation of Web Applications. *IEEE Internet Comp.*, 12(5):13–15, 2008.

[60] R. Berbner, M. Spahn, N. Repp, O. Heckmann, and R. Steinmetz. Heuristics for qos-aware web service composition. In *IEEE Int. Conf. on Web Services (ICWS)*, pages 72–82, 2006.

[61] S. Bhiri, O. Perrin, and C. Godart. Ensuring required failure atomicity of composite web services. In *Int. Conf. on World Wide Web (WWW)*, pages 138–147. ACM, 2005.

[62] S. Bhiri, O. Perrin, and C. Godart. Extending workflow patterns with transactional dependencies to define reliable composite web services. In *Advanced Int. Conf. on Telecommunications and Int. Conf. on Internet and Web Applications and Services (AICT/ICIW)*, pages 145–150, 2006.

[63] D. Booth and C. K. Liu. Web Services Description Language (WSDL) Version 2.0 Part 0: Primer. Technical report, W3C, 2007. `http://www.w3.org/TR/2007/REC-wsdl20-primer-20070626` - Ext. on Nov. 2015.

[64] A. Bouguettaya, Q. Z. Sheng, and F. Daniel. *Web Services Foundations*. Springer, 2014.

[65] A. Brogi and S. Corfini. SAM: A semantic web service discovery system. In *Knowledge-Based Intelligent Inf. and Eng. Sys.*, pages 703–710, 2007.

[66] A. Brogi, S. Corfini, and R. Popescu. Semantics-based composition-oriented discovery of web services. *ACM Trans. on Internet Techn. (TOIT)*, 8(4):19, 2008.

[67] D. Bunting, M. C. O. Hurley, M. Little, J. Mischkinsky, E. Newcomer, J. Webber, and K. Swenson. Web Services Transaction Management (WS-TMX) ver1. 0. Technical report, Arjuna Tech. Ltd., Fujitsu Limited, IONA Tech. Ltd., Oracle Corp., and Sun Microsystems, Inc, 2003. `http://www.immagic.com/eLibrary/ARCHIVES/TECH/ARJUNA/A030728M.pdf` - Ext. on Nov. 2015.

[68] F. Cabrera, G. Copeland, B. Cox, T. Freund, J. Klein, T. Storey, and S. Thatte. Web Services Transaction (WS-Transaction). Technical report, BEA Systems, IBM, and Microsoft, 2002. `http://xml.coverpages.org/WS-Transaction2002.pdf` - Ext. on Nov. 2015.

[69] Y. Cardinale and M. Rukoz. A framework for reliable execution of transactional composite web services. In *Int. Conf. on Management of Emergent Digital EcoSystems*, pages 129–136, 2011.

[70] R. Casado, M. Younas, and J. Tuya. A Generic Framework for Testing the Web Services Transactions. In *Advanced Web Services*, pages 29–49. Springer, 2014.

[71] A. Ceponkus, P. Furniss, A. Green, S. Dalal, and M. Little. Business Transaction Protocol. Technical report, OASIS, 2002. `https://www.oasis-open.org/committees/business-transaction/` - Ext. on Nov. 2015.

[72] M. Champion, C. Ferris, E. Newcomer, and D. Orchard. Web Services Architecture, W3C Working Draft 14 November 2002, 2002. `http://www.w3.org/TR/2002/WD-ws-arch-20021114/` - Ext. on Nov. 2015.

[73] L. Chen, J. Wu, H. Jian, H. Deng, and Z. Wu. Instant recommendation for web services composition. *Services Computing, IEEE Transactions on*, 7(4):586–598, 2014.

[74] M. Chen, P. Poizat, and Y. Yan. Adaptive composition and qos optimization of conversational services through graph planning encoding. In *Web Services Foundations*, pages 423–449. Springer, 2014.

[75] M. Chen and Y. Yan. QoS-aware Service Composition over Graphplan through Graph Reachability. In *IEEE Int. Conf. on Services Computing (SCC),*, pages 544–551, 2014.

[76] J. Cheng, C. Liu, M. Zhou, Q. Zeng, and A. Yla-Jaaski. Automatic Composition of Semantic Web Services Based on Fuzzy Predicate Petri Nets. *IEEE Trans. on Automation Science and Eng.*, 12(2):680–689, 2015.

[77] N. Cheniki, A. Belkhir, and Y. Atif. Mobile services discovery framework using DBpedia and non-monotonic rules. *Computers & Electrical Engineering*, 2016.

[78] P. K. Chrysanthis and K. Ramamritham. Synthesis of extended transaction models using ACTA. *ACM Trans. on Database Sys. (TODS)*, 19(3):450–491, 1994.

[79] S. Cirani, L. Davoli, G. Ferrari, R. Léone, P. Medagliani, M. Picone, and L. Veltri. A scalable and self-configuring architecture for service discovery in the internet of things. *IEEE Internet of Things Journal*, 1(5):508–521, 2014.

[80] C. Colombo and G. J. Pace. Recovery within long-running transactions. *ACM Comp. Surveys (CSUR)*, 45(3):28, 2013.

[81] W. W. W. Consortium et al. Simple object access protocol (SOAP), 2000. `http://www.w3.org/TR/soap/` - Ext. on Nov. 2015.

[82] O. Curé, H. Naacke, M. A. Baazizi, and B. Amann. On the Evaluation of RDF Distribution Algorithms Implemented over Apache Spark. In *Int. Workshop on Scalable Semantic Web Knowledge Base Systems (SSWSISWC)*, pages 16–31, 2015.

[83] O. Curé, H. Naacke, T. Randriamalala, and B. Amann. LiteMat: a scalable, cost-efficient inference encoding scheme for large RDF graphs. In *IEEE Int. Conf. on Big Data (Big Data)*, pages 1823–1830, 2015.

[84] L. M. da Silva, R. Braga, and F. Campos. Composer-Science: A semantic service based framework for workflow composition in e-Science projects. *Information Sciences*, 186(1):186–208, 2012.

[85] Z. Ding, J. Liu, Y. Sun, C. Jiang, and M. Zhou. A transaction and qos-aware service selection approach based on genetic algorithm. *Systems, Man, and Cybernetics: Systems, IEEE Transactions on*, 45(7):1035–1046, 2015.

[86] Z. Ding and Y. Sun. Performance evaluation of composite web service based on transaction. In *Int. Conf. on Service Sciences (ICSS)*, pages 214–219. IEEE, 2013.

[87] R. Duffin. Topology of series-parallel networks. *Journal of Mathematical Analysis and App.*, 10:303–313, 1965.

[88] S. Dustdar, R. Pichler, V. Savenkov, and H.-L. Truong. Quality-aware service-oriented data integration: requirements, state of the art and open challenges. *ACM SIGMOD Record*, 41(1):11–19, 2012.

[89] S. Dustdar and W. Schreiner. A survey on web services composition. *Int. journal of web and grid services*, 1(1):1–30, 2005.

[90] S. Elbassuoni, M. Ramanath, R. Schenkel, and G. Weikum. Searching RDF Graphs with SPARQL and Keywords. *IEEE Data Eng. Bull.*, 33(1):16–24, 2010.

[91] A. K. Elmagarmid. *Database transaction models for advanced applications*. Morgan Kaufmann Publishers Inc., 1992.

[92] D. Eppstein. Parallel recognition of series-parallel graphs. *Information and Computation*, 98(1):41–55, 1992.

[93] Y.-Y. FanJiang and Y. Syu. Semantic-based automatic service composition with functional and non-functional requirements in design time: A genetic algorithm approach. *Information and Software Tech.*, 56(3):352–373, 2014.

[94] T. A. Farrag, A. I. Saleh, and H. A. Ali. Semantic web services matchmaking: Semantic distance-based approach. *Computers & Electrical Eng.*, 39(2):497–511, 2013.

[95] L. Frank, R. U. Pedersen, C. H. Frank, and N. J. Larsson. The CAP theorem versus databases with relaxed ACID properties. In *Int. Conf. on Ubiquitous Information Management and Communication*, page 78. ACM, 2014.

[96] T. Freund and M. Little (Eds.). Web Services Business Activity (WS-BusinessActivity). Technical report, OASIS, 2007. `http://docs.oasis-open.org/ws-tx/wstx-wsba-1.1-spec/wstx-wsba-1.1-spec.html` - Ext. on Nov. 2015.

[97] N. Gamez, J. El Haddad, and L. Fuentes. Managing the Variability in the Transactional Services Selection. In *Int. Workshop on Variability Modelling of Software-intensive Systems*, page 88. ACM, 2015.

[98] N. Gamez, J. El Haddad, and L. Fuentes. SPL-TQSSS: A Software Product Line Approach for Stateful Service Selection. In *IEEE Int. Conf. on Web Services (ICWS)*, pages 73–80, 2015.

[99] L. Gao, S. D. Urban, and J. Ramachandran. A survey of transactional issues for web service composition and recovery. *Int. Journal of Web and Grid Services*, 7(4):331–356, 2011.

[100] J. M. García, D. Ruiz, and A. Ruiz-Cortés. Improving semantic web services discovery using SPARQL-based repository filtering. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:12–24, 2012.

[101] H. Garcia-Molina and K. Salem. *Sagas*, volume 16 (3). ACM, 1987.

[102] M. Garriga, C. Mateos, A. Flores, A. Cechich, and A. Zunino. RESTful Service Composition at a Glance: a Survey. *Journal of Network and Computer App.*, 2015. In Press. Available online 13 Dec. 2015.

[103] Gartner. Gartner Reveals Five Business Process Management Predictions for 2010 and Beyond, 2010. Press Release, `http://www.gartner.com/newsroom/id/1278415` - Ext. on April. 2016.

[104] Gartner. Gartner Says That Consumers Will Store More Than a Third of Their Digital Content in the Cloud by 2016, 2012. Press Release, `http://www.gartner.com/newsroom/id/2060215` - Ext. on Oct. 2015.

[105] S. C. Geyik, B. K. Szymanski, and P. Zerfos. Robust dynamic service composition in sensor networks. *IEEE Trans. on Services Comp.*, 6(4):560–572, 2013.

[106] S. C. Geyik, B. K. Szymanski, and P. Zerfos. Robust dynamic service composition in sensor networks. *Services Computing, IEEE Transactions on*, 6(4):560–572, 2013.

[107] A. Ghari Neiat, A. Bouguettaya, and T. Sellis. Spatio-Temporal Composition of Crowdsourced Services. In *Service-Oriented Computing*, pages 373–382. Springer, 2015.

[108] A. Goldman and Y. Ngoko. On graph reduction for qos prediction of very large web service compositions. In *IEEE Int. Conf. on Services Comp. (SCC)*, pages 258–265, 2012.

[109] D. Gorgan, V. Bacu, T. Stefanut, D. Rodila, and D. Mihon. Earth Observation application development based on the Grid oriented ESIP satellite image processing platform. *Comp. Standards & Interfaces*, 34(6):541–548, 2012.

[110] J. Gray. The Transaction Concept: Virtues and Limitations (Invited Paper). In *Int. Conf. on Very Large Data Bases (VLDB)*, pages 144–154, 1981.

[111] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques.* Morgan Kaufmann Publishers Inc., 1992.

[112] M. Grobe. RDF, Jena, SparQL and the 'Semantic Web'. In *ACM SIGUCCS fall Conf.: communication and collaboration*, pages 131–138. ACM, 2009.

[113] K. Grolinger, M. A. Capretz, A. Cunha, and S. Tazi. Integration of business process modeling and Web services: A survey. *Service Oriented Comp. and App.*, 8(2):105–128, 2014.

[114] J. E. Haddad. Optimization Techniques for QoS-Aware Workflow Realization in Web Services Context. In *Inter. Workshop on Resource Discovery (RED) - LNCS 6799 (2012)*, pages 134–149, 2010.

[115] T. Haerder and A. Reuter. Principles of Transaction-oriented Database Recovery. *ACM Comput. Surv.*, 15(4):287–317, Dec. 1983.

[116] R. Hamadi and B. Benatallah. A petri net-based model for web service composition. In *Australasian Database Conf. -Volume 17*, pages 191–200, 2003.

[117] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Sys. Science and Cybernetic*, 4(2):100–107, 1968.

[118] I. Horrocks, P. F. Patel-Schneider, and F. Van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Web semantics: science, services and agents on the World Wide Web*, 1(1):7–26, 2003.

[119] A. F. Huang, C.-W. Lan, and S. J. Yang. An optimal QoS-based Web service selection scheme. *Information Sciences*, 179(19):3309–3322, 2009.

[120] A. Immonen and D. Pakkala. A survey of methods and approaches for reliable dynamic service compositions. *Service Oriented Comp. and App.*, 8(2):129–158, 2014.

[121] I. O. f. S. ISO. *ISO 8402: Quality Management and Quality Assurance-Vocabulary.* International Organization for Standardization, 1994.

[122] V. Issarny, N. Georgantas, S. Hachem, A. Zarras, P. Vassiliadist, M. Autili, M. A. Gerosa, and A. B. Hamida. Service-oriented middleware for the future internet: state of the art and research directions. *Journal of Internet Services and App.*, 2(1):23–45, 2011.

[123] M. C. Jaeger, G. Rojec-Goldmann, and G. Mühl. Qos aggregation for web service composition using workflow patterns. In *IEEE Int. Conf. on Enterprise distributed object comp.*, pages 149–159, 2004.

[124] C. Jatoth, G. Gangadharan, and R. Buyya. Computational intelligence based qos-aware web service composition: A systematic literature review. *IEEE Trans. on Services Comp.*, PP(99), 2015.

[125] K. Jensen. *Coloured petri nets*. Springer, 1987.

[126] W. Jiang, D. Lee, and S. Hu. Large-scale longitudinal analysis of soap-based and restful web services. In *IEEE Int. Conf. on Web Services (ICWS)*, pages 218–225, 2012.

[127] W. Jiang, T. Wu, S.-L. Hu, and Z.-Y. Liu. QoS-aware automatic service composition: A graph view. *Journal of Comp. Science and Tech.*, 26(5):837–853, 2011.

[128] W. Jiang, C. Zhang, Z. Huang, M. Chen, S. Hu, and Z. Liu. QSynth: A Tool for QoS-aware Automatic Service Composition. In *IEEE Int. Conf. on Web Services (ICWS)*, pages 42–49, 2010.

[129] A. Jungmann and F. Mohr. An approach towards adaptive service composition in markets of composed services. *Journal of Internet Services and App.*, 6(1):1–18, 2015.

[130] Z. Kaoudi and I. Manolescu. RDF in the Clouds: A Survey. *The VLDB Journal*, 24(1):67–91, 2015.

[131] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[132] M. Klusch. Service discovery. In *Encyclopedia of Social Network Analysis and Mining*, pages 1707–1717. Springer, 2014.

[133] M. Klusch, P. Kapahnke, S. Schulte, F. Lecue, and A. Bernstein. Semantic Web Service Search: A Brief Survey. *KI-Künstliche Intelligenz*, pages 1–9, 2015.

[134] N. Kokash and V. D'Andrea. Evaluating quality of web services: A risk-driven approach. In *Business Inf. Sys.*, pages 180–194. Springer, 2007.

[135] S. Kona, A. Bansal, M. B. Blake, S. Bleul, and T. Weise. WSC-2009: A Quality of Service-Oriented Web Services Challenge. In *IEEE CEC*, pages 487–490, 2009.

[136] O. Kondratyeva, N. Kushik, A. Cavalli, and N. Yevtushenko. Evaluating Quality of Web Services: A Short Survey. In *IEEE Int. Conf. on Web Services (ICWS)*, pages 587–594, 2013.

[137] M. Koutraki, D. Vodislav, and N. Preda. Deriving intensional descriptions for web services. In *ACM Int. on Conf. on Information and Knowledge Management*, pages 971–980, 2015.

[138] K. Kritikos, B. Pernici, P. Plebani, C. Cappiello, M. Comuzzi, S. Benbernou, I. Brandic, A. Kertész, M. Parkin, and M. Carro. A survey on service quality description. *ACM Comp. Surveys (CSUR)*, 46(1):1, 2013.

[139] D. Lee, J. Kwon, S. Lee, S. Park, and B. Hong. Scalable and efficient web services composition based on a relational database. *Journal of systems and Software*, 84(12):2139–2155, 2011.

[140] J. Lee, S.-J. Lee, and P.-F. Wang. A Framework for Composing SOAP, Non-SOAP and Non-Web Services. *IEEE Trans. on Services Comp.*, 8(2):240–250, 2015.

[141] Y.-J. Lee. Algorithm for Automatic Web API Composition. In *Int. Conf. on Building and Exploring Web Based Env. (WEB)*, pages 57–62, 2013.

[142] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, et al. DBpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 5:1–29, 2014.

[143] A. L. Lemos, F. Daniel, and B. Benatallah. Web Service Composition: A Survey of Techniques and Tools. *ACM Comput. Surv.*, 48(3):33:1–33:41, Dec. 2015.

[144] B. Li, Y. Xu, J. Wu, and J. Zhu. A petri-net and qos based model for automatic web service composition. *Journal of Software*, 7(1):149–155, 2012.

[145] G. Li, L. Liao, D. Song, and Z. Zhang. Self-Adaptive Web Service Composition Based on Stochastic Context-Free Grammar. In *IEEE Int. Conf. one-Business Engineering (ICEBE)*, pages 139–144, 2014.

[146] J. Li, Y. Yan, and D. Lemire. Full solution indexing using database for qos-aware web service composition. In *IEEE Int. Conf. on Services Comp. (SCC)*, pages 99–106. IEEE, 2014.

[147] L. Li, C. Liu, and J. Wang. Deriving Transactional Properties of Composite Web Services. In *IEEE Int. Conf. on Web Services (ICWS)*, pages 631–638, 2007.

[148] C.-C. Lin, X.-L. Liu, and S.-M. Yuan. Reversible data hiding for vq-compressed images based on search-order coding and state-codebook mapping. *Information Sciences*, 293:314–326, 2015.

[149] M. Little. Transactions and Web services. *Comm. of the ACM*, 46(10):49–54, 2003.

[150] M. Little. Web services transactions: Past, present and future. In *XML Conf. and Exposition*, 2003.

[151] M. Little and A. Wilkinson (Eds.). Web Services Atomic Transaction (WS-AtomicTransaction) Version 1.2). Technical report, OASIS, 2009. `http://docs.oasis-open.org/ws-tx/wstx-wsat-1.2-spec.html` - Ext. on Nov. 2015.

[152] A. Liu, L. Huang, Q. Li, and M. Xiao. Fault-tolerant orchestration of transactional web services. In *Web Information Systems (WISE)*, pages 90–101. Springer, 2006.

[153] A. Liu, Q. Li, L. Huang, and M. Xiao. Facts: A framework for fault-tolerant composition of transactional web services. *IEEE Trans. on Services Comp.*, 3(1):46–59, 2010.

[154] A. Liu, H. Liu, Q. Li, L.-S. Huang, and M.-J. Xiao. Constraints-aware scheduling for transactional services composition. *Journal of Comp. Science and Tech.*, 24(4):638–651, 2009.

[155] H. Liu, W. Zhang, K. Ren, C. Liu, and Z. Zhang. A risk-driven selection approach for transactional web service composition. In *Int. Conf; on Grid and Cooperative Comp. (GCC)*, pages 391–397. IEEE, 2009.

[156] H. Liu, Z. Zheng, W. Zhang, and K. Ren. A Global Graph-based Approach for Transaction and QoS-aware Service Composition. *KSII Transactions on Internet & Information Systems (TIIS)*, 5(7):1252–1273, 2011.

[157] Y. Liu, A. H. Ngu, and L. Z. Zeng. Qos computation and policing in dynamic web service selection. In *Int. World Wide Web Conf. on Alternate track papers & posters*, pages 66–73. ACM, 2004.

[158] H. Ludwig, A. Keller, A. Dan, R. P. King, and R. Franck. Web Service Level Agreement (WSLA) Language Specification. Technical report, W3C, 2003. `http://www.research.ibm.com/wsla/WSLASpecV1-20030128.pdf` - Ext. on Nov. 2015.

[159] S. Luo, B. Xu, and Y. Yan. An accumulated-qos-first search approach for semantic web service composition. In *IEEE Int. Conf. on Service-Oriented Computing and Applications (SOCA)*, pages 1–4, 2010.

[160] C. Lv, W. Jiang, S. Hu, J. Wang, G. Lu, and Z. Liu. Efficient Dynamic Evolution of Service Composition. *IEEE Trans. on Services Comp.*, PP(99), 2015.

[161] Z. Maamar, N. C. Narendra, D. Benslimane, and S. Subramanian. Policies for context-driven transactional web services. In *Advanced Information Systems Eng.*, pages 249–263. Springer, 2007.

[162] Z. Maamar, S. Subramanian, P. Thiran, D. Benslimane, and J. Bentahar. An approach to engineer communities of web services: Concepts, architecture, operation, and deployment. *Int. Journal of E-Business Research (IJEBR)*, 5(4):1–21, 2009.

[163] D. Martin, M. Burstein, D. Mcdermott, S. Mcilraith, M. Paolucci, K. Sycara, D. L. Mcguinness, E. Sirin, and N. Srinivasan. Bringing semantics to web services with OWL-S. *World Wide Web*, 10(3):243–277, 2007.

[164] S. A. McIlraith, T. C. Son, and H. Zeng. Semantic web services. *IEEE intelligent systems*, 16(2):46–53, 2001.

[165] S. Mehrotra, R. Rastogi, H. F. Korth, and A. Silberschatz. A Transaction Model for Multidatabase Systems. In *Int. Conf. on Distributed Comp. Sys. (ICDC)*, pages 56–63, 1992.

[166] S. A. D. Midouni, Y. Amghar, and A. Chikh. A Full Service Approach for Multimedia Content Retrieval. In *Int. Conf. on Model and Data Engineering MEDI*, pages 125–137, 2014.

[167] E. Miller. An introduction to the resource description framework. *Bulletin of the American Society for Information Science and Technology*, 25(1):15–19, 1998.

[168] M. Moghaddam and J. G. Davis. Service selection in web service composition: A comparative review of existing approaches. In *Web Services Foundations*, pages 321–346. Springer, 2014.

[169] R. H. Möhring, M. Skutella, and F. Stork. Scheduling with AND/OR precedence constraints. *SIAM Journal on Computing*, 33(2):393–415, 2004.

[170] L. Mokdad, J. Ben-Othman, and A. Abdelkrim. Performance evaluation of composite Web services. *EAI Endorsed Trans. Indust. Netw. & Intellig. Syst.*, 2(4):e4, 2015.

[171] S. B. Mokhtar, N. Georgantas, and V. Issarny. COCOA: COnversation-based service COmposition in pervAsive computing environments with QoS support. *Journal of Systems and Software*, 80(12):1941–1955, 2007.

[172] L. Momtahan, A. Martin, and A. Roscoe. A taxonomy of web services using csp. *Electronic Notes in Theoretical Computer Science*, 151(2):71–87, 2006.

[173] F. Montagut, R. Molva, and S. T. Golega. Automating the composition of transactional web services. *Int. Journal of Web Services Research*, 5(1):24, 2008.

[174] F. Montagut, R. Molva, and S. T. Golega. The pervasive workflow: A decentralized workflow system supporting long-running transactions. *IEEE Trans. on Sys., Man, and Cybernetics, Part C: App. and Reviews*, 38(3):319–333, 2008.

[175] F. Moo-Mena, J. Garcilazo-Ortiz, L. Basto-Díaz, F. Curi-Quintal, and F. Alonzo-Canul. Defining a self-healing qos-based infrastructure for web services applications. In *IEEE Computational Science and Engineering Workshops (CSEWORKSHOPS)*, pages 215–220. IEEE, 2008.

[176] S. Murugesan. Understanding Web 2.0. *IT professional*, 9(4):34–41, 2007.

[177] L. D. Ngan and R. Kanagasabai. Semantic Web service discovery: state-of-the-art and research challenges. *Personal and ubiquitous computing*, 17(8):1741–1752, 2013.

[178] H. Nguyen Vu. *Prise en compte des relations spatiales dans les bases d'images symboliques.* PhD thesis, Université Paris-Dauphine / CNAM, 2011.

[179] A. M. Omer. *A framework for Automatic Web Service Composition based on service dependency analysis.* PhD thesis, TU Dresden, 2011. `http://www.qucosa.de/fileadmin/data/qucosa/documents/7074/Abrehet_Mohammed_Omer_PhD_dissertation.pdf` - Ext. on Oct. 2015.

[180] A. Ordóñez, V. Alcazar, O. M. C. Rendon, P. Falcarin, J. C. Corrales, and L. Z. Granville. Towards automated composition of convergent services: A survey. *Computer Communications*, 69:1–21, 2015.

[181] M. T. Özsu. A survey of rdf data management systems. *Frontiers of Computer Science*, 10(3):418–432, 2016.

[182] F. Paganelli, T. Ambra, and D. Parlanti. A QoS-aware service composition approach based on semantic annotations and integer programming. *Int. J. of Web Info. Sys. (IJWIS)*, 8(3):296–321, 2012.

[183] M. Palmonari, A. Sala, A. Maurino, F. Guerra, G. Pasi, and G. Frisoni. Aggregated search of data and services. *Information Systems*, 36(2):134–150, 2011.

[184] M. P. Papazoglou and W.-J. Van Den Heuvel. Service oriented architectures: approaches, technologies and research issues. *The VLDB journal*, 16(3):389–415, 2007.

[185] D.-S. Park, J.-S. Park, T. Kim, and J. H. Han. Image indexing using weighted color histogram. In *Int. Conf. on Image Analysis and Processing*, pages 909–914. IEEE, 1999.

[186] J. Pathak, S. Basu, and V. Honavar. Assembling composite web services from autonomous components. *Emerging Artificial Intelligence Applications in Comp. Eng.*, 160:394–405, 2007.

[187] C. Pautasso, O. Zimmermann, and F. Leymann. Restful web services vs. big'web services: making the right architectural decision. In *Int. Conf. on World Wide Web (WWW)*, pages 805–814. ACM, 2008.

[188] C. Pedrinaci and J. Domingue. Web services are dead. long live internet services. Technical report, Knowledge Media Institute-The Open University, W. Hall, M. Keynes, MK7 6AA, UK, 2010. SOA4All White Paper, `http://cordis.europa.eu/docs/projects/cnect/9/215219/080/deliverables/001-WhitePaperv23.pdf` - Ext. on Nov. 2015.

[189] C. Pedrinaci, D. Liu, M. Maleshkova, D. Lambert, J. Kopecky, and J. Domingue. iserve: a linked services publishing platform. In *Workshop on Ontology Repositories and Editors for the Semantic Web (ORES)*, volume CEUR 596, 2010.

[190] J. L. Peterson. Petri nets. *ACM Comput. Surv.*, 9(3):223–252, 1977.

[191] F. Plastria. Formulating logical implications in combinatorial optimisation. *European Journal of Operational Research*, 140(2):338–353, 2002.

[192] A. Portilla, G. Vargas-Solar, C. Collet, J.-L. Zechinelli-Martini, and L. García-Bañuelos. Contract based behavior model for services coordination. In *Web Information Systems and Tech.*, pages 109–123. Springer, 2008.

[193] N. Preda, G. Kasneci, F. M. Suchanek, T. Neumann, W. Yuan, and G. Weikum. Active knowledge: dynamically enriching RDF knowledge bases by web services. In *ACM SIGMOD Int. Conf. on Management of Data (SIGMOD)*, pages 399–410. ACM, 2010.

[194] E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. Technical report, W3C, 2008. `https://www.w3.org/TR/rdf-sparql-query/` - Ext. on March 2016.

[195] Z. Qian, S. Lu, and L. Xie. Colored petri net based automatic service composition. In *IEEE Asia-Pacific Service Comp. Conf.*, pages 431–438, 2007.

[196] K. Rajaram and C. Babu. Deriving reliable compositions using cancelable web services. *ACM SIGSOFT Software Eng. Notes*, 39(1):1–6, 2014.

[197] K. Rajaram, C. Babu, and A. Adiththan. Specification of transactional requirements for web services using recoverability. *Int. Journal of Inf. Tech. and Web Eng. (IJITWE)*, 8(1):51–65, 2013.

[198] K. Rajaram, C. Babu, and A. Ganesan. DILT: A Hybrid Model for Dynamic Composition and Execution of Heterogeneous Web Services. In *Distributed Comp. and Internet Tech.*, pages 239–244. Springer, 2015.

[199] É. Ranisavljević, F. Devin, D. Laffly, and Y. Le Nir. A dynamic and generic cloud computing model for glaciological image processing. *Int. Journal of Applied Earth Observation and Geoinformation*, 27:109–115, 2014.

[200] J. Rezaei. Best-worst multi-criteria decision-making method. *Omega*, 53:49–57, 2015.

[201] L. Richardson and S. Ruby. *RESTful web services.* "O'Reilly Media, Inc.", 2008.

[202] P. Rodriguez-Mier, M. Mucientes, and M. Lama. A dynamic qos-aware semantic web service composition algorithm. In *ICSOC - LNCS Vol. 7636*, pages 623–630, 2012.

[203] P. Rodriguez Mier, M. Mucientes, and M. Lama. A hybrid local-global optimization strategy for qos-aware service composition. In *IEEE International Conferrence on Web Services (ICWS)*, pages 735–738, 2015.

[204] P. Rodriguez-Mier, M. Mucientes, and M. Lama. Hybrid Optimization Algorithm for Large-Scale QoS-Aware Service Composition. *IEEE Trans. on Services Comp.*, PP, 2015.

[205] P. Rodríguez-Mier, M. Mucientes, J. C. Vidal, and M. Lama. An optimal and complete algorithm for automatic web service composition. *Int. Journal of Web Services Research (IJWSR)*, 9(2):1–20, 2012.

[206] P. Rodríguez-Mier, C. Pedrinaci, M. Lama, and M. Mucientes. An Integrated Semantic Web Service Discovery and Composition Framework. *IEEE Trans. on Services Comp.*, 1, 2015. PrePrints, doi:10.1109/TSC.2015.2402679.

[207] M. Rukoz, Y. Cardinale, and R. Angarita. Faceta*: Checkpointing for transactional composite web service execution based on petri-nets. In *Int. Symp. on Advances in Transaction Processing (ATP) - Procedia Computer Science*. Elsevier, 2012.

[208] H. Saboohi and S. A. Kareem. Failure recovery of world-altering composite semantic services-a two phase approach. In *Int. Conf. on Information Integration and Web-based Applications & Services*, pages 299–302. ACM, 2012.

[209] M. Saleem and A.-C. N. Ngomo. HiBISCuS: Hypergraph-Based Source Selection for SPARQL Endpoint Federation. In *The Semantic Web: Trends and Challenges*, pages 176–191. Springer, 2014.

[210] M. L. Sbodio, D. Martin, and C. Moulin. Discovering Semantic Web services using SPARQL and intelligent agents. *Web Semantics: Science, Services and Agents on the World Wide Web*, 8(4):310–328, 2010.

[211] E. Schouten. Big Data 'as a Service', 2012. `http://edwinschouten.nl/2012/09/19/bigdata-as-a-service/` - Ext. on Nov. 2015.

[212] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. Fedx: Optimization techniques for federated query processing on linked data. In *The Semantic Web–Int. Semantic Web Confe. (ISWC)*, pages 601–616. Springer, 2011.

[213] W. Serrai, A. Abdelkrim, L. Mokdad, and Y. Hammal. An efficient approach for Web service selection. In *IEEE Symp. on Comp. and Communication (ISCC)*, pages 167–172, 2016.

[214] Q. Z. Sheng, B. Benatallah, M. Dumas, and E. O.-Y. Mak. SELF-SERV: a Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment. In *Int. Conf. on Very Large Data Bases (VLDB)*, pages 1051–1054. VLDB Endowment, 2002.

[215] Q. Z. Sheng, X. Qiao, A. V. Vasilakos, C. Szabo, S. Bourne, and X. Xu. Web services composition: A decade's overview. *Information Sciences*, 280:218–238, 2014.

[216] A. Signoroni, M. Pezzoni, C. Tonoli, and R. Leonardi. A comparison of state-of-the-art technologies for irreversible compression of large medical datasets. In *Int. Symp. on Computer-Based Medical Systems (CBMS)*, pages 1–6. IEEE, 2012.

[217] F. Slaimi, S. Sellami, O. Boucelma, and A. Ben Hassine. Crowdsourcing for Web Service Discovery. In *On the Move to Meaningful Internet System (OMT) Confederated Int Conf. - LNCS Vol. 9415*, pages 451–464, 2015.

[218] A. Strunk. Qos-aware service composition: A survey. In *IEEE European Conf. on Web Services (ECOWS)*, pages 67–74, 2010.

[219] S. Subramanian, P. Thiran, N. C. Narendra, G. K. Mostefaoui, and Z. Maamar. On the enhancement of bpel engines for self-healing composite web services. In *Int. Symp. on Applications and the Internet*, pages 33–39, 2008.

[220] C.-a. Sun, E. El Khoury, and M. Aiello. Transaction management in service-oriented systems: Requirements and a proposal. *IEEE Trans. on Services Comp.*, 4(2):167–180, 2011.

[221] C.-A. Sun, X. Zhang, Y. Shang, and M. Aiello. Integrating transactions into bpel service compositions: An aspect-based approach. *ACM Trans. on the Web (TWEB)*, 9(2):9, 2015.

[222] L. Sun, H. Dong, F. K. Hussain, O. K. Hussain, and E. Chang. Cloud service selection: State-of-the-art and future research directions. *Journal of Network and Computer App.*, 45:134–150, 2014.

[223] Y. Syu, Y.-Y. FanJiang, J.-Y. Kuo, and S.-P. Ma. Towards a genetic algorithm approach to automating workflow composition for web services with transactional and qos-awareness. In *IEEE World Congress on Services (SERVICES)*, pages 295–302, 2011.

[224] Y. Syu, Y.-Y. Fanjiang, J.-Y. Kuo, and S.-P. Ma. A Review of the Automatic Web Service Composition Surveys. In *IEEE Int. Conf. on Semantic Comp. (ICSC)*, pages 199–202, 2014.

[225] M. Taheriyan, C. A. Knoblock, P. Szekely, and J. L. Ambite. Rapidly Integrating Services into the Linked Data Cloud. In *Int. Conf. on The Semantic Web (ISWC) - Volume Part I*, pages 559–574, 2012.

[226] B. Thuraisingham, L.-J. L. Zhang, and L. Moser. Emerging Web Services. *IEEE Trans. on Services Comp.*, 8(3):341–342, 2015.

[227] H. T. Tran and G. Feuerlicht. Service Repository for Cloud Service Consumer Life Cycle Management. In *Service Oriented and Cloud Comp.*, pages 171–180. Springer, 2015.

[228] B. Upadhyaya, Y. Zou, H. Xiao, J. Ng, and A. Lau. Migration of SOAP-based services to RESTful services. In *IEEE Int. Symp. on Web Systems Evolution (WSE)*, pages 105–114, 2011.

[229] J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series parallel digraphs. In *ACM Symp. on Theory of comp.*, pages 1–12. ACM, 1979.

[230] W. M. van der Aalst. Business Process Execution Language. In *Encyclopedia of Database Systems*, pages 288–289. Springer, 2009.

[231] W. M. Van Der Aalst and A. H. Ter Hofstede. YAWL: yet another workflow language. *Information Sys.*, 30(4):245–275, 2005.

[232] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow patterns. *Distributed and parallel databases*, 14(1):5–51, 2003.

[233] A. V. Vathsala and H. Mohanty. A survey on checkpointing web services. In *Int. Workshop on Principles of Eng. Service-Oriented and Cloud Sys.*, pages 11–17. ACM, 2014.

[234] Á. Villalba, J. L. Pérez, D. Carrera, C. Pedrinaci, and L. Panziera. servIoTicy and iServe: A Scalable Platform for Mining the IoT. In *Int. Conf. on Ambient Systems, Networks and Technologies (ANT), Int. Conf. on Sustainable Energy Information Technology (SEIT) - Procedia Computer Science (Elsevier)*, pages 1022–1027, 2015.

[235] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad. A survey on trust and reputation models for Web services: Single, composite, and communities. *Decision Support Sys.*, 74:121–134, 2015.

[236] W. Wang, S. De, R. Toenjes, E. Reetz, and K. Moessner. A Comprehensive Ontology for Knowledge Representation in the Internet of Things. In *IEEE Int. Conf. on Trust, Security and Privacy in Comp. and Comm. (TrustCom)*, pages 1793–1798, 2012.

[237] W. Wang, L. Wang, and W. Lu. A Resilient Framework for Fault Handling in Web Service Oriented Systems. In *IEEE Int. Conf. on Web Services (ICWS)*, pages 663–670, 2015.

[238] T. Weise, M. B. Blake, and S. Bleul. *Web Services Foundations*, chapter Semantic Web Service Composition: The Web Service Challenge Perspective, pages 161–187. Springer, 2014.

[239] B. Wu, C.-H. Chi, and S. Xu. Service selection model based on qos reference vector. In *IEEE Congress on Services*, pages 270–277, 2007.

[240] J. Wu and F. Yang. Qos prediction for composite web services with transactions. In *Int. Conf. on Service-Oriented Comp. (ICSOC)*, pages 86–94. Springer, 2007.

[241] Q. Wu and Q. Zhu. Transactional and QoS-aware dynamic service composition based on ant colony optimization. *Future Generation Computer Sys.*, 29(5):1112–1119, 2013.

[242] Y. Wu, C.-G. Yan, Z. Ding, G.-P. Liu, P. Wang, C. Jiang, and M. Zhou. A Multilevel Index Model to Expedite Web Service Discovery and Composition in Large-Scale Service Repositories. *IEEE Trans. on Services Comp.*, 9(3):330–342, 2016.

[243] M. XSharifi, A. Manaf, A. Memariani, H. Movahednejad, H. Md Sarkan, and A. Dastjerdi. Multi-criteria consensus-based service selection using crowdsourcing. In *Advanced Information Networking and Applications Workshop (WAINA)*, pages 114–âĂŞ120, 2014.

[244] Y. Yan and M. Chen. Anytime qos-aware service composition over the graphplan. *Service Oriented Comp. and App.*, 9(1):1–19, 2015.

[245] Y. Yan, M. Chen, and Y. Yang. Anytime QoS Optimization over the PlanGraph for Web Service Composition. In *ACM Symp. on Applied Comp. (SAC)*, pages 1968–1975, 2012.

[246] Y. Yan, B. Xu, and Z. Gu. Automatic Service Composition Using AND/OR Graph. In *E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, 2008 10th IEEE Conference on*, pages 335–338. IEEE, 2008.

[247] Y. Yang, M. Dumas, L. García-Bañuelos, A. Polyvyanyy, and L. Zhang. Generalized aggregate Quality of Service computation for composite services. *Journal of Systems and Software*, 85(8):1818–1830, 2012.

[248] Y. Yin, B. Zhang, X. Zhang, and Y. Zhao. A self-healing composite web service model. In *IEEE Asia-Pacific Services Comp. Conf. (APSCC)*, pages 307–312, 2009.

[249] J. J.-W. Yoo, S. Kumara, D. Lee, and S.-C. Oh. A Web Service Composition Framework Using Integer Programming with Non-functional Objectives and Constraints. In *IEEE CEC/EEE*, pages 347–350, 2008.

[250] K. P. Yoon and C.-L. Hwang. *Multiple attribute decision making: an introduction*, volume 104. Sage publications, 1995.

[251] J. Yu, Q. Z. Sheng, J. Han, Y. Wu, and C. Liu. A semantically enhanced service repository for user-centric service discovery and management. *Data & Knowledge Eng.*, 72:202–218, 2012.

[252] Q. Yu, X. Liu, A. Bouguettaya, and B. Medjahed. Deploying and managing Web services: issues, solutions, and directions. *The VLDB Journal*, 17(3):537–572, 2008.

[253] T. Yu and K.-J. Lin. Service selection algorithms for web services with end-to-end qos constraints. *Information Systems and E-Business Management*, 3(2):103–126, 2005.

[254] T. Yu, Y. Zhang, and K.-J. Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. on the Web (TWEB)*, 1(1):6, 2007.

[255] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Trans. on Software Eng.*, 30(5):311–327, 2004.

[256] W. Zhang, Y. Yang, S. Tang, and L. Fang. Qos-driven service selection optimization model and algorithms for composite web services. In *Annual Int. Conf. on Computer Software and App.*, volume 2, pages 425–431. IEEE, 2007.

[257] Z. Zhang, S. Chen, and Z. Feng. Semantic Annotation for Web Services Based on DBpedia. In *IEEE Int. Symp. on Service Oriented System Eng. (SOSE)*, pages 280–285, 2013.

[258] Z.-l. Zhang, F. Hong, and H.-j. Xiao. A colored petri net-based model for web service composition. *Journal of Shanghai University (English Ed.)*, 12:323–329, 2008.

[259] Z. Zheng and M. R. Lyu. An adaptive qos-aware fault tolerance strategy for web services. *Empirical Software Eng.*, 15(4):323–345, 2010.

[260] Z. Zheng and M. R. Lyu. Selecting an optimal fault tolerance strategy for reliable service-oriented systems with local and global constraints. *IEEE Trans. on Comp.*, 64(1):219–232, 2015.

[261] Z. Zheng, H. Ma, M. R. Lyu, and I. King. Qos-aware web service recommendation by collaborative filterin. *IEEE Transactions Services Comp.*, 4(2):140âĂŞ152, 2011.

[262] Z. Zheng, Y. Zhang, and M. R. Lyu. Investigating QoS of real-world web services. *IEEE Trans. on Services Comp.*, 7(1):32–39, 2014.

[263] A. Zimmermann. Colored Petri Nets. *Stochastic Discrete Event Systems: Modeling, Evaluation, Applications*, pages 99–124, 2008.

[264] G. Zou, Q. Lu, Y. Chen, R. Huang, Y. Xu, and Y. Xiang. Qos-aware dynamic composition of web services using numerical temporal planning. *IEEE Trans. on Services Comp.*, 7(1):18–31, 2014.

[265] M. Zur Muehlen, J. V. Nickerson, and K. D. Swenson. Developing web services choreography standards-the case of REST vs. SOAP. *Decision Support Sys.*, 40(1):9–29, 2005.

Appendix A

Lists of Figures and Tables

List of Figures

List of Tables

Appendix B

Abbreviation, symbol and definition tables

Table B.1: Acronym and Abbreviation table

Abbreviation	**Description**
CPN	Colored Petri Net
CS	Composite Service
OWL	Web Ontology Language
PN	Petri Net
QoS	Quality of Service
RDF	Resource Description Framework
TP	Transactional Property
WS	Web Service
WSC	Web Service Challenge
WSDL	Web Services Description Language
WSLA	Web Service Level Agreement

Table B.2: Table of symbols

Symbol	**Description**
s	A service
$in(s)$	The input set needed by service s to be invoked
$out(s)$	The out set produced by service s after its execution
$e(s)$	The execution time of service s
$t(s)$	The throughput of service s
$c(s)$	The cost of service s
$r(s)$	The reliability of service s
$Score(s)$	The QoS score associated with service s
$q_j(s)$	The jth QoS criterion associated with service s
w_j	The weight associated with q_j.
$S_c(s)$	The set of service components composing a composite service s
sc_i	The ith component of a composite service s
S	Set of vertices in a graph representing services
D	Set of of vertices in a graph representing the inputs and the outputs of services
$G = (V, U)$ with $V = S \cup D$	A ServiceData graph (see Def. 2.3.3)
$G^c = (V^c, U^c)$	A sub graph of graph G
$G = (V, A)$	A service dependency graph (see Def. 2.3.5)
(P, T, F)	A Petri Net with P, a set of places, T, a set of transitions and F, flow relation (see Def. 2.3.4)
p	Pivot transactional property (see Def. 2.6.4)
c	Compensatable transactional property (see Def. 2.6.5 and 2.6.9)
pr	Pivot retriable transactional property (see Def. 2.6.6)
cr	Compensatable retriable transactional property (see Def. 2.6.6 and and 2.6.10)
a	Atomic transactional property (see Def. 2.6.8)
ar	Atomic retriable transactional property (see Def. 2.6.10)
$TP(s)$	The transactional property of service s
$\mathcal{Q}$	Composition query (see Def. 2.7.1)
$WF_{\mathcal{Q}}$	Abstract workflow associated with $\mathcal{Q}$ (see Def. 2.3.2 and 2.7.1)
$W_{\mathcal{Q}}$	Set of QoS weights associated with $\mathcal{Q}$ (see Def. 2.5.5 and 2.7.1)
$TP_{\mathcal{Q}}$	Transactional property of the expected composite service (see Section 2.6 and 2.7.1))
$I_{\mathcal{Q}}$	Set of inputs associated with $\mathcal{Q}$ (see Def. 2.2.1 and 2.7.1)
$O_{\mathcal{Q}}$	Set of the expected outputs associated with $\mathcal{Q}$ (see Def. 2.2.1 and 2.7.1)
A_i	ith activity of a workflow (see Def. 2.7.2)

Table B.3: Definition table

N°	**Defined term**	**Page**
2.2.1	Syntactic description of service	6
2.3.1	Composite service/component service	8
2.3.2	Workflow	8
2.3.3	ServiceData graph	9
2.3.5	Service Dependency graph	10
2.3.4	Service Petri Net	10
2.4.1	Service directory/registry/repository	11
2.5.1	Execution time	12
2.5.2	Throughput	12
2.5.3	Cost	12
2.5.4	Reliability	12
2.5.5	QoS Score	13
2.6.1	Backward recovery	14
2.6.2	Forward recovery	14
2.6.3	Semantic recovery	15
2.6.4	Pivot service	15
2.6.5	Compensatable non-composite service	15
2.6.6	Pivot or compensatable non-composite service	15
2.6.7	Transactional (non-composite) service	15
2.6.8	Atomic service	16
2.6.9	Compensatable composite service	16
2.6.10	Pivot or compensatable composite service	16
2.6.11	Transactional composite service	16
2.7.1	Composition query	18
2.7.2	Top-down component service selection	18
2.7.3	Bottom-up component service selection	18
2.7.4	QoS-aware service selection	19
2.7.5	Transactional-aware service selection	19
2.7.6	QoS and Transactional-aware service selection	19
2.7.7	Dynamic fault-tolerant composite service execution	19
3.1.1	Top-down QoS and Transactional-aware composition problem	24
4.1.1	Bottom-up QoS and Transactional-aware composition problem	31