Grant Agreement N° 215483

Title:	*Survey of Quality Related Aspects Relevant for Service-based Applications*
Authors:	*Salima Benbernou (UCB), Ivona Brandic (TUW), Manuel Carro (UPM), Marco Comuzzi (Polimi), Elisabetta Di Nitto (Polimi), Maha Driss (INRIA), Julia Hielscher (UniDue), Raman Kazhamiakin (FBK), Gabor Kecskemeti (SZTAKI), Attila Kertesz (SZTAKI), Kyriakos Kritikos (UOC), Andreas Metzger (UniDue), Hassina Meziane (UCBL), Andrea Mocci (Polimi), Barbara Pernici (Polimi), Pierluigi Plebani (Polimi), Sagar Sen (INRIA), Fabrizio Silvestri (CNR), Branimir Wetzstein (USTUTT)*
Editor:	*Barbara Pernici (Polimi), Andreas Metzger (UniDue)*
Reviewers:	*Manuel Carro (UPM), Marco Pistore (FBK), Dragan Ivanovic (UPM)*
Identifier:	*Deliverable # PO-JRA-1.3.1*
Type:	*Deliverable*
Version:	*1*
Date:	*15 July 2008*
Status:	*Final*
Class:	*External*

Management Summary

Quality related aspects relevant for service-based applications cover a broad field of research, including work on quality modeling, QoS and SLA negotiation, as well as constructive and analytical quality assurance (like testing, monitoring and static analysis). This deliverable provides a survey of this broad field of “service quality” and identifies the key areas where research contributions are currently available. Based on this survey of the state of the art, important and emerging research challenges are identified that could be pursued in the future in order to close several of the gaps which emerge from the current state of the art on “service quality”.

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement n° 215483 (S-Cube).

Members of the S-Cube consortium:

University of Duisburg-Essen (Coordinator) – UniDue	Germany
Tilburg University – Tilburg	Netherlands
City University London – CITY	U.K.
Consiglio Nazionale delle Ricerche – CNR	Italy
Center for Scientific and Technological Research – FBK	Italy
French Nat'l Institute for Research in Computer Science and Control – INRIA	France
The Irish Software Engineering Research Centre – Lero	Ireland
Politecnico di Milano – Polimi	Italy
MTA SZTAKI – Computer and Automation Research Institute – SZTAKI	Hungary
Vienna University of Technology – TUW	Austria
Université Claude Bernard Lyon – UCBL	France
University of Crete – UOC	Greece
Universidad Politécnica de Madrid – UPM	Spain
University of Stuttgart – USTUTT	Germany

Published S-Cube documents

These documents are all available from the S-Cube Web Portal at http://www.s-cube-network.eu/

The S-Cube Deliverable Series

Vision and Objectives of S-Cube

The Software Services and Systems Network (S-Cube) will establish a unified, multidisciplinary, vibrant research community which will enable Europe to lead the software-services revolution, helping shape the software-service based Internet which is the backbone of our future interactive society.

By integrating diverse research communities, S-Cube intends to achieve world-wide scientific excellence in a field that is critical for European competitiveness. S-Cube will accomplish its aims by meeting the following objectives:

- Re-aligning, re-shaping and integrating research agendas of key European players from diverse research areas and by synthesizing and integrating diversified knowledge, thereby establishing a long-lasting foundation for steering research and for achieving innovation at the highest level.
- Inaugurating a Europe-wide common program of education and training for researchers and industry thereby creating a common culture that will have a profound impact on the future of the field.
- Establishing a pro-active mobility plan to enable cross-fertilisation and thereby fostering the integration of research communities and the establishment of a common software services research culture.
- Establishing trust relationships with industry via European Technology Platforms (specifically NESSI) to achieve a catalytic effect in shaping European research, strengthening industrial competitiveness and addressing main societal challenges.
- Defining a broader research vision and perspective that will shape the software-service based Internet of the future and will accelerate economic growth and improve the living conditions of European citizens.

S-Cube will produce an integrated research community of international reputation and acclaim that will help define the future shape of the field of software services which is of critical for European competitiveness. S-Cube will provide service engineering methodologies which facilitate the development, deployment and adjustment of sophisticated hybrid service-based systems that cannot be addressed with today's limited software engineering approaches. S-Cube will further introduce an advanced training program for researchers and practitioners. Finally, S-Cube intends to bring strategic added value to European industry by using industry best-practice models and by implementing research results into pilot business cases and prototype systems.

S-Cube materials are available from URL: http://www.s-cube-network.eu/

Contents

Chapter 1

Introduction

1.1 Motivation

In the last ten years, the Service Oriented paradigm for Information Systems development has emerged as a powerful solution to provide seamless integration between organizations that provide their services as web-enabled software services (e.g., Web Services [8]). In the beginning, the interests of researchers and practitioners has converged on the functional aspects of those software services and their description. Because of the increasing agreement on the implementation and management of the functional aspects of those services, such as the adoption of WSDL for service description, SOAP for Web service messaging, or WS-BPEL for Web service composition, the interests of researchers is shifting toward the 'non-functional' or quality aspects of web-enabled software services [192].

Standard, XML-based protocols which are used to describe service interfaces, to classify services, and to implement service messaging enable the *open world* perspective on service-based applications, where services can dynamically be bound and invoked. Service requestors do not need to know in advance which service they will be actually invoking during the operation of the service-based application, since service binding can be made dynamically according to the requestors' functional and quality requirements. Due to this *loose coupling* between service requestors and providers in a service-based application [194], the management and assurance of the quality aspects of services becomes of utmost importance. Service requestors need to be given guarantees by the service provider on the quality with which a service will be provided. In addition, a service requestor must be provided with tools to monitor the fulfilment of quality guarantees.

The open world perspective enables services to collaborate in highly distributed environments, cutting across the boundaries of various organizations (including enterprises, governmental and non-profit organizations). Those services are often provisioned in the context of short-term, volatile and thus *highly dynamic* business relationships between service providers and requestors [194]. A typical case is the one of on-demand computing, in which customers invoke services only on the basis of spot needs, without establishing any kind of longer term relationships with their partners.

In general, contracts (i.e., formal agreements) between service providers and requestors on quality aspects of services have to be established [59]. For service-based applications that support dynamic business relationships, the contract life-cycle should be as much as possible automated. Otherwise the effort for establishing and managing the contracts might be a prohibitive factor in establishing the business relationships.

Generally, the life-cycle of 'electronic' contracts constitutes three main phases [202, 270]:

- *Contract definition*: The contract definition phase concerns the establishment of a model for the definition of contract terms, which is understood and shared by the contracting parties. This is usually achieved through the definition of a contract template, which is then instantiated in an actual contract that reflects the domain dependent interests of providers and customers.

- *Contract establishment*: The contract establishment phase concerns the set of activities required for the instantiation of the contract template in an actual contract. This may involve the selection of the contract partner, among a set of potential partners, and the negotiation of the contract terms between the selected partner and the service customer.

- *Contract enactment*: The contract enactment concerns the execution of the contract and the monitoring of its satisfaction. In the specific context of contracts on quality aspects, this activity coincides with the monitoring of the satisfaction of quality guarantees negotiated by a service provider and the service requestor.

1.2 Aims and Focus of the Deliverable

This survey aims at reviewing relevant literature in the context of *quality of service (QoS)* contract establishment and enactment for service-based applications.

In particular, we restrict our attention to QoS contracts, or more general those parts of Service Level Agreements (SLAs) which deal with statements about the levels of services' quality on which the service requestor and the providers reach an agreement. This survey does not focus on other aspects of the contracts, i.e., parties' identification, legal obligations, or contract unfulfillment penalties, which are also aspects covered in SLAs or general contracts.

For what concerns contract establishment, we focus on QoS negotiation as the primarily means to automatically establish contracts on QoS between service requestors and providers.

Regarding contract enactment, we review relevant techniques for checking quality, such as the analysis and formal verification of service specifications, service testing, and the monitoring of QoS during service execution.

The review of models, techniques, and tools for QoS negotiation and quality assurance requires a preliminary understanding on how quality of a service or a service-based application can be described and modeled. On the one hand, quality represents the *object* of QoS negotiation and the features of the adopted QoS model may have a strong impact on the negotiation protocols and strategies that can be implemented to run negotiation and adopted by negotiation participants, respectively. On the other hand, only quality aspects that have been described or modeled can then be verified, either statically (e.g., by means of formal verification) or dynamically (e.g., by means of testing or monitoring).

According to the aforementioned rationale, this deliverable will address the following three major aspects of quality in service-based applications:

- **QoS description.** We argue that obtaining a holistic taxonomy of QoS aspects in service-based applications represents a very challenging task. Relevant quality aspects may differ (i) according to the services' application domain, e.g., domain specific quality for B2B applications is different than quality of services intended for retail customers, and (ii) on the basis of the type of services involved in the system, e.g., specifying quality for grid services in a scientific computing application is different than describing quality for application level services in a virtual travel agency scenario. Therefore, we first introduce a high level taxonomy of QoS dimensions, built on the basis of previous work on the quality of service components, and we then focus on the review of meta-models that can be used to describe service QoS in different scenarios and on the languages proposed, both in academia and industry, to describe instances of such meta-models.

- **QoS negotiation.** The review of relevant literature in the field of QoS negotiation in service-based applications shows high dependability on the classification of QoS aspects provided in the previous part. In particular, we discern between three main different approaches to QoS negotiation in service-based applications. First, we review relevant work in the field of application level QoS dimensions negotiation. In this case, QoS negotiation is usually performed adopting state of the

art techniques mostly drawn from the agent-based computing literature. Besides the need to establish QoS contracts that can be monitored at runtime, QoS negotiation often becomes a means to efficiently select services on the basis of non-functional requirements. Second, trust and security of service access and usage – which are among QoS aspects that could be negotiated at application level – show peculiar characteristics. While, in fact, performance related or domain specific QoS negotiation represent a classical multi-attribute negotiation problem, trust and privacy negotiation requires a different paradigm, oriented toward the managed and secure disclosing of credentials among parties that provide and use a service. Third, QoS negotiation and resource allocation represents a fundamental issue in grid services management. While the description of QoS aspects in grid services is not critical, the focus, in this last case, is on negotiation protocols to achieve efficient and manageable resource allocation.

- **Quality assurance.** To assure the desired quality of a service-based application, two complementary strategies can be employed: *constructive* and *analytical* quality assurance. Where the goal of constructive quality assurance is to prevent the introduction of faults (or defects) while the artifacts are created (in the sense of 'correctness by construction'), the goal of analytical quality assurance is to uncover faults in the artifacts after they have been created. In this deliverable we will cover the state of the art in analytical quality assurance for service-based applications. Constructive quality assurance will be addressed by deliverable PO-WP-JRA-1.1.1 ('State of the art report on software engineering design knowledge and Survey of HCI and contextual Knowledge'). Our review will cover three major classes of approaches for analytical quality assurance in service-based applications: (i) *Testing*, the goal of which is to (systematically) *execute* services or service-based applications with predefined inputs in order to uncover failures, (ii) *Monitoring*, which observes services or service-based applications as well as their context during their *current* execution, (iii) *Static Analysis*, the aim of which is to systematically *examine* (without execution) an artifact (e.g., a service specification) in order to determine certain properties or to ascertain that some predefined properties are met.

The overview covers literature from different approaches. The main research areas which have been covered are the following: web service, semantic web services, grid, and software engineering approaches to quality of service in web services. In the area of web services, QoS has been studied in depth in all three aspects covered in the deliverable. Semantic web service approaches have been applied mainly for description, and, to some extent to negotiation of service quality. Grid research has been focusing manly in resource management and therefore description and negotiation have been studied. The problem of the problem of quality assurance has been studied mainly in software engineering and web service research.

The remainder of this deliverable is structured as follows. Chapter 2 reviews relevant work in the field of QoS description; Chapter 3 focuses on SLA negotiation; Chapter 4 surveys relevant contributions to quality assurance for service-based applications and thus identifies contributions for the enactment of QoS contracts. The three chapters aims at identifying gaps and overlaps which might require further research. Finally, Chapter 5 identifies relevant future research directions that emerge from our review of the relevant work in QoS description, negotiation and assurance.

1.3 Relationships with other S-Cube Deliverables

Several of the topics addressed in this deliverable are closely related to the topics addressed in other work packages of S-Cube, or are addressed – with a different focus – in those work packages. The relationships of this deliverable with other S-Cube deliverables include:

- *PO-JRA-1.1.1* ('State of the art report on software engineering design knowledge and Survey of HCI and contextual Knowledge') & *PO-JRA-2.2.1* ('Overview of the state of the art in compo-

sition and coordination of services'): Where Section 4 of this deliverable focuses on analytical quality assurance techniques and methods, constructive quality assurance approaches are covered in PO-JRA-1.1.1 (specifically, process models and design methods) and PO-JRA-2.2.2 (including automated and QoS-aware service-compositions).

- *PO-JRA-1.2.1* ('State-of-the-Art report, gap analysis of knowledge on principles, techniques and methodologies for monitoring and adaptation of SBAs'): The definition and description of quality aspects from Section 2.2 identifies the kind of quality aspects what can be monitored during service execution. In addition, Section 4 of this deliverable builds on the survey results of PO-JRA-1.2.1 with respect to monitoring. In order to understand the dependencies between monitoring and other analytical quality assurance techniques and methods, we have classified these results according to the classification framework introduced in Section 4.3 of this deliverable. This was necessary, as PO-JRA-1.2.1 followed different goals in its survey classification and thus a different classification framework was applied.

- *PO-JRA-2.2.1* ('Overview of the state of the art in composition and coordination of services'): QoS negotiation, reviewed in Chapter 3, can be regarded as a means for guaranteeing local and global quality constraints in service compositions. For what concerns the verification of service compositions, we have chosen to narrow the selection of the survey in this deliverable to these works which are more related with non-functional properties such as performance prediction, structural complexity of service compositions, etc., while other approaches which lean more towards checking correctness of service compositions are examined in the deliverable PO-JRA-2.2.1.

- *CD-IA-1.1.1* ('Comprehensive overview of the state of the art on service-based systems'): The objective of CD-IA-1.1.1 is to build a comprehensive overview of the state of the art on service based systems. The review of relevant research on QoS negotiation and quality assurance in SBAs made in this deliverable constitutes one of the pillars of the aforementioned integrated research action. In addition, by actively contributing to CD-IA-1.1.1, the terminology used in this deliverable as well as other deliverables will be integrated, with the aim of supporting a common understanding and consistent use of terminology throughout the work packages of S-Cube.

1.4 Review Methodology

The review of relevant research in the fields of QoS definition, QoS negotiation and quality asurance in SBAs has followed a systematic approach. Both general conference and journals on services and software engineering have been covered, and in addition, special research area publication sites have also been covered.

First, we conducted a systematic review of major academic publications that address the above fields in their subject coverage. Specifically, we reviewed work, starting from year 2000, in the following publications:

- **Conference Proceedings**: Int. Conf. on Software Engineering (ICSE), Int. Conf. on Web Services (ICWS), Int. Services Computing Conf. (SCC), European Conf. on Web Services (ECOWS), Int. Enterprise Distributed Object Computing Conf. (EDOC), World Wide Web Conf. (WWW), Int. Conf. on Service Oriented Computing (ICSOC), Int. Conf. on Business Process Management (BPM), Int. Conference on Software Engineering (ICSE), International Symposium on Software Testing and Analysis (ISSTA), International Symposium on Software Testing and Analysis (ISSTA)

- **Academic Journals and Magazines**: IEEE TSE, ACM TOSEM

- **Digital libraries**: ACM Digital library on a keyword basis (e.g., SOA, SBS, SOA and monitoring, verification, analysis), SCOPUS, INSPEC

Other contributions for the literature review derive from the expertise of the different partners that contributed to this section of the deliverable. These include relevant work published in domain specific conferences and workshops proceedings or in specialized academic journals, all of which fall outside the aforementioned list of academic publications systematically reviewed.

Chapter 2

QoS and SLA definition

2.1 Introduction

In the present chapter the problem of specifying QoS has been analyzed. The approaches in the literature have been focusing on two main aspects: the classification of quality dimensions in service-based systems and the definition of metamodels or ontological approaches to enable the definition of the quality characteristics. Several models and languages have been proposed in the research literature and by standardization groups. Section 2.2 discusses the problem of the definition of services' non-functional aspects and, in particular, QoS. Section 2.3 revises proposals, both from academia and industry, for expressing service QoS. In particular, we focus our attention on languages for expressing QoS and on semantic approaches for defining QoS dimensions and related properties and relationships. General observations are presented in Section 2.4. Based on those observations, potential research challenges on QoS and SLA definition will be identified and summarized in Chapter 5.

2.2 Approaches to Non-functional Requirements and Quality Definition in Service-based Applications

Quality dimensions (a.k.a. quality attributes or quality parameters) express capabilities or requirements. By grouping a set of relevant quality dimensions, a service can be defined by its quality that states how the service work. About the quality dimensions, in the literature we can identify two main efforts:

- *Cataloguing the list of relevant dimensions*. Considering relevant QoS dimensions, metrics measure quantifiable QoS attributes in the applications, system, and the resources. They are further divided into performance metrics, security levels, and the relative importance. Policies describe the system behavior specifications.
- *Defining a meta-model for describing a quality dimension*. Some work defines meta-models for characterizing a QoS dimensions, stating the necessity to have a common way for expressing quality of service dimensions. Having a shared vision, service providers and service users can communicate using the same language: providers can effectively describe the potential of their services, at the same time, users can be aware of these offerings and can compare and select the most suitable service with respect to current requirements. Thus, mechanisms for service discovery, service selection, and service negotiation can be defined.

Providing a certain level of quality of various services is becoming a key issue in several currently running projects. To guarantee some level of quality is far not easy in these heterogeneous, distributed and highly dynamic systems. Previously, most distributed computing provisions followed a so-called best effort operation: a service provider attempts to provide a consistent, available service but makes

little or no guarantees on that provision. For example users are able to specify the minimum level of hardware required to run their jobs. These requirements are matched by the resource management system to available resources or the request is queued until a resource with the specified requirements becomes free. However, within this type of service provisioning users may need additional guarantees apart from those capabilities of the resources.

At the moment, most service-based applications agree and enforce these guarantees manually, often using telephone, email or web forms to contact system administrators who then perform the steps necessary to reserve grid resources [196]. Two examples are the UK National Grid Service (NGS) and the US TeraGrid. The NGS provides a web form for its users to complete which is then forwarded to system administrators. On the US TeraGrid advance reservation is performed through a telephoned or emailed agreement to system administrators, who manually create a dedicated job submission queue that is used during the period the resources have been reserved. Such manual solutions do not allow dynamic resource provisioning. The use of agreements can allow providers to reduce costs and allow them to make more efficient use of their resources by being able to plan ahead and set up new resources when there is higher demand. The vision of SOA developers is to allow loosely-coupled, shared services to be assembled into new applications that provide a greater value than the sum of its parts.

However, when composing these new applications, new problems arise: How can a proper service be chosen? How can the end-user be assured that the service will do what it advertises? This requires information about the services qualities and then an enforceable agreement of the QoS agreed between the customer and the provider, which may be on aspects of the system not just associated with its performance. The advertising and guarantee of those qualities through the formation of an enforceable contract that allows the end user to discover the quality and standard of service provisioning and then have confidence in the delivery of that service. Automated advance reservation of resources will require the use of agreements to allow the resource provider to plan its resource allocation and make the most efficient use of their resources. This can be achieved by the provider migrating less important computational tasks to cheaper or less powerful resources. This in turn can help the service provider minimize its costs thereby increasing their profits. Other examples of the benefits in using agreements in services are that by providing a guaranteed service, a service provider then can introduce different charges depending on the different levels of service guaranteed. A key component to support this type of dynamic service provisioning is the ability to negotiate, monitor and conclude a contract for service usage in the form of a Service Level Agreement. This is required in order that the obligations and expectations of the service customer and provider are clearly set out and can be monitored and assessed during the contracts lifetime and/or after the contract has completed. Contracts are one of the foundations of commerce as they form a legally binding exchange of promises between parties that can be enforced using legal recourse.

Service Level Agreements (SLAs) are used in different domains and on different levels to establish agreements on the quality of a service (QoS) between a service provider and a service consumer. An agreement defines a relationship between two parties that is dynamically established and dynamically managed. The objective of this relationship is to deliver a service by one of the parties. Each of the two parties can be either an initiator of, or a responder to the agreement. In the agreement each party agrees on the respective roles, rights and obligations. A provider in an agreement offers a service according to conditions described in the agreement. A consumer enters into an agreement with the intent of obtaining guarantees on the availability of one or more services from the provider. Agreements can also be negotiated by entities acting on behalf the provider or the consumer. An agreement creation process usually consists of three steps: The initiator retrieves a template from the responder, which advertises the types of offers the responder is willing to accept. The initiator then makes an offer, which is either accepted or rejected by the responder.

In its most basic description, an SLA is a document containing agreed terms for service provisioning [43]. The contents of this document are usually agreed between a service provider (the entity providing computational resources) and its customer (the consumer of those services), though there are situations where an SLA is agreed between the customer and a third-party broker (an entity arranging a

service for the customer and not providing any service itself) and/or between a broker and the provider.

According to a recent technical report [196], the SLAs life-cycle is composed of the following stages: first a general SLA template is formed then advertised by the provider. An individual SLA is negotiated then agreed on the basis of this template. The negotiation phase is separate to the agreement phase because a negotiation may not always lead to an agreement. Then, because the service described in the SLA may start some time after the SLA was agreed (for example in the case of advance reservation) the agreement and execution phases are also separated. The execution phase is where resources are commissioned and deployed to complete the agreed service. Whilst the service is executing (or active) the SLA is in the assessment stage where, periodically, the performance of the service is reviewed against what was agreed in the SLA. Finally, when the service has completed, it is determined if the SLA was met and what the costs to the customer are and what penalties may apply to the provider for breaking the terms of the SLA. According to each of the parties policies after this stage is completed the SLA may be archived, there is usually a limitation period, where the SLA record must be kept because it is a legal document describing how services were provided. The archive phase also provides a mechanism for providing an audit trail for later reference. To introduce SLA usage to a service environment, the following conditions should be performed:

- SLA publication and discovery: Within a distributed environment, customers need to find what services are being offered by service providers and what the qualities of their services are. There must exist a common mechanism for the service provider to publish the capabilities of their services and resources. As in the real world, these advertisements may be published to all possible users or targeted to a certain set of customers, such as those who are prompt at settling accounts or those that have used specialised resources in the past.

- Protocols for SLA formation: This means how SLAs are negotiated between the parties involved in the provisioning of services. SLAs are part of a formal contract for service provisioning. This formal contract is only as legally enforceable as the protocol used to form that contract. The protocol should promote interoperability between customers and service providers, protect the service provider from denial-of-service (DoS) attacks and truly perform negotiations (defined as a multi-round discussion aimed at reaching an agreement). The advertising and agreement processes used in the negotiation process should have the capability to re-negotiate a contract.

- SLA representation: Like the protocol used to form SLAs, the choice of SLA representation also affects the interoperability between a service provider and its potential customers. For example, if the service provider chooses a proprietary format to represent their SLAs, it must expect the customer to understand the content and its meaning before the customer can commit to a contract based on that content: if the customer insists on using its own format the service provider will not be able to interpret it. Either situation leads to the agreement process breaking down before it has begun because neither party can understand the other. Therefore it is important to define, how the QoS terms, guarantees, payment and penalty terms are represented within the SLA.

- SLA monitoring, evaluation and accounting: Once an agreement is formed between a service provider and a customer the resources provided must be monitored and their capabilities and performance constantly evaluated to ensure that they are meeting the QoS guarantees agreed in the SLA. The ability to account and audit resource use is closely linked to the monitoring function, therefore the accounting and evaluation must take place together to determine what is currently happening in the system as a whole.

In the present section we focus mainly on SLA representation, together with necessary support for publication and discovery.

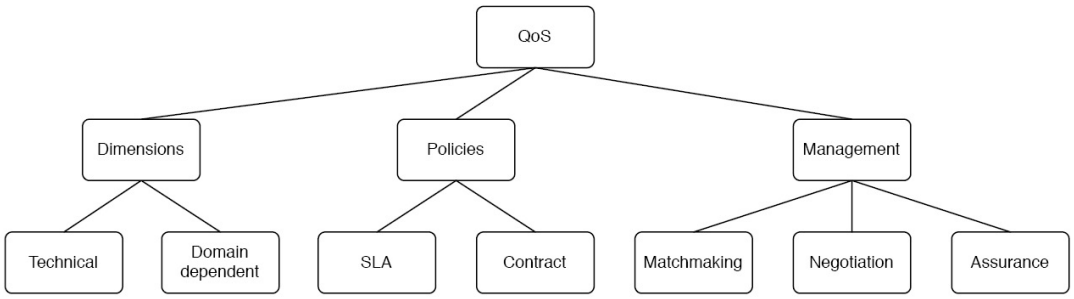

Figure 2.1: QoS relevant aspects

2.2.1 Quality Dimensions and their Characterization

In a Service Oriented Architecture, service providers need to characterize their services defining both the offered functionalities and the offered quality. At the same time, users not only express their requirements by listing the desired functionalities, but also define a minimum level of quality that the service must ensure. The main issue is due to the subjectiveness of the quality: the quality of a service from a provider standpoint might be different than the quality from the user standpoint. At the same time, the same quality level might be sufficient for a given user and not enough for another one. Thus, an effective quality of service definition must mediate between the intrinsic subjectiveness of quality definition and the objectiveness required whenever we have to compare different standpoints. Extending the taxonomy proposed by Sabata et al. [218], Figure 2.1 shows all the aspects that are relevant for defining the quality of service and, in particular, the quality of Web services. At the first level, we distinguish between how to define the QoS, i.e., *quality dimensions*, how users and providers agree on the quality of service, i.e., *policies*, and which are the mechanisms for managing quality dimensions and policies, i.e. *management*.

Possession of knowledge of QoS offered by some service, in combination with ability to request a certain level of QoS for a particular service, implies an unavoidable need for QoS expression.

However, widely adopted service description languages (in particular WSDL) do not include expression of QoS properties along with other properties of a service. Before service providers and clients can make any decision based on quality of a service, both sides have to agree on a QoS model, which describes basic QoS parameters and their possible combinations.

This section reviews relevant approaches for QoS modeling and description in the fields of Web services, Semantic Web services, and Grid services.

For what concern Web services, we first review models for QoS description proposed by practitioners and standardization consortia, for creating agreement on QoS aspects between companies that endorse the service oriented paradigm. Second, we revise relevant contributions in the context of XML-languages that can be adopted for the interoperable exchange of quality specifications.

Quality dimensions provide a shared knowledge about how a quality can be defined in terms of names, metrics, relationships and dependencies among quality dimensions. The set of quality dimensions that are considered relevant for a service will be used by all the actors involved to express the requirements and capabilities, to compare them, and to select the best service. In the literature, several work propose list of quality dimensions classified in different ways. For instance, similarly to the ISO 9126, Sommerville [236] identifies three main categories of non-functional requirements (i.e., process requirement, product requirements and external requirements) that cover the whole software life-cycle. Focusing on the product quality specification, we need to consider two main issues: the *technical quality dimensions* and the *domain dependent quality dimensions*.

Technical quality dimensions include all the quality dimensions that characterize the service provisioning and that are relevant regardless of the kind of service. For instance, the service availability is considered an important quality aspect even if we are considering a travel-reservation service, rather

than a bank account service. About the the Web service community, Ran [204] identifies a set of technical quality dimensions, grouped in 4 main categories (runtime, transaction, configuration management, and security), that are considered relevant for describing a Web service: i.e, scalability, capacity, performance, reliability, availability, flexibility, exception handling, integrity, regulatory, supported standard, stability, cost, completeness. In Chung et al. [56] 161 dimensions are listed and some of them deeply introduced according to the NFR (Non-Functional Requirements) Framework proposed by the same authors. These dimensions refer to the accuracy, security, and performance requirement categories. In case the final product of a service are data, information quality literature has a rich set of approaches for identifying and then classifying the quality dimensions. *Data quality* can be measured along the following dimensions [206, 240]:

- for data views: Relevance, granularity and level of detail;
- for data values: Accuracy, consistency, currency and completeness;
- for data presentation: Format and ease of interpretation;
- general dimensions: Privacy, security, trust, and ownership

Trust and security dimensions are getting more and more attention in the service literature.

The growing number of accessible services in open distributed systems, calls for security enforcement. Examples of such systems include the World Wide Web, grid computing systems, and distributed intelligent systems, and ad-hoc networks used by joint military task forces [107] [213]. The security needs can potentially include guarantees of confidentiality, integrity, authentication, authorization, availability, auditing and accountability [187, 14]. A significant issue for service-based applications is that if they are publicly accessible, such as over the Internet, there is no way of knowing in advance all the users that will be accessing the service. To ensure that users of a Web service gain appropriate access when no relationship exists between the user and the service provider is a challenging task which requires a flexible approach to access control which is called *trust* mechanism.

Some surveys have been provided regarding trust; [107] provides an overview of trust in Internet, [14] discusses trust in computer science and semantic web, [213] examines trust management.
[14] organizes trust research in four areas: (1) policy-base trust, using policies to establish trust, focused on managing and exchanging credentials and enforcing access polices; (2) using reputation to establish trust where past interactions or performance for an entity are combined to assess its future behaviour; (3) general models of trust, where trust models are useful for analyzing human and agentized trust decision; (4) trust in information resources which is related whether the web resources and web sites are reliable.

Our study is focusing on the first area. Traditional access control models rely on knowing requester identities in advance [85]. Service-based applications typically have large and dynamic requester populations. This means that requesters' identities are seldom known in advance. Most existing Web applications deal with strangers by requiring them to first register an identity at the Web site. Such approaches do not fit into the Web service philosophy of dynamically choosing services at run-time. So traditional assumptions for establishing and enforcing access control regulations no longer hold. Trust negotiation is an access control model that addresses this issue by avoiding the use of requester identities in access control policies [281, 33]. Instead, access is granted based on trust established in a negotiation between the service requester and the provider. In this negotiation - called a trust negotiation - the requester and the provider exchange credentials. Credentials are signed assertions describing attributes of the owner. Examples of credentials include membership documents, credit cards, and passports. The attributes of these credentials are then used to determine access. For instance, a requester may be given access to resources of a company by disclosing a credential proving she is an employee of that company. This example shows that the requester identity is not always needed to determine access. Credentials are typically implemented as certificates.
To summarize, the goal of a trust negotiation is to find a sequence of credentials $(C1, ..., Ck, R)$, where

R is the resource to which access was originally requested, such that when credential Ci is disclosed, its policy has been satisfied by credentials disclosed earlier in the sequence or to determine that no such credential disclosure sequence exists (more details about trust negotiation will be discussed in section 3.3).
In recent years, trust negotiation has been proposed as a novel authorization solution and as an automated technique for the establishment of trust and the enforcement of need-to-know between entities in open-system environments, in which resources are shared across organizational boundaries.

Domain dependent quality dimensions includes specific aspects that are relevant for a set of service and are used by service requesters to select which is the best service among a set of functionally equivalent ones.

The policies define an agreement between service users and service providers. Such an agreement includes the quality dimensions relevant for both the actors, and the values that are admissible during the service exploitation. Usually, the agreement is defined according to several configurations, i.e., *Service Level Agreement, SLA*. In addition, the SLA can be enriched also specifying responsibilities about the assurance of the quality level, and the entail penalties in case the agreement is not fulfilled.

Finally, the *management* includes all the mechanisms for supporting users during the service selection (*matchmaking*), to obtain the policies (*negotiation*) and to ensure their fulfillment (*assurance*).

2.2.2 Metamodels and Ontologies

In the presence of multiple Web Services with overlapping or identical functionality, service requesters need objective QoS criteria to distinguish one service from another. It is argued in [149] that it is not practical to come up with a standard QoS model that can be used for all web services in all domains. This is because QoS is a broad concept that can encompass a number of context-dependent non-functional properties such as privacy, reputation and usability. Moreover, when evaluating QoS of web services, domain specific criteria must be taken into consideration. For example, in the domain of phone service provisioning, the penalty rate for early termination of a contract and compensation for non-service, offered in the service level agreement are important QoS criteria in that domain. Therefore, an extensible QoS model must be proposed that includes both the generic and domain specific criteria. In addition, new domain specific criteria should be added and used to evaluate the QoS of web services without changing the underlying computation (i.e. matchmaking and ranking) model. Last but not least, the semantics of QoS attributes/concepts must be described in order to ensure that both WS provider and consumer talk about the same QoS attribute/concept.

Sometimes, generic QoS attributes with the same name like “application availability” may have different meanings to the parties that describe them (network level of the hosting system, application that implements the service) or they may be computed differently. Other times, domain-dependent QoS attributes may have the same name but obviously different meaning. So it is important to describe QoS attributes/concepts not only syntactically, but also semantically in order to have a better discovery (matchmaking) process with high precision and recall.

Two main approaches have been proposed in the literature:

- Metamodels and
- Ontologies.

Metamodels use a model-based approach to represent the characteristics of quality attributes or dimensions and to relate them to services and to their use by interested parities. Several models have been proposed as a basis for QoS metamodeling, including UML and Object-Role Model. The characteristic of these approaches is to provide a high level, semi-formal description of quality, focusing on syntactic aspects and leaving the semantic aspects informally defined.

Ontologies provide a formal, syntactic, and semantic description model of concepts, properties and relationships between concepts. They give meaning to concepts like QoS attributes, QoS offers, domains,

services so that they are human-understandable and machine-interpretable while they provide the means for interoperability. Moreover, they are extensible as new concepts, properties or relationships can be added to an ontology. In addition, Semantic Web techniques can be used for reasoning about concepts or for resolving or mapping between ontologies. These techniques can lead to syntactic and semantic matching of ontological concepts and enforcement of class and property constraints (e.g. type checking, cardinality constraints, e.t.c.). Therefore, by providing semantic description of concepts and by supporting reasoning mechanisms, ontologies cater for better discovery process with high precision and recall. Last but not least, ontologies can help specialized agents/brokers in performing very complex reasoning tasks like WS discovery or mediation.

For the above reasons, many ontology-based approaches have been proposed and implemented. Most of these approaches focus only on the QoS definition part of WSs for the purposes of WS discovery while only one [182] goes one step further producing a semantic language for WS agreements. All of these approaches differ in their expressiveness [138], connection to a WS functional description language and the occurrence or not of an implementation framework supporting WS discovery. In this section we focus on upper level ontologies, while semantic quality languages are described in the following section.

Non-functional service properties based on the Object-Role Modelling (ORM) [189]

O'Sullivan et al. [189] proposes a formal description of non-functional service properties based on the Object-Role Modelling (ORM). Using this approach, the authors define foundational concepts as: service provider, temporal model, locative model, service availability, obligations, price, payments, discounts, penalities, rights, language, trust, quality, security. About the qualify, Figure 2.2 shows the definition of a QoS dimension and the relationships with the other concepts introduced in the paper.

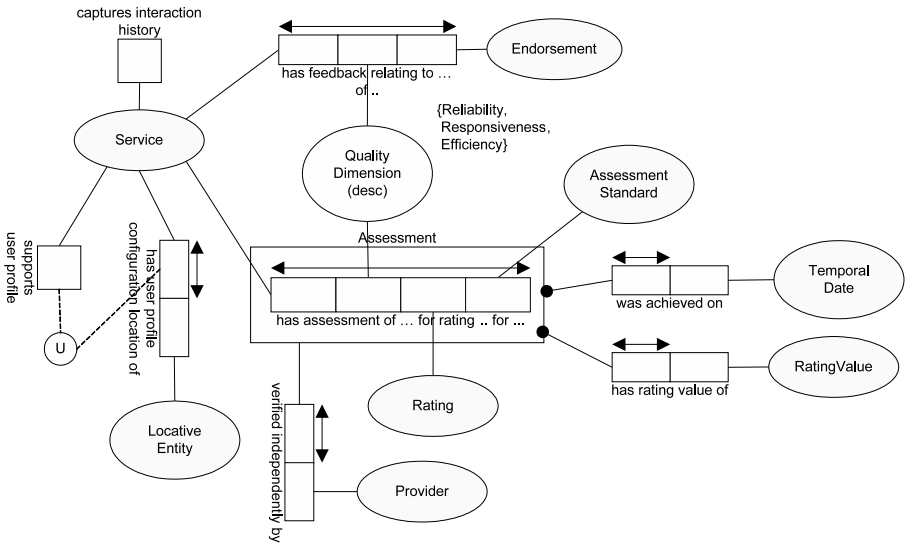

Figure 2.2: Service quality

QoS Modelling Language [95]

QML (QoS Modelling Language) [95] is another research effort for the specification of a QoS meta-model. It was designed according to some basic principles for the support of QoS specification. It contains the following constructs:

- *contract type*: Definition of a QoS dimension that includes definitions for the metrics of this QoS dimension;

- *contract*: Gives particular values/constraints to the fields of a contract type. This is where the idea of contract inheritance is implemented;
- *profile*: One service is associated with many (QoS) profiles. Each profile consists of one list of contracts/requirements.

Each contract may describe constraints for a QoS dimension either for the whole service or just for one service operation. But for every QoS dimension, at most one constraint will be valid for one operation of the functional interface of the service. One (QoS) Service Profile P is matched with one client profile Q if all contracts of P conform to all the contracts of Q. Contract conformance is translated into constraint conformance. Considering its expressivity, QML conforms to many of the requirements set in [138] mentioned in Section 2.3. However, it is only an abstract language not tied to any particular formalism. In addition, there is no implemented framework supporting it.

OMG metamodel to define quality of web services [245]
In [245] the OMG group does not actually propose a model to define quality of web services. It proposes a UML profile (i.e., a definition of entities and their relations) with with generic quality of services reflecting non-functional characteristics can be uniformly represented. This model is intended to represent the concepts contained in a *metamodel* (an abstract description) which defines what quality of service is and how it can be used. This metamodel (which is in itself described using UML) is decomposed into:

QoS characteristics This submetamodel includes the *names* (constructors) of the non-functional characteristics in the QoS model (e.g., latency, throughput); the *dimensions* in which each characteristics is measured (e.g.: reliability can be measured in MTBF, time to repair, etc), as well as the direction of order in the domain, its units, associated statistics, etc.; the possibility of grouping several characteristics (e.g., the performance *category*); the description of the values that quantifiable QoS characteristics can take, and others.

QoS constraints This makes it possible to express the limits the values that some QoS characteristic can take, and also to specify whether this limits are offered (guaranteed by a provider) or requested (needed by a client). UML classes make it possible to represent a QoS contract which links offered and requested QoS levels between two participants, including the degree to which the constraints are to be satisfied (e.g., *Guarantee*, *Best-Effort*, ...) and their extent (e.g., *end-to-end*). Constraints can be composed of other constraints.

QoS Level This defines the different modes in which a system can be. Depending on, e.g., available resources, a different execution level can be jumped to if continuing the execution in the current one is not possible. The submetamodel defines the abstract classes to represent levels, transitions between them, and when those transitions have to take place.

The metamodel is complemented with a profile which extends the metamodel in a constrained way, with a catalog of some categories which group characteristics (e.g., performance characteristics, security characteristics), expressed through UML models. This proposal does not provide any hint on how to calculate QoS for a given service. However, it does provide a structured representation of the different traits than can be needed to state / calculate the QoS of a services, and also on the relationship between these traits.

2.3 QoS Modeling and Related Languages

This section reviews relevant approaches for QoS modeling and description in the fields of Web services, Semantic Web services, and Grid services.

For what concern Web services, we first review models for QoS description proposed by practitioners and standardization consortia, for creating agreement on QoS aspects between companies that endorse the service oriented paradigm. Second, we revise relevant contributions in the context of XML-languages that can be adopted for the interoperable exchange of quality specifications.

QoS requirements for web services [138]

Kritikos and Plexousakis [138] explicitly define the set of requirements that a Web service description needs to satisfy. In detail, the requirements are the following.

- *An extensible and formal semantic QoS model*: the QoS model has to include both domain independent QoS dimensions, and domain specific QoS dimensions. In addition, new domain specific criteria should be added and used to evaluate QoS without changing the underlying computation (i.e. matchmaking and ranking) model. Finally, the semantics of QoS concepts must be described in order to have terms/concepts with specific meaning for both WS requesters and providers.
- *Standards compliance*: QoS-based WS description model has to be compliant with already widely-accepted standards. In this way, it will be easily adopted by the research community. In addition, it will use all freely-available tools related to these standards for its development.
- *Syntactical separation of QoS-based and functional parts of service specification*: QoS specifications should be syntactically separated from other parts of service specifications, such as interface definitions. This improves reusability and flexibility in describe several service with the same QoS or a service with different level of QoS.
- *Support refinement of QoS specifications and their constructs*: QoS specifications should not only be reused but also refined: create a new WS QoS offering by referencing an older one and by adding constraints like refinement of an older QoS restriction or creation of a new one.
- *Allow both provider and requester QoS specification*: It should be possible to specify both the QoS properties that clients require and the QoS properties that services provide. These two aspects should be specified separately.
- *Allow fine-grained QoS specification*: It should be possible to specify QoS properties/metrics at a fine-grained level. As an example, performance characteristics are commonly specified for individual operations. A QoS model must allow QoS specifications for interfaces, operations, attributes, operation parameters, and operation results.
- *Extensible and formal QoS metrics model*: For each domain, the attributes are important inputs to the overall QoS of a service. Some attributes are common across domains and some are specific to domains. Each attribute is measured with the help of a metric. Each attribute/metric has the following aspects:
  - The value set for the metric (and its allowed value range).
  - The domains that this attribute belongs to.
  - The weight of the metric relative to its domain and user preferences.
  - The characteristic of the function from metric values to overall QoS values.
  - The temporal characteristic of the metric value.
  - There must be a description (mathematical or otherwise formal) of how a QoS metric value of a complex WS can be derived from the corresponding QoS metric values of the individual WSs that constitute the complex one.
  - A set of reference ontologies: e.g., ontology of measurement units, ontology of currency units, ontology of measured properties and ontology of measurement methods.

- *Allow classes of service specification*: Class of Service means the discrete variation of the complete service and QoS provided by one WS.

UDDI approaches [205]

The UDDI (`http://www.uddi.org/pubs/uddi_v3.htm`) WS standard is dedicated to the description and discovery of Web Services. However, it is based on the tModel concept which leads to purely syntactic queries. In addition, there is no QoS description of offers or demands in the UDDI description model.

In [205], an extension to UDDI is proposed. A new data structure type - called qualityInformation - is added to the UDDI model that represents description of QoS information about a particular WS. This proposed data structure is under the businessService data structure type, in addition to bindingTemplate data structure type, which provides binding information for a particular service. The qualityInformation data structure type also refers to tModels as references to QoS taxonomies which also need to be defined in the extended UDDI registry. These taxonomies define the new terminologies or concepts about the proposed QoS information, which do not exist in the existing UDDI registries.

The main disadvantage of this approach is that there is no actual description about the contents of the qualityInformation data structure type and its referenced tModels. In addition, it relies on the UDDI technology and UDDI's tModel, so it can be used only for syntactic matchmaking of QoS terms and specifications.

Calculation of QoS values [69]

The work described in [69] proposes calculation of values of those QoS attributes of a WS that it is not likely to be provided/measured by service providers or third-party software modules. These attributes are usually measured by the values users submit from their experience of using the particular service. So the calculation of the value of the described QoS attribute will be a function of what value it should had taken and what value was eventually offered. This work proposes that we should take into account only the ratings of users that have the same expectation for the value of a QoS attribute as with the one of the requesting user. This expectation does not only depend on the advertised value of a QoS attribute but also on other (contextual) factors like previous user experience from the service, the service usage cost, and recommendations from friends. So this expectation depends on the context of the user. This work presents many disadvantages. First of all, QoS calculation is based on triples stored in a registry. These triples have the form (expected value, perceived value, rating) and are related to a QoS attribute. However, all parts of the triple take values from the [0, 1] space of real numbers. In addition, we trust all users who submitted ratings. Moreover, user expectation is only modeled by the expected value of a user. So what is needed is better and richer representation of user expectations and better techniques for matching expectations. Last but not least, this work does not explain what happens when the expectation for a QoS attribute of the requesting user is totally different from the ones collected.

Modeling WS reputation [163]

In [163], an architecture and model of Web Service reputation (QoS) is presented. It is proposed that for successful description of QoS, three challenges must be dealt with: a) definition of a QoS conceptual model for WS attribute; b) semantics to QoS service attributes must be added; c) reputations should consider time and history of endorsements and ratings. Based on the above requirements/challenges, a conceptual model of WS reputation is proposed which is used for the calculation of a WS reputation and is influenced on the following factors: a) relative weights given to QoS service attributes by the requesting user; b) QoS attribute aggregation algorithm for each QoS attribute; c) the set of endorsers of the service and the list of trusted endorsers of the user; d) the history of the service; e) damping for the rating such as older ratings matter less than newer ratings. The suggested conceptual model includes a model of QoS attributes. In this model, for each attribute the following aspects are defined: a) its type and allowed values; b) the domains it belongs to, along with a weight of this attribute in relation to the

enclosing domain and user preferences; c) the characteristic function from attribute values to ratings; d) the time characteristics of the values of this attribute.

The main disadvantages of this work are the following. First of all, the reputation of a WS is calculated and not it's QoS. Reputation should be considered as one QoS attribute. Another disadvantage is that there is no explicit clarification of how the reputation of a WS is calculated. In addition, concepts like QoS constraints and QoS offers and demands are not modeled. Last but not least, the QoS metrics conceptual model does not contain the classes and properties described in [248].

WSDL extension for constraint specification [67]

In [67], there is an extension to WSDL in order to include constructs for constraint specification of WSs. By using these constructs, a service provider can specify meaningful attributes for characterizing a WS and its operations. Attribute constraints and inter-attribute constraints can be explicitly defined. In addition, a Constraint Satisfaction Processor was developed for matching the requirements given in a service request against the constraints of registered services in the service discovery process. This processor and some additional components are integrated with the IBM's UDDI registry to form a Constraint-based Broker. The drawbacks of this approach are the following: a) The QoS model is not rich enough as it does not cover every possible aspect. b) The semantics of QoS metrics and of other entities is missing. c) The constraint specification language is not rich enough as it only allows unary attribute constraints and simple inter-attribute constraints of the form "if A then B". This language does not also include linear and non-linear attribute functions.

Extensible QoS model [149]

The research work described in [149] discusses the need for an extensible QoS model that not only contains general but also domain-specific QoS criteria. It sustains that QoS must be represented to users according to user preferences and users should express accurately their preferences with this QoS model without resorting to complex coding of user-profiles. It also suggests that QoS computation must be fair and open for providers and requesters. Then it proposes an extensible QoS model. However, this model is just a description of some general QoS criteria and other criteria that can be grouped and not a formal ontological description of QoS constraints regarding some metrics that are also defined by suitable ontologies.

QRL[58]

QRL [58] is a simple but powerful text-based language used mostly to describe complex constraints on QoS metrics and to analyze how the assessment of QoS metrics will take place based on their values or range of values. Its main highlights are: a) Both QoS offers and demands contain not only constraints about their capabilities but also requirement constraints on the capabilities of the other party. So it follows a symmetric approach of QoS-based WS description for both providers and requesters; b) The QoS demand has a specific part that realizes the specification of the utility function of a QoS metric. So the selection part of the WS discovery process becomes more easier to implement; c) It relies on the powerful OPL (`http://www.ilog.com/products/oplstudio/`) language for describing Mathematical or Constraint Programming Problems.

Its main drawbacks are: a) a) the language is not XML based, and its full specification is not given; b) it is not quite expressive with respect to its QoS model; c) QoS parameters and their metrics are described in a single common text-based catalog which is of course not easily maintained; d) it is transformed to OPL so it relies on this language and its specific capabilities.

Oasis WSQM [244]

Oasis proposes in its Web Service Quality Model (WSQM) [244] a model for quality of services in which three components interplay:

Factor	Definition
Availability	$1 - \frac{Down\ time}{Unit\ time}$
Successability	$\frac{Number\ of\ response\ messages}{Number\ of\ request\ messages}$
Accessability	$\frac{Number\ of\ acknowledgements\ received}{Number\ of\ request\ messages}$
Max. Throughput	$\frac{\max Completed\ requests}{Unit\ time}$

Table 2.1: Some Service Level Measurement Quality factors for WSQM.

- Quality Factor: the different dimensions and measures of the quality.
- Quality Associates: the organizations or people related to inspection, loading, provision, and use of web services. These associates may be providers, developers, stakeholders, etc.
- Quality Activity: all the actions which can be taken by the Quality associates related to ensuring the stability of the web service, such as (outstandingly) contracting, but also search, delegation, monitoring, etc.

The overall WSQM is defined (but not formally) through the conglomerate of several *quality dimensions*, namely the *business value quality*, the *service level measurement quality*, the *interoperability quality*, the *business processing quality*, the *manageability quality*, and the *security quality*. Every of them is decomposed int a set of *quality sub-factors*, and associated quality contracts and quality associates and (in some cases) standards are identified.

Business Value Quality This measures the value that using a web service brings to the business itself. It is dependent on the type of business, and is decomposed into *business suitability*, *business effect*, and *business recognition level*. Hints to calculate these sub-factors, and how to compose them together, are given.

Service Level Measurement Quality This deals with the user perception of the web service, and is divided into performance (i.e., response time, maximum throughput) and stability (i.e., availability, successability, and accessibility). Numerical formulas are given to calculate these indicators (see Table 2.1).

Interoperability Quality This measures the degree of compatibility of a web service with established standards, such as SOAP, WSDL, and UDDI.

Business Processing Quality This factor is a measure of the swiftness and accuracy with which the business process is actually being executed. This is divided into three sub-factors: reliability of messaging (where unreliable communications happen, and where certain properties of the messaging system have to be guaranteed), transactionality (i.e., A.C.I.D. properties), and collaborability (i.e., the ability to include execution of distributed web services within a business process).

Manageability Quality This factor measures the quality of the web service from the viewpoint of the maintainer or developer. It includes subfactors such as introspectability, controlability, and notifiability of the web service and of the platform, measured both taking into account the possibility that these can be done and taking into account wether they are actually done. These are, clearly, those more closely related with other classic notions of software quality.

Security Quality This last factor deals with the ability to fend off or avoid altogether unauthorized accesses to the system and to provide integrity of the data. Security is subclassified into several

subfields (data confidentiality, data integrity, authentication, access control, non-repudiation, accessibility, audit trail, privacy) and two sub-factors are identified: security at the transport level and at the message level. For each of them a large series of components, together with the technologies enabling them, are identified (e.g., use of TLS or IPSEC to ensure user authentication at the transport level, or SOAP messages signed with XML-DISG at the message level). Several security profiles are defined depending on which quality factors are guaranteed by a web service.

WSQM is a quite wide model, tackling quality of service from the point of view of several parties. However, as a result of the breadth of the approach, we can object to lack of well defined indicators for objective comparison of quality levels for some factors.

HP language to specify service level agreements [219]

HP Labs have devised a language to specify service level agreements in a flexible, extensible way [219]. Their proposal focuses on the ability to express QoS constraints taking into account that they may be seen from several points of view:

- **When** should a service check for SLA conformance (e.g., right after every invocation, as average of n invocations, ...)?
- **Which** inputs are necessary for checking SLA conformance?
- **Where** is the conformance monitored (e.g., by the provider, by the client)?
- **What** are the factors monitores and *How* are they actually meassured?

Within all these points of view, this proposal is somewhat better suited for time-related constraints, than for other interesting kinds of constraints. Table 2.2 shows an abstract syntax for their specification. `Clause` contains the exact information regarding the expected performance. *measuredItem* defines what is being measured (e.g., messages, operations, ports, etc.); *evalWhen* specifies when it has to be measured; *evalOn* specifies the range of data on which the evaluation takes place (e.g., delivery time of a message, or average over some time span); *evalFunc* is the function which is applied to the data to obtain the final QoS evaluation; finally, *evalAction* is the operation to be executed upon measurement (which could take, for example, corrective actions).

The concrete syntax used in the specification is contained in a XML schema.

IBM WSLA [131]

IBM's WSLA model [131] aims at being a generic framework on top of which different specific QoS models can be built. As such, it is a rich framework which tries to provide basic blocks to specify metrics, constraints, SLA parameters, etc. The framework also makes it possible to delegate part of the SLA enforcement to third parties and aims at achieving seamless integration with state-of-the-technique E-Commerce systems. Similarly to [219], specifications following the WSLA model try to state what are the SLA parameters, which service they are related to, how they are computed, and which metrics are used. Unlike other proposals, the aim of the model is to make this as flexible and customizable as possible.

The SLA definitions are contained in a document which extends the WSDL document(s) corresponding to the service(s) being monitored. These SLA definitions can be applied either to separate operations, or refer to the web service as a whole – even to compositions of web services, and cover negotiation, deployment, SLA measurement and reporting, corrective actions (when necessary), and termination.

IBM's WSLA framework [131] has an associated language, WSLA, extensively specified both in syntax and runtime semantics in [152]. WSLA documents have three main sections: the **Parties** section,

identifying all the parties taking part in a SLA; the **Service Description**, which specifies the characteristics of the service and its observable parameters; and the **Obligations**, which defines guarantees and constraints on SLA parameters.

The language itself is XML-based and covers all the concepts in the framework, including, for example, the possibility of defining metrics, where the party containing the source of the data for the metric can be expressed, the function to be applied to give the final result (and its units) can be written, and the metric can be composed of other metrics.

The WSLA monitoring services are automatically configured to enforce an SLA upon receipt. The SLA management life-cycle of WSLA consists of five stages:

- Negotiation/Establishment: In this stage an agreement between the provider and the consumer of a service is arranged and signed. An SLA document is generated.
- SLA Deployment: The SLA document of the previous stage is validated and distributed to the involved components and parties.
- Measurement and Reporting: In this stage the SLA parameters are computed by retrieving resource metrics from the managed resources and the measured SLA parameters are compared against the guarantees defined in the SLA.
- Corrective Management Actions: If an SLO has been violated, corrective actions are carried out. These actions can be to open a trouble ticket or automatically communicate with the management system to solve potential performance problems. Before all actions regarding the managed system the Business Entity of the service provider is consulted to verify if the proposed actions are allowable.
- SLA Termination: The parties of an SLA can negotiate the termination the same way the establishment is done. Alternatively, an expiration date can be specified in the SLA.

WS-Policy [20]

IBM's WS-Policy [20] is a W3C recommendation which aims at describing a model (and an XML-based language) with the ability to express different types of policies (*policy assertions*) and their composition. The language design is not made concrete at the level of individual policy assertions, but it focuses more on the combination of several (maybe nested) policies to generate more complex policies. Among the combination patterns in the model we can cite:

- Policy intersection,
- Requirement that a least one out of a non-empty collection of policies is enforced,
- Requirement that all policies in a non-empty collection of policies are enforced.

The model permits referring to policies using an URI, and has also the notion of "normal form" of policies (since, due to the combination patters above, several ways of writing the same combination are possible). In order to have shorter policy expression, additional attributes are also defined (for example, to express that a policy is optional). A series of XML transformation operators are defined so that compactly expressed policies can be transformed into a *normal form*, which can then be used to, for example, compare different policies.

WS-Agreement [277]

Many approaches use the WS-Agreement for SLA representation. The work carried out by the Grid Resource Allocation Agreement Protocol (GRAAP) Working Group of the Open Grid Forum [2] has led to the development of WS-Agreement [277], a specification for a simple generic language and protocol to establish agreements between two parties. It defines a language and a protocol to represent the services of providers, create agreements based on offers and monitor agreement compliance at runtime. The agreement structure is composed of several distinct parts: Name, Context and Terms of Agreement. The latter is also divided in service description terms and guarantee terms. Service descriptions terms mainly describe the functionality to be delivered under the agreement. The guarantee terms denote the assurance on service quality for each item mentioned in the service description terms section of the WS-Agreement. In grid resource management such assurances may be denoted as a parameter (constant) or bounds (min/max) on the availability of part or the whole of the resource. In the WS-Agreement, such assurances are referred to as Service Level Objectives (SLOs); in a domain specific to computation services provision, they are usually expressed as values. Each SLO may refer to one or more business values, called Business Value Lists (BVLs). This list expresses different value aspects of an SLO. The other two types of guarantee terms are Qualifying Conditions and Importance, which have a similar function to SLO and BVL, respectively.

Web Service Offerings Language (WSOL) [249, 250]

WSOL [249, 250] (Web Service Offerings Language) is a WSDL-compatible language to express the QoS that a given service can offer as well as QoS constraints. Functional, non-functional constraints, and access rights[1] can be expressed within WSOL in a homogenous way that is non-intrusive to the WSLD description, using QoS constraints. A QoS constraint contains specification of what QoS metrics are monitored, as well as when and by what entity, and usually describe QoS guarantees.

QoS constraints can be grouped in classes of services, termed *service offerings* ((also possible within the metamodel of [245]) which bundle together related QoS constraints; different classes of QoS can be separately applied to a single web service. One advantage in doing that is that changes in the environment conditions (due to, e.g., network problems or mobility) can be worked around by renegotiating the QoS with the same service which was being accessed, without the need to start another search and composition phase – unless, of course, the alternative service offering is not satisfactory.

The available types of constraints are defined in an XML schema, and these may usually refer to arithmetic and boolean operations. However, metrics and measurement units are assumed to be defined in an external ontology. Interestingly enough, WSOL makes it possible to define, besides post-conditions of the web service operations, constraints that have to be checked some time (to be defined) after an operation takes place, periodically, etc.

The (dynamic) relationship between service offerings is not represented in WSOL. It is, rather, represented in a specific XML format as triplets

$$\langle ServOff_1, ContrState, ServOff_2 \rangle$$

where $ServOff_1$ and $ServOff_2$ are the initial and replacement service offerings and $ContrState$ are the constraint and statements which were not satisfied by $ServOff_1$.

Among the shortcomings of this proposal we can cite the integration of constraint dimensions and the need to improve the specification of relationships between service offerings to support both easier and more flexible specification and dynamic adaptation. Additionally, there is no specification of the QoS demand of the consumer and the ontologies for metrics are, to the best of our knowledge, not yet developed.

[1] As well as management statements (management responsibility, prices, monetary penalties) and different reusability constructs.

```
SLA             =  Dateconstraint Parties SLO*
Dateconstraint  =  Startdate Enddate Nextevaldate
SLO             =  Daytimeconstraint Clause*
Dateconstraint  =  Day* Time*
Clause          =  MeasuredItem EvalWhen EvalOn EvalFunc EvalAction
MeasuredItem    =  Item*
Item            =  MeasuredAt ConstructType ConstructRef
```

Table 2.2: Abstract Scheme of a SLA Specification.

Ontologies for QoS [248]

[248] describe that for the specification of constraints for QoS metrics, five ontologies must be developed from which the most important (the top one) is the metrics ontology. They describe the structure and involved elements in four out of the five ontologies. However, they just stayed on the requirements for the specification of the metrics ontologies. They did not develop any ontology. In addition, the requirements specified are incomplete according to the requirements posed in [138]. For example, the 'metric' class consists only of five attributes while other important attributes/properties are missing. As another example, they imagine that metrics should be related to each other. However, they do not describe all the types of relationships that can appear between QoS metrics.

OWL-S [241]

The OWL-S [241] ontology is a semantic approach for the description of Web Services. It has many advantages in respect with the other WS description standards but it does not describe QoS offers or demands. It only contains an attribute used for rating a WS. However, as it is an ontological approach, it can be extended in order to describe QoS offers or demands.

In [285], DAML-S (OWL-S) Web Service description language is extended to include a QoS specification ontology called DAML-QoS. This is achieved by the following: a) A *ServiceProfile* element is associated to many QoS profiles (service offerings); b) External ontologies in DAML for metrics and units are referenced or developed; c) Existence of a *BasicQoSProfile* containing all the basic metrics and ability to inherit/extend this type of profile to provide constraints and/or include custom-made metrics. DAML-QoS is supported by an implemented QoS-based WS discovery framework.

The deficiencies of this research effort are the following: a) the proposed ontology is quite limited, while the QoS ontology vocabulary is absent; b) The QoS metrics values are restricted to have the set of natural numbers as their range for better reasoning support. However, this leads to imprecision and errors up to one half of measurement unit. Moreover, this flaw is actually the result of a misuse of OWL's cardinality constraints.

Relations among QoS attributes [163, 164]

Work analyzed in [164] is actually a continuation of the work in [163]. The requirements of the work in [163] have been translated to a quite expressive ontology language. Its main highlight is the formalization of relationships between QoS attributes. When a QoS attribute depends on another one, then either its values influence the values of the other with a specific impact or the values of these attributes change in a parallel or in inverse parallel way. A framework using the ontology to support dynamic web services selection is also outlined.

The main drawback of the proposed ontology is the lack of a metric model. In addition, this ontology lacks both an openly available implementation and links to a semantic description WS language like OWL-S.

QoSOnt and upper level ontologies [74, 264, 137]

QoSOnt [74] is another carefully designed ontology for semantic QoS-based WS description. Its main features are: a) ability to measure QoS attributes by many metrics; b) application of a QoS metrics to either whole WS or a single operation; c) approach to unit conversion using SWRL rules; and d) direct connection to OWL-S.

This is a very good approach but not a complete one. In addition, it is not accompanied by a formal WS discovery framework.

The work in [264] proposes an upper-level ontology that uses the main features of the ontologies produced by the work of [164, 74]. In addition, a QoS ontology vocabulary (actually a mid-level ontology) has been designed for domain-independent QoS properties. This is a rich ontology that is also connected to OWL-S. However, it lacks information on how QoS constraints are specified and it is not publicly available. In addition, it is not supported by a WS discovery framework.

OWL-Q [137] is an upper ontology carefully designed based on specific requirements [138]. It is composed on many facets, each capturing a particular aspect of QoS-based WS description. It is also directly connected to OWL-S and is supported by a QoS-based WS discovery framework. This ontology is publicly available in: `http://www.csd.uoc.gr/~kritikos/OWL-Q.zip`. A mid-level ontology has also been designed for domain-dependent QoS metrics. In addition, this ontology is now enriched [140] with SWRL rules in order to support unit transformation, statistical metric value derivation, semantic property inference and matching of QoS metrics.

Enriched WS-Agreement [182]

The work in [182] semantically enriches the WS-Agreement language in order to develop a semantic framework for matching agreement specifications of providers and requester automatically. The WS-Agreement language is extended in important areas such as the *SLO* and *QualifyingCondition* with the addition of the expression, predicate, parameter, and value tags as defined in the WSLA specification [131]. In addition, four ontologies are used to provide a commonality of terms between agreement parties and to provide rich domain knowledge to the search engine so that it may achieve the best possible match results: 1) an OWL ontology for representing the WS-Agreement schema; 2) a OWL-based QoS ontology encompassing QoS concepts used in guarantees; 3) a third OWL ontology representing domain knowledge; 4) the OWL Time ontology [114] for representing temporal constructs such as *endTime*, *interval*, *dayOfWeek*, and *seconds*. Moreover, this approach uses SWRL rules in order to: a) transform one *SLO* to another one that is semantically similar but syntactically heterogeneous with respect to the first one; b) to compare *SLO*s according to the semantics of a domain specific predicate; c) to derive new domain knowledge by e.g. producing a new *SLO* from one or more other *SLOs*; d) to enable user assertions over subjective personal preferences. Last but not least, it must be stated that this extended language is connected to WSDL-S (`www.w3.org/Submission/WSDL-S/`) and is supported by a complete semantic QoS-based WS discovery framework. This work has the following deficiencies: a) QoS metrics are not modeled at all; b) *SLO*s of guarantees are expressed in terms on unary constraints (i.e., containing just one QoS concept); c) Although timing conditions are expressed in *guarantee*s, this does not happen with the whole *alternative*.

WSMO-QoS [271]

WSMO-QoS [271] is an upper level ontology complementary to the WSMO (`www.wsmo.org`) semantic language for functional WS description. It is also directly connected to WSMO. Besides this upper-level ontology, a vocabulary of general domain-independent QoS attributes has also been developed. WSMO-QoS is a very rich ontology capturing many aspects of QoS metric description. It includes and allows many metric value types (linguistic, numeric, boolean), dynamic calculation of metrics values, the attachment of units to metrics, unit transformation functions, the expression of the tendency of the metric's value from the user's perspective and grouping of QoS attributes. This ontology is supported by a QoS-aware selection framework of WSs. The main deficiencies of this ontology are the following: a)

there is a one-to-one mapping of QoS attributes and metrics, which is incorrect; b) no measurement modeling (functions and measurement directives of metrics are not expressed); c) only equality constraints on metrics are allowed, which is quite restrictive; d) not publicly available yet.

onQoS-QL [103]

The onQoS-QL [103] ontology is very rich encompassing all appropriate aspects of QoS-based WS description in almost the same way as OWL-Q. It is also supported by a semantic QoS-based WS discovery framework. Its main highlights are: a) the use of scales for restricting the metric value types; b) the use of unary and binary predicates for constructing metric constraints; c) metric constraints have both retrieval and ranking semantics; d) many important types of measurement processes are supported. The main drawbacks of this work are: a) no connection to a functional WS description language; b) measurement process is external and not specifically defined with the use of metric functions and measurement directives; c) only unary and binary metric comparison predicates are used for expressing QoS constraints.

Mapping requirements from higher layers onto the underlying network [246]

The research effort described in [246] analyzes what must be enclosed into the QoS information for a WS request or advertisement with the help of a QoS ontology. Important elements of this ontology are *QoSInfo* and *QoSDefinition*. *QoSInfo* describes standard or user-defined *serverQoSMetrics* and *transportQoSPriorities* and the values they will take. It also references protocols used by a WS for security and transaction support. The *QoSDefinition* element describes QoS information (*QoSInfo*) either for the whole service or for every operation of the service. Additionally, it includes information about protocols supporting service management and QoS monitoring and about the trusted third-parties that will participate in these protocols. It ends with the price for the usage of this service supporting the QoS offer. One WS advertisement is related to many service offers (*QoSDefinition*) while one service request enquires one particular service offer. One important feature of this research effort is that it supports the mapping of QoS requirements from higher layers onto the underlying network in terms of the Internet model. This mapping is achieved by the help of proxies (residing at the provider and consumer) and by the existence of a QoS-aware network. QoS network parameters are given as guidelines to QoS-aware routers while the client proxy calculates the network performance by taking into account the server performance information provided by the server proxy.

This research work comes with three main deficiencies. First of all, there is not a complete and accurate description of QoS constraints as QoS constraints are just equality constraints. Secondly, metrics ontologies are not developed, but are just referenced. Finally, this work is not backed up by WS discovery framework.

2.4 Observations

An executive summary of the characteristics of the QoS models we have reviewed in this section can be found in the Table 2.3.

Table 2.4 gives a summary of QoS languages and their characteristics, following the classification presented in [139].

Finally, table 2.5 summarizes and compares the main semantic approaches for QoS-based WS description according to four dimensions:

1. Expressivity: Is the QoS (metric) model rich enough?
2. Connection: Does the approach connect to a WS functional description language?
3. Framework: Is there a QoS-based WS discovery framework supporting the semantic approach?

Proposal	Summary
WSQM	Relates several quality factors from different aspects, including additional value brought by web services to the organization. No formal description. Composition of factors not defined.
HP SLA	Defines a framework to which makes it possible specify what and when should be measured in the service compositions and how the measurements should be composed together to give a numeric overall quality.
WSLA	This is a flexible framework to express QoS models which is rich enough to give syntax to precisely describe what basic QoS measures are and how to compose them.
Ws-Policy	It can express policies (QoS classes, where each class is defined by a series of indicators). Operations between policies (to define new policies) are defined.

Table 2.3: Summary of characteristics of QoS models.

Proposal	OMG	WSOL	WSLA	WS-Agreement
Formal	—	√	√	√
Extensible	—	√	√	×
Standards compliant	√	√	√	√
Separate QoS language	√	√	√	√
QoS refinement	√	√	—	—
Provider / requester QoS	√	×	—	√
Class of services	√	√	×	×

Table 2.4: Summary of characteristics of QoS expression languages

4. Agreement: Does the approach describe WS agreements and SLAs?

Table 2.5: Comparison of Semantic QoS-based WS Description Approaches

Approaches	Expressivity	Connection	Framework	Agreement
[248]	Medium	–	No	No
OWL-S	Low	OWL-S	No	No
DAML-QoS	Medium	OWL-S	Yes	No
[164]	Medium	–	No	No
[264]	Medium	OWL-S	No	No
OWL-Q	High	OWL-S	Yes	No
[182]	Medium	WSDL-S	Yes	Yes
WSMO-QoS	Medium	WSMO	Yes	No
onQoS-QL	High	–	Yes	No
[246]	Low	No	No	Medium

Chapter 3

QoS Negotiation in Service-based Applications

Relevant research concerning automated negotiation of services' QoS is reviewed in this chapter. We tailor the discussion of QoS negotiation in SBAs around (i) application level quality negotiation, (ii) QoS negotiation in grid services, and (iii) mechanisms for trust and security negotiation.

In current practice the QoS of a Web service is usually statically defined. Policies or, more generally, quality documents are attached to Web services at publication time. Such documents are then retrieved and analyzed once a service is requested. In Web service selection, quality documents are matched against the quality requirements expressed by applications or users requesting a service, whereas, in service composition, information about quality of a Web service can be used to make a decision on whether to consider or not a Web service for executing a process task. Such an evaluation is made on the ability of a service offer to satisfy local and global quality constraints that users can express on tasks or the whole process, respectively.

In theory, the above mentioned approach represents a *take-it-or-leave-it* or *one-shot* negotiation of Web service QoS. In other words, the service requestor is forced to accept the QoS offer made by the provider and there is no opportunity for setting the QoS profile of a service at runtime, on the basis of providers and requestors preferences, costs model, or willingness to pay. However, Web services are usually provided in a loosely coupled and highly variable environment. On the one hand, Web services can be offered with multiple QoS profiles and, on the other hand, the offered QoS may vary according to variable conditions, such as network load or the number of requests served in a given time instant. The interests of both service providers and requestors on the QoS of a Web service may also be different. For instance, the costs sustained by a Web service provider to increase the availability of its services may not be balanced by a comparable increase of the service requestors' benefits in obtaining a service with higher availability. The on-the-fly setting of QoS profiles on the basis of the service oriented architecture users' needs can be instantiated through the adoption of automated negotiation frameworks in the context of SBAs.

This section reviews relevant literature in the field of QoS negotiation in SBAs. Specifically, Section 3.1 revises research on application level Web service QoS negotiation, classifying contributions on the basis of the QoS model adopted for negotiation, the supported negotiation protocols, and the chosen architectural style adopted for the implementation of the negotiation infrastructure. Adopting similar classification criteria, Section 3.2 discusses the issue of service QoS negotiation in Grid computing, where negotiation is mostly employed as an admission control mechanism for resource reservation. In a loosely coupled environment, service requestors and providers are not likely to know each other advance, since business relationships can be flexibly implemented on-the-fly. This is why we consider the establishment of trust as a paramount issue in SBAs. Finally, Relevant work in the field of trust and security negotiation is discussed in Section 3.3. In Chapter 5 potential future research streams that flow from our review of academic research on Web service QoS negotiation are summarized.

3.1 QoS Negotiation in Web Services and Semantic Web Services

The objective of this section is to revise relevant literature on QoS negotiation in the Web service context. In particular, the literature on Web service QoS negotiation is constituted by several isolated contributions. The heterogeneity of contributions in this field can be ascribed to different motives. First, negotiation is not a *native* research issue of Web services, since it has been studied since 50 years by microeconomics and, more recently, by the literature on multiagent systems. Moreover, QoS negotiation represents a tool for improving the management aspects of Web service-Based architectures. In particular, typical issues in loosely coupled environments management, such as service discovery and selection, composition, and monitoring, raise different issues concerned with negotiation of QoS. Finally, QoS negotiation can be implemented according to different paradigms, such as broker-based architectures and multiagent systems. Each implementation paradigm introduces specific issues that must be dealt with while tackling the Web service QoS negotiation problem.

In order to provide a common background for classifying research on Web service QoS negotiation, we start from understanding the nature of the negotiation problem in the Web service context. As underlined in the previous sections, QoS of a Web service is usually defined by multiple attributes, which span from performance related to domain specific QoS dimensions. Hence, the negotiation of Web service QoS can be usually intended as a multiattribute negotiation problem [120]. For what concerns negotiation participants, negotiation can be either one-to-one or multiparty. Specifically, the participants involved in Web service QoS negotiation are the service requestor, who requires a service with a certain level of quality, and one or more service providers, which provide services with variable quality.

More specifically, our literature classification is grounded on the three basic elements that define an automated negotiation problem [120, 82], that is:

- **Negotiation Object.** It defines the features of the *object* that is under negotiation. While, for instance, in a planning problem agents may negotiate on which actions need to be taken in the future, in the Web service context the object of negotiation is always QoS. Therefore, as already underlined, negotiation is always multiattribute because of the multiattribute nature of Web service QoS;
- **Negotiation Protocol.** It is constituted by the set of rules that define what is allowed in a negotiation and how negotiation participants can exchange messages;
- **Negotiation Strategy.** It defines the decision models of negotiation participants. A decision model defines how negotiators generate new offers, when they accept offers, or when to withdraw from negotiation.

Starting from the above mentioned framework, we propose three classifications of research contributions in the field of Web service QoS negotiation research. First, we classify research contributions according to the features of the quality description on which negotiation is performed. Secondly, we focus on the negotiatio protocols that can be instantiated by the proposed solutions. Finally, we classify contributions according to the architectural paradigm chosen for their implementation. A summary of the classification made in this section is reported in Table 3.3.

The first classification is based on the nature of the negotiation object, i.e., how to define the QoS dimensions on which negotiation is performed. In particular, we make a distinction between approaches to negotiation that rely on a *fixed* set of QoS dimensions and other approaches that adopt *extensible* ways to define QoS. When QoS dimensions are fixed, their number, types, and metrics cannot be modified according to the domain in which QoS negotiation occurs. QoS dimensions, in this case, are usually performance related, since they are usually independent from the application domain. Conversely, extensible QoS usually rely on the definition of a model, either declarative or ontological, which can be used to define and use ad-hoc QoS dimensions.

Table 3.1: Fixed vs. extensible negotiable QoS definition: advantages and drawbacks.

	Fixed set of QoS dimensions	**Extensible QoS model**
Advantages	Real world use cases specification; Rigorous definition of QoS metrics and evaluation methods.	Easiness in including domain specific QoS; Facilitated QoS lifecycle management; In tune with the open world perspective of the SOA.
Drawbacks	Domain specific QoS constrained to the chosen use case; No QoS lifecycle management.	Lack of real world use cases.

The research in [52, 279], for instance, presents an architecture for executing QoS negotiations between service requestors and providers in an Internet applications sourcing scenario. The negotiation considers a fixed set of QoS dimensions, i.e., the throughput, the availability, and the latency of service provisioning.

Declarative extensible QoS models are considered in [57, 72, 106]. QoS dimensions can be defined in terms of name, metric, unit of measure, and evaluation method. An ontological model for QoS description is proposed in [142], where a QoS ontology is required to validate the content of policies that define Web service SLAs.

Most of the approaches presented in the literature rely on extensible QoS models, that can be ether declarative or ontological. In particular, declarative QoS models [106, 172] include QoS definitions into policies that are usually attached to published services. Ontological models [142] organize the elements that define a negotiable QoS dimension, such as name, metric, and monitoring methods, in an ontology, usually expressed in OWL-S.

It has to be noticed that the extensible QoS models reviewed so far do not allow the dynamic definition of QoS dimensions while negotiation is executing. In other words, relevant QoS dimensions for a given domain should be defined priori to the negotiation architecture deployment, since it is not possible for negotiating participants to on-the-fly define new QoS dimensions.

Table 3.1 summarizes the principal benefits and drawbacks of the two alternatives concerning QoS models. The main benefit of considering fixed QoS sets is the opportunity to adopt real world use cases with a rigorous definition of relevant QoS characteristics. Conversely, extensible QoS models trade off the adoption of real world use cases in favor of flexible and generalizable QoS descriptions that are easier to be managed, i.e., updated, modified, or revised, over time.

For what concerns the negotiation protocol, we discern approaches that support 1:1 negotiation, between the service requestor and a single service provider, or 1:N negotiation, between the service requestor and N service providers.

On the one hand, the case of 1:1 negotiation applies to the case of automated SLA establishment [57, 106]. In this case, the service requestor has already chosen a service but, since such service can be provided with variable quality, the QoS of the Web service can be automatically negotiated at runtime.

On the other hand, 1: N negotiation applies to the issues of Web service discovery and selection [99], when the service requestor must choose among a set of functionally equivalent Web services that can be distinguished only by their variable QoS profile. A utility-based approach to QoS-based Web service selection is proposed in [170]. In particular, although no actual negotiation algorithms are provided, the authors propose a service selection method which maximizes the utility of the Web service consumer while guaranteeing costs constraints.

In case of service compositions, current solutions for Web service QoS automated negotiation rely on the coordination of a set of bilateral negotiations between the service requestor and the services involved in the composition [72, 52]. In particular, [52] proposes two different methods for dealing with such a coordination. The *negotiate-all-then-enact* approach involves a round of bilateral negotiation before

enacting the service compositions. The objective is for the service requestor to obtain agreements on QoS that satisfy his or her global QoS requirements. The second approach is the *step-by-step-negotiate-and-enact*, in which the QoS of a service is negotiated right before its individual enactment. This second case increases the complexity of obtaining QoS agreements which satisfy the service requestor global QoS requirements.

Semantic-based negotiation mechanisms and protocols have been often inspired by the agent community literature. In [141] a survey on approaches for multi-attribute negotiation in Artificial Intelligence is introduced. In this field, the goal is to design appropriate models with automated and tractable negotiation mechanisms such that autonomous agents can deal with. For instance, Faratin et al. [83] presents a formal account of a negotiating agent's reasoning component to generate the initial offer, to evaluate the incoming proposal, and to generate the counter proposal. About the protocols, the FIPA Communicative Act Library Specification [88] is considered as a foundational approach for defining specialized negotiation protocols. An example of architecture based on this specification is discussed by Chhetri et al. [53].

Focusing on the semantic Web community, In [54] Chiu et al. discuss how ontology can be helpful for supporting the negotiation. In particular, the authors highlight how shared and agreed ontologies provide common definitions of the terms to be used in the subsequent negotiation process. In addition, they propose a methodology to enhance the completeness of issues in requirements elicitation. Lamparter et al. [143] introduce a model for specifying policies for automatic negotiation of Web services. In this case, the starting point is the upper ontology DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering) [162]. On this basis, this work proposed a policy ontology that also includes the user preferences specifications and an additional ontology for expressing the contracts.

About the use of ontology for specifying the agreement among parties, Oldham et al. [183] present reasoning methods for the components of an agreement which must be compatible for quality matches. This approach is based on WS-Agreement and takes advantage of Semantic Web methods to achieve rich and accurate matches. With the same goal Uszok et al. [265] have developed KAoS policy ontology that allows for the specification, management, conflict resolution, and enforcement of policies within the specific contexts established by complex organizational structures.

We argue that providing a review of negotiation strategies proposed in the literature is out of scope in this paper, since the analysis of negotiation strategies belongs to different fields of inquiry, such as agent-based computing and microeconomics. Moreover, we want to stress that the reviewed approaches to Web service QoS negotiation are not innovative from the point of view of negotiation strategies, since they rely on well known families of strategies, such as concession-based [57], learning-based [52], or search-based strategies [72].

We end our discussion with a classification based on the architectural style adopted for the implementation of the architecture that supports the design and the execution of negotiations. In particular, we have identified two main approaches for implementing QoS negotiations, i.e., *broker-Based* and *multiagent* architectures.

Broker-based architectures imply the existence of a third party, i.e., a broker, which executes QoS negotiation on behalf of service requestors and providers [57, 170]. The negotiation participants strategy is made known to the broker by means of policies. Once having read the policies, the broker is able to simulate the whole negotiation process. The need for including a broker to execute the negotiation has been initially introduced in [157]. The architecture of a broker for supporting Web service QoS negotiation is described in [57]. On the requestor side, negotiation can be either automated, by means of policies, or manually executed by human actors. [106] and [172] propose frameworks for QoS negotiation based on policy matching. Although the description of an architecture is out of scope in the policy framework definition, the authors hypothesize the existence of a broker which execute QoS negotiation and policy matching.

Multiagent architectures exploit Multi-Agent Systems (MAS) to execute Web service QoS negotiations. Specifically, negotiation is executed by agents that negotiate on behalf of external actors, that is,

Table 3.2: Broker-based vs. multiagent QoS negotiation architectures: advantages and drawbacks.

	Broker-based	**Multiagent**
Advantages	Reduced negotiation communication overhead; No need to create new agents/services for negotiating.	Customizable negotiation strategies embedded in agents; Increased flexibility in implementing complex negotiation protocols; Strategies not disclosed to third parties.
Drawbacks	Trust and security (information disclosure to third party); Need for complex policy model definitions; Lack of scalability (the broker may become a bottleneck).	Communication overhead (message exchange among agents); Need to integrate agent-based platforms into Web service architectures; Often a coordinator of negotiating agents is required; Effort required for building and managing negotiating agents.

web service providers and requestors. The underlying multiagent system is usually built according to the FIPA specification [87]. Negotiation agreements are then translated into XML documents which can be easily managed by the Web service architecture.

[52, 279] propose a multiagent-Based negotiation system for coordinated negotiations in service compositions. A coordinator manages the message exchange among service providers and requestors' negotiating agents built from service offers and requirements, respectively. A multiagent negotiation system that supports multiple protocols is proposed in [72]. A rule-based system implements different negotiation protocols, such as auctions and 1:1 negotiations, while message exchange is managed with an event-based approach. Negotiating agents implement strategies for generating or accepting offer and withdrawing from negotiation on the behalf of their owners, i.e., service requestors and providers. Finally, the Collaborative Agreement for Automatic Trading (CAAT) is a multiagent negotiation system built on top of the FIPA specification for managing negotiations in Web service choreography [178]. CAAT is described in terms of the agents' interaction protocol, which is based on an ontology of concepts, and supports bilateral negotiation and moderated negotiations, i.e., bilateral negotiation moderated by an external third party.

Table 3.2 summarizes the principal benefits and drawbacks of the two alternatives for Web service QoS negotiation implementation. The main benefit of broker-based solutions is to reduce the negotiation communication overhead, since negotiation strategies and offers can be automatically generated by the broker. At the same time, the need to declare strategies, e.g., behavioral and pricing models, to a third party introduces of privacy and security that will be further analyzed in the sections to come. Agent-based solutions have higher communication overhead and need to rely on an underlying multiagent platform in order to be implemented. The main advantage of agent-based solutions is the customizability and flexibility, since service requestors and providers can create ad hoc agents that reflect their personal negotiation strategies.

3.2 Negotiation Protocols in Grid Computing

The shifting emphasis of the Grid towards a service-oriented paradigm, as well as trends in application service delivery to move away from tightly coupled systems towards structures of loosely coupled, dynamically bound systems has led to the adoption of Service Level Agreements as a standard concept by which work on the Grid can be allocated to resources and enable coordinated resource management. In the context of Grid and Web services, the current understanding of the community is that such an SLA is essentially an electronic contract, which is expected to be negotiated almost fully automatically by different processes and, as such, much be machine readable and understandable. As a result, there has

Table 3.3: Comparison of approaches to Web service QoS negotiation.

	QoS Model		Neg. protocols			Architecture	
	Fixed QoS	Ext. QoS	1:1	1:N	others	Broker-based	Agent-Based
Chhetri et al. [52]	✓				✓ (WS composition)		✓
Yan et al. [279]	✓				✓ (WS composition)		✓
Comuzzi and Pernici [57]		✓	✓			✓	
Di Nitto et al. [72]		✓	✓	✓			✓
Gimpel et al. [72]		✓	Not specified			✓ policy matching	
Lamparter et al. [142]		✓ ontology	Not specified			✓ policy matching	
Mukhi and Plebani [172]		✓	Not specified			✓ policy matching	
Menasce and Dubi [170]		✓		✓ WS selection and composition		✓	

been a significant amount of research, in recent years, on various topics related to SLAs.

Service Negotiation and Acquisition Protocol (SNAP) [61]

The first promising negotiation protocol in the field of Grid Computing was the Service Negotiation and Acquisition Protocol (SNAP) [61]. The negotiation *objects* are tasks (QoS user requests) and resources (its characteristics). This work proposes three different types of SLAs:

1. *Task service level agreements* (TSLAs) in which one negotiates for the performance of an activity or task. It characterizes a task in terms of its service steps and resource requirements.

2. *Resource service level agreements* (RSLAs) in which one negotiates for the right to consume a resource. An RSLA can be negotiated without specifying for what activity the resource will be used. These two SLAs can be regarded as negotiation *protocols*.

3. *Binding service level agreements* (BSLAs) in which one negotiates for the application of a resource to a task. The BSLA associates a task, defined either by its TSLA or some other unique identifier, with the RSLA.

By combining these agreements in different ways, a high variety of resource management approaches can be represented including: batch submission, resource brokering, co-allocation and co-scheduling. These variations define the negotiation *strategies*. They have also shown an SLA usage scenario for resource brokering, which they called Community Scheduler Scenario. They defined a community scheduler as an entity that acts as a mediator between the user community and the grid resources: activities are submitted to the scheduler rather than to the end resource, and the activities are scheduled onto resources in such a way as to optimize the communitys use of its resource set. A Grid environment may contain many resources, all presenting an RSLA interface as well as a TSLA interface. Optimizing the use of resources across the community served by the scheduler is only possible if the scheduler has some control over the resources used by the community. Hence the scheduler negotiates capacity guarantees via RSLAs with a pool of underlying resources, and exploits those capabilities via TSLAs and BSLAs. This set of agreements abstracts away the impact of other community schedulers as well as any local workloads, assuming the resource managers enforce SLA guarantees at the resources. Community scheduler services present a TSLA interface to users. Thus a user can submit a task to the scheduler by negotiating a TSLA, and the scheduler then turns to a resource by binding this TSLA against one of the existing RSLAs. The scheduler may also offer an RSLA interface. This would allow applications to co-scheduler activities across communities, or combine scheduled resources with additional non-community resources. The various SLAs offered by the community scheduler result in a very flexible resource management environment. Users in this environment can interact with community and resource-level schedulers as appropriate for their goals and privileges.

In the area of Grid Computing SLA usage is the most important in the field of Resource Management. The following use cases gathered in a technical report [227] describe the need for agreements:

- Agreement on resource usage: For a presentation with live demonstration of an application the necessary compute resources to run the application have to be available at the time of the presentation. In a normal cluster environment where the nodes of the cluster are used under a space-sharing policy, the probability of finding a free slot that matches the requirements of a user immediately is low, thus his job usually will be queued and executed later. In order to have the resources required at the time of the presentation the presenter needs to reserve the resources in advance to be sure that they can be used for the demonstration at the time foreseen. This reservation can be expressed as a Quality of Service and an SLA may be issued where the reservation is fixed.

- Agreement on multiple QoS parameters: In an environment consisting of several clusters operated in different administrative domains SLAs might be used for co-allocation or the resource allocation

for workflows. A typical use-case is the co-allocation of multiple compute resources together with the network links between these resources with a dedicated QoS to run a distributed parallel application. The user specifies his request and resource broker starts the negotiation with the local scheduling systems of the compute resources and with the network resource management system in order to find a suitable time-slot, where all required resources are available at the same time. Once a common time-slot is identified the broker requires the reservation of the individual resources. This reservation can also be expressed as a Quality of Service and an SLA may be issued where the reservation is fixed. Another use-case is a workflow spanning across several resources. The only difference to the use-case described before is the kind of temporal dependencies: While for the distributed parallel application the resources must be reserved for the same time, for the workflow use-case the resources are needed in a given sequence. Thus a scheduler needs to negotiate the reservations such that one workflow component can be executed on the required resource after the preceding component is completed.

- Grid Scheduler interoperation: As there is no single orchestrating service or grid scheduler in a grid spanning across countries and administrative domains yet, so we have to deal with multiple instances of independent grid schedulers. Using resources from different domains requires co-ordination across multiple sites. There are two approaches either directly trying to negotiate with respective local scheduling systems or negotiation with the respective local broker. The former solution requires local policies allowing a remote broker to negotiate with local schedulers, which is in general not the case. In the second case there is one access point to the local resources, which then negotiates on behalf of the initiation broker. As the second approach also has a better scalability than the first one the OGF Grid Scheduling Architecture Research Group (GSA-RG) [3] decided to consider this approach for the definition of a Grid Scheduling Architecture. For the communication between the different grid schedulers a language and a protocol to create SLAs was selected to achieve the necessary interoperability while at the same time resulting in SLAs at the end of the negotiation process that can be combined by the initiating scheduler into one single agreement with his client.

The work presented in [222] provides a deeper investigation of SLA usage for job scheduling in Grids. Jobs, submitted for execution to high-performance computing resources, are associated with an SLA. This SLA is negotiated between a client (e.g., a user or a resource broker) and a provider (the owner of a resource with its own local scheduler) and contains information about the level of service agreed between the two parties, such as acceptable job start and end times. Users negotiate and agree an SLA with a resource broker. Brokers negotiate and agree an SLA with users; these SLAs may be mapped to one or more SLAs, which are negotiated and agreed with local resources and their schedulers. Finally, local schedulers need to schedule the work that is associated with an SLA which they agreed to (the constraints associated with such an SLA, agreed by a resource, may be stored locally in the resource, in some kind of a resource record). It is also noted that a single SLA agreed between a user and a broker may translate to multiple SLAs between the broker and different local resources to serve the user's request (for example, this could be the case when the SLA between a user and a broker refers to a workflow application with several tasks that are executed on different resources. In such case, the user may want to set constraints for the workflow as a whole and the broker may have to translate it to specific SLAs for individual tasks, to indicate the possible differences between these two types of SLA, the terms meta-SLA and sub-SLA are used. Furthermore, this SLA-based view for job submission, may still allow the submission of jobs that are not associated with an SLA; however, no guarantees about their completion time would be offered in this case.

To introduce SLA usage to job scheduling the following challenges need to be solved:

- SLA vocabulary: The vision of SLA based scheduling assumes that the SLAs themselves are machine readable and understandable. This implies that any agreements, between the parties con-

cerned, for a particular level of service need to be expressed in a commonly understood (and legally binding) language.

- SLA negotiation: It is envisaged that SLAs may be negotiated between machines and users or only between machines. In this negotiation some commonly agreed protocol needs to be followed. This protocol needs to take into account both the nature of the distributed systems and networks which are used for the negotiation (for example, what if an offer from one party is not received by the other party due to a network failure), and should abide by appropriate legal requirements. In addition, during negotiation, machines should be able to reason about whether an offer is acceptable and possibly they should be able to make counter-offers.

- Scheduling: Given that, currently, scheduling of jobs on high-performance compute resources is mostly based on priority queues (with the possible addition of backfilling techniques), the use of SLAs would require the development of a new set of algorithms for efficient scheduling, which would be based on satisfying the terms agreed in the SLA. The existence of efficient scheduling algorithms would be of paramount importance to estimate capacity and reason on the possible acceptance (by a local resource) of a new request to make an SLA.

- Constitutional Aspects: In the context of any SLA based provision, sooner or later, the need for dispute resolution may arise. In addition, users may also be interested in the reliability of specific brokers; for example, how likely is that a broker will honour an SLA (even if breaking the SLA would require the broker to pay a penalty). This issue of modeling reputation may also be related to the approaches followed for pricing and/or penalties when agreeing SLAs.

These requirements and the previously introduced use cases reveal the relevant properties of SLA usage in Grids. The negotiation *objects* are the static resource characteristics (eg. CPU, memory, disk), the resource capabilities (eg. licenses, reservation, price, software) and the user requirement properties (eg. time, cost constraints). The negotiation *protocol* and *implementation* are depend on the actual solution, we previously mentioned the WS-Agreement and the WSLA. Both are defined by XML Schemas. The negotiation *strategies* usually depend on the actual protocol, but this is the point, where solutions using the same protocols may differ. SLA-based grid resource management relies heavily on the renegotiation of the agreement (which can also regarded as a negotiation strategy). It generally means reconsideration of the quality and the level of service, such as processor, memory or bandwidth requirements, resource reservations, and so on. Renegotiation requires user involvement during the job execution. Therefore the most important issue is to reduce the amount of user interaction. In [221] researchers provided an interesting solution to achieve this goal within the WS-Agreement framework. The basic idea is to regard guarantee terms as functions to increase the flexibility of the agreement. They introduced a list of universal variables (ie. current wall clock time, network bandwidth) and a list of predefined functions (ie. min/max bound, list average, Gaussian function). As a result the Service Level Indicatiors/Objectives in the SLAs are described not as constants but as functions of universal variables. In this way the terms of the WS-Agreement are no longer constants or independent range-values. Such an agreement has an infinitely large number of outcomes, therefore it is able to describe the entire guarantee terms of an SLA. This way of SLA usage provide a richer term set for grid applications and reduce the need for renegotiation.

QoWL [43]

[43] presents an approach for high level Grid workflow specification that considers a comprehensive set of QoS requirements. Besides performance related QoS, it includes economical, legal and security aspects. For instance, for security or legal reasons the user may express the location affinity regarding Grid resources on which certain workflow tasks may be executed. the QoS-aware workflow system provides support for the whole workflow life cycle from specification to execution. Workflow is specified

graphically, in an intuitive manner, based on a standard visual modeling language. A set of QoS-aware service-oriented components is provided for workflow planning to support automatic constraint-based service negotiation and workflow optimization. For reducing the complexity of workflow planning, a QoS-aware workflow reduction technique is introduced.

Amadeus [44]

In [44] the Amadeus framework is presented, which is a holistic service-oriented environment for QoS-aware Grid workflows. Amadeus considers user requirements, in terms of QoS constraints, during workflow specification, planning, and execution. Within the Amadeus environment workflows and the associated QoS constraints are specified at a high level using an intuitive graphical notation. A set of QoS-aware service-oriented components is provided for workflow planning to support automatic constraint-based service negotiation and workflow optimization. Amadeus framework uses static and dynamic planning strategies for workflow execution in accordance with user-specified requirements.

End-to-end QoS for compute-intensive medical simulation services [31]

In [31] a Grid infrastructure is presented addressing the issue of offering end-to-end QoS in the form of explicit timeliness guarantees for compute-intensive medical simulation services. Within the GEMSS project [1], parallel applications installed on clusters or other HPC hardware may be exposed as QoS-aware Grid services for which clients may dynamically negotiate QoS constraints with respect to response time and price using Service Level Agreements. The used infrastructure infrastructure is based on standard Web services technology and relies on a reservation based approach to QoS coupled with application specific performance models.

3.3 QoS Negotiation in Security

In this section, two specific aspects of QoS negotiation in security are discussed: (1) trust negotiation and (2) privacy negotiation.

Features of trust negotiation system

The goal of trust negotiation is to enable strangers to access sensitive data in open environments without violating the data's access policy. In trust negotiation, two parties establish trust through iterative disclosure of credentials and requests for credentials to verify properties of the negotiating parties. To carry out a trust negotiation, parties usually uses strategies implemented by an algorithm. Most of the research in this area focuses on protecting resources and credentials and assumes that policies can be freely disclosed. Research in trust negotiation has focused on a number of important issues including languages for expressing resource access policies such as Trust-X, X-TNL, Trustbuilder, Cassandra (e.g., [27, 32, 35, 148]), protocols and strategies for conducting trust negotiations (e.g., [282, 5, 136, 110]), and logics for reasoning about the outcomes of these negotiations (e.g., [197, 275]). The foundational results presented in these papers have also been shown to be viable access control solutions for real world systems through a series of implementations (such as those presented in [34, 5, 274]) which demonstrate the utility and practicality of these theoretical advances.

Based on the current work, we can state, a trust negotiation system may include four components:

- *Trust negotiation policy module*
- *Credential manager module*
- *Strategies negotiation module*
- *Trust negotiation policy management module*

Let us define now the requirements needed for each components. We propose some dimensions to evaluate the current relevant negotiation models. However, those dimensions are not limited and can be augmented due to the increasing thought in this area.

Trust negotiation policy language requirements
A trust negotiation policy language is a set of syntactic constructs including credentials and policies and how they encode access control needs [34, 33]. Here we discuss the characteristics that may have policy language.

Policy specification
Trust negotiation policies specify which credentials and other resources to disclose at a given state of the trust negotiation, and the conditions to disclose them. The specification can be formal, using logic, algebra, etc., or semi-formal.

Specification of the combination The language may be expressive enough to contain operators combining credentials characterizing a given subject.

Authentication of multiple identities
Under this approach to automate trust establishment, each party can have many identities, each corresponding to identify that a particular credential issuer uses to designate that individual. For instance, my purse today includes my driver's license number, my credit card number, my library card number etc. I may be asked to prove that I possess several of these identities during runtime (trust establishment).

Sensitive Policy protection
The protection can be handled at the policy level or system level in the sense that by analyzing policies' content, outsiders might infer sensitive information about parties.

Trust negotiation policy lifecycle management
The characteristics that must be considered, is the trust negotiation policy lifecycle management. It might include how to update trust negotiation policies in a consistent manner and how to cope with dynamic policy evolution, that is, the change to a policy while there are active negotiations based on the policy being modified.

Credential manager requirements

Type of disclosure
The context aware applications will reveal credentials in the correct context.The disclosure needs some actions to be specified that will satisfy conditions.

Automation
The credential exchange can range from automatic (no user intervention is needed), over semi automated (some aspects use tools, and other need user intervention), to manually.

Credential chain discovery
The credentials used in a negotiation can be available locally or discovered at run time.

Ownership
Some systems use various security protocols(outside) during negotiation, when a remote credential is received asking for verification to prove the ownership. Some other systems integrate negotiation framework with the existing tools and systems.

Strategies negotiation requirements
Dynamicity
In some domains such as e-Health, not all entities are willing to negotiate credentials or disclose access policies directly to strangers regardless of negotiation strategies and instead prefer to negotiate and disclose sensitive information only to strangers within what we refer to as a circle of trust. The need is to describe how locally trusted intermediary parties can provide multiple negotiation and delegations hops to protect credentials and access policies [7].

Privacy protection mechanisms
Privacy is one of the major concerns of users when exchanging information through the Web and thus we

believe that trust negotiation systems must effectively address privacy issues in order to be widely applicable. Some research papers investigate privacy in the context of trust negotiations. [94, 38, 239, 186] propose a set of privacy-preserving features for inclusion in any trust negotiation system. [226] proposes the identification of privacy vulnerabilities in trust negotiation.

Automatic

Exchange of attribute credentials is a means to establish mutual trust between strangers wishing to share resources or conduct business transactions. Automate trust negotiation to regulate the exchange of sensitive information during this process is important [118, 273].

Bilateral trust establishment

It is important not only for a service provider to trust the clients requesting its services, but for clients to trust the services that they choose to interact with.

Scalability

How much trust establishment can be automated in a highly scalable manner? Can it be made ubiquitous?

Analyzing existing trust negotiation for web services and grid computing

We surveyed and analyzed several approaches based on the requirements presented earlier.

Web services

Existing efforts in the area of trust negotiation have not been standardized and do not fit into any authorization standard such as the eXtensible Access Control Markup Language (XACML). In [166], investigates how XACML can fit into trust authorization management systems by exploring existing concepts, and where necessary, extending them to accomplish the goal. The authors proposed XACML Trust Authorization Framework (XTAF), which is a loosely coupled architecture with a trust component that protects authorization information (policies and credentials) layered such that it integrates seamlessly into any XACML compliant authorization engine with minimal effort.They authors show how the XACML policy language can be used to support bilateral exchange of policies and credentials, and protect unauthorized access to services. They also introduce a Trust Authorization Service Handler (TASH) to handle trust and privacy of authorization information. This supports runtime bilateral authorization operations between two or more parties.

In [233], authors propose a model-driven approach to trust negotiation in Web services. The framework, called Trust-Serv, features a trust negotiation policy model based on state machines. More importantly, this model is supported by both abstractions and tools that permit life cycle management of trust negotiation policies, an important trait in the dynamic environments that characterize Web services. In particular, they provide a set of change operations to modify policies, and migration strategies that permit ongoing negotiations to be migrated to new policies without being disrupted.

In [230] a WS-ABAC model is proposed to use attributes associated with subject, object and environment, and service parameters for access control measures in Web services environment. WS-ABAC model extends the traditional RBAC model to gain many advantages from its attributes-based capability. In this model, authorization decision is based on the attributes of the related entities. The ATN mechanism is used to provider needed attributes and addresses the disclosure issue of the sensitive attributes when the attribute conditions are not satisfied. So it can protect users privacy. Only when the user does not give needed attributes for authorization decision, the access request is rejected.

Redesigning and restandardizing existing protocols to make authorization decisions to meet the needs of large-scale open systems, is a time consuming To address this problem in [145, 187, 146], authors propose a system called Traust, a stand-alone authorization service that allows for the adoption of trust negotiation in a modular, incremental, and grassroots manner, providing access to a wide range of resources without requiring widespread software or protocol upgrades. It uses current prototype Trust-X or TrustBuilder to allow clients to establish bilateral trust.

In [185], the authors investigate the problem of trust in Semantic Web Services. They include trust policies in WSMO-Standard, together with the information disclosure policies of the requester, using the

Peertrust language [100]. Peertrust provides the means to perform trust negotiation and delegation. As the matchmaker needs to have access to the requester and provider policies, in order to match not only the requester functional requirements but also trust information, the authors proposed a distributed registry and matchmaker architecture that allows the service providers to keep their policies private, thus not forcing them to disclose sensitive information.
In [76], is proposed a security-by contract as a novel negotiation framework where services, needed credentials, and behavioral constraints on the disclosure of privileges are bundled together and that clients and servers have a hierarchy of preferences among the different bundles. While the protocol supports arbitrary negotiation strategies, two concrete strategies (one for the client and one for the service provider) make it possible to successfully complete a negotiation when dealing with a co-operative partner and to resist attacks by malicious agent to "vacuum-clean" the preference policy of the honest participant.
The Web Services Trust Language (WS-Trust) uses the secure messaging mechanisms of WS-Security to define additional primitives and extensions for the issuance, exchange and validation of security tokens. WS-Trust also enables the issuance and dissemination of credentials within different trust domains.

The goal of WS-Trust [174] is to enable applications to construct trusted [SOAP] message exchanges. In order to secure a communication between two parties, the two parties must exchange security credentials (either directly or indirectly). However, each party needs to determine if they can "trust" the asserted credentials of the other party. This specification defines extensions to WS-Security for issuing and exchanging security tokens and ways to establish and access the presence of trust relationships. Using these extensions, applications can engage in secure communication designed to work with the general Web Services framework, including WSDL service descriptions, UDDI businessServices and bindingTemplates, and SOAP messages.

Grid computing

In [147] is proposed a novel trust negotiation framework, TOWER, which integrates distributed trust chain construction of trust management and aims to enhance the grid security infrastructure. The approach leverages attribute-based credentials to support flexible delegation, and dynamically constructs trust chains. A novel TRust chAin based Negotiation Strategy (TRANS) is proposed to establish trust relationship on the fly by gradually disclosing credentials according to various access control policies. It is implemented in CROWN grid. In [116] is presented ROST an original scheme of Remote and hOt Service deployment with Trustworthiness. By dynamically updating runtime environment configurations, ROST avoids restarting the runtime system during deployment. In addition, we include trust negotiation in ROST, which greatly increases the flexibility and security of the CROWN Grid.
In [13] is analyzed several classes of trust and their use in Grids: service provision, resource access, delegation, certification and context trust. Current technologies for managing trust have been also discussed. The concept of Virtual Organizations is central to Grids. The authors have enriched the classical VO lifecycle with trust management activities. Trust values and trust policies are created before starting the VO identification phase. In the VO Identification phase, trust information such as reputation could be taken into account when selecting potential VO candidates. The VO formation phase includes all activities related to trust negotiation. During VO operation, trust values are computed and distributed among the VO participants. In VO dissolution, trust information such credentials and access rights are revoked to avoid misuse of the information.
An efficient method of hidden credentials by reusing randomness in a way that does not compromise the security of the system is proposed in [42]. The number of elliptic curve operations required depends only on the number of credentials relevant to a transaction and is constant over a change in policy size or complexity. A monotonic secret splitting scheme is also proposed, where the relevant shares and the corresponding boolean expression are only revealed as relevant pairs of shares are discovered. Reducing prefix length in our secret splitting scheme increases anonymity but also increases overhead and the probability of a runaway table.
The paper [217] discusses the adaptive trust negotiation and access control (ATNAC) framework. It ad-

dresses the problem of access control in open systems by protecting itself from adversaries who may want to misuse, exhaust or deny service to resources. A federated security context allows Grid participants to communicate their security appraisal and make judgments based on collective wisdom and the level of trust among them.

Comparing the approaches

To better assess the current state of the art, in this section the requirements introduced in section 3.3 is mapped into three tables that provide a higher-level view of the detailed discussions provided in Section 3.3 on the basis of the requirements we listed earlier. All the approaches have their own drawbacks and advantages. None of the proposals are complete, even though current approaches address significant subsets of relevant requirements.

Comparing policy languages

Figure 3.1 describes the comparison of the presented approaches with respect to the policy language requirements. The figure depicts the nature of the policy specification, whether the combination on credentials can be supported by the policy language or not, the existence of multiple authentication, the protection of sensitive data is provided in the policy level and finally the management of policy life cycle is discussed.

The combination of credentials is considered as important and there is a way to express it in the specification language in almost existing existing framework. The authentication of multiple identities is almost missing.

Moreover, life cycle management of policies that is, the creation, evolution, and management of policies is an often overlooked part of policy model design. Policies are rarely set in stone when first defined. Such aspect is totally absent in the current framework except in Trust-Serv.

Furthermore, there are no existing approaches addressing the trust negotiation cross all the layers of the service based systems with respect to a variety of information discussed earlier. The proposed approaches address the trust negotiation in web service.

Approaches	Trust-Serv	Traust	WS-ABAC	XTAF	PeerTrust	WS-Trust
Specification	Formal	Semi	Formal	semi-formal	Semi	Syntactic
Combination	Yes	Yes	No	Yes	Yes	Yes
Multiple-authentication	No	No	No	No	No	Yes
Sensitive-policy-protection	No	Yes	No	Yes	Yes	Yes
Management-evolution	Yes	No	No	NO	No	No

Figure 3.1: Comparing policy specification language

Comparing credential management

Figure 3.2 describes the comparison of the presented approaches with respect to the credential management requirements. The figure depicts the types of disclosure, the automation when tackling the credentials, the discovery of the credential chain, and finally the ownership of the credential.

The table shows that in the trust negotiation research in service, the trend is toward performing actions while disclosing credentials in an automatic way.

The mechanism of the credentials' discovery during the negotiation is more and less seldom. In fact, during run time system should include tool for chain discover to retrieve at run time credentials that are not locally cached.

Most of current systems try to integrate the negotiation framework with the exiting tools and systems in order to maximize the control of the security while the data are exchanged.

No existing systems address how to obtain credentials, assuming that the entity disclosing credentials has its own method to obtain and cache the credentials locally.

Approaches	Trust-Serv	Traust	WS-ABAC	XTAF	PeerTrust	WS-Trust
Type-disclosure	actions	Actions	Actions	context+actions	Actions	Actions
Automation	Auto	Auto	Semi	semi	Auto	Semi
Chain discovery	Yes	No	No	No	Yes	Yes
Ownership	No	Integ(Tokens)	Integ(RBAC)	Integ(XACML)	No	INte(Tokens)

Figure 3.2: Comparing credential manager

Comparing systems and strategies

Figure 3.3 describes the comparison of the presented approaches with respect to the strategy of negotiation requirements. The figure depicts the dynamicity of the negotiation, the automation, the presence of the mechanism of privacy protection during the negotiation, the scalability an finally the existence of bilateral trust establishment.

This table shows that the trend of the strategies is a dynamic and an automatic. The credentials are disclosed during run time not locally but can be found dynamically or by combining credentials.

The current systems are increasingly aware of protecting sensitive data exchanged during the negotiation why they integrate mechanism taking care that. Furthermore, also the bilateral trust establishment is considered in the current systems because it is important to let a service provider and customer trust mutually, not only the provider which means how can the provider trusts the client while purchasing online and providing credit card number.

The scalability is an important criteria in distributed systems as are SBA. However, in the presented approaches, no complexity and no consistency of the credential discovery mechanisms are discussed, which are very important in the distributed systems. Moreover, the existing approaches are only research paradigms, no standardization is pointed out.

Approaches	Trust-Serv	Traust	WS-ABAC	XTAF	PeerTrust	WS-Trust
Dynamicity	Dynamic	Run-time	No	No	Yes	No
Automatic	Auto	Semi		semi-auto	Auto	semi
Privacy protection mechanism	No	Yes	No	Yes	Yes	Yes
scalability	Yes	Yes	Yes	No	Yes	Yes
bilateral	No	Yes	No	No	Yes	Yes

Figure 3.3: Comparing strategies

Privacy negotiation in SBA

Privacy violation is a serious and pressing problem for Internet users that requires immediate solution. Negotiation records are usually confidential and private, and owners may not want to reveal the details of these records. In [284], a privacy preserving negotiation learning scheme is introduced, which incorporates secure multiparty computation techniques into negotiation learning algorithms to allow negotiation parties to securely complete the learning process on a union of distributed data sets.
[79] identifies the risks of privacy invasion during the setup of interactive multimedia applications, and introduce three schemes to solve the problem of protecting the users privacy, with varying degree of complexities. The first approach is based on the use of a trusted third party; this has been a common approach for the Public Switched Telephony Network (PSTN). The second approach is based on the trust relationship between the communicating parties, and the third approach is based on primitives from the field of secure distributed computation.

[200] has presented the necessity of negotiation about privacy principles in a relationship between service provider and customer. Modeling the users individual utility maximization can take into account the multi-dimensionality of privacy; the service provider may wish to reduce the negotiation space in a way that suits the given business scenario. It proposed two new elements that follow the structure of the current P3P 1.1 grouping mechanisms and allow software-supported negotiations in E-Commerce.
E-commerce systems are increasingly concerned with data protection, they follow a property-based approach to privacy which leads to privacy negotiation and bargaining upon the base of the data subjects consent. After considering the technological and regulative strategies of protecting consumer privacy, [184] discusses the shortcomings of that approach and claims that, as long as a general privacy culture has not yet evolved in the (web) society, it might collide with the notion of data protection as a fundamental right.
Due to automation of infrastructure both users and services have many agents acting on their behalf. Respectively in pervasive systems one of the most problematic concerns arises on user right to maintain privacy. [41] is focused on instruments to enable and maintain privacy through a subtle fusion of different privacy enabling techniques. The authors present a conceptual privacy enabling architecture of infrastructural pervasive middleware that uses trust management, privacy negotiation and identity management during the inter-entity communication life cycle.

In order to take into account the privacy concerns of the individuals, organizations (e.g., Web services) provide privacy policies as promises describing how they will handle personal data of the individual. However, privacy policies do not convince potential individuals to disclose their personal data, do not guarantee the protection of personal information, and do not provide how to handle the dynamic environment of the policies. [30], introduces a framework based on an agreement as a solution to these problems. It proposes a *privacy agreement* model that spells out a set of requirements related to requestor's privacy rights in terms of how service provider must handle privacy information. It defins two levels in the agreement: (1) policy level and (2) negotiation level. A formal privacy model is described in the policy level to provide upon it a reasoning mechanism for the evolution. The framework supports a life cycle management in the negotiation level of the agreement, hence, the privacy evolution is handled in this level.

Chapter 4

Quality Assurance for Service-based Applications

4.1 Introduction and Overview

To achieve the desired quality of a service-based application, two complementary kinds of techniques and methods can be employed: *constructive* and *analytical* quality assurance techniques and methods. The goal of constructive quality assurance techniques and methods is to prevent the introduction of faults (or defects) while the artifacts are created. Examples for such techniques include code generation (model-driven development), development guidelines, as well as templates. The goal of analytical quality assurance techniques and methods is to uncover faults in the artifacts after they have been created. Examples for analytical quality assurance techniques are reviews and inspections, formal correctness proofs, testing, as well as monitoring.

This deliverable will cover the state of the art in analytical quality assurance techniques and methods for service-based applications. Techniques and methods for constructive quality assurance will be addressed e.g. by deliverable PO-WP-JRA-1.1 (which focuses on process models and design methods) and by deliverable PO-JRA-2.2.2 (which includes automated and QoS-aware service-compositions).

This part of the deliverable provides a comprehensive survey of the state of the art in analytical quality assurance for service-based applications, for which we will survey, summarize and classify the existing contributions in the literature. Furthermore we will identify gaps and overlaps in state of the art in order to identify open research issues that should be addressed in future research.

The remainder of this part of the deliverable is structured as follows. Section 4.2 introduces fundamental terminology and important classes of quality assurance techniques and methods. In Section 4.3 the classification framework which we have employed to categorize the surveyed papers is described. Section 4.4 summarizes and classifies surveyed contributions from the literature. Section 4.5 consolidates the results from Section 4.4 and – based on the classification framework – identifies gaps and overlaps in the research contributions.

Finally, based on the findings in Section 4.5, Chapter 5 will highlight potential research challenges in analytical quality assurance of service-based applications.

4.2 Fundamentals

In order to structure the survey results, we sub-divide the analytical quality assurance techniques and methods into the three major classes: *testing*, *monitoring* and *static analysis*. These classes have been proposed in the software quality assurance literature (e.g., see [173, 167, 188, 101]) and have been used in a recent overview of quality assurance approaches for service-based applications (cf. [23]).

Figure 4.1 provides an overview of these classes and their sub-classes which are explained in the following sub-sections.

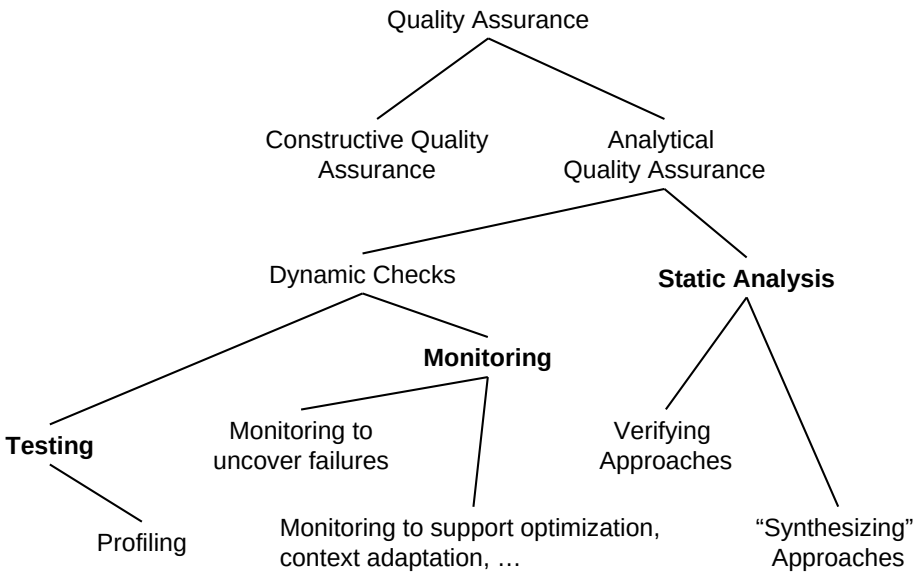

Figure 4.1: Overview of quality assurance approaches

Testing

The goal of testing is to (systematically) *execute* services or service-based applications[1] in order to uncover failures (cf. [173, 167, 188, 101]).

During testing, the service or service-based application which is tested is fed with concrete inputs and the produced outputs are observed. The observed outputs can deviate from the expected outputs with respect to functionality as well as quality of service (e.g., performance or availability). When the observed outputs deviate from the expected outputs, a failure of the service or the service-based application is uncovered.

Failures can be caused by faults (or defects) of the test object. Examples for faults are a wrong exit condition for a loop in the software code that implements a service, or a wrong order of the service invocations in a BPEL specification. Finding such faults typically is not part of the testing activities but is the aim of *debugging* (e.g., cf. [167, 101]).

A special case of testing is profiling. During profiling, a service or a service-based application can be systematically executed in order to determine specific properties. As an example, during profiling the execution times of individual services in a service composition could be measured for 'typical' or 'extreme' inputs in order to identify optimization potentials.

Testing cannot guarantee the absence of faults, because it is infeasible (except for trivial cases) to test all potential concrete inputs of a service or service-based application. As a consequence, a sub-set of all potential inputs has to be determined for testing (e.g., cf. [188]). The quality of the tests strongly depends on how well this sub-set has been chosen. Ideally this sub-set should include concrete inputs that are representative for all potential inputs (even those which are not tested) and it should include inputs that – with high probability – uncover failures. However, in cases where choosing such an ideal sub-set typically is infeasible, it is important to employ other quality assurance techniques and methods which complement testing.

[1] In the remainder of this part of the deliverable, we follow the convention of PO-JRA-1.1 'State of the art report on software engineering design knowledge and Survey of HCI and contextual Knowledge' and use the term service-based application as a synonym for service composition.

Monitoring

Monitoring observes services or service-based applications during their *current* execution, i.e. during their actual use or operation (cf., [68]). In addition, the context of a service or a service-based application can be monitored. This context can include other systems, the execution platform (hardware, operating systems, etc.) and the physical environment (e.g., sensors or actuators).

Monitoring can address different goals (cf. Deliverable PO-JRA-1.2.1 'State-of-the-Art report, gap analysis of knowledge on principles, techniques and methodologies for monitoring and adaptation of SBAs'). As an example, monitoring techniques can be employed to support the optimization of a service-based application during run-time. Further, monitoring can be used to enable the context-driven run-time adaptation of a service-based application. Also, monitoring may be used to uncover failures during the current execution of a service or service-based application.

In contrast to testing and static analysis, which aim at providing more or less general statements about services or service-based applications, monitoring always provides statements about their current execution (i.e., about current execution traces). Thereby, monitoring can uncover failures which have escaped testing, because the concrete input that lead to the current execution trace might have never been tested.[2] Also, monitoring can uncover faults which have escaped static analysis, because static analysis might have abstracted from a concrete aspect which was relevant during the current execution. Monitoring therefore provides a complementary measure to ensure the quality of a service-based application and thus "can be used to provide additional defense against catastrophic failure" [68].

Static Analysis

The aim of static analysis (e.g., see [80, 101]) is to systematically *examine* an artifact in order to determine certain properties or to ascertain whether some predefined properties are met. In the remainder of this deliverable, we refer to the first kind of approaches as *synthesis* approaches and to the latter kind of approaches as *verification* approaches. Analysis can be applied at several stages in the development cycle, and therefore examples for artifacts which can be subject to analysis include requirement documents, design specifications, interface descriptions, and code.

Examples of static analysis include formal techniques and methods, such as data flow analysis, model checking, execution in abstract domains, symbolic execution, type checking, and correctness proofs, which are all usually characterized because they compute properties that are in many cases approximations of the more concrete properties, but which, in this case, are safe, in the sense that the lack of accuracy must not lead to an error for the intended use of the analysis. Informal approaches, such as reviews, walkthroughs, and inspections, are as well examples of static analysis.

In contrast to testing (or monitoring), where individual executions of the services or service-based applications are examined, analysis can examine classes of executions [101]. Thus, analysis can lead to more universal statements about the properties of the artifacts than testing (or monitoring).

In order to achieve these more universal statements, static analysis – unlike testing or monitoring – does not execute the artifacts which are being examined, since termination (which is theoretically ensured when the system has a finite state space) is usually a necessary condition for a successful analysis. However, systems may have a state space so large (or infinite) as to make traversing it unfeasible. In those cases static analysis resorts to working with *safe approximations* of the actual system semantics, which makes the system actually under analysis effectively finite, but different from the initial one.

Those approximations can be very sophisticated and take the form of, e.g., relations between inputs and outputs which approximate the system behavior in the domain of the analysis. When these approximations capture the properties of interest faithfully enough, then the results, even if not as accurate as they could be, are useful – and correct. Yet, as approximations might abstract away from some relevant concrete details, aspects might be overlooked [101] or simply not be captured faithfully enough. Thus

[2] As explained above, only a sub-set of all potential inputs can be tested.

static analysis can complement the other classes of quality assurance techniques and methods but typically will not be enough, if used in isolation, in order to give a complete picture of the whereabouts of the execution of a computational system.

4.3 Classification Framework

The papers which are surveyed in this part of the deliverable are categorized in order to understand common concepts and thus identify potential gaps and overlaps (see Section 4.5). A number of "dimensions" are used for this categorization and constitute our classification framework.

During the process of surveying the papers and based on discussions with other S-Cube work packages, this framework has been continuously evolved in order to cover all relevant dimensions.

The "dimensions" of this classification framework are described in the following sub-sections.

Major Class of Quality Assurance Technique or Method (*Class*)

As has been described in Section 4.2, we will distinguish between three major classes of quality assurance techniques and methods for service-based applications: testing, monitoring and static analysis. For an individual technique it can well be possible that it will be classified to fall into more than one of these major classes.

Quality Characteristics which are Addressed (*Quality*)

Different kinds of quality characteristics – also known as QoS dimensions, quality attributes or quality parameters – can be addressed by the quality assurance techniques and methods.

An important quality characteristics is functional *correctness*, i.e. assuring that the functionality expected from the service or service-based application is met.[3]

Other quality characteristics that are relevant for service-based applications include performance, privacy, security or availability. Chapter 2 of this deliverable discusses those quality characteristics for service-based applications in more detail.

Moment During the Life-Cycle (*Life-Cycle*)

This dimension classifies the techniques and methods according to the moment during the life-cycle at which they can be / should be employed. For the purpose of this deliverable, we distinguish between two major phases in the life-cycle: *design* (before the application is being deployed) and *operation* (while and after the system has been deployed).

More detailed software life-cycle models for service-based applications will be introduced in deliverable PO-JRA-1.1.1 ("State of the art report on software engineering design knowledge and Survey of HCI and contextual Knowledge").

Research Discipline (*Discipline*)

This dimensions states which research disciplines has proposed the discussed solution (this can typically be identified by the affiliation of the authors or the conference / journal in which the paper has been published).

Those disciplines include (but are not limited to) Business Process Management (*BPM*), Service-oriented Computing (*SOC*), Software Engineering (*SE*) and Grid Computing (*Grid*).

[3] Often, a distinction between functional and non-functional (quality) characteristics is made. Following the ISO/IEC 9126 standard, we subsume "functionality" under "quality".

Relevant Layer of the Service-based Application (*Layer*)

A check can involve artifacts on different layers of a service-based application. These layers are Business Process Management (*BPM*), Service Composition and Coordination (*SCC*) and Service Infrastructure (*SI*).

Artifact that is Checked (*Artifact*)

This dimension classifies the technique or method according to the artifact that is checked (i.e., analyzed, tested or monitored).

The entity which is checked, can include – besides others – an atomic service (which does not invoke other services), a composed/aggregated service (or a service-based application), a service registry, or the context of a service or a service-based application. As an example, many monitoring approaches (see Section 4.4.2) observe changes in the context of the system in order to produce monitoring results and to enable the adaptation of the service-based application.

Artifact Against which the Entity is Checked (*Reference*)

In order to check an entity, a reference artifact is needed against which the check is performed. As an example, when performing the test of an atomic service, the service implementation could be tested against the service interface.

Examples for such reference artifacts are service interfaces, service specifications, or Service-level Agreements (SLAs).

Level of Formalization (*Formality*)

The following levels of formalization of a quality assurance technique and method can be distinguished: *formal*, *semi-formal*, or *non-formal* techniques and methods.

The results of a formal techniques or methods are achieved by means of mathematical methods. This requires that the input artifacts to such formal techniques or methods have a formal representation. Examples for formal techniques include model checking, correctness proofs or symbolic execution.

A semi-formal technique or method rests on a language that contain formal as well as informal elements. Thus those artifacts are not completely amenable to mathematical methods.

Examples for non-formal techniques are reviews or inspections. Although the process for inspections is rigorously defined, the steps during inspection do not rely on mathematical precision.

Degree of Automation (*Automation*)

Quality assurance techniques and methods can consist of individual steps. An individual step can be amenable to automation (i.e., it can be executed by a software tool) or it cannot be automated, because it requires a human to perform a creative task.

The degree of automation of a technique or method can thus range from *fully automated*, over *partially automated* to *not automated*.

In a fully automated technique or method no manual steps have to be performed. During the application of a partially automated technique or method, not all the steps are performed by tools and thus some steps require “user” intervention.

Means of Validation (*Validation*)

This dimension aims at classifying how the proposed technique or method has been (empirically) validated.

The following “levels” of empirical evaluation, taken from [283], are used for that purpose:

- *Controlled Experiment:* All of the following exist: Random assignment of treatments to subjects. Large sample size. Hypotheses formulated. Independent variable selected. Random sampling.
- *Quasi Experiment:* One or more of points in Controlled Experiment are missing.
- *Case Study:* All of the following exist: Research question stated. Propositions stated. Unit(s) of analysis stated. Logic linking the data to propositions stated. Criteria for interpreting the findings provided. Performed in a real world situation.
- *Exploratory Case Study:* One or more of points in Case Study are missing.
- *Experience Report:* All of the following exist: Retrospective. No propositions (generally). Does not necessarily answer how or why. Often includes lessons learned.
- *Meta-Analysis:* Study incorporates results from previous similar studies in the analysis.
- *Example Application:* Authors describing an application and provide an example to assist in the description, but the example is "used to validate" or "evaluate" as far as the authors suggest.
- *Survey:* Structured or unstructured questions given to participants.
- *Discussion:* Provided some qualitative, textual, opinion-oriented evaluation. E.g. compare and contrast, oral discussion of advantages and disadvantages.

4.4 Survey Results

This section comprises the results of the literature survey for the three major classes of analytical quality assurance techniques and methods for service-based applications. The results for each of these classes are presented in subsections 4.4.1 - 4.4.3.

Each subsections provides an overview of the survey procedure which has been followed.

The contributions which have been identified as being relevant are summarized according to the following "template" (which has been inspired in parts by [235]):

- *Problem*: The problem which is addressed by the paper.
- *Explanation*: The reasons for the existence of the problem (i.e., "why is the problem a problem?").
- *Solution*: The solution to the problem as presented in the paper.
- *Benefits*: The claimed or demonstrated benefits of the solution when applied to the problem.
- *Validation*: Description on how the solution and its benefits have been validated.

A categorization of the paper with respect to the dimensions of the classification framework will be presented (in overview form) in Section 4.5.

4.4.1 Testing

A systematic literature review (cf. [133]) has been performed to gather the relevant work on testing service-based applications.

The research question which was addressed by this study was: "What is the state-of-the-art in testing of service-based applications?" Subquestions, which we kept in mind when discovering the field of testing service-based applications were:

- Which techniques, mechanisms and methodologies are used for testing service-based applications?

- Which challenges or opportunities are arising from testing of service-based applications?
- Where are the differences between "normal" testing and testing of service-based applications?
- Which new approaches or adjustments / adaptations are needed for service-based application testing?

The sources indicated in Section 1.4 of the introduction have been searched using the search string "(test OR testing) AND (service OR SOA)". For each search result, its relevance has been assessed and only the relevant results have been summarized. Additional references have been identified by following the citations of existing publications. The results of the systematic review are summarized in the following.

An Approach for Specification-based Test Case Generation for Web Services [109]

Problem: How can specification-based testing of web services be performed?

Explanation: Specification-based testing of web services is important for consumer and broker because the web service source code is unavailable for them. They get only descriptions or specifications of the web services they want to use.

Solution: Hanna and Munro describe in [109] a method for specification-based testing of web services. Their approach is built on WSDL and XML Schema data types. First a XML Schema data types model is built. This model is used to generate test cases based on boundary value analysis. The test cases are stored in XML files. For the processing of the approach a cooperation between broker and consumer is necessary. The broker builds the model, generates the test cases and stores the XML test case file with the service description. The consumer retrieves the original WSDL file and the XML test case file. He can add test cases (based on his domain knowledge). Then he uses the test cases to send SOAP messages to the web service under test and analyzes the responses.

Benefits: A specification-based testing method is applied to web services using XML Schema data types models and XML test case files.

Extending WSDL to Facilitate Web Services Testing [256]

Problem: WSDL files provide no dependence information.

Explanation: WSDL files can be seen as a specification for testing web services. A WSDL specification contains the number of inputs and outputs, the type and order of inputs and outputs and how web services should be invoked. This information is not sufficient for testing.

Solution: In [256] Tsai et al. propose four kinds of extensions for WSDL files:

- input-output dependency: Can be generated by dependence analysis inside a web service. "May help to eliminate unnecessary test cases for carrying out regression test."
- invocation sequence: Provides tracing information among web services. Can be useful in path testing and data flow testing.
- hierarchical functional description: Improve functional and regression testing and enable automation.
- concurrent sequence specifications: Calling sequences, concurrent behaviors and timing aspects are captured.

Benefits: Support of testing web services with WSDL files as specification.

Coyote: An XML-Based Framework for Web Services Testing [261]

Problem: Testing web services is difficult.

Explanation: Web services are distributed and often only WSDL specifications are available.

Solution: The authors propose an XML-based object-oriented testing framework called Coyote. The framework is composed of test master and test engine. The test master specifies test scenarios and test cases, performs the analysis and converts WSDL specifications into test scenarios. The test engine interacts with the web service under test and provides tracing information.

Benefits: A framework for web service testing.

WSDL-Based Automatic Test Case Generation for Web Services Testing [18]

Problem: In [18] the authors address three problems: (1) Testing of web services needs to be automated; (2) Test cases must be generated based on standard specifications; (3) Test cases should be documented in XML-based standard format.

Explanation: Services are published, bound, invoked and integrated at runtime and only have a programmable interface. So automation of the testing process without user interaction is essential. "Web services propose a fully specification-based process using XML-based standards." These standards should be used for the generation of test cases. The use of XML-based standards also for the documentation of test cases is useful consistency of the process.

Solution: In [18] Bai et al propose a technique for generation test cases automatically based on WSDL specifications. They present their technique for individual and combination operations of atomic services. The technique consists of four steps: First the test data is generated from WSDL message definitions. Then test operations are generated based on the analysis of the parameters in the WSDL file. The third step is the generation of operation flows to test a sequence of operations by operation dependency analysis. At the end the test specification is build as test cases encoded in XML files.

Benefits: Technique for automatic test of atomic web services based on WSDL files.

Validation: The authors present an experiment on 356 services from which the WSDL specifications are given. 8200 test cases are generated from which 7500 are successful exercised. The remaining test cases mostly failed due to errors in the WSDL files.

Swiss Cheese Test Case Generation for Web Services Testing [253, 257]

Problem: How to check for the trustworthiness of web services.

Explanation: Web services are based on UDDI which is not responsible for the quality of services. So the trustworthiness / vulnerability of web services is a problem for the users of web services. Traditional dependability techniques must be redesigned to handle the dynamic features of web services.

Solution: The authors of [253] present Swiss Cheese test case generation. The OWL-S specification of a web service is converted to scenarios. After that boolean expressions are extracted. With this boolean expressions a K-map is formed and Hamming distances and boundary counts are computed. Finally a Swiss Cheese map is created in which each cell at least belongs to a test case. Positive and negative test cases are generated. At the end the test cases can be ranked.

In [257] the Swiss Cheese test case generation is based into a framework for verifying web services by completeness and consistency analysis.

Benefits: The approach is specification-based which deals with the lack of code when using web services. The creation of positive and negative test cases is a possibility to provide high insurance.

Validation: The authors show the generation of test cases with their approach in an example.

Ontology-Based Test Case Generation for Testing Web Services [272]

Problem: Testing of dynamically and automatically constructed web services.

Explanation: Automatically constructed web services cannot be tested with traditional offline and manual testing techniques.

Solution: Wang et al present a model-based approach with automatically generated test cases based on the OWL-S web services process specification. "OWL-S is first transformed into a Petri-Net model to provide a formal representation of the structure and behavior of the service under test." Then test cases are generated based on the Petri-Net model. "Test cases are generated from two perspectives: test process generation based on the Petri-Net behavior analysis; and test data generation based on ontology reasoning."

Benefits: This approach has following characteristics:

1. "it applies the semantic WS process model to the model-based WS testing process;"
2. "it uses the Petri-Net model to capture the structure and behavior of OWL-S process and defines the Petri-Net ontology for carrying the semantic information;"
3. "it investigates the ontology-based test knowledge representation and reasoning-based test generation."

Validation: "A prototype system TCGen4WS is implemented to support the proposed approach."

Automated Functional Conformance Test Generation for Semantic Web Services [195]

Problem: Testing of semantic web services.

Explanation: "The persistent state of the world (in terms of domain instances) of the web services needs to be accounted for in order to derive the test suite."

Solution: "For each web service, our approach produces testing goals which are refinements of the web service preconditions using a set of fault models. A novel planner component accepts these testing goals, along with an initial state of the world and the web service definitions to generate a sequence of web service invocations as a test case. Another salient feature of our approach is generation of verification sequences to ensure that the changes to the world produced by an effect are implemented correctly. Lastly, a given application incorporating a set of semantic web services may be accessible through several interfaces such as direct invocation of the web services, or a Graphical User Interface (GUI). Our technique allows generation of executable test cases which can be applied through both interfaces."

Benefits: "We have presented an automated approach to generate functional conformance tests for semantic web services which are defined using the Inputs, Outputs, Preconditions, Effects (IOPEs) paradigm."

Validation: "We also present results which compare two approaches: an existing manual approach without the formal IOPEs information and the IOPEs-based approach reported in this paper. These results indicate that the approach described here leads to substantial savings in effort with comparable results for requirements coverage and fault detection effectiveness."

Generating test cases for web services using extended finite state machine [132]

Problem: "It is hard to test web services."

Explanation: "Web services are distributed applications with numerous aspects of runtime behavior that are different from typical applications."

Solution: "This paper presents a new approach to testing web services based on EFSM (extended finite state machine). WSDL (web services description language) file alone does not provide dynamic behavior information. This problem can be overcome by augmenting it with a behavior specification of the service. Rather than domain partitioning or perturbation techniques, we choose EFSM because web services have control flow as well as data flow like communication protocols. By appending this formal model of EFSM to standard WSDL, we can generate a set of test cases which has a better test coverage than other methods. Moreover, a procedure for deriving an EFSM model from WSDL specification is provided to help a service provider augment the EFSM model describing dynamic behaviors of the Web service."

Benefits: "First, this paper introduces a new Web service testing method that augments WSDL specification with an EFSM formal model and applies a formal technique to Web service test generation. Second, using the EFSM based approach, we can generate a set of test cases with a very high test coverage which covers both control flow and data flow."

Validation: "The method is applied to an industry level example and the efficacy is showed in terms of test coverage and fault detection."

Contract-based Testing for Web Services [63]

Problem: In [63] Dai et al. describe that the problems of testing services derive from the fact that services are invoked instead of integrated and thus providers can evolve a service without the knowledge of the users.

Explanation: Lack of notification on changes are a problem for users of a service because they want to be sure that the service functions as stipulated when first using the service. The authors conclude that the system "has to be tested dynamically and automatically at runtime without human interaction".

Solution: Dai et al. present an approach which uses contracts as formal agreements between users and providers containing rights and obligations for both sides. The contracts are described using the OWL-S process model. For generating test cases two parts are suggested: the generation of valid test data and the generation of the test process. Both generation processes are based on contracts. In addition, contracts contain enough information on the expected output for using them as test oracles. Combined with monitoring, the functionality of the service-based application can be checked.

Benefits: Using contract-based testing in the way the authors suggest, is a possibility to automatically check the functionality of a service by derived test cases and oracles. Test cases and oracles are derived from the contracts which can be seen as specification of the service.

Towards Contract-based Testing of Web Services [111]

Problem: Heckel and Lohmann deal with the problem that the flexible service-oriented architecture "requires new mechanisms to ensure that service requester and service provider can work together".

Explanation: In [111], the authors state that due to the loose coupling and the distribution of services, service requestors often bind to services at run-time. In the authors' point of view this prevents integration testing of a service-based application.

Solution: The authors propose using Design by Contract for web services. For the interoperability of service providers and requesters the concept of Design by Contract, which comes from component-based systems, should be complemented by the use of required and provided contracts. A provided contract specifies pre- and post-conditions of the service. A required contract specifies the information the requester is willing to provide and the situation he wants to achieve at the end. For the representation of the contracts Heckel and Lohmann propose the use of graph transformation rules. Criteria "whether a provided contract of a service provider fulfils the required contract of a service requester" are mentioned in a previous paper by the authors.

Furthermore the authors propose the usage of their provided and required contracts for testing a web service. Pre-conditions of the provided contract can be the source for the generation of test inputs whereas post-conditions can serve as test oracles. If other services are needed for the execution of the service under test, the required contracts of the service under test can simulate the behaviour of this required service.

Benefits: The approach of Heckel and Lohmann provides a possibility for testing services against their specification when the specification is available as provided and required contracts in form of graph transformation rules.

Testing BPEL-based Web Service Composition Using High-level Petri Nets [75]

Problem: Testing of BPEL-based web services.

Explanation: "BPEL-based web service composition essential has dynamic aspect such as recomposition, re-configure, and dynamic binding during execution. This aspect makes behavior analysis and testing of BPEL-based web service composition software significantly complicated."

Solution: "This paper proposes a technique for analysis and testing BPEL-based Web service composition using high-level Petri nets. To illustrate how these compositions are verified, the relationships between BPEL-based Web service composition and high-level Petri nets is constructed. By analyzing the structure of Web service composition based on BPEL, the corresponding HPN is constructed. The dynamism and occurrence are presented in HPN with guard expression with coloured token. After translation, the equivalent HPN of the Web service composition based on BPEL can be verified on existing mature tool, and the related researches on HPN, e.g. testing coverage and reduction techniques that have been studied deeply, can be employed in testing of Web service composition based on BPEL, optimized test case can be generated based on the HPN translated."

Benefits: "This paper proposes a method to model the workflow of web service composition based on BPEL using HPNs and testing method of web service composition based on BPEL using HPNs. ... The main problem around testing web service composition based on BPEL, the verification and the quality of the test cases generated, can be resolved."

Validation: "An example is provided to illustrate the translation ruled and the automatic verify progress."

On Combining Multi-formalism Knowledge to Select Test Models for Model Transformation Testing [228]

Problem: "How can we select models for testing web services?"

Explanation: In [228] the authors present a method to automatically select test models given any meta-model as input. The selected models are used to test model transformations. Model transformations are ubiquitous in Model-driven Engineering as they automate important software development steps. Testing and validating them is thus a crucial activity for successful MDE. Knowledge to test model transformations can come from various sources and be expressed in different formalisms. However, combining this knowledge to a set of consistent constraints is a challenge. Using this set of constraints to select test models is another difficult problem. In this paper, the authors identify sources of testing knowledge and present several transformations to a *common constraint language* of first-order relational logic. The constraints are solved leading to a selection of qualified test models from the input domain of a model transformation. The selected models not only conform to the input meta-model but are also specialized to conform to test model objectives, partition knowledge, and the transformation pre-condition. These test models serve as input for black-box testing of a model transformation. They outline a black-box testing tool Cartier that uses Alloy as the relational logic language to represent combined knowledge. They illustrate their approach using the Unified Modeling Language Class Diagram to Relational Database Management Systems transformation as a running example.

Benefits: The approach in [228] can be easily extended to automatic selection of WSDL instances called test models. QoS properties can be obtained, or inconsistencies detected in a service by feeding it with several hundred automatically selected test models.

Probabilistic QoS and Soft Contracts for Transaction based Web Services [207]

Problem: "How can we qualify test cases of Web Serivces?"

Explanation: Every test model to a web service needs a response that states whether a contract was satisfied by the input test case. This response is given by a function called a *test oracle* [258]. In [207] the authors present probability distributions of QoS properties such as response time as soft contracts that can act as test oracles. In particular they present the idea of obtaining a QoS contract for a web service orchestrations and choreographies leading to composed services. These soft contracts are better suited to incremental testing compared hard contracts (e.g., response time always less than 5 msec). Allowing progressive selection of test models using the method described in [228]. They implement their approach

using the TOrQuE tool. Experiments on TOrQuE show that overly pessimistic contracts can be avoided and significant room for safe overbooking exists.

Benefits: The benefits of using soft contracts as test oracles allows for the incremental development of test models that can help test orchestrations. Hard contracts do not give a direction to the test model selection process.

Model-based functional conformance testing of web services operating on persistent data [232]

Problem: "WSDL standard does not allow behavioral specification (such as pre- and postconditions) of web services in the presence of persistent data."

Explanation: For web services which operate in the presence of persistent data, "the result of invoking a web service depends not only on the user input (such as order entry), but also on the state of the persistent data (such as current inventory level of the ordered product). Testing of such web services poses challenges because the persistent data state of the system needs to be accounted for in order to derive the test suite."

Solution: "In this paper, we propose the use of existing test generation techniques based on Extended Finite State Machine (EFSM) specification to address the generation of functional conformance testes for web services which operate on persistent data."

Benefits: "The novel contribution of this paper is an algorithm which translates a WSDL-S behavioral specification of operations of a web service into an equivalent EFSM representation which can be exploited to generate an effective set of test cases."

Scenario-Based Web Service Testing with Distributed Agents [255]

Problem: Testing of web services has to be seen as a whole process.

Explanation: Testing Web services is difficult, because they are loosely coupled, have a dynamic behavior, are invoked by unknown parties, can have concurrent threads and object sharing and different parties (client, broker and provider) are involved in testing. These difficulties have to be considered all together to find an adequate solution.

Solution: Tsai et al propose in [255] a web service testing framework (WSTF). WSTF includes the following features:

- enhanced WSDL: Four kinds of extensions of WSDL files as described in [256].
- scenario-based testing as a specification-based testing technique which can be applied to WSDL files: First the scenarios are build for sub-systems. Then the interaction of different sub-systems is considered to form an overall system scenario.
- automated test script generation based on the scenario specification
- automated distributed test execution including test monitors and distributed agents

Benefits: An automated framework for testing of web services is presented.

Validation: The authors describe an example of a supply chain management system. All seeded bugs are found with the framework.

Group testing for web services [254, 263, 252, 262, 15, 259]

Problem: A " large number of web services is available on the internet". In particular for the same specification different implementations are possible and likely.

Explanation: For the large number of web services available on the internet testing needs to be performed. This is a huge effort in time and cost.

Solution: In [254, 263] progressive group testing is suggested as solution. The presented technique can be used for unit and integration testing. Two phases are proposed: (1) prescreening; (2) runtime group testing.

In the prescreening phase the goal is to eliminate "unlikely-to-win web services" as fast as possible. Test cases are generated and with the dependencies among the test cases a hierarchy is build. At the end of the prescreening phase multiple winning web services with ranking are selected.

In the runtime group testing phase the best candidates identified in the prescreening phase are integrated in the live system. Further testing is performed with these and newly arriving web services. Given a set of functionally equivalent web services. Input for one of these services is forwarded to all of them. The results from all services are voted (by a voting service). By comparing the output of one service with the weighted majority output (as oracle) faults can be detected.

In [252] the authors update their group testing technique to a framework called ASTRAR (Adaptive Service Testing and Ranking with Automated oracle generation and test case Ranking). The prescreening is done in the so called training phase while the volume testing phase performs runtime group testing.

The ASTRAR framework is extended in [262]. Here the two phase process is extended by introducing a windowing mechanism. The web services are divided into subsets called windows. The testing process then is not simultaneous any more because the web services are tested window by window. This method improves testing by saving test runs.

In [15] group testing with windowing is extended with an adaptive mechanism. This adaptive mechanism provides the possibility to adapt test cases to the continuously changing services (in case of updates or redeployment).

In [259] a stochastic voting for group testing is proposed. The majority-voting scheme is designed to distinguish correct but slightly variant output from truly incorrect output. This paper proposes a hierarchical classification based on simulated annealing and multi-dimensional Chi-square statistical techniques to analyze data to see if a majority can be reached.

Benefits: Group testing provides the possibility to test functional equivalent services and to receive a ranking between these services.

Validation: Case studies are performed to examine for example the relationships among training size, target size and test cost.

Perturbation-based testing for Web Services [181, 65, 278]

Problem: Does the interaction between two web services function correctly?

Explanation: Web services are located on different servers which belong to different companies. The web services interact by passing messages and data (through XML, SOAP).

Solution: In [181] data perturbation is used to test the interaction of web services. Request messages are modified. With the resending of the modified request the messages are used as test cases. The analysis of the response messages reveals uncorrect behavior.

The authors distinguish between data value perturbation - modifies values in SOAP - and interaction perturbation - modifies messages in RPC and data.

In [278] further perturbation operators for XML schema are described.

[65] builds upon the work of Offutt and Xu. They introduce new perturbation operators for the modification of SOAP messages.

Benefits: Interaction test between two web services.

Validation: In [181] an empirical study is presented where the test method found 14 of the inserted 18 faults. The authors of [65] show the function of their approach with the developed tool SMAT-WS.

BPEL Unit Testing [165, 153]

Problem: Test of BPEL compositions

Explanation: The problem with testing BPEL compositions is numerous external dependencies. These dependencies are based on the web services that are accessed by the BPEL composition.

Solution: The authors present an approach for unit testing BPEL processes: BPELUnit. The framework contains the following steps:

- test specification: the WSDL specification is used to generate test cases
- test organization: the test cases are grouped into test suites with links to external artifacts
- test execution: the BPEL process can be tested by simulation or by real-life deployment
- test results: a report of successful test cases, failures and errors is generated

Benefits: A framework for testing BPEL processes.
Validation: The authors describe the implementation of their framework.

Biased covering arrays for progressive ranking and composition of Web Services [46]
Problem: "Which services are the most appropriate to combine into an application?"
Explanation: When building a composed service, a choice between several services that provide the same functionality must be exercised. In order to test which one of alternative services is the most adequate one, the services have to have been published.
Solution: The authors of [46] build upon the group testing techniques of Tsai et al. [254, 263, 252, 262, 15]. "Group Testing is applied to narrow down the number of prospective candidates for each web service that may later be combined into a final composite service." Then "interaction testing may be applied to generate economically sized test suites that further evaluate the candidate WS." Therefore biased covering arrays are used.
Benefits: The results of group testing techniques are used to further investigate web services which could be integrated in a composed service.
Validation: An example for the use of biased covering arrays is shown.

Testing of service-oriented architectures - a practical approach [78]
Problem: Testing services is time and cost consuming.
Explanation: "Whenever a change in one of the source code parts arises a new test has to be done and very often there is no stable test environment. This then leads to the problem of setting up the whole test environment for every single change. In such cases the investment for testing can explode."
Solution: The automation of the test process can lead to a decrease of costs and time for testing. In [78] an approach for automated testing of services is presented including a Meta language in XML for the definition of test cases. The authors focus on the presentation of a prototype implementation called SITT (Service Integration Test Tool). SITT "has the possibility to test and monitor if certain workflows between multiple service endpoints really behave as described with the XML Meta language."
Benefits: Reduction of costs and time spend for testing services.
Validation: A real-world application scenario from the domain of Telecommunications providers, namely Mobile Number Portability is presented to show the features of SITT.

An abstract workflow-based framework for testing composed web services [122]
Problem: "Testing composed web services impose many challenges to existing testing methods, techniques, and tools."
Explanation: "Structural-based testing approaches have been thoroughly researched for traditional applications; however, they have not yet been examined, as a methodology, for testing composed web services."
Solution: The authors introduce in [122] "a formal model for an abstract-based workflow framework that can be used to capture a composed web service under test." The structural-based workflow framework uses workflow graphs for simple composed and complex composed web services. Additionally a set of applicable structural-based testing criteria to the framework is defined.
Benefits: Use of structural-based testing techniques for testing of composed web services.

A test bed for web services protocols [203]

Problem: "Transactions across composed web services are usually non-trivial and require the use of some pre-agreed or standard protocols."

Explanation: "Proper specification and implementation of these transaction protocols are critical for the correct execution and termination of transactions."

Solution: In [203] a test bed based on conformance checking for automatically testing a given web service protocol implementation is proposed. Especially the number of entities that can participate is considered because it might not always be the same number. In this approach "sequences of messages (outputs) are checked against a formal model rather than a single message at a time". "The architecture of the test bed consists of three main phases: envelope capture, parsing and conformance checking." "Messages from actual implementations are logged over a period of time and parsed and checked for conformance against a dynamically adaptable model of the WS protocol in operation."

Benefits: A test bed for testing transaction protocols including protocol implementations where the number of entities that can participate is not always the same.

Validation: An example is presented to illustrate the test bed.

WSDLTest - A Tool for Testing Web Services [234]

Problem: "A significant barrier to the use of Web services is the problem of testing them."

Explanation: "Regardless of where they come from, no one can ensure that the web service components will work as one might expect. Even those that are bought may not fit exactly to the task at hand. The fact that they are not compatible can lead to serious interaction errors."

Solution: "One of the solutions to deal with the problem lies in the ability to simulate the usage of the services. Requests must be generated and responses must be validated automatically in a fast and reliable manner. To accomplish this goal, we have developed a tool called WSDLTest. WSDLTest is part of a larger complex tool set - DataTest - for generating and validating system test data."

Benefits: "The WSDLTest tool generates Web service requests from the WSDL schemas and adjusts them in accordance with the pre-condition assertions written by the tester. It dispatches the requests and captures the responses. After testing it then verifying the response contents against the post-condition assertions composed by the tester."

Validation: The Experience with WSDLTest in an e-Government project is reported.

Automatic Conformance Testing of Web Services [112]

Problem: "Can the client trust the implementation of the service description?"

Explanation: In [112] Heckel and Mariani start their research with service descriptions which are supplemented by adding a behavioral specification of the service consisting of graph transformation rules. These graph transformation rules can model the behavior of the provided service and the requirements of requesters. "Then, service discovery includes the matching of these models" and only "if the provided model satisfies the requirements, binding is allowed". It remains the problem that the model may be suitable (for the requirements of the service requester) but the actual implementation behind be faulty.

Solution: To overcome the problem, the authors introduce high-quality service discovery which adds automatic testing to the approach of behavioural matching with graph transformation rules. Then "the registration of services is allowed only if testing is passed".

Benefits: "Clients that use a high-quality service discovery agency have the guarantee that any discovered Web Service has passed the testing phase, therefore it can rely on both the interface compatibility and the implementation of the service."

Validation: For validating their approach the authors performed experiments with two simple web services - each containing one single operation - and with the more complex Amazon Web Service. The GT rules for the web services are derived from the provided descriptions/documentations of the services. For one of the simple services, the approach found a fault in the implementation. By testing the Amazon Web Service a fault in the specification was revealed. After modifying the GT rules the web service

passed the test. So Heckel and Mariani draw the conclusion that their technique "is useful with respect to the complexity of current web services".

WebSob: Automated Robustness Testing of Web Services [158, 159, 160]

Problem: How can black-box robustness testing be performed for web services?

Explanation: In [158, 159] and [160] Martin, Basu and Xie look at web service testing from the consumer point of view. Consumers mostly don't have the possibility to get implementation details of the used web services. Thus consumers can only perform black-box testing. Moreover robustness of the web service is a problem for the consumer. The web service has to handle input parameters which contain consumer specific information. If the web service is not robust enough to handle these inputs it is possible that unauthorized instances retrieve consumer specific information.

Solution: The authors propose the framework "WebSob" for automated robustness testing of web services. Based on the WSDL description made available by the provider of the web service, the framework contains the following steps:

1. code generation: to implement a service consumer
2. test generation: use of existing unit test generation tools for Java. Besides random values focus on extreme and special values.
3. test execution: runs generated tests
4. response analysis: selection of tests whose responses indicate possible robustness problems. The selected tests have to be inspected manually. The authors classified four types of exceptions that indicate possible robustness problems: "404 File Not Found", "405 Method Not Allowed", "500 Internal Server Exception" and "Hang".

Benefits: The authors present a framework for automated black-box testing of web services based on WSDL descriptions which pursues robustness against extreme or special input values.

Validation: In their evaluation Martin, Basu and Xie try to answer the following research questions: Can WebSob generate test cases with only the WSDL file provided as input? And can WebSob identify potential robustness problems in web services? For their evaluation they used 35 freely available web services. It was possible to generate tests and some responses indicated possible robustness problems.

Regression testing approach of Ruth and Tu [215, 214, 216]

Problem: Web services undergo rapid modifications which have to be supported by rapid verification.

Explanation: For each modification of a web service two aspects have to be examined: (1) Do the modified parts function correctly? (2) Does the modification have effects on the unmodified functions?

For the second aspect it is common practice to use regression testing and to run previously generated test cases again. Yet, this is not affordable for complex systems and it is not rapid.

Solution: In [215] the authors propose to apply safe regression test selection (RTS) to web services. The approach is based on the safe RTS algorithm by Rothermel and Harrold for monolithic applications and requires control-flow graphs.

In [214] the approach of [215] is extended so that the safe RTS technique can be automated. Another extension is the handling of multiple concurrent modifications which is described in [216].

Benefits: The authors present an approach to apply safe regression test selection to web services. With this approach the reduction of testing time and of costs (for calling web services) is possible.

Validation: For their validation Ruth and Tu use a case study of a purchase order system. The system consists of three web services and an application. After modifications in one node of the CFG, the use of safe RTS for web services leads to a reduction of 68% of the tests for the whole system.

Distributed functional and load tests for Web services [224]

Problem: System-level testing of service-based applications.

Explanation: "System-level testing considers functionality and load aspects to check how a system performs for single service requests and scales as the number of service requests accessing/using it increases."

Solution: "This paper presents a flexible test framework including functional, service interaction and load tests. It is generic in terms of being largely independent of the system to be tested. The paper discusses the automation of the test framework with the Testing and Test Control Notation TTCN-3 and also presents an implementation of the test framework using a TTCN-3 toolset."

Benefits: "The paper presented a flexible test framework for Web services realized in TTCN-3."

Validation: "The test framework is exemplified for Web service tests and demonstrates distributed functional and load tests for a specific Web service."

A multi-agent based framework for collaborative testing on Web services [17]

Problem: Testing of web services.

Explanation: "Testing is a challenge due the dynamic and collaborative nature of web services."

Solution: "This paper introduces an on-going project on a multi-agent based framework to coordinate distributed test agents to generate, plan, execute, monitor and communicate tests on WS. Test agents are classified into different roles which communicate through XML-based agent test protocols. Test Master accepts test cases from Test Generator, generates test plans and distributed them to various test groups. A set of test agents that implement a test plan are organized into a test group, which is coordinated by a Test Coordinator. Test Runners execute the test scripts, collect test results and forwards the results to Test Analyzer for quality and reliability analysis. The status of the test agents are monitored by the Test Monitor. Test agents are dynamically created, deployed and organized. Through the monitoring and coordinating mechanism, the agents can re-adjust the test plan and their behavior at run-time to be adaptive to the changing environment."

Benefits: "To address the challenges of collaborative and dynamic service-oriented testing, this paper presents the MAST framework for testing services with agent-based technology."

The audition framework for testing Web services interoperability [36]

Problem: There is a "urgent need for methodologies supporting Web services reliable interaction, and in particular deal with testing concerns."

Explanation: "Service oriented architectures and Web services are emerging technologies, which have overall inherited problems and advantages from the component-based approach, but exacerbated the aspects of loose coupling, distribution and dynamism of 'components', here elements furnishing published services on external client requests."

Solution: "We propose a framework that extends UDDI registry role from the current one of a 'passive' service directory, to also sort of an accredited testing organism, which validates service behaviour before actually registering it. This testing stage (called audition) mainly focuses on interoperability issues, so to facilitate the coordination among services registered at the same UDDI. The audition needs to rely on a Web service specification augmented with information on how the service has to be invoked. We propose that this information is given in the form of a protocol state machine, which is a newly introduced behaviour diagram of the UML 2.0."

Benefits: The authors "have proposed a general framework in which the UDDI registering role is extended to also play the role of an external testing organism which validates the WS conformity to the published (augmented) interface, and its ability to interact with other available services. The underlying idea is that before being registered at a UDDI, a WS must pass an audition, in which its dynamic behaviour is tested, especially regarding coordination aspects in the interaction with other services."

A Framework for Testing Web Services and Its Supporting Tool [169]

Problem: "With the increase of the popularity of Web services, more and more Web applications are developed with this new kind of components."

Explanation: "This new way of software development brings about new issues for software testing, which has been widely recognized as a realistic means for ensuring the quality of software systems."

Solution: "In this paper, we focus on facilitating the testing of Web services. In particular, we propose a framework for testing Web services, which can help a tester of Web services in two ways: firstly, it can help the tester to acquire effective test data; and secondly, it can help the tester to execute the test data for the testing."

Benefits: "In this paper, we have proposed a framework for automating the testing of Web services. In our framework, we mainly focus on test data acquisition, test data selection, and test data execution."

Validation: "We also discuss some issues for the implementation of its supporting tool. Furthermore, we report an experimental study of our framework. The results can validate the effectiveness of our approach."

Testing Web services [231]

Problem: Testing of web services.

Explanation: "Web services are based on communication protocols, service descriptions, and service discovery and are built on top of existing Web protocols and based on open XML standards. Web services are described using Web Services Description Language (WSDL), and the universal description, discovery, and integration directory provide a registry of Web services descriptions. Testing Web services is important for both the Web service provider and the Web service user."

Solution: "This paper proposes a technique for testing Web services using mutation analysis. The technique is based on applying mutation operators to the WSDL document in order to generate mutated Web service interfaces that are used to test the Web service. For this purpose, we define mutant operators that are specific to WSDL documents."

Benefits: A technique for testing web services based on mutation analysis is presented. The authors identified nine mutation operators which can be applied to WSDL documents.

Validation: An example shows the procedure to use the new mutation testing technique.

A Model-Driven Approach to Discovery, Testing and Monitoring of Web Services [150]

Problem: Development of high-quality service-based applications and the embedding of development and verification in the service-lifecycle.

Explanation: "Established technology for providing, querying and binding services is largely based on syntactic information. The lack of semantic information in service descriptions prevents reliable automatic integration of services."

Solution: A "framework for developing high-quality service-based applications adressing both the verification problem as well as its embedding in the service life-cycle" is presented. The approach "aims to guarantee high-quality service-oriented applications by refining the classical life-cycle of service registration, discovery and usage". Registration: "Only tested web services should be allowed to participate in high-quality service-based applications." (see [112]) Discovery: "Based on the extension of service descriptions to include behavioural specifications, service discovery can match descriptions against behavioural requirements." Usage: "Service models are used to automatically generate monitors that are able to continuously verify the behaviour of web service implementations."

Benefits: The "framework adresses coherent and model-driven development of high-quality service-based applications and its embedding into the service life-cycle".

Validation: The framework has been used with several case studies. In this paper the results of the studies are summarized.

Generation of Conformance Test Suites for Compositions of Web Services Using Model Checking [98]

Problem: Testing compositions of web services is complex.

Explanation: The complexity of testing compositions of web services relies on their distributed nature and asynchronous behaviour.

Solution: A new testing method for compositions of web services is proposed. "A formal verification tool (the SPIN model checker) is used to automatically generate test suites for compositions specified in an industry standard language: BPEL. Adequacy criteria is employed to define a systematic procedure to select the test cases. Preliminary results have been obtained using a transition coverage criterion."

Benefits: The main contribution of the presented research is the "definition of a new method to obtain conformance test suites for compositions of web services".

A simple approach for testing web service based applications [243]

Problem: "Several aspects make testing web services a challenging task."

Explanation: "The heterogeneity of web services that uses different operating systems and different server containers makes the dynamic integration of these services a non-easy task. Moreover, web services do not have user interfaces to be tested and therefore they are hard to test manually."

Solution: "This paper presents a technique for building reliable Web applications composed of Web services. All relevant Web services are linked to the component under test at the testing time; thus, the availability of suitable Web services is guaranteed at invocation time. In our technique, a Web application and its composed components are specified by a two-level abstract model. The Web application is represented as task precedence graph (TPG) and the behavior of the composed components is represented as a timed labeled transition system (TLTS). Three sets of test sequences are generated from the WSDL files, the TLTS and the TPG representing the integrated components and the whole Web application. Test cases are executed automatically using a test execution algorithm and a test framework is also presented. This framework wraps the test cases with SOAP interfaces and validates the testing results obtained from the Web services."

Benefits: "One contribution of this paper is that it allocates all suitable web services that fulfill a component during the testing of the web application rather than during invocation time. This will give a wide range for rapid selection of available web services at invocation time. Another contribution is the full-coverage test cases that are generated form the WSDL files and the TLTS of the integrated components and the TPG of the whole web application. A third contribution is the Test-execution algorithm that generates two log files. Finally, a testing framework is presented, it supports both execution and test scenario management."

Search-based Testing of Service Level Agreements [73]

Problem: In [73] Di Penta et al. deal with the problem that the violation of a Service Level Agreement (SLA) - negotiated between service provider and service consumer - "would cause lack of satisfaction for the consumer and loss of money for the provider".

Explanation: A violation of an SLA (Service Level Agreement) —i.e., that the provider cannot hold the agreed QoS level— is undesirable for both sides. So, the SLA should be tested before offering. This reduces the possibility that the SLA will be violated during the service usage.

Solution: The presented solution is to use Genetic Algorithms (GAs) for the generation of test data. The approach for testing SLAs is presented from two perspectives: a white box approach which can be used by integrators for which the composition source code is available and a black box approach for all other groups which do not have access to the source code. The white box approach starts with the identification of potentially QoS-risky paths (which "are likely to exhibit high values for upper-bounded QoS attributes and low values for lower-bounded QoS attributes"). After this identification, a GA is used to generate test cases that cover the path and violate the SLA. The GA considers combinations of inputs and bindings (between abstract and concrete services). Mutation, crossover and selection operators

are adapted to services. The presented fitness function considers the distance of an individual to QoS constraint violation and the coverage of the QoS-risky paths. In the black box approach QoS-risky paths cannot be computed so the fitness function only considers the QoS constraint distance.

Benefits: Testing a SLA before offering it can increase satisfaction of service consumers and can prevent service providers from losing money because of violated SLAs.

Validation: For the evaluation of the effectiveness of the presented testing approach two case studies are introduced. The results provide evidence that the approach is able to generate test cases for the violation of SLAs.

Using Test Cases as Contract to Ensure Service Compliance across Releases [45, 198]

Problem: In [45] Bruno et al. look at the problems that occur because services are used but not owned. Thus services are out of the users control and the user cannot decide on migrating to a new version of the service and is not always aware of changes the service provider makes.

Explanation: Users want to be sure that the service "delivers over the time the desired function" and always meets the QoS requirements. Without control over changes, users cannot enforce these points and have to trust service providers instead.

Solution: The authors present an approach appropriate for regression testing by the user. The method has its roots in component-based software testing and uses test cases as a contract between the user and the provider of the service. The idea of the approach is to bind test cases and a set of QoS assertions to a service and to test after a certain time if the test cases and assertions still hold.

Benefits: The approach offers a possibility to stipulate quality of service levels and functionality between service user and provider.

Validation: The approach of Bruno et al. is validated by an empirical study. For five releases of two open source systems, test cases are run against all releases of each system. For both systems some test cases failed and showed that there were changes in a new release that could be problematic for the users of an older release.

A metamorphic testing approach for online testing of service-oriented software applications [51]

Problem: "The criteria to define functional correctness of the service may vary according to its environment."

Explanation: "A service-oriented application may bind dynamically to its supportive services. For the same service interface, the supportive services may behave differently. A service may also need to realize a business strategy, like best pricing, relative to the behavior of its counterparts and the dynamic market situations."

Solution: "This article proposes a metamorphic approach for online services testing. The off-line testing determines a set of successful test cases to construct their corresponding follow-up test cases for the online testing. These test cases will be executed by metamorphic services that encapsulate the services under test as well as the implementations of metamorphic relations. Thus, any failure revealed by the metamorphic testing approach will be due to the failures in the online testing mode."

Benefits: "Using our online testing methodology, testers builds a bridge between the test oracle available in the offline testing mode and the test oracle problem encountered in the online testing mode."

Validation: "We have also conducted an experiment to evaluate the feasibility of our proposal. The experimental results encouragingly indicate that, on average, when the set of original test cases are unknown to be successful, an extra 16% effort to check test results and a 13% reduction of failure detection are observed. This supports our proposal that original test cases should be (much) better to be successful, particularly when the checking is (much) less costly when it can be conducted in the offline testing mode."

Dynamic Reconfigurable Testing of Service-Oriented Architecture [16]

Problem: "SOA (Service-Oriented Architecture) presents unique requirements and challenges for test-

ing."

Explanation: "Dynamic reconfiguration in SOA software means that testing need to be adaptive to the changes of the service-oriented applications at runtime."

Solution: "This paper presents a ConfigTest approach to enable the online change of test organization, test scheduling, test deployment, test case binding, and service binding. ConfigTest is based on our previous research on the MAST (Multi-Agents-based Service Testing) framework [17]. It extends MAST with a new test broker architecture, configuration management and event-based subscription/notification mechanism. The test broker decouples test case definition from its implementation and usage. It also decouples the testing system from the services under test. With the configuration management, ConfigTest allows the test agents to bind dynamically to each other and build up their collaborations at runtime. The event mechanism enables that a change in one test artifact can be notified to all the others which subscribe their interests to the change event."

Benefits: "ConfigTest proposed in this paper addresses the issues with the loose coupling test broker architecture, flexible configuration specification, and an effective event-based subscription / notification mechanism. ConfigTest can deal with various change scenarios."

Validation: "This paper presents and analyzes the collaboration diagrams of various testing reconfiguration scenarios and illustrates the ConfigTest approach with an example of service-based book ordering system."

Quality analysis of composed services through fault injection [97]

Problem: The goal of this work is to present a systematic testing method for service-based Cooperative Information System (CIS) processes. The method is based on fault injection during process execution.

Explanation: The analysis is performed on processes where the service composition structure is fixed, with a well defined process schema. Is is assumed that faults occur one at a time and that for services the WSDL description of the service interface is available. The testing is of two types: a) black-box testing, when the service implementation code is not accessible, and b) white-box testing, when service code is accessible, and in particular when information on used data sources used by the service is also visible.

Solution: The method is based on fault injection during process execution. Two types of faults are considered: data faults and time delays. Data faults are frequent in a CIS, since both data redundancy, which is a possible cause of faults due to misalignment of data, and the probability of out-of-date values increase in a CIS composed of hundred nodes with replicated data and external services. The execution time of the composed service depends both on the performance of the single component services and on their composition. Hence, failures (visible erroneous process behavior caused by a fault) might be caused by delays in process execution. The method is to assess how the process reacts to a systematic set of faults injected during its execution. The developed support tool uses fault injection to test composed services through the method. Experimental results are shown.

Benefits: The analysis of fault consequences can be used at design time to suggest possible process or service improvements, or to support service management with appropriate repair actions. This approach contributes to increase the quality of a composed service, since it enables early detection of data faults and their correction. This technique allows repairing the process with no modifications of the process structure.

Validation: Experimental results on a reference example show how design quality of a composed service can be tested. The analysis of the effect of data and time delays are the outcome.

4.4.2 Monitoring

The comprehensive and detailed survey on the monitoring activities in the life-cycle of service based application is presented and detailed in Deliverable PO-JRA-1.2.1. In this section we will summarize and classify the survey results relevant from the quality provisioning perspective.

The related work on monitoring of service-based application were collected through a systematic review process that covers several relevant areas, such as service-oriented computing, grid computing, business process management.

In that review the goal is to reflect the state of the art in monitoring of service-based applications. In particular, the survey aims at

- providing a taxonomy of the state-of-the-art monitoring principles, concepts, and methodologies;
- performing a comparison of these approaches and the monitoring methodologies from different areas of information systems, such as software engineering, databases, distributed systems, business intelligence, etc;
- identifying gaps and overlaps in the current research activities and reveal the challenges and perspectives for this research topic.

With respect to monitoring in SOA, the presented survey is based on and extends a previous work in [102]. In that survey the authors concentrated on run-time monitoring of Web services and service compositions. The reviewed work presents both the research approaches towards monitoring of Web services and Web service-based compositions. The presented survey follows this approach and extends the list of relevant work with recent advances in this area. Apart from the results from [102] (26 papersworks), the relevant work were identified through the list of sources presented below, and from the related work mentioned in the corresponding sections of papers. We remark, however, that the results of the serch partially overlap with those in extracted from [102]; we have reviewed only the most relevant work from the whole set of papers and reports.

Assumption-based monitoring of service compositions [199, 21]

The papers [199, 21] focus on run-time checking of the behavioral requirements (expressed in a special temporal logic-based notation) on the services participating to the composition, and collecting certain statistical and non-functional information over their execution.

Monitoring Conversational Web Services [39]

The work presents an approach for checking at run-time that the actual behavior of the conversational service complies with the expected behavior represented in a special algebraic notation that expresses functional constraints on the service evolution.

Smart Monitors for Composed Services [24]

A technique for monitoring functional and non-functional assertions over BPEL process activities is presented. Assertions may be specified either in programming language directly, or using special assertion language. The monitored process is modified in a way to interact with a dedicated monitoring service to provide the relevant data.

Dynamo [25, 26, 22]

Dynamo framework proposes an expressive monitoring language WSCoL for expressing functional and non-functional requirements (assertions and invariants) on the BPEL processes. The framework allows for collecting information from external sources (services). In [22] the language is extended with the possibilities to express more complex (temporal) properties over the executions of underlying processes.

Monitoring for diagnosis [12]

Monitoring approach in this work has a goal of collecting and reporting a fault diagnosis information, rather than simply detecting property violation. The approach is based on a distributed framework that extends the functionality of the services with the diagnosis evaluation and management facilities.

Monitoring privacy-agreement compliance [29]

The work addresses run-time monitoring of compliance of the privacy agreement defining the users privacy rights and their possible handling by the service provider. The privacy properties (obligations and data rights) are formally in temporal logic specifications and the corresponding state machines, and then monitored at run-time using the events reported in the system log.

Requirements monitoring based on event calculus [238, 155, 156]

The authors approach the problem of monitoring service-based applications for conformance to a set behavioral requirements expressed in a rich event calculus-based specification language, and deals with service events, quality metrics, temporal constraints, etc. A run-time logic inference engine is used to detect violations of the properties.

Monitoring security patterns [237, 134]

The problem of monitoring security properties is addressed. The security properties are specified as patterns and formally represented in event calculus-based notation. The run-time checking of the properties relies on the approach defined above.

Performance Monitoring for utility computing [84]

The work deals with monitoring of SLA. The contract patterns are formally represented in event calculus and monitored using a specific framework.

Planning and monitoring execution with business assertions [144]

In the work monitoring is used to detect failures in the execution of customized business process in order to dynamically re-plan and adapt the process execution to the user requirements.

Automated SLA Monitoring [220]

The work proposes an approach for monitoring compliance of the service execution to the predefined contract information (SLA). The formalized contract statements are monitored by intercepting service interactions and process logs. The management of the monitored information and the analysis of compliance is performed by a dedicated platform.

WSLA [131]

An industrial approach proposed by IBM for monitoring SLA information is presented. The framework relies on a comprehensive model of contracts, QoS metrics and their aggregation, and on a sophisticated, multi-functional run-time environment.

Cremona [151]

Cremona is an industrial run-time platform for negotiating, managing, and monitoring SLA agreements. The agreements are defined using WS-Agreement notation. Multi-layered and extensible execution platform and API support the monitoring and management actions.

Colombo [60]

Colombo is an industrial platform for the development and enactment of services and service-based applications. Apart from many other facilities, it provides a way to check and enforce service policies expressed in WS-Policy notations. The policies may be attached to services, service operations, and even exchanged message types.

Query-based business process monitoring [28]

The approach targets the problem of monitoring BPEL processes. The monitoring queries are defined using visual pattern-based notations compliant with BPEL to define when the report should be provided, and using report specifications defining the information to be reported. For monitoring the queries are transformed into BPEL processes that collect information on the monitored processes from a low-level observer.

Model-driven development of monitored process [171]

The work proposes a way to develop SOA-based business processes with integrated monitoring information utilizing a model-driven approach. The authors have created a metamodel for modeling of process performance metrics (PPIs) based on BPMN process elements. The process augmented with monitoring primitives is automatically generated.

Model-driven development for BPM [55]

The work focuses on efficient development and deployment of monitoring information in BPM. The proposal relies on a meta-model extended with the concepts of metrics, business actions and events. The model is transformed into observer specification (for monitoring and evaluating the metrics) and a and data warehouse (to query and visualize information).

Probing and monitoring WS-BPEL processes [208]

The work deals with the problem on how to extract events from a BPEL process in order to enable auditing in an interoperable way. The authors propose a way to augment the BPEL processes with auditing activities, and present a set of strategies and mechanisms for collecting the relevant probes at different functional layers.

iBOM [50]

in this work the authors deal with the problem on how to create a monitoring solution, which not only enables to measure KPIs, but also to understand the causes of undesired KPI values, and prediction of future values. The approach is based on the combination of business activity monitoring with data mining to explain the monitored results.

An Agent-based Architecture for BAM [119]

The work present a multi-layered agent-based architecture for providing continuous, real-time analytics for business processes.

Fuzzy mining [108]

The work proposes an approach for automated extraction of the behavioral specification of the business process from the actual execution logs of the system. The approach relies data mining techniques for the log analysis.

Conformance checking with ProM [211, 268]

The presented approach allows for checking (at run-time or posteriori) the conformance of the observed business process execution with respect to the actual process specification and characteristics. The monitoring is based on mining and analyzing information from process logs.

Process mining for security [267]

The approach allows for auditing security-critical properties of the process executions. First, the correct model of process is extracted using logs without security violations. Second, the actual executions are verified for conformance to that model using process mining techniques.

Deriving protocol models from logs [180]

The approach to extraction of service interaction protocol models based on monitoring interaction logs is proposed. The approach is semi-automated and provides a way to interact with the designer in order to refine and correct the extracted model.

Timed transition discovery from Web service conversations [71]

The approach targets extraction of behavior model of the service in terms of temporal constraints (timed transitions) between relevant service interactions and events.

Grid Monitoring Architecture (GMA) [247]

GMA is an abstraction of the essential characteristics needed for scalable high performance monitoring on a large distributed computational Grid. It provides the standard specification of the grid monitoring architecture, the components and their roles, the communication models, without, however, defining the implementation.

SCALEA-G [251]

A unified monitoring and performance analysis tool for the grid is presented. The tool provides flexible facilities for the definition and instrumentation of sensors, access to previously monitored information. The information may be collected both at the system (middleware) level and at the application level.

Globus MDS-2, MDS-4 [62, 93]

In Metacomputing Directory Service (MDS) architecture the monitoring information collected by distributed information providers is collected and stored in a dedicated aggregate directory services. The proposed architecture allows for distributed, standardized and easily extendable implementation of grid monitoring, while suffering from serious performance problems. These problems were taken into account and partially resolved in version MDS-4, where the corresponding directory services are re-implemented using Web service standards and solutions.

R-GMA [89]

R-GMA is a relational database implementation of GMA that can be used not only as a monitoring solution, but as a generic information source. It allows for managing monitoring information providers defined as database providers, stream data providers, or providers of historical data. The information queries are expressed in SQL-like notation and collect the historical data, ongoing events, or latest events of certain type.

MonALISA [179]

MonALISA is a monitoring solution based on peer-to-peer Jini platform. The platform is used for dynamic discovery, loading and replication of relevant common information. The data collection is performed by a special engine, which dynamically loads and controls the monitoring modules. The engine allows also for aggregating the previous information and is equipped with a powerful management user interface.

GridICE [9]

GridICE is a multi-layer centralized grid monitoring platform that is capable of observing simple and composite resource metrics. The collected data is made available for consumers by the publisher service, while the notifications, statistics and periodic reports are provided by the notification service.

4.4.3 Analysis

Systematic literature review has been performed to gather the relevant work (cf. [133]).

The basic research question to be addressed by this survey is: "What is the state-of-the-art in analysis of service-based applications?" Examples of subquestions, which have been addressed while considering the field of service-based applications analysis, are:

- Which techniques, mechanisms and methodologies are used for analyze service-based applications?
- Where are the differences between classical analysis approaches and testing of service-based applications?
- Which new approaches or adjustments / adaptations are needed for service-based application analysis?

The sources indicated in Section 1.4 of the introduction have been searched using the search string "(verification OR analysis) AND (service OR SOA)". have been searched using the given search strings. The number of results that have been found are also given. Each search result has been manually filtered to consider only relevant results. Moreover, some top references have been found just by considering previous state-of-the-art chapters written in journal papers.

The relevant references are summarized in the following sub-sections.

Given the very wide field that analysis is, we have chosen to narrow the selection of the survey in this deliverable to these papers which are more related with non-functional properties such as performance prediction, structural complexity of service compositions, etc., while other approaches which lean more towards checking correctness of service compositions are examined in the first deliverable of JRA-2.2.

One pragmatic reason to perform such a split is, of course, to avoid duplicated reviews if at all possible. However, in some cases, this could not be avoided due to the relevance of the work under consideration. However, the different approach taken by these two deliverables can make up for the possible overlapping and, in any case, examine these pieces of work under different, and hopefully complementary, lights.

QoS Analysis with PEPA models [104, 105]

Problem: Both papers analyze the problem of quality specification and analysis of service-oriented architectures by using a model-driven method based on UML.

Explanation: Service-oriented Applications in general must be developed by taking into account both the problem of scalability and security. In this work, both problems are analyzed, but in this summary we focus only in the problem of scalability and thus system performance tuning. During the development of a service-based application, non-functional and performance-related qualities must be considered as an integral part of the development process, in order to integrate functional analysis with performance analysis, and, when it is possible, by automating most of those processes.

Solution: The proposed solution uses well-known UML diagrams such as state diagrams and sequence diagrams, which are suitable for model-driven development. Those diagrams are colored with performance-related characteristics of modeled systems, and they are automatically translated to PEPA, a common used stochastic process algebra. In [105], authors propose the analysis of retrieved PEPA models with the multi-terminal binary decision diagram (MTBDD)-based PRISM stochastic model checker.

Benefits: Main benefits of these approaches include the use of UML-based diagrams, which are easy to extend with performance related quantities; moreover, a model-driven approach allows the direct translation to PEPA models, which are then analyzed with a state-of-the-art stochastic model checker as PRISM.

Validation: The approach is evaluated by illustrating two examples: in [104], the authors validate their approach by using a web-based application example; in [105], the approach is validated on a real-world problem from the domain of mobile telephony.

Complexity analysis of BPEL Web Processes [49]

Problem: The work by Cardoso [49] faces the problem of measuring the complexity of BPEL Web Processes.

Explanation: The complexity of BPEL descriptions, when used for describing WS Processes, can interfere with maintenance, understandability and effectiveness of both the WS-based application and development. Thus, it can be useful to measure the complexity of such descriptions in term of metrics derived by static analysis of the BPEL descriptions. In fact, an high complexity is more error prone, and exposes developers to difficult in maintenance and deployment.

Solution: The approach uses three different static-analysis derived metrics to predict and measure the complexity of a BPEL description. The complexity of a BPEL description is given by the following metrics: control-flow complexity, data-flow complexity and resource complexity. The author exposes ideas about each different metric, but then he focuses on the single control-flow complexity, which takes into account the presence of splits, joins, loops, starting and ending points. The complexity metric is similar by other state-of-the-art metrics for source code, and thus it should be easily interpreted by developers.

Benefits: The main benefit of the approach is the possibility of detecting early increases on the complexity of a BPEL description during the development, even if the complexity measure obtained by the static-analysis computation must be interpreted accurately by developers.

Validation: The approach is evaluated with a motivating example on the field of banking. Examples of different loan processes are provided, as found during their evolution inside the overall application description, leading to higher complexity. This motivating example shows how the approach can be used to detect moments in the development process where a process of re-engineering should be done to improve the maintainability and understandability of the process descriptions.

Performance Modeling of WS-BPEL-Based Web Service Compositions [212]

Problem: The paper analyzes QoS aspects of web service orchestrations built with WS-BPEL descriptions from the point of view of the service integration.

Explanation: Classic QoS analysis of SoA application is applied in the context of this paper in order to select an optimal set of services to orchestrate a WS composition by filling a WS-BPEL description of the process, which is the aim of the service integrator. The non-functional contract between the integrator and the third party service providers is composed of a set of Service Level Agreements (SLAs). The overall problem is the development of a performance analysis and a model for the evaluation of the quality of processes created by using WS-BPEL.

Solution: First, the approach introduces a mathematical model, based on operational research, which describes the performance of composed web service processes written in the WS-BPEL language. The approach introduces a distributed infrastructure which is able to detect if the introduction of a new instance affects (and how) the performance of the web service provider nodes, and detect if SLAs are violated or fulfilled.

Benefits: The approach introduces a well-defined mathematical model to calculate the expected values of performance parameters of WS-BPEL compositions.

Validation: The approach is not evaluated with real examples.

Performance Prediction of Web Service workflows [161]

Problem: The general problem analyzed by the authors is the performance prediction of service-based applications. The issue is faced by considering from both the user and service provider points of view.

Explanation: Web Service based application play a very important role in the general SoA architecture. Moreover, they can be selected and composed in order to create highly complex applications. In this case, the Business Process Execution language (BPEL) can be used to express such compositions and interactions. Although, a very important factor to decide how composition can be instantiated, that is, which actual services must be selected, is the whole performance of the BPEL description. The prediction of the BPEL workflow performance can be also useful to detect if a given composition is able to provide the requirements about the quality of service.

Solution: Authors propose an integrated framework to resolve the issue of performance prediction and assessment of workflows expressed in the BPEL language. The starting point for the prediction is composed of annotated BPEL and WSDL specifications, from which authors derive performance bounds on response time and throughput. Thus, users are able to assess the efficiency of the BPEL workflow, and service providers can, for example, adapt their compositions by estimating performance gains of different upgrades to existing systems.

The formal background used to describe and derive performance bounds is based on the operational laws of Queuing Network Analysis. The bounds can be used to detect and inspect bottlenecks at the system specification level. The approach is suited for resolving a lot performance-related issues without the need for providing too many details, because the bounds can be obtained without deriving the whole performance model.

The approach can be applied both at design time and at runtime. In the first case, the approach allows to select service-based on their expected performance, or to calculate the expected overall system performance. In the latter case, it can be used to reconfigure the system, for example to deal at runtime with changes of users requirements or with modification of the environment. A more complex, detailed and precise methodology can be applied off-line at the whole resource level, to update the performance information published with the services.

Benefits: The main benefit of the proposed approach is that performance bounds can be obtained with little computational effort. Thus, the client is able to answer a lot of common performance-related questions which usually arise during the deployment cycle. It also allows runtime reconfiguration with relatively minimal effort.

Validation: The whole approach is validated to a case study as a motivating example in the area of grid computing. Web Services are executed to perform jobs in a computational grid. A BPEL workflow describes a closed workload on the Grid. In the example, the approach is able to detect that the bottleneck is the network, and this result can be used to improve the overall performance of the system.

Adaptive Service Composition in Flexible Processes [11]

Problem: Authors use static and dynamic analysis techniques to face the problem of modeling and solving the runtime WS selection problem on complex service oriented systems described as business processes in BPEL.

Explanation: The problem of service selection arises whenever complex applications, described as processes invoking services, need to select their composing elements from a set of functionally equivalent services which differ for nonfunctional characteristics (QoS parameters). The problem can be defined as the selection of the best set of available services at runtime. Constraints are both process-related ones, and end-user preferences.

Solution: The approach introduces a complete modeling technique to the WS selection problem at runtime based on integer linear programming (optimization problem). The new modeling approach to the service selection problem is based on diverse contributions:

- static analysis of BPEL by loop peeling, leading to an optimization on the management of loops in BPEL descriptions;
- the use of negotiation whenever a feasible solution to the optimization problem cannot be found;
- a class of global constraints allowing the execution of stateful WS.

Benefits: The main benefit of the approach is the ability to find the optimal solution of the WS selection problem; moreover, the loop peeling is effective when global QoS constraints are strong.

Validation: The model and the optimization algorithm have been tested on a set of randomly generated process instances, in order to detect the benefits and effectiveness of each contribution. Parameters for evaluation include strong constraints on global QoS parameters, which show, for example, the importance and effectiveness of loop peeling with such constraints.

Analysis of Interacting BPEL Web Services [96]

Problem: The problem illustrated and faced by the authors consist of a set of techniques for the analysis of composite WS specified in BPEL.

Explanation: The analysis illustrated in the paper consider interactions of composite web services as conversations, that is, the sequence of messages which have been exchanged by the services. Compositions are described, as usual, with the BPEL language, against which some behavioral properties must be model-checked.

Solution: The approach translates BPEL specifications of composite web services to an intermediate representation, and then to the target verification language Promela, which can be used, together with a LTL property, as input to the SPIN model checker. The intermediate representation used by the authors is guarded automata augmented with unbounded queues for incoming messages; guards are expressed by using XPath expressions, handling rich data manipulation. In order to deal with unbounded input queues for incoming messages, the authors propose a set of conditions which guarantees that a large class of composite web services can be completely verified by using a finite state model checker as SPIN.

Benefits: The approach is able to verify LTL properties on BPEL-specified web service compositions using a well-known model checker as SPIN.

Validation: The approach is validated by using the "Loan processing Example" from the BPEL language specification.

A model checking approach to verify BPEL4WS workflows [40]

Problem: Authors analyze the problem of formal verification of workflow-based compositions of web services specified in BPEL4WS.

Explanation: As in many fields of Software Engineering, the problem of practical formal verification by using model checking can be also used to verify functional properties in SoA based applications.

Solution: The solution proposed by the authors addresses the problem by translating WS compositions described in BPEL4WS to BIR, the source language of Bogor, a state-of-the-art extensible model checker. First, the methodology can be used to verify deadlock freedom from WS compositions. Moreover, additional properties to be verified can be specified by using WS-CoL and LTL. In the first case, WS-CoL allows the predication on variables containing data both inside and outside the process; they can be verified by using assert statements in the BIR language. LTL properties can be verified by using two ad-hoc Bogor extensions.

Benefits: The approach provides a full coverage of BPEL4WS constructs. Moreover, generated models for model checking are compact.

Validation: The evaluation is done by using a suite of workflows coming, for example, from the BPEL4WS standard and from other WS-based applications. Results show improvement in state space on generated models.

Modeling Web Service Orchestration [168]

Problem: This work addresses the problem of regulating distributed service-based workflows evolution.

Explanation: Current network technologies allow the development of new interaction business paradigms, such as virtual enterprises: different companies pool together their services to offer more complex, added-value products and services. Systems supporting such models are commonly referred to as Cooperative Information Systems (CIS's); various approaches are proposed for the design and development of CIS's: schema and data integration techniques, agent-based methodologies and systems, business process coordination and service-based applications. By using a service-based approach, the cooperative system consists of different distributed applications which integrate the E-services offered by different organizations. Such integration raises issues regarding service composability, correctness, synchronization and coordination.

Solution: [168] proposes PARIDE framework (Process-based framework for oRchestratIon of Dynamic E-services) to define a common conceptual component model (and the related description language) for E-services, and the notions of compatibility and dynamic substitution of E-services based on the concept of cooperative process. PARIDE framework was experimented within the VISPO (Virtual-district Internet-based Service PlatfOrm) Project. VISPO has the goal of allowing participation in the activities of the district by invoking a series of available E-services according to the rules defined in the orchestration schema. Orchestration engines allow controlling the correct evolution of the process, and alerting the current process responsible if exceptions are raised. PARIDE adopts a Petri Net-based model to ensure the description of the orchestration of E-services, and the related design of distributed orchestration engines. Besides, it provides analysis techniques based on the Petri Nets in order to address specific issues such as deadlocks, possible timeouts, configuration reachability, etc.

Benefits: The E-service orchestration model proposed in this paper provides a mechanism for supporting control of process evolution in terms both of control and data flows, and for distributing and assigning process responsibilities.

Validation: An example stemming from the "Italian E-government" scenario is used to demonstrate the application of the approach to the service composition problem.

Workflow Verification [266][4][210]

Problem: In these papers Aalst et al. address the problem of the verification and the analysis of service-based workflows. The work particularly focus on analyzing soundness and conformance properties that any workflow process should satisfy.

Explanation: Service-based workflow systems facilitate the everyday operation of business processes by taking care of the logistic control of work. In contrast to traditional information systems, they attempt to support frequent changes of the workflows at hand. Therefore, the need for analysis methods to verify the correctness of workflows models and the conformance of the execution traces to these models are becoming more prominent.

Solution: Based on a Petri-net-based representation of workflows, Aalst in [266] provides techniques to verify soundness property, e.g., a workflow is sound if and only if, for any case, the process terminates properly, i.e., termination is guaranteed, there are no dangling references, and deadlock and livelock are absent. The correctness of a process can be decided by partitioning the workflow into sound subprocesses. A Petri-net-based workflow analyzer called Woflan is proposed to support the application of the approach. In [210], Aalst et al. are interested in providing answers to conformance problem due to the coexistence of event logs and process models of business workflows. They use Petri nets to model processes; this is argued by the fact that Petri nets are formal and have associated analysis techniques to easily parse any event log. [210] shows that conformance has two dimensions: fitness (the event log may be the result of the process modeled) and appropriateness (the model is a candidate from a structural and behavioral point of view). Metrics measuring fitness and appropriateness are supported by the Conformance Checker, a tool which has been implemented by the ProM Framework. [266] [4] [210]

show benefits of the use of Petri nets for specifying and analyzing business workflows, but they don't treat these issues in cases of distributed service-based applications.

Benefits: The results presented in [266] give workflow designers a handle to construct correct workflows. [266] proposes a workflow analyzer, based on standard Petri-nets-based analysis tools, which can be used by people not familiar with Petri-net theory. This workflow analyzer interfaces with existing workflow products such as Staffware, COSA, METEOR, and Protos. Authors, in [210], provide definition of conformance. They propose metrics to measure this property. These metrics are supported by a tool called Conformance Checker.

Validation: The applicability and advantages of these approaches are demonstrated through a set of case studies.

Modeling and Model Checking Web Services [225]

Problem: Schlingloff et al. address the problem of checking correctness of composite web service processes.

Explanation: In this work, the original goal of modeling BPEL processes with Petri nets is to give the language BPEL4WS a formal semantic, and to compare the applicability of several formalisms for this task (e.g. Abstract State Machines).

Solution: The work described in [225] shows how to build Petri net models of web services formulated in the BPEL4WS specification language. The main goal is to define an abstract correctness criterion, called usability and to study the automated verification according to this criterion. This work relates correctness of web service models to the model checking problem for alternating temporal logics.

Benefits: This approach provides results concerning computer aided usability analysis. The analysis covers scenarios of central as well as distributed usability.

Validation: No tool is developed to concretize this approach and no case study is provided to show the applicability of the work.

Model-Checking Verification for Reliable Web Service [175]

Problem: The author is interested in how much the software model checking technique can be used as a basis for raising reliability of Web service, Web service flow descriptions in particular.

Explanation: Model checking is a technique for the verification and validation of software systems. This technique is applied to software requirement specification and design specification and aims to increase the reliability and productivity from early stages of the software development. In this work, the author attempts to test model checking technique in the case of distributed service-based applications.

Solution: SPIN model-checker is used in [175] to verify a set of properties related to business flows, described by the WSFL workflow. SPIN provides a specification language Promela that describes the target system to be a collection of Promela processes (automata) with channel communications. The flow description written in WSFL is translated into Promela, the input specification language of SPIN. The properties to be checked are reachability, deadlock-freedom, or application specific progress properties. The application specific properties are expressed as formulas of LTL (Linear Temporal Logic), which are also fed into SPIN.

Benefits: The experience shows that faulty flow descriptions can be identified with model checking technique. This technique is also very helpful in studying alternative semantics of the WSFL in regard to the handling of dataflows.

Validation: A simple case study is used for demonstrating the usefulness of the model checking technique in the case of distributed service-based applications.

Modeling and Verifying Web Service Orchestration by means of the Concurrency Workbench [135]

Problem: In this work, authors address the problem of how to exploit verification techniques like model checking, preorder checking and equivalence checking to model and verify web service orchestrations.

Explanation: The Concurrency Workbench (CWB) is a generic and customizable verification tool. The CWB supports model checking, preorder checking and equivalence checking. Originally, the CWB was designed for the verification of the Calculus of Communicating Systems (CCS). However, the CWB can be customized to support languages other than CCS. To support a new language, a considerable number of modules need to be implemented. This would be a time consuming task if not for the Process Algebra Compiler (PAC). The PAC is a tool that generates these modules from a specification of the syntax and semantics of the language. In the work presented in [135], authors show how the CWB and the PAC can be exploited to model and verify web service orchestration.

Solution: [135] exploits CWB and PAC to model and verify web service orchestration. A new calculus, named the BPE-calculus and based on BPEL4WS, expresses web service orchestration. The operational semantics of BPE-calculus is used as input of the PAC to produce modules for the CWB. The latter provides a powerful verification tool for BPE-processes supporting model checking, preorder checking and equivalence checking.

Benefits: Authors introduce a new calculus, named the BPE-calculus, that contains the main control flow constructs of BPEL4WS. They propose a verification tool, based on CWB and PAC, for BPE-processes.

Validation: Authors present an empirical evaluation of the implemented tool.

Specification-Based Verification and Validation of Web Services and Service-Oriented Operating Systems [260][117]

Problem: These papers examine the problem of collaborative verification and validation of distributed service-based applications.

Explanation: Current web services testing techniques assume that unit testing has been adequately performed by the web service providers. This assumption does not hold if trustworthy services need to be composed based on the searched and found web services over the Internet. Limiting the sources of web service providers breaches the idea of the open platform on which web services are based upon. These papers examine the importance of dynamic web services unit test, which is significantly different from software unit testing due to the unavailability of source code and the runtime feature.

Solution: [260] examines the importance of dynamic web service unit test, which is significantly different from software unit testing due to the unavailability of source code and the runtime feature. This paper proposes three techniques to perform unit testing: specification-based test case generation, collaborative testing, and group testing, which can enforce the trustworthiness of the service-based application components before they are integrated into the new composite web service.

[117] is a continuation of the work presented in [260]. [117] proposed an integrated process to automatically translate OWL-S specification of composite web service into a C-like specification language. C-like specification is processed by BLAST model checker. BLAST performs model checking of composite web service, generates positive and negative test cases during model checking, and tests the web service using the test cases. The existing techniques applied in this integrated process do not meet the requirements of composite web service verification and testing. So, enhancement in OWL-S and PDDL are proposed to better facilitate test case generation.

Benefits: The main benefits of the work presented in [117] are: a) Applying BLAST to WS and evaluating atomic WS while previous approaches treat atomic WS as a black box. b) Extension of BLAST to handle concurrency in precise OWL-S semantics. c) Extension of OWL-S and PDDL (Planning Domain Definition Language) to better facilitate both positive and negative test case generation.

Validation: No validation is provided to show the applicability of the work.

Model Checking with Abstraction for Web Services [229]

Problem: This article addresses a problem of verifying the applications that implement the web services and develops techniques for modeling and verification of low level languages used for the implementation of web services. Particularly, authors addresse the state explosion problem of model checking techniques. They propose to use abstraction data techniques to face this problem. They apply this solution in the case of distributed service-based applications.

Explanation: Web-services are highly distributed programs, and thus, prone to concurrency related errors. Model checking is a powerful technique to identify flaws in concurrent systems. However, the existing model checkers have only very limited support for the programming languages and communication mechanisms used by typical implementations of web services. This work presents a formalization of communication semantics geared for web services, and an automated way to extract formal models from programs implementing web services for automatic formal analysis. The formal models are analyzed by means of a symbolic model checker that implements automatic abstraction refinement.

Solution: SatAbs [229] is a tool enabling analysis of service-based applications. It relies on model checking techniques to identify eventual flaws in such concurrent systems. [229] formalizes the semantics of a PHP-like language and enables modeling of both synchronous and asynchronous communication between services. The resulting models are amenable to verification using model checking.

Benefits: The implementation proposed in [229] is able to show safety properties of a combination of multiple PHP scripts running in parallel. Most steps of the analysis loop are done in a thread-modular manner, and thus, the analysis scales in the number of threads.

Validation: No validation is provided to show the applicability of the work.

Transforming BPEL into annotated Deterministic Finite State Automata for Service Discovery [276]

Problem: The problem illustrated and faced by the authors is service discovery for BPEL which requires adequate formal model to support matchmaking of state dependent services.

Explanation: Web services advocate loosely coupled systems, although current loosely coupled applications are limited to stateless services. The reason for this limitation is the lack of a method supporting matchmaking of state dependent services exemplarily specified in BPEL. In particular, the sender's requirement that the receiver must support all possible messages sent at a certain state are not captured by models currently used for service discovery.

Solution: The work described in [276] presented the concept of matching business processes in loosely coupled architectures. It proposes a transformation from BPEL to annotated Deterministic Finite State Automata aDFA. The transformation represents messages that might be sent by a party at a particular state as messages that must be supported by the corresponding receiving party. This explicit modeling of mandatory transitions of a sender and optional transitions of a receiver is the main contribution of this approach.

Benefits: Service discovery is a hot research issue. In particular, a lot of work is dedicated to define service descriptions addressing specific aspects required for service discovery. All the proposed approaches extend the currently established service repository being based on UDDI and none of them has addressed the issue of process annotation as presented in this work.

Validation: The validation is done by considering a simple case study.

Specification and Validation of the Business Process Execution Language for Web Services [81]

Problem: This work addresses the problem of enriching business process models specified with BPEL4WS specification with operational semantics.

Explanation: This work formally defines an abstract operational semantics for the BPEL language. Formalization of language semantics serves two main purposes: a) To eliminate deficiencies hidden in

natural language descriptions, for instance, such as ambiguities, loose ends, and inconsistencies. b) To establish a platform for experimental validation of key language attributes by making abstract operational specifications executable on real machines.

Solution: An abstract operational semantics for BPEL4WS in terms of a real-time distributed abstract state machine DASM is proposed in [81]. The BPEL abstract machine is organized into three basic layers reflecting different levels of abstraction. The top layer, called abstract model, provides an overview and defines the modeling framework comprehensively. The second layer, called intermediate model, specifies the relevant technical details and provides the full DASM model of the core constructs of the language. Finally, the third layer, called execution model, provides an abstract executable semantics of BPEL. By analytical means, i.e. the ability to formally reason about critical language properties, and experimental validation, i.e. through simulation and testing, several coherence and consistence properties (e.g. correlation and synchronous receive/reply messages) are checked in order to improve the quality of the language definition.

Benefits: The proposed hierarchical structuring of the BPEL abstract machine into three layers reflects a clear separation of concerns, enhances intelligibility, and enables a tighter integration of the formal and the informal language description so that they effectively complement each other.

Validation: For representing the approach a case study from the e-commerce domain is applied.

Compatibility Verification for Web Service Choreography [90]

Problem: In [90], authors address the problem of verifying process interactions for coordinated web services composition.

Explanation: Web Service workflow languages aim to fulfil the requirement of a coordinated and collaborative service invocation specification to support long running and multi-service transactions. Amongst the key issues in the design and implementation of components in this architecture style for critical business applications, is the formation of compositions as a series of interacting workflows and how transactions of activities interact to support the underlying business requirements. This issue is collectively wrapped up in the term Web Service Choreography. These interacting workflows can be constructed using various emerging standards and managed by multiple parties in the domain of interest and as such the task of linking these activities across workflows within this domain is crucial. This work considers the issues in web service choreography by expanding on earlier work discussing modeling local web service compositions [91] which focused on the analysis and verification of behavior models of web service compositions implemented in the BPEL4WS specification.

Solution: This work proposes to use finite state machine representations of web service orchestrations to analyze process interactions of web service compositions. The aim of this analysis concentrates on the compatibility of processes that take part in the complete composition environment. With an analysis process, the implementers of the composition can be assured that interaction will be compatible and therefore match the requirements of interaction. Using the Labeled Transition System Analyzer LTSA, three levels of compatibility for component compositions are analyzed: a) Interface Compatibility: focusing on semantics of correlating invocations against receiving and replying messages between partner processes. b) Safety Compatibility: assurance that the composition is deadlock free and is checked against partial correctness of transitions. c) Liveness Compatibility: assurance against starvation of progress (that the service process eventually terminates) and that messages received are served on a first-come-first-served basis.

Benefits: The proposed approach is supported by a suite of cooperating tools for specification, formal modeling and providing verification results from orchestrated web service interactions.

Validation: To prove the effectiveness of the approach proposed in [90], a case study from the e-commerce domain is applied.

Modeling Component Connectors in Reo by Constraint Automata [19]

Problem: Authors, in this work, address the problem of the coordination in composing web services.

Explanation: Coordination models and languages close the conceptual gap between the cooperation model used by the constituent parts of an application and the lower-level communication model used in its implementation. The current interest in constructing applications by combining existing software components necessitates attention to the so-called glue-code. Using components, thus, means understanding how they individually interact with their environment, and specifying how they should engage in mutual, cooperative interactions in order for their composition to behave as a coordinated whole.

Solution: [19] introduces constraint automata as a formalism to describe the behavior and possible data flow in coordination models that connect anonymous components to enable their coordinated interaction. Constraint automata are used as an operational model for Reo, an exogenous coordination language for compositional construction of component connectors based on a calculus of channels. Constraint automata make modeling subtle timing and input/output constraints of Reo connectors possible, specifically their combined mix of synchronous and asynchronous transitions.

Benefits: The theory of constraint automata yields a rich basis for formal verification of coordination mechanisms.

Validation: Authors present simple examples to prove the applicability of the proposed approach. The efficiency of this work is proven theoretically.

A Model-Checking Verification Environment for Mobile Processes [86]

Problem: In this work, Ferrari et al. study the problem of the verification of the behavior of mobile systems.

Explanation: A global computing system is a network of stationary and mobile components. The primary features of a global computing system are that its components are autonomous, software versioning is highly dynamic, the network's coverage is variable and often its components reside over the nodes of the network (WEB services), membership is dynamic and often ad hoc, without a centralized authority. Global computing systems must be made very robust since they are intended to operate in potentially hostile dynamic environments. This means that they are hard to construct correctly and very difficult to test in a controlled way. In this area, formal analysis techniques and the corresponding verification technologies are important to gain confidence in correct behavior and to weed out bugs and security hazards before a system is deployed.

Solution: [86] exploits History Dependent automata HD-automata as a basis for the design and development of verification toolkits for reasoning about the behavior of mobile systems. A verification environment, called HD-Automata Laboratory HAL, is used to exploits HD-automata of systems specified in the π-calculus. The HAL environment includes modules that implement decision procedures to calculate behavioral equivalences; it includes as well modules that support verification by model-checking of properties expressed as formulae of temporal logics. The construction of the model-checker takes direct advantage of the finite representation of π-calculus specifications. HAL checks liveness and safety properties.

Benefits: The distinguishing and innovative feature of this approach is that the translation mapping is driven by the finite-state representation of the system (the π-calculus process) to be verified.

Validation: To illustrate the effectiveness and usability of this approach, Ferrari et al. consider case studies that allow them to demonstrate some common verification patterns that arise frequently when reasoning about p-calculus specifications.

Describing and Reasoning on Web Services using Process Algebra [223]

Problem: In this work, Salan et al. study the problem of modeling and verifying web service composition using a Process Algebra.

Explanation: Web services are an emerging and promising area involving important technological challenges. Some of the main challenges are to correctly describe web services, to compose them adequately and/or automatically, and to discover suitable services working out a given problem. In this work, authors propose a framework to develop and reason about web service compositions at an abstract level.

Solution: The approach proposed in [223] uses Process Algebra called CCS as an abstract representation means to describe, compose and reason (simulation, property verification, correctness of composition) on service-based applications. The techniques, used to check whether a service-based application described in process-algebraic notations respects temporal logic properties (e.g. safety and liveness properties), are referred to as model checking methods.

Benefits: The strength of this approach is to work out all issues related to the description, the composition and the reasoning on web services at an abstract description level, based on the use of existing approaches and tools, while keeping a two-way connection with the application layer.

Validation: A case study is used to illustrate how Process Algebra can be used to specify concrete web service composition problems and to reason about such interacting processes using existing verification tools.

LTSA-WS: A Tool for Model-Based Verification of Web Service Compositions and Choreography [92]

Problem: In this work, Foster et al. discuss the problem of verifying web service compositions and choreography.

Explanation: Web service composition languages such as the BPEL4WS aim to fulfill the requirement of a coordinated and collaborative service invocation specification to support running transactions and multi-service scenarios. However, a composition alone does not fulfill the requirement of an assured collaboration in cross-enterprise service domains. Participating services must adhere to policies set out to support these collaborative roles with permissions and obligations constraining the interactions between services. This issue is wrapped up in the term Web Service Choreography. The design and implementation of service components in this interaction style must support the original policies as defined by the service owners. Therefore, of clear interest is the need of process verification of web service choreography.

Solution: LTSA-WS is a tool implementing a model-based approach to verifying service-based applications. This tool supports verification of global properties (e.g. absence of deadlock and liveness) created from design specifications and implementation models to confirm expected results from the viewpoints of both the designer and implementer. Scenarios are modeled in UML, in the form of Message Sequence Charts, and then compiled into the Finite State Process FSP process algebra to concisely model the required behavior. BPEL4WS implementations are mechanically translated to FSP to allow an equivalence trace verification process to be performed.

Benefits: The LTSA-WS plug-in and specifically the version supporting BPEL4WS, is being considered for use on a number of medium sized case studies.

Validation: To evaluate this tool, a series of case studies are tested.

Formal Verification of Web Service Composition [209]

Problem: In this work, authors address the problem of analyzing and verifying web service compositions.

Explanation: Current Web services composition proposals, such as BPML, BPEL4WS, WSCI, and OWL-S, provide solutions for describing the control and data flows in Web service composition. However, such proposals remain at the descriptive level, without providing any kind of mechanisms or tool support for analysis and verification.

Solution: The work presented in [209] proposes an event-based approach for checking both functional and non-functional requirements of web service compositions. The properties to be monitored are

specified using the Event Calculus formalism. Functional requirements are initially extracted from the specification of the composition process that is expressed in WSBPEL. This ensures that they can be expressed in terms of events occurring during the interaction between the composition process and the constituent services that can be detected from the execution log. Non-functional requirements (e.g. such as security policies) to be checked are subsequently defined in terms of the identified detectable events by service providers.

Benefits: Using events is very attractive. Main advantages are: a) Business processes leave their business events in event logs. b) It exists various work for checking event-based specifications consistency.

Validation: For representing the approach, a case study from the e-commerce domain is applied.

Execution Semantics for Service Choreographies [66]

Problem: This work discusses the problem of analyzing web service choreographies.

Explanation: A service choreography is a model of interactions in which a set of services engage to achieve a goal. Choreographies have been put forward as a starting point for building service-oriented systems since they provide a global picture of the system's behavior. In previous work we presented a language for service choreography modeling targeting the early phases of the development lifecycle. This work proposes a service interaction modeling language and provides execution semantics for this language in terms of a mapping to p-calculus. This formal semantics provides a basis for analyzing choreographies.

Solution: [66] presents a new approach that proposes to define a service interaction modeling language as well as techniques for analyzing and relating global and local models of service interactions. This work introduced a formal semantics for a service interaction modeling language, namely Let's Dance, which supports the high-level capture of both global models (i.e. choreographies) and local models of service interactions. The semantics is defined by translation to π-calculus.

Benefits: Let's Dance, which supports the high-level capture of both global models (i.e. choreographies) and local models of service interactions.

Validation: A simple composition is used to illustrate the approach. The efficiency of this work is proven theoretically.

Integrating Business Requirements and Business Processes [128, 127]

Problem: The work address the problem of augmenting business process models with the models of strategic goals and requirements, and of verifying the compliance between these models.

Explanation: In order to support continuous changes and adaptation of business process models to business goals and requirements it is necessary to explicitly relate the strategic goals and requirements to the business process models. The ability to formally analyze how the changes in the requirements affect the process models enables the requirements traceability and improves the reliability of the system through its continuous evolution.

Solution: The proposed solution is to adopt the explicitly associate the strategic models representing business requirements and goals to the business process models and then provide a formal analysis support for the verification of their compliance. For modeling business requirements of Web service compositions the authors adapt the Tropos methodology, that provides notations to capture the goals of the participants, their mutual dependencies and expectations. The Formal Tropos language is then used for the formal definition of these requirements and goals. The requirements specifications defined in Tropos are then integrated into the business process specifications, represented in BPEL, in the form of activity annotations.

In order to perform formal analysis of these models, the requirements specifications and the annotated process models are translated into an internal unified representation, and then the analysis is performed using model checking techniques.

Benefits: Explicit integration of the strategic requirements models with the business process models and the subsequent formal analysis of their compliance allows for the requirements traceability along the evolution of the service-based application.

Validation: For representing the approach a case study from the e-health domain is applied. The authors also present the results of the experimental evaluation of the analysis at different phases of the approach.

WS-VERIFY: Formal Analysis of WS Compositions [130, 129, 126, 123]

Problem: In this work, Kazhamiakin et al. address the problem of the verification and analysis of Web service compositions defined as a set of stateful behavioral models against various functional requirements.

Explanation: The necessity to detect requirements violations and problems in the specifications of the composition behavior is an important issue in the service-oriented design and requires high level of formalisation and automation support. Such analysis, however, has to tackle several problems specific to service composition behavior. First, the service communications are essentially asynchronous and rely on complex message management systems. Second, the service specifications extensively use complex data structures and operations. Third, the correctness of a wide class of systems strongly relies on a compliance of the time-related composition requirements.

Solution: In [130] a unified framework for the formal verification of Web service compositions. The framework integrates the methods and techniques for modeling and analyzing specific aspects, such as asynchronous interactions [129], data-flow properties [126] and qualitative/quantitative time characteristics [123].

The framework relies on a formal model, where (i) each service is defined as state transition system that represents the control flow of the stateful service execution, data evolution and manipulation, and time characteristics of the performed activities; (ii) services interact with each other through a parametric communication model which allows for capturing literally any adopted distributed messaging systems; (iii) temporal logics are exploited for the specification of various behavioral requirements.Based on this formal model, a set of novel analysis approaches that deal with the issues specific to the Web service domain are proposed. First, the authors present a technique to associate with a Web service composition the most adequate communication model, i.e., the one that is sufficient to correctly capture all possible behaviors of the composition, while being as efficient as possible in the analysis. Second, an analysis approach that takes into account the data flow among the component processes is presented. The approach exploits abstraction techniques for modeling only relevant data flow aspects. The approach allows not only for the verification of data-enriched specifications, but also for dealing with the data incompleteness. Third, the authors present a timed analysis approach that allows for the representation, verification of simple and complex time commitments and composition requirements, and for the computation of duration bounds where these properties are satisfied, using the specific duration calculus formalism.

The underlying realization of the presented approaches relies on the model checking techniques. The approaches are implemented and integrated in a WS-VERIFY analysis toolkit that allows for automated verification of the service compositions.

Benefits: The possibility to model and automatically verify service composition specification taking into account the specific aspects of the service-based applications allows for considerable improvement of the composition correctness.

Validation: Various approaches and techniques defined within the framework are implemented in a formal analysis toolkit. The applicability and advantages of these approaches is demonstrated through a set of case studies from different domains. The corresponding experimental results show the effectiveness of the analysis.

Choreography Analysis [124, 125]

Problem: The work address the problem of the formal analysis of the service choreography specifications, in particular the problem of the choreography realizability and the implementation conformance.

Explanation: Service choreographies aim at representing global observable behavior of the collaborating services. The realizability problem addresses the possibility to "project" the choreography on the participants so that the behavior of the projection composition is guaranteed to correspond to the original choreography. The conformance analysis instead aims at checking whether the observable behavior generated by the service implementations corresponds to the choreography specification. These complementary forms of analysis are equally necessary for the correct implementation of the Web service collaborations.

Solution: The presented solution relies on a formal model that allows for defining both the prescribed choreography behavior and the behavior of the composition of the implementing services. The formalism is given in terms of state transition systems communicating through a certain model of message queues. In order to address the realizability problem, the authors present a hierarchy of realizability notions that allow one to efficiently analyze whether the given choreography can be implemented and under which conditions. In order to address the problem of conformance, the authors formally define the notion of conformance relation between the choreography and the composition implementation, extended with a set of constraints on the information alignment between partners. The corresponding analysis algorithms are presented.

Benefits: The choreography realizability analysis is an important activity in the design of reliable service compositions. The conformance analysis additionally allows one to check at run time whether the new services may participate to the composition without violating the prescribed choreography specification.

Validation: The approaches are demonstrated on a set of simple scenarios, without however practical implementation and empirical evaluation.

Petri Net-based Analysis of WS-BPEL processes [191, 190, 269]

Problem: The authors address the problem of the behavioral verification of the composed services defined in BPEL language, in particular, the deadlock analysis, detecting non-viable parts, or concurrent message consumption.

Explanation: BPEL language is de-facto standard for implementing executable business processes on top of Web services. The necessity to carry out verification activities before the process deployment requires (i) correct and unambiguous language formalization and (ii) the techniques for the efficient and automated checking of relevant correctness properties.

Solution: The presented approach defines a rigorous and detailed scheme for the formalization of BPEL structures. For this purpose a special class of Petri nets, called workflow nets, is adopted. In this way the formal semantics of BPEL is defined. Based on this formalism, the authors define two tools that in combination allow for the automated verification of BPEL. The BPEL2PNML tool translates the BPEL process into the Petri Net Modeling Language (PNML), and the resulting model is analyzed with the WofBPEL tool. Within the approach the following forms of analysis have been defined: detection of unreachable BPEL activities, detection of multiple concurrently enabled activities that may consume the same type of message, and determining for every reachable state of the process, which message types may be consumed in the rest of the execution.

Benefits: The presented approach defines the most complete and well-formed semantics of BPEL among all the work devoted to the analysis of BPEL processes. The presented correctness properties enable some important forms of analysis and optimization techniques.

Validation: The viability and importance of the presented analysis technique is motivated and demonstrated on a comprehensive case study.

Verification of WS Compositions with Service/Resource Nets [242]

Problem: In [242] Tang et al. target the analysis of the Web service composition workflows. In particular, the concurrent resource access and linear time analysis problems are studied.

Explanation: In order to design reliable service compositions, techniques and tools for their formal analysis are required. In order to adequately describe the composition model, the underlying formalism should be able to represent time properties, resources (i.e., piece of data or even a service), and conditions. Given these components, it is possible to carry out important forms of analysis. In particular, it is possible to see whether the same resource is being concurrently accessed by several activities, or to infer the duration of the composition process.

Solution: In their work the authors introduce a special Petri Net-based formalism, called Service/Resource Net (SRN) that is able to capture not only the control flow of the composition workflow, but also other relevant aspects of the composition, such as time bounds of activities, resources (different data variables or services), and conditions. The authors also show how the service composition may be represented in this formalism. When the composition is translated into the SRN model, and the target net is simplified it is possible to perform traditional analysis methods applicable to Petri Nets. Beyond these methods, it is possible to perform certain specific analysis techniques, namely resource matching and linear temporal inference. Resource matching problem amounts to the analysis of situations when several subflows of the composition model attempt to access the same resource simultaneously. It is shown how to address the problem using bigraph maximal matching algorithm. Linear temporal inference amounts to the computation of the time bounds for the whole workflow based on the time constraints specified for its subactivities. A set of specific rules are introduced in order to perform the computation.

Benefits: The ability to deal with resources, time, and conditions increases the expressiveness of the composition representation, and as a consequence improves the analysis capabilities.

Validation: A simple case study is used to demonstrate the application of the approach to the service composition problem.

CP-Net Based Verification for WS Compositions [280]

Problem: In [280] Yi and Kochut address the verification of the service composition behavior with a particular focus on the problem of protocol conformance, that is "when a partner service with a complex conversation protocol is incorporated in a larger composite Web service, the protocol must be enforced properly."

Explanation: In Web service composition the integration of different services should be done efficiently and correctly. One of the key issues here is to provide a precise and reliable way of integrating conversation partner into the composition specification. Accordingly, the composition design framework should provide the means for the correctness analysis of such integration.

Solution: The authors present a design and verification framework for service compositions based on the Coloured Petri Nets (CP-Nets) formalism. Given its expresiveness, the formalism allows for the representation of data types and data manipulations, as well as the concurrency issues and interprocess synchronization. The proposed formal model enables specification of both the complex conversation protocols of single partners and the overall composition model.

The proposed approach suggest an incremental and iterative design of service composition starting from the conversation protocol of the partners. Besides the generic CP-Net-based forms of analysis (boundedness, liveness, reachability), it is possible to perform the analysis of the protocol conformance. When the resulting model is verified, the corresponding business process specification (in BPEL) may be automatically derived from the CP-Net model.

Benefits: The possibility to verify the protocol conformance problem at design time is an important property for the design of reliable service compositions.

Validation: A simple case study is used to represent the formalism and the approach.

Web Service Interfaces [37]

Problem: In [37] Beyer et al. address the problems of checking service compatibility (two or more services are compatible within a composition) and service substitutivity (one service can be safely replaced with another).

Explanation: A Web service often depends on other Web services, which have been implemented by different vendors, and their correct usage is governed by rules. Such rules may constrain data types and service signatures, but they may also express temporal constraints on the service invocations. In order to verify/enforce these rules, specific forms of analysis are necessary.

Solution: The presented solution proposes an approach to verify that within a composition one service can correctly collaborate with another or may be substituted by another service according to the predefined set of rules. In order to specify these rules, a special formalism, Web service interface language, is presented. The interface defines three kinds of rules on the Web service use: (i) it specifies the service signature rules (methods and their types of input / output parameters); (ii) consistency rules, i.e., propositional constraints on method calls and output values that may occur in a Web service conversation; and (iii) protocol rules, i.e., temporal constraints on the ordering of method calls. For each kind of rules a specific logic-based notation is presented and formalized.

The problem of the interface compatibility and substitutivity is defined separately for each of the interface kind, as well as the corresponding inference rules and algorithms for their checking. This permits an incremental analysis approach, where the verification starts with simpler signature constraints and ends with the complex protocol rules.

Benefits: The possibility to perform compatibility and substitutivity checks is an important analysis capability both at design-time and at run-time when the services are discovered and bound. The use of interface constraints instead of complete implementation descriptions simplifies the analysis and allows for more efficient algorithms.

Validation: The approach is explained with a simple service composition scenario. Its efficiency is proven theoretically.

Semantic Based Verification of BPEL4WS Abstract Processes[77]

Problem: In [77] Duan et al. discuss the problem of correctness analysis of abstract protocols that define distributed service-based processes, and the possibility to synthesize the complex process model from a set of elementary tasks.

Explanation: Abstract processes represent the behavioral interface of a composition component and therefore defines the restrictions on the component use. The abstract process corresponds to a workflow, where elementary tasks correspond to the basic Web service calls. When the preconditions and effects of the elementary tasks are known it is possible to check whether the whole process is compatible with this information, and, moreover, construct from elementary tasks new abstract workflows satisfying the required properties.

Solution: The presented solution relies on a logical model, which allows for associating preconditions and effects with the atomic service calls and with the complex control flow constructs. The preconditions and effects are given as boolean expressions over state propositions that hold in source and target states of the process respectively. Given a process model and a set of semantic annotations of the elementary tasks, it is possible to infer the precondition and effect of the whole process.

The proposed analysis approach consists of (i) recursive inference of the strongest assertions at every control point of the process, and (ii) checking whether the task preconditions are compatible with these assertions. The overall workflow is correct when the expected postcondition is compatible with the inferred one.

Benefits: The analysis of abstract processes allows one for reasoning on the composition correctness regardless implementation details of the underlying executable processes.

Verification of Data-Driven Web Services[70]

Problem: In [70] Deutch et al. study the problem of verification of the compositions of Web Service peers which interact asynchronously by exchanging messages. The correctness properties are given in the form of Linear Temporal First-Order Logic and Conversation Protocols.

Explanation: The behavioral verification of service composition problem is a complex analysis problem due to various features of the underlying service models, such as asynchronous and lossy interactions, data, object cardinality. These features may seriously restrict the applicability of any analysis techniques and should be carefully studied.

Solution: In this work the authors present a formalism for specifying compositions as a set of asynchronously interacting Web service peers. Each peer is represented with a local database, a control flow model, input output queues, and reaction rules. The composition The reaction is described by queries over the database, internal state, user input and received messages. The composition model is parametric with respect the queue bounds, message loss, openness, etc. The correctness properties of the composition have the form either of temporal first order logic sentences (e.g., ordering constraints on the action execution) or Büchi automata representing the expected conversation protocol.

Using on the proposed formalism, the authors discuss its expressiveness and the complexity of the verification problem under different variants of the above parameters. They also identify syntactic restrictions on the peer and property specifications which, under appropriate communication semantics, guarantee an acceptable decidability of verification, and show that the restrictions are quite tight: even slight relaxations thereof lead to undecidability.

Finally the authors discuss modular verification, where the correctness is checked when the complete specification of other peers is not available or partial. Modular verification is useful when some peers are provided by autonomous parties unwilling to disclose implementation details, or when verification of a partially specified composition is desired.

Benefits: The presented formal framework allows for identifying a practically appealing and fairly tight class of specifications for which verification is efficiently decidable. Based on this formalisation it is possible to devise an underlying implementation platform and engineering approaches, where the verification becomes feasible thus improving the correctness of the system.

A Logic-Based Verification of Contracts in Semantic Web Services[64]

Problem: In [64] Davulcu et al. address the problem of automated verification of contract compliance between services taking into account dynamic aspects of the service and contract descriptions.

Explanation: Service contracts describe mutual expectations and commitments of the interacting participants. The dynamic aspect of contracts amounts to the workflow models of the interaction protocols seen from the point of view of each participant. The problem of automated contracting deals with the analysis of compliance between the expectations and commitments between the partners and therefore the verification of the compatibility of the workflow models.

Solution: In order to capture the dynamic aspects of contracting the authors propose a logic called CTR-S. The logic permits representation of the composition participants in terms of workflow, while the desired properties (expectations) are represented as constraints. The automated contracting is therefore a problem of verifying that the expectation is enforceable by the workflow model. A corresponding model and proof theories for capturing the workflow constructs (sequence, concurrent execution, external/internal choices, etc.) and execution constraints (single or serial occurrence of events, their arbitrary nested logical combinations) are developed and represented. Finally, the authors present the reasoning algorithms for the automated verification of the workflows against expectation constraints.

Benefits: The possibility to model and automatically verify complex contracting properties and constraints is important for the dynamic service composition, where the services are discovered and bound at run-time.

Validation: A simple case study is used for demonstrating purposes. The complexity of the analysis algorithms are formally proven.

Automated Model Checking and Testing for Composite Web Services [117]

Problem: In [117] Huang et al deal with the problem that existing model checking approaches treat atomic WSs as black-boxes.

Explanation: The process-oriented models of model checking approaches successfully capture the temporal properties among atomic WSs. However, if the internal structure of each atomic WS is blank in the model specification, then it is inherently hard to describe and check more delicate properties involving the effect and output of each atomic WS.

Solution: The presented solution consists of the following steps: a) use OWL-S to bound the behavior of atomic WSs; b) convert OWL-S to the C-like language of BLAST [113]; c) enhance BLAST for checking concurrency in OWL-S; d) embed data-bound and safety temporal properties in the C-like code; e) use BLAST for model-checking and positive test case generation; f) use of the typological algorithm of [258] and the positive test cases as input for the generation of the corresponding negative test cases. For converting OWL-S to the C-like code, firstly OWL-S's structural constructs are converted and then the PDDL2.1 logical formulas used for expressing conditions in these constructs. Fortunately, most constructs had natural correspondence to the C-like language except from *split*, *split + join*, *choice* and *unordered* constructs, which were taken care of in a special way. For checking OWL-S concurrency, BLAST was enhanced to cope with the synchronization semantics of *split + join* by exhausting every interleaving of the concurrent threads that matter. This was achieved by extending the traditional partial order reduction approach for state transition systems to control flow automata and by incorporating the summarizing technique of [201] to reduce the complexity of the interleaving. For embedding properties, data-bound properties are concerted to predicate bound properties which in turn are converted to reachability computation. Safety temporal properties are expressed by the *NEXT* and *UNTIL* temporal operators and embedded in a form of special functions.

Benefits: Data-bound and safety temporal properties are embedded in the composite WS specification and then verified by the generation of positive and negative test cases.

Validation: For the evaluation of the effectiveness of the presented analysis approach one big case study is used. The results show that the approach is able to generate positive and negative test cases for verifying data-bound properties embedded in the OWL-S specifications of composite WSs.

Simulation, Verification and Automated Composition of Web Services[176]

Problem: In [176] Narayanan and McIlraith deal with enabling markup and automated reasoning technology to describe, simulate, automatically compose, test and verify Web service compositions.

Explanation: The success of the WS paradigm has led to a proliferation of available WSs. These WSs can be combined in a composition in order to perform a complex task. The big challenge is to describe and prove properties of these WSs in order to: a) test the composed system by simulating its execution under different input conditions; b) to logically verify certain maintenance and safety conditions associated with the composed WS; c) to automatically compose WSs.

Solution: The authors take the DAML-S ontology for describing the capabilities of WSs and define the semantics for a relevant subset of DAML-S in terms of a first-order logical language. The basic idea for composing and verifying is to first map salient aspects of the DAML-S model into a situation calculus, and from there into the Petri net domain. Then, they relate the complexity of various WS tasks (simulation, verification, performance analysis) to the expressiveness of DAML-S by providing proofs for their claims. Most importantly, they show that the complexity of the reachability problem for DAML-S Process Models is PSPACE-complete.

Benefits: DAML-S is eventually mapped to Petri-nets. Many powerful analysis techniques have been developed for Petri-nets. These techniques can be used for the simulation, validation and verification

of Composite WSs described in DAML-S. In addition, the authors prove the complexity of various WS analysis tasks with respect to various restricted forms of DAML-S.

Validation: A DAML-S interpreter has been implemented that translates DAML-S markups to the simulation and modeling environment KarmaSIM [177]. The KarmaSIM tool allows for interactive simulation and supports various verification and performance analysis techniques like reachability analysis, deadlock detection, invariant computations, quantitative analysis techniques and most-likely paths. A sample DAML-S network having a variety of non-free constructs as well as loops has been constructed, translated and simulated by the KarmaSIM tool.

Towards a formal verification of OWL-S process models[10]

Problem: In [10] Ankolekar et. al. deal with the verification of the interaction protocol of WSs.

Explanation: Verification of the interaction protocol of WSs can be used to prove that the protocol to be advertised is indeed correct (e.g. does not contain deadlocks) and to guarantee additional properties, e.g. purchased goods are not delivered if a payment is not received.

Solution: The authors work extends the work analyzed in [176] in three directions. *First*, a model of WS data flow is provided in addition to control flow. In this way, the verification procedure can detect harmful interactions between data and control flow. For instance, if a choice process is realized by two atomic processes and a data flow link exists from the output of the first atomic process to the input of the second one, the result is a legal OWL-S process model which is flawed. *Second*, an OWL-S process model is translated into a simpler model of the PROMELA specification language [115] that preserves all the essential behavior to be verified. Actually, it must be noted that only *processes*, *control flow*, *concurrency*, *data flow* and *loops* are fully modeled while *conditions* and *inputs/outputs* are partially modeled in PROMELA. More specifically, the authors claim that checking of data values could be deferred to a type-checker or semantic reasoner. Thus, they do not represent values or data-types of inputs and outputs and limit themselves to modeling assignment. Moreover, OWL-S *preconditions* and *effects* are not modeled as it is claimed that these constructs do not affect the execution and verification of the Process Model and they are just warnings. *Third*, the PROMELA specifications are fed into the SPIN [115] model checking tool for automatically verifying that the interaction protocol satisfies the claims. If SPIN verifies that a claim is true, then the claim is true in the corresponding OWL-S Process Model. Otherwise, if the protocol contains an error, the model-checker identifies a counter-example. The claim may still hold in the Process Model, so the counter-example must be analyzed and simulated in the actual Process Model. If this does produce faulty behavior, then a bona fide bug has been discovered, else a spurious bug has been identified.

Benefits: This work can check various claims about an OWL-S Process Model. These properties include the values of certain variables at certain points in the code and true statements that can be made about execution states or the paths of execution. In addition, as the entire state space of a verification model is searched, this work can also identify unreachable or dead code in a Process Model.

Validation: Several properties of the execution of the Amazon OWL-S Process Model were verified by conducting five tests on this work's implementation. For each test the size of the model constructed by Spin, the time taken in seconds to construct the model and the time for verification were measured. The experiment showed that the verification of OWL-S Process Models that did not contain any loops could be done very effectively while the addition of loops caused exponential increase in number of states and verification time.

Towards efficient verification for process composition of semantic web services[154]

Problem: In [154] Luo et. al. deal with the verification of semantic WS compositions.

Explanation: OWL-S provides no way to validate a WS composition. In practice, WS composition is usually a complex and error-prone process whether happening at design time or at runtime. Moreover,

if an erroneous composition plan is executed without being previously verified, deployment of the WS composition process often results in runtime errors, which need to be repaired on-the-fly at high costs.

Solution: The authors of this work propose an analysis and verification technique based on Colored Petri Nets (CPNets) [121]. The main idea is to define a composite process in a three-level specification: *interface net*, *orchestration net* and *composition net*, and to specify the control and data flow by mapping control constructs of OWL-S into the CPNet representation. Then the constructed CPNet model is simulated and validated to detect the composition errors, such as deadlocks, and to verify whether the composition process does have certain expected dynamic properties.

The *interface net* identifies each component service composed of some semantically related subprocesses as a unique functional object and specifies how to participate in data exchange with other services. The *orchestration net* describes the dynamic behaviors of a component service by focusing on its internal operational nature. *Composition net* defines the coordination by means of a two-layer hierarchy: *global model* and *local model*. The *global model* is used to describe the choreography process which denotes the interaction protocol among heterogeneous participants distributed over the Web and it is represented by a collection of interface nets of the participating services, and a collection of links which specify the interaction rules. The *local model* is used to describe the inner orchestration process of each component service and it is represented by the above orchestration net. The model constructed by the mapping of OWL-S constructs is proven to be adequate for the description of the given composition through the use of the Composition Adequacy Verification (CAV) algorithm.

Finally, a OWL-S Process Model is translated to an equivalent CPNet, CPN tools can be used to simulate it in order to test whether the composition process will be executing as desired. In addition, state space analysis of CPNets can be carried out to formally validate and verify the functional correctness of the composition under consideration.

Benefits: This work can be used to detect composition errors such as deadlocks and to verify that the composition process exposes certain expected dynamic properties like *reachability*, *boundedness*, *liveness*, *fairness* and *home*.

Validation: The authors have devised a fictitious scenario of an OWL-S WS composition based on a travel plan. They have implemented a translator that transforms OWL-S markups to the simulation and modeling environment CPN Tools.After their OWL-S process model is translated into a CPNet model, it is executed to simulate the performance of the composition process. The results show that the composition is not erroneous.

4.5 Observations

This section classifies the approaches summarized above according to the classification framework from Section 4.3. Based on this classification, we will summarize our major observations.

The classification of the results as well as the discussion of our observations will be done separately following the three major classes of quality assurance techniques and methods (i.e., testing, monitoring and static analysis). For each class, the contributions will be summarized in two tables. The first table (Table-A) contains the *Class*, *Quality*, *Life-Cycle*, *Discipline* and *Layer* dimension of the classification framework. The second table (Table-B) shows the *Artifact*, *Reference*, *Formality*, *Automation* and *Validation* dimension.

4.5.1 Testing

The surveyed papers and their classification shows several areas of interest in the testing community.

The majority of the approaches that has been surveyed addresses the problem of generating test cases for testing services and service-based applications. This includes deriving test cases from service descriptions (“black box”), based on WSDL for instance, and deriving test cases from service compositions (“white box”), based on BPEL for example. Approaches are presented for deriving (or generating)

test inputs, as well for determining the expected outputs (test oracles). In addition, several of these approaches include techniques for executing the test cases, once these have been defined.

Several other approaches aim at providing solutions to support the regression testing of service-based applications. A regression test, i.e., re-testing the application, is essential due to its dynamic nature (the service composition can change due to the evolution of the services or other changes in the context of the service-based application). Interestingly, although dynamics as an important characteristic of service-based applications is acknowledged and many of the testing techniques support the automatic generation and execution of test cases, there are but a few contributions that actually propose performing tests during the actual operation of the system. It appears that run-time quality assurance currently is basically left to monitoring.

Only very few isolated approaches provide solutions for specific QoS characteristics. However, as it has been motivated in Section 2 of this deliverable, QoS is an important aspect of the contracts between service providers and consumers. Thus, assuring whether the services conform with the agreed levels of quality is an essential activity.

4.5.2 Monitoring

The observations are briefly summarized with focus on analytical quality assurance. A detailed discussion of the observations can be found in deliverable PO-JRA-1.2.1.

Research activities on monitoring has provided very rich and comprehensive set of approaches that cover wide range of goals, problem aspects, and information types, as well as the different components of the SBA architecture. The monitoring problem is considered from various stakeholder perspectives and it addresses functional and also QoS aspects.

Another important trend is to consider more and more aspects during the monitoring task, such as different types of information, sources, types of event, their distribution. However, only few approaches go beyond a particular monitoring problem or provide a wider perspective on the application execution. Furthermore, there are no existing approaches that cross all the functional layers of a service-based application, consider evolutionary aspects of executions with respect to a variety of information, etc.

The evidence that the survey provides shows that only a few approaches deal with the central aspect for monitoring, which is observing the context of the service-based application. Existing approaches mainly focus on the events that are internal to the service-based application.

4.5.3 Analysis

The survey results, as shown in the previous sections, show that a lot of research effort has been spent by the community to build the current state of the art in analysis of service-based applications from a quality assurance point of view. Different formal approaches, ranging from Petri nets to process algebras, and different quality characteristics, ranging from non-functional to functional properties, and for different forms of service-based applications specifications, have been proposed and analyzed by the research community.

The problem has been faced from the general perspective of software analysis for years, gaining major and mature results. The contributions we analyzed show that software analysis techniques and methods can be adapted to service-based applications. In fact, for example, many of the state of the art approaches include classic verification methodologies, such as model checking, applied to the service oriented world. Moreover, other approaches face the problem of analyzing non-functional characteristics of service-based applications, which are critical in open-world architectures.

Despite the great effort in the area, an evident problem can be found in the lack of integrated methods for the overall analysis of functional and non-functional properties of service-based applications. In other words, an interesting line of research would be the creation of a set of comprehensive, integrated techniques and methods able to encorporate different aspects of quality analysis at different levels of

Table 4.1: Table-A for Testing

Contribution	*Class*	*Quality*	*Life-Cycle*	*Discipline*	*Layer*
[109]	testing	correctness	design	SE, SOC	SCC
[256]	testing	functionality	design	SOC	SI
[261]	testing	functionality	design	SOC	SCC
[18]	testing	functionality	design	SOC	SCC
[253, 257]	testing	trustworthiness, robustness	design	SOC, SE	SCC
[272]	testing	functionality	design	SOC, SE	SCC
[195]	testing	functionality, conformance	design	SE, SOC	SCC
[132]	testing	functionality	design	SE, SOC	SCC
[63]	testing, monitoring	functionality	operation	SOC	SI
[111]	testing	functionality, interoperability	design	SE	SI
[75]	testing	functionality	design	SOC	SCC, BPM
[228]	testing	functionality	operation	SE	SCC
[207]	testing	several quality characteristics	operation	SE	SCC
[232]	testing	functionality, conformance	design	SE, SOC	SCC
[255]	testing, monitoring	functionality	whole life-cycle	SOC, SE	SCC, SI
[254, 263, 252, 262, 15, 259]	testing, monitoring	functionality	design and operation	SOC	SCC
[181, 65]	testing	functionality	design	SE	SI
[165, 153]	testing	functionality	design	SOC	SCC, BPM
[46]	testing	functionality	design	SE	SCC
[78]	testing, monitoring	functionality	design	SOC	SCC
[122]	testing	functionality	design	SOC	SCC
[203]	testing	conformance	design	SOC	SCC
[234]	testing	functionality	design	SE	SCC
[112]	testing	conformance	registration	SE	SI
[158, 159, 160]	testing	robustness	design	SOC, SE	SCC
[215, 214, 216]	testing	functionality	execution	SOC, SE	SCC
[224]	testing	functionality and performance load	design	SE	SCC
[17]	testing, monitoring	functionality	design	SOC, SE	SCC
[36]	testing	interoperability	registration	SE	SCC
[169]	testing	functionality	design	SE	SCC
[231]	testing	functionality	design	SE	SCC
[150]	testing, monitoring	reliability, conformance	whole life-cycle	SE	SCC
[98]	testing	functionality	design	SE, SOC	SCC
[243]	testing	correctness, reliability, availability	design	SOC	SCC
[73]	testing	all QoS characteristics	operation	SE, SOC	SCC
[45, 198]	testing, monitoring	arbitrary QoS characteristics	operation	SOC, SE	SCC
[51]	testing	functionality	operation	SOC	SCC
[16]	testing	functionality	operation	SOC, SE	SCC
[97]	testing	functionality	design	SOC, IS	BPM

abstraction. Such techniques should take into account the specific layers of a service-based application, and face coherence issues among different models and aspects of the same application.

Another critical issue is the analysis problem in the emerging area of open-world service-based applications, where the interaction with third-party entities is fundamental. In these architectures, dynamic analysis of the behavior of third-party services can be useful to enhance the overall dependability of service-based applications.

Table 4.2: Table-B for Testing

Contribution	*Artifact*	*Reference*	*Formality*	*Automation*	*Evaluation*
[109]	service, service composition	WSDL description	semi-formal	automated	discussion
[256]	service	WSDL spec.	-	-	-
[261]	service, service composition	WSDL spec.	semi-formal	automated	-
[18]	(atomic) service	WSDL spec.	formal	automated	experience report
[253, 257]	web service	OWL-S spec.	formal	non-automated	example appl.
[272]	web service	OWL-S spec.	formal	automated	example appl.
[195]	semantic web services	inputs, outputs, preconditions, effects (IOPEs)	semi-formal	automated	experience report
[132]	web service	EFSM based on WSDL spec.	formal	non-automated	example appl.
[63]	services, service compositions	OWL-S process model	semi-formal	automated	discussion
[111]	(atomic) service	provided and required contracts	formal	automated	discussion
[75]	web service composition	BPEL spec.	semi-formal	automated	discussion
[228]	service, service composition	provided and required contracts	formal	automated	discussion
[207]	service composition	probability distribution functions	formal	automated	discussion
[232]	web services which operate in the presence of persistent data	EFSM based on WSDL-S specification	formal	automated	discussion
[255]	service composition	WSDL description	semi-formal	automated	example appl.
[254, 263, 252, 262, 15, 259]	group of services	majority output	semi-formal	automated	example
[181, 65]	test of interaction between two web services	XML and SOAP messages	semi-formal	automated	example appl.
[165, 153]	BPEL process	WSDL spec.	semi-formal	automated	example appl.
[46]	composed web service	other web services	formal	non-automated	example appl.
[78]	service and service workflow	XML test description	semi-formal	automated	example appl.
[122]	composed service and workflow	-	formal	non-automated	example
[203]	web service protocol	formal model of the web service protocol	formal	automated	example appl.
[234]	web service	WSDL spec.	non-formal	automated	experience report
[112]	(atomic) service	specification (as graph-transformation rules)	formal	automated	experience report
[158, 159, 160]	service, service composition	WSDL spec.	semi-formal	automated	explor. case study
[215, 214, 216]	service, service composition	control-flow graph	formal	automated	example appl.
[224]	service-based application	XML description	non-formal	automated	discussion
[17]	atomic and composed services	service spec.	semi-formal	automated	discussion
[36]	web service	protocl state machine spec.	semi-formal	automated	discussion
[169]	composed web service	WSDL spec.	semi-formal	automated	example appl.
[231]	web service	WSDL spec.	semi-formal	automated	example
[150]	service-based application	UML models and graph transformation rules	formal	automated	discussion
[98]	compositions of web services	BPEL spec.	formal	automated	discussion
[243]	web service based application	WSDL spec., TLTS, TPG	formal	automated	discussion
[73]	service composition	SLA	semi-formal	automated	case study
[45, 198]	services, service compositions	WSDL description	semi-formal	automated	example appl.
[51]	service-based application	oracle from offline test	formal	automated	experience report
[16]	service-oriented application	-	semi-formal	automated	example
[97]	service, service composition	WSDL, BPEL	semi-formal	automated	discussion

Table 4.3: Table-A for Monitoring

Contribution	*Class*	*Quality*	*Life-Cycle*	*Discipline*	*Layer*
[199, 21]	monitoring	behavior	run-time (operation)	SOC	SCC
[39]	monitoring	behavioral conformance	run-time (operation)	SOC	SCC
[24]	monitoring	functional correctness	run-time (operation)	SOC	SCC
[25, 26, 22]	monitoring	behavioral correctness	run-time (operation)	SOC	SCC
[12]	monitoring	behavioral correctness	run-time (operation)	SOC	SCC
[29]	monitoring	privacy agreements	run-time (operation)	SOC	SI
[238, 155, 156]	monitoring	behavioral correctness, QoS (e.g., perfomance)	run-time (operation)	SOC	SCC
[237, 134]	monitoring	compliance to security requirements	run-time (operation)	SOC	SI, SCC
[84]	monitoring	correctness with respect to SLA properties	run-time (operation)	SOC, Grid	SI
[144]	monitoring	behavioral correctness	run-time (operation)	SOC, BPM	SCC, BPM
[220]	monitoring	SLA, QoS (security, performance, reliability, cost)	run-time (operation)	SOC	SI
[131]	monitoring	SLA, QoS (performance, reliabiltiy)	run-time (operation)	SOC	SI
[151]	monitoring	SLA, various observable QoS	run-time (operation)	SOC	SI
[60]	monitoring	functional correctness	run-time (operation)	SOC	SI
[28]	monitoring	relatively any characteristic of a process	run-time (operation)	BPM	BPM
[171]	monitoring	performance, KPI	run-rime (operation)	BPM	SCC
[55]	monitoring	business metrics, KPIs	run-time (operation), post-mortem	BPM	BPM
[208]	monitoring	relatively any characteristic of a process	run-time (operation)	BPM	SCC
[50]	monitoring	business parameters	run-time (operation)	BPM	BPM
[119]	monitoring	business properties and metrics	run-time (operation)	BPM	BPM
[108]	monitoring	behavior	post-mortem	BPM	BPM
[211, 268]	monitoring	behavioral correctness	run-time (operation), post-mortem	BPM	BPM
[267]	monitoring	security (security-critical behavior)	run-time (operation), post-mortem	BPM	BPM
[180]	monitoring	behavior	post-mortem	SOC	SCC
[71]	monitoring	behavioral properties	post-mortem	SOC	SCC
[247]	monitoring	Performance	run-time	Grid	SI
[251]	monitoring	Performance	run-time	Grid	SI
[62, 93]	monitoring	Various domain-specific information (QoS, resource properties, available services)	run-time, post-mortem	Grid	SI
[89]	monitoring	Domain-specific information	run-time, post-mortem	Grid	SI
[179]	monitoring	Domain-specific information, performance	run-time, post-mortem	Grid	SI
[9]	monitoring	Domain-specific information, performance	run-time, post-mortem	Grid	SI

Table 4.4: Table-B for Monitoring

Contribution	*Artifact*	*Reference*	*Formality*	*Automation*	*Evaluation*
[199, 21]	composition of BPEL processes	behavioral composition requirements	formal	automated	explor. case study
[39]	composition of BPEL processes	conversation constraints	formal	automated	explor. case study
[24]	BPEL process	func. assertions on BPEL activities	formal	automated	explor. case study
[25, 26, 22]	BPEL process	func. and non-func. assertions, temporal requirements	formal	automated	explor. case studies
[12]	global behavior of service composition	compliance with local service execution models	formal	automated	explor. case study
[29]	use of privacy information by service provider	privacy agreement properties (rights and obligations)	formal	automated	discussion
[238, 155, 156]	BPEL process, service composition execution	func. and non-func. assertions, temporal requirements	formal	automated	explor. case study, experiments
[237, 134]	service composition execution	security properties (patterns)	formal	automated	explor. case study
[84]	BPEL process	func. and non-func. assertions, temporal requirements	formal	automated	discussion
[144]	business process	func. requirements and assertions	formal	automated	explor. case study
[220]	service execution	proprietary SLA properties and assertions	-	automated	discussion
[131]	service execution	proprietary SLA properties and assertions	-	automated	discussion
[151]	service execution	WS-Agreement properties and assertions	-	automated	discussion
[60]	services and message exchanges	WS-Policy spec.	-	automated	discussion
[28]	BPEL process	visual behavioral queries and report spec.	formal	automated	explor. case study
[171]	BPEL process	process performance indicators	-	automated	explor. case study
[55]	business process	business metrics	-	automated	discussion
[208]	BPEL process	process audit spec.	-	automated	discussion
[50]	business process	metrics, KPIs	-	automated	discussion
[119]	business process	metrics, KPIs	-	automated	discussion
[108]	process execution logs	-	formal	automated	explor. case study
[211, 268]	business process execution logs	process spec.	formal	automated	explor. case study, experiments
[267]	business process execution logs	security-correct process spec.	formal	automated	explor. case study
[180]	service interactions	-	formal	semi-automated	experiments
[71]	service interactions	-	formal	automated	discussion, experiments
[247]	service messages	Domain-specific metrics	-	automated	discussion
[251]	TCP streams	XPath queries over monitoring data	-	automated	TBD
[62, 93]	specific data providers	specific information queries	-	automated	discussion, case studies
[89]	data streams, specific providers	SQL-like queries	-	automated	discussion
[179]	specific data providers	regular expression predicates	-	automated	discussion
[9]	specific data providers	domain-specific metrics	-	automated	TBD

Table 4.5: Table-A for Analysis

Contribution	*Class*	*Quality*	*Life-Cycle*	*Discipline*	*Layer*
[104, 105]	analysis (verification)	QoS (scalability, security)	design	SE, SOC	SCC
[49]	analysis (synthesis)	code complexity	design	SE, SOC	SCC
[212]	analysis (synthesis)	QoS (many)	design	SE, SOC	SCC
[161]	analysis (synthesis)	QoS (many)	design, execution	SE, SOC	SCC
[11]	analysis (synthesis)	QoS (many)	design, execution	SOC, SE	SCC
[96]	analysis (verification)	functional behavior	design	SE, SOC	SCC
[40]	analysis (verification)	behavior	design	SE, SOC	SCC
[168]	analysis (verification)	Deadlock, Timeouts, Reachability	Execution	SOC and BPM	SCC
[266, 4, 210]	analysis (verification)	Correctness, Conformance	Design, Execution	BPM	SCC
[225]	analysis (synthesis)	Usabiliy	Design	SOC	SCC
[175]	analysis (verification)	Dataflows properties (e.g. deadlocks, reachability)	Design	SE, SOC	SCC
[135]	analysis (verification)	Deadlocks	Execution	BPM, SOC	SCC
[260, 117]	analysis (verification)	Reachability property, Safety temporal property, Predicate-bound property	Verification and Validation	SE, SOC	SCC
[229]	analysis (verfication)	Safety properties	Execution	SOC	SCC
[276]	analysis (synthesis)	Exchanged message properties	Design	BPM, SOC	SCC
[81]	analysis (verification)	Correlation and synchronous receive/reply messages	Execution	BPM, SOC	SCC
[90]	analysis (verification)	Interface Compatibility, Safety Compatibility, Liveness Compatibility	Design	BPM, SOC	SCC
[19]	analysis (verification)	Synchronous and asynchronous transition properties	Design	SE, SOC	SCC
[86]	analysis (verification)	Behavioral and safety properties	Design	SE, SOC	SCC
[223]	analysis (verification)	Safety and liveness properties	Design	SE, SOC	SCC
[92]	analysis (verification)	Deadlock, liveness	Design, execution	SOC	SCC
[209]	analysis (verification)	Security policies	Execution	SOC	SCC
[66]	analysis (synthesis)	Reachability	Design, execution	SOC	SCC
[128, 127]	analysis (verification)	behavioral correctness	design	BPM, RE	BPM, SCC
[130, 129, 126, 123]	analysis (verification)	behavioral correctness	design	BPM, SOC	SCC
[124, 125]	analysis (verification)	behavioral correctness	design	SOC	SCC
[191, 190, 269]	analysis (verification)	behavioral correctness	design	BPM	SCC, BPM
[242]	analysis (verification)	behavioral correctness	design	SOC	SCC
[280]	analysis (verification)	protocol conformance	design	SOC	SCC
[37]	analysis (verification)	behavioral correctness	design, execution	SOC	SCC, SI
[77]	analysis (verification)	behavioral correctness	design	SOC	SCC
[70]	analysis (verification)	behavioral correctness	design	SOC	SCC
[64]	analysis (verification)	behavioral correctness	design	SE, SOC	SCC
[117]	analysis (verification)	behavioral correctness	design	SE, SOC	SCC
[176]	analysis (verification), simulation	behavioral correctness	design	SE, SOC	SCC
[10]	analysis (verification)	behavioral correctness	design	SE, SOC	SCC
[154]	analysis (verification), simulation	behavioral correctness	design	SOC	SCC

Table 4.6: Table-B for Analysis

Contribution	*Artifact*	*Reference*	*Formality*	*Automation*	*Evaluation*
[104, 105]	colored UML statecharts, sequence diagrams	QoS requirements	formal	automated	explor. case studies
[49]	WS-BPEL descriptions	developer interpretation, complexity standards	formal	automated	explor. case study
[212]	WS-BPEL descriptions	SLAs	formal	automated	not evaluated
[161]	BPEL, WSDL descritions	QoS requirements	formal	automated	explor. case study
[11]	BPEL descriptions	QoS parameters	formal	automated	test cases
[96]	BPEL descriptions	LTL properties	formal	automated	explor. case study
[40]	BPEL4WS compositions	LTL properties, WS-CoL, deadlock freedom	formal	automated	explor. case studies
[168]	Petri nets	Orchestration schemas	formal	automated	explor. case study
[266, 4, 210]	Petri nets	Workflow models	formal	automated	explor. case study
[225]	Petri nets	BPEL4WS specifications	formal	not automated	-
[175]	Promela processes, LTL formula	WSFL specifications	formal	automated	explor. case study
[135]	BPE-calculus	BPEL4WS specifications	formal	automated	empirical evaluations
[260, 117]	C-like, PDDL	OWL-S specifications	formal	automated	-
[229]	PHP	Safety properties	formal	automated	-
[276]	aDFA	BPEL4WS specifications	formal	automated	explor. case study
[81]	DASM	BPEL4WS specifications	formal	automated	explor. case study
[90]	LTS	BPEL4WS specifications	formal	automated	explor. case study
[19]	Constraint automata	Graphs	formal	automated	theoretical proves
[86]	HD-automata	π-calculus processes	formal	automated	explor. case studies
[223]	CCS	BPEL4WS specifications	formal	automated	explor. case studies
[92]	FSP	BPEL4WS specifications	formal	automated	empirical case studies
[209]	Event Calculus	BPEL4WS specifications	formal	not automated	empirical case studies
[66]	π-calculus	Let's Dance	formal	not automated	Theoretical proves
[128, 127]	business process specification in BPEL	formal Tropos requirements specifications	formal	automated	explor. Case Study
[130, 129, 126, 123]	BPEL descriptions	behavioral composition requirements	formal	automated	explor. case studies
[124, 125]	composition of stateful services	choreography specification	formal	automated	Simple Scenarios
[191, 190, 269]	BPEL descriptions	predefined execution properties	formal	automated	explor. Case Study
[242]	formal model of the composition specification	predefined execution properties	formal	automated	explor. case study
[280]	formal model of the service composition	generic correctness properties, and conversation protocol of a participant	formal	automated	explor. case study
[37]	WS interface models	other Web service interface models	formal	automated	Simple Scenario, Theoretical proves
[77]	abstract process models	elementary service preconditions and process postconditions (goals)	formal	automated	not evaluated
[70]	WS composition model	temporal properties (LTL), conversation protocols	formal	not automated	not Evaluated
[64]	protocols of the composition participants	dynamic properties of service contracts	formal	automated	explor. case Study
[117]	OWL-S Composite WS Descriptions	temporal properties	formal	automated	explor. case study
[176]	AML-S Composite WS Descriptions	safety conditions	formal	automated	explor. case study
[10]	OWL-S Composite WS Descriptions	temporal properties	formal	automated	explor. case study
[154]	OWL-S Composite WS Descriptions	reachability, liveness, fairness properties	formal	automated	Explor. case study

Chapter 5

Research Perspectives

This final chapter of the deliverable highlights relevant research issues for the future, which have been identified based on the above survey results and the observations on quality related aspects relevant for service-based application. The results of the survey presented in Chapter 2 put the accent on several directions to establish a research agenda for the near future concerning Quality of Service (QoS) modeling and specification, as well as QoS negotiation in service-based applications. The observations of the survey on analytical quality assurance for services and service-based applications – as presented in Chapter 4 – have uncovered several gaps in the current state of the art. Drawing from these observations and from the issues reported in [193], the following important research challenges can be identified.

QoS Modeling

For what concerns QoS modeling, our survey has uncovered the lack of a well established and standard QoS model for services. In addition, most of the research approaches do not offer a rich, extensible, and semantic QoS model. None of the approaches presented in our survey model or publicly provide a rich set of quality dimensions that could be used for expressing QoS capabilities or requirements (the most flexible being probably WSLA, which however, does not provide operations on QoS profiles). As a result, QoS capabilities and requirements and service SLAs are described by many different formalisms. The differences between these formalisms limit the fulfillment of the vision of automated and precise QoS-based service matchmaking and selection and QoS-aware service composition.

Hence, we argue that a first research direction concerns the development of a standard QoS model, describing every relevant aspect of QoS for services, including metrics, units, measurement functions and directives, constraints, value types, etc. In addition, this QoS model should encompass a rich set of domain-dependent and global quality dimensions and should be extensible so as to allow the addition of new quality dimensions when it is needed (e.g., for a new application domain). Last but not least, this standard QoS model should be semantically enriched in order to be machine-processable and machine-interpretable.

Such a comprehensive QoS model for services requires a suitable QoS language to be used in complex service-based applications, in which services can be invoked and composed with variable QoS profiles. Such a rich language (encompassing a well established service QoS model) should be capable of expressing QoS capabilities and SLAs by using functions, operators and comparison predicates on QoS metrics. It should also allow the description of composition rules for every possible combination of composition constructs and QoS metrics. Moreover, it should allow the description of different QoS offerings for the same functional offering of a service; i.e. it should be able to describe profiles.

QoS Negotiation

We identify two main streams for short-term research on service QoS negotiation. First, we underline the issue of automated SLA establishment in service compositions. The review shows that most of the

current work in this field concerns the negotiation between a service consumer and a service provider or the set of providers of functionally equivalent services. Proposals for managing complex 1-to-N negotiation with services involved in the same service composition are still at their infancy and need further development. Second, research efforts should be devoted to the analysis of innovative negotiation strategies explicitly tailored to the requirements of service-based applications. As of now, the participants to the QoS negotiation in service-based applications adopt state-of-the-art strategies, drawn from research on agent-based computing. We argue that more efficient and flexible solutions to the negotiation problem become feasible when negotiation strategies take into account the features of negotiation objects and protocols in service-based applications.

QoS negotiation can also be used to extend the capabilities of service composition tools. QoS agreement gives of course best results when global optima have been achieved. This is in general difficult, especially because complex QoS expressions can give rise to optimization functions which are difficult or impossible to treat mathematically. Approximations, such as the ones achieved with genetic algorithms, simulated annealing, or planning strategies, appear to be the only feasible resort at this moment. Ensuring that the optimum found is global (or, at least, not too far from the global one) is an issue to be dealt with. Additionally, if the architecture allows for dynamic re-negotiation (within the same service, for which services should offer different QoS classes, or selecting another service), the cost of negotiation, e.g., planning, equation solving, etc., should be gauged against the expected gain achieved with the new service.

In Grid computing SLA management, we have observed that QoS models and SLA usage for resource provisioning in current Grids is quite similar to the approaches used for service-based applications. Some of the models and implemented tools are even used in both fields (an example is WS-Agreement). Concerning the solutions available for Grid computing, we can state that the adopted and proof-of-concept implementations are still premature. We found promising theoretical approaches that fit SLA usage to current Grid systems, but there is no common mechanism for SLA advertisement and discovery. WS-Agreement seems to be a good candidate, but existing solutions using this form still cannot interoperate. Regarding SLA negotiation, WS-Agreement still cannot deliver the solution: in most cases it can only be used for a simple offer-accept interaction. The future directions should identify a commonly accepted approaches for advertising and discovery. Recent experiences both in the Service and Grid fields should be taken into account in order to arrive at a widely accepted and interoperable solution.

Run-time Quality Assurance

Services are often provisioned in the context of short-term, volatile and thus highly dynamic business relationships between service providers and requestors which are not known at design time. Thus, services will have to be enabled to collaborate in highly distributed environments, cutting across the boundaries of various organizations (including enterprises, governmental and non-profit organizations). The aggregation of services to build service-based applications is very likely to happen mostly at run-time in the near future. Which means that the actual functional behavior and quality aspects of the service-based application will have to be determined during its actual operation. Moreover, determining the relevant aspects of the context in which the service-based application is executed becomes a run-time issue. As an example, the types of users which interact with the service-based application, or the actual situation or location in which the application is operated is only determined at run-time.

There is thus a strong need for quality assurance techniques that can be applied while the service-based application is in operation. Currently, typically monitoring techniques are employed for assuring the quality of an application during its operation. The problem with monitoring is that it only checks the current (actual) execution. It does not allow to pro-actively uncover faults which are introduced, e.g. due to a change in the application, if they are not leading to a failure in the current execution. Other techniques, such as testing or static analysis, examine sets of classes of executions and thus would allow to pro-actively uncover such faults before they lead to an actual failure during the operation of

the system. Therefore, standard and consolidated "off-line" software quality assurance techniques (like testing and analysis) should be extended such that they are applicable while the application operates ("on-line" techniques).

One important requirement of those "on-line" techniques, however, is that their overhead and costs should not be an overkill to the point that they become unpractical for this reason. Overheads and costs that incur under any execution environment typically include time, computational resources (e.g., processing power and memory) as well as communication resources (e.g., network bandwidth). Therefore, work on making the "on-line" techniques as light-weight as possible is needed.

Finally, as (self-)adaptation of service-based applications becomes an essential characteristic, there is a strong need to assure that the adaptation of a service-based application behaves as expected. This requires specific testing and analysis techniques to verify the adaptation behavior; e.g. this could include checking whether the behavior after the adaptation conforms to the expected behavior before the adaptation took place (in order to keep backwards compatibility). Those techniques and methods to assure the correctness of adaptations are badly lacking at the moment.

Assuring Quality Characteristics

In a dynamic business scenario, the contract life-cycle should be automated as much as possible, in order to allow organizations to dynamically change service providers (business partners) or to re-negotiate SLAs (see above). That requires that QoS aspects need to be checked during the operation, e.g. by monitoring the QoS characteristics, in order to determine whether the new service provider meets the desired QoS or whether there is a need for re-negotiating the SLAs. As this survey has revealed, there are only few and isolated research contributions on assuring QoS aspects. There is thus a strong need for novel techniques and methods that address QoS characteristics in a comprehensive and end-to-end fashion across all layers of a service-based application. In addition, approaches that consider the context of a service-based application and its impact on QoS are needed in order to pave the way towards context-aware service-based application.

Due to the dynamicity of the world in which service-based applications operate, techniques are needed to aggregate individual QoS levels of the services involved in a service composition in order to determine and thus check the end-to-end QoS during run-time. This aggregation will typically span different layers of a service-based application and thus a common understanding of what the different QoS characteristics mean within and across these layers is needed (also see above). In fact, the definition of a QoS taxonomy will be addressed in the follow-up deliverable CD-JRA-1.3.2: "Quality reference model for SBA".

Synergies between Approaches

The survey results have shown that first attempts are made to exploit potential synergies between the different classes of analytical quality assurance techniques. As an example, testing can be used as preventive mechanism for finding faults while monitoring is used during the operation of the service-based application. A combination of both techniques can compensate the weaknesses of the single approaches [48, 47]. Further examples are the use of dedicated tests during the operation of the service-based applications in order to determine the conformance to SLAs, the use of static analysis tools (like model checkers) to derive test cases, or the analysis of monitoring logs ("post-mortem" analysis or audit).

Despite these initial attempts, synergies that can be achieved by joining and integrating different kinds of techniques have not been fully exploited. For example, research can be directed at facilitating the use of monitoring results as input for run-time verification. Or, testing could be combined with monitoring in such a way that when a deviation is observed during monitoring, dedicated test cases are executed in order to pinpoint the issues that lead to the deviation (also see next sub-section).

Debugging and Diagnosis

In this deliverable, the focus has been on dynamic techniques and methods for uncovering failures and on static techniques for uncovering faults. Obviously, once a failure has been identified, the cause for this failure, i.e. the fault in the artifact, needs to be uncovered. This is the realm of debugging and diagnosis. In this area, we see many open issues in the context of quality assurance. Specifically, this leads to issues on how one can interact with services for the purpose of debugging without this interaction having side-effects on the current execution (actual operation) of the service-based application. This also means that current service interfaces must be enhanced for testability, diagnosis and debuggability, which however must be well balanced with desired characteristics such as information hiding, encapsulation and loose coupling.

* * *

Quality related aspects relevant for service-based applications cover a broad field of research, including work on quality modeling and specification, QoS and SLA negotiation, as well as constructive and analytical quality assurance (like testing, monitoring and static analysis). This deliverable gave an overview of this broad field of "service quality" and identified the key areas where research contributions are currently available. Based on this survey of the state of the art, important and emerging research challenges were highlighted to be pursued during future research in order to close several of the gaps which emerge from the current state of the art on "service quality".

Bibliography

[1] The GEMSS project: Grid-enabled medical simulation services, EU IST project, ist-2001-37153. `http://www.gemss.de/`.

[2] OGF grid resource allocation agreement protocol working group website:. `https://forge.gridforum.org/sf/projects/graap-wg`.

[3] OGF grid scheduling architecture research group website. `https://forge.gridforum.org/sf/projects/gsa-rg`.

[4] Proceedings of the international conference on business process management, BPM. In Wil M. P. van der Aalst, Arthur H. M. ter Hofstede, and Mathias Weske, editors, *Business Process Management*, volume 2678 of *Lecture Notes in Computer Science*. Springer, 2003.

[5] *Interactive Access Control for Web Services*. Kluwer, 2004.

[6] *2007 IEEE International Conference on Services Computing (SCC 2007), 9-13 July 2007, Salt Lake City, Utah, USA*. IEEE Computer Society, 2007.

[7] O. Ajayi, R. Sinnott, and A. Stell. Dynamic trust negotiation for flexible e-health collaborations. In *Proceedings of the 15th ACM Mardi Gras conference (MG)*, pages 1–7, New York, NY, USA, 2008. ACM.

[8] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures, and Applications*. Springer, 2004.

[9] Sergio Andreozzi, Natascia De Bortoli, Sergio Fantinel, Antonia Ghiselli, Gian Luca Rubini, Gennaro Tortone, and Maria Cristina Vistoli. GridICE: a monitoring service for grid systems. *Future Generation Computer Systems*, 21(4):559–571, April 2005.

[10] Anupriya Ankolekar, Massimo Paolucci, and Katia P. Sycara. Towards a formal verification of owl-s process models. In Yolanda Gil, Enrico Motta, V. Richard Benjamins, and Mark A. Musen, editors, *International Semantic Web Conference*, volume 3729 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2005.

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

[12] Liliana Ardissono, Roberto Furnari, Anna Goy, Giovanna Petrone, and Marino Segnan. Fault tolerant web service orchestration by means of diagnosis. In *Software Architecture, Third European Workshop, EWSA 2006*, pages 2–16, 2006.

[13] A. Arenas, M. Wilson, and B. Matthews. On trust management in grids. In *Proceedings of the 1st international conference on Autonomic computing and communication systems (Autonomics'07)*, pages 1–7, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[14] Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semant.*, 5(2):58–71, 2007.

[15] X. Bai, Y. Chen, and Z. Shao. Adaptive web services testing. In *31st Annual International Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 233–236, 2007.

[16] X. Bai, D. Xu, G. Dai, W. . Tsai, and Y. Chen. Dynamic reconfigurable testing of service-oriented architecture. In *Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 368–375, 2007.

[17] Xiaoying Bai, Guilan Dai, Dezheng Xu, and Wei-Tek Tsai. A multi-agent based framework for collaborative testing on Web services. In *The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, 2006 and the 2006 Second International Workshop on Collaborative Computing, Integration, and Assurance.SEUS 2006/WCCIA 2006*, page 6, 2006.

[18] Xiaoying Bai, Wenli Dong, Wei-Tek Tsai, and Yinong Chen. WSDL-Based Automatic Test Case Generation for Web Services Testing. In *Proceedings of the IEEE International Workshop on Service-Oriented System Engineering (SOSE)*, pages 215 – 220. IEEE Computer Society, 2005.

[19] Christel Baier, Marjan Sirjani, Farhad Arbab, and Jan Rutten. Modeling component connectors in reo by constraint automata. *Sci. Comput. Program.*, 61(2):75–113, 2006.

[20] Siddharth Bajaj, Don Box, Dave Chappell, Francisco Curbera, Glen Daniels, Phillip Hallam-Baker, Maryann Hondo, Chris Kaler, Dave Langworthy, Anthony Nadalin, Nataraj Nagaratnam, Hemma Prafullchandra, Claus von Riegen, Daniel Roth, Jeffrey Schlimmer, Chris Sharp, John Shewchuk, Asir Vedamuthu, cınalp Ümit Yal and David Orchard. *Web Services Policy Framework (WS-Policy)*. IBM, March 2006.

[21] Fabio Barbon, Paolo Traverso, Marco Pistore, and Michele Trainotti. Run-time monitoring of instances and classes of web service compositions. In *IEEE International Conference on Web Services (ICWS 2006)*, pages 63–71, 2006.

[22] Luciano Baresi, Domenico Bianculli, Carlo Ghezzi, Sam Guinea, and Paola Spoletini. A timed extension of WSCoL. In *2007 IEEE International Conference on Web Services (ICWS 2007)*, pages 663–670, 2007.

[23] Luciano Baresi and Elisabetta DiNitto. *Test and Analysis of Web Services*. Springer-Verlag GmbH, 2007.

[24] Luciano Baresi, Carlo Ghezzi, and Sam Guinea. Smart monitors for composed services. In *Service-Oriented Computing - ICSOC 2004, Second International Conference*, pages 193–202, 2004.

[25] Luciano Baresi and Sam Guinea. Towards dynamic monitoring of WS-BPEL processes. In *Service-Oriented Computing - ICSOC 2005, Third International Conference*, pages 269–282, 2005.

[26] Luciano Baresi, Sam Guinea, and Pierluigi Plebani. WS-Policy for service monitoring. In *Technologies for E-Services, 6th International Workshop, TES 2005*, pages 72–83, 2005.

[27] M. Becker and P. Sewell. Cassandra: Distributed access control policies with tunable expressiveness. In *POLICY '04: Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks*, page 159, Washington, DC, USA, 2004. IEEE Computer Society.

[28] Catriel Beeri, Anat Eyal, Tova Milo, and Alon Pilberg. Monitoring business processes with queries. In *Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 603–614, 2007.

[29] Salima Benbernou, Hassina Meziane, and Mohand-Said Hacid. Run-time monitoring for privacy-agreement compliance. In *Service-Oriented Computing - ICSOC 2007, Fifth International Conference*, pages 353–364, 2007.

[30] Salima Benbernou, Hassina Meziane, Yin Hua Li, and Mohand-Said Hacid. A privacy agreement model for web services. In *IEEE SCC* [6], pages 196–203.

[31] S. Benkner, G. Engelbrecht, S.E. Middleton, I. Brandic, and R. Schmidt. End-to-End QoS support for a medical grid service infrastructure. *New Generation Computing, Special Issue on Life Science Grid Computing*, 2007.

[32] E. Bertino, E. Ferrari, and A. Squicciarini. X-TNL: An XML-based language for trust negotiations. In *POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, page 81, Washington, DC, USA, 2003. IEEE Computer Society.

[33] E. Bertino, E. Ferrari, and A. Squicciarini. Trust negotiations: concepts, systems, and languages. *Computing in Science and Engineering*, 6(4):27–34, 2004.

[34] E. Bertino, E. Ferrari, and A. Squicciarini. Trust negotiations: Concepts, systems, and languages. *Computing in Science and Engineering*, 06(4):27–34, 2004.

[35] E. Bertino, E. Ferrari, and A.C. Squicciarini. Trust-X: A peer-to-peer framework for trust establishment. *IEEE Transactions on Knowledge and Data Engineering, TKDE*, 16(7):827 – 842, 2004.

[36] A. Bertolino and A. Polini. The audition framework for testing Web services interoperability. In *Proceedings. 31st Euromicro Conference on Software Engineering and Advanced Applications*, pages p. 134–42, 2005.

[37] D. Beyer, A. Chakrabarti, and T. A. Henzinger. Web Service Interfaces. In *Proceeding of the International Conference on World Wide Web (WWW)*, 2005.

[38] A. Bhargav-Spantzel, A. C. Squicciarini, and E. Bertino. Trust negotiation in identity management. *IEEE Security and Privacy*, 5(2):55–63, 2007.

[39] Domenico Bianculli and Carlo Ghezzi. Monitoring conversational web services. In *IW-SOSWE'07*, 2007.

[40] Domenico Bianculli, Carlo Ghezzi, and Paola Spoletini. A model checking approach to verify BPEL4WS workflows. In *Proceedings of the 2007 IEEE International Conference on Service-Oriented Computing and Applications (IEEE SOCA 2007), Newport Beach, USA*, pages 13–20. IEEE Computer Society Press, June 2007.

[41] A. J. Blazic, K. Dolinar, and J. Porekar. Enabling privacy in pervasive computing using fusion of privacy negotiation, identity management and trust management techniques. In *First International Conference on the Digital Society (ICDS 2007), 2-6 January 2007, Guadeloupe, French Caribbean*, page 30. Springer, 2007.

[42] R. W. Bradshaw, J. E. Holt, and K. E. Seamons. Concealing complex policies with hidden credentials. In *Proceedings of the 11th ACM conference on Computer and communications security CCS '04*, pages 146–157, Washingtion, DC, USA, 2004. ACM.

[43] I. Brandic, S. Pllana, and S. Benkner. An approach for the high-level specification of QoS-aware grid workflows considering location affinity. *Scientific Programming Journal*, 14(3-4):231–250, 2006.

[44] I. Brandic, S. Pllana, and S. Benkner. Specification, planning, and execution of QoS-aware grid workflows within the Amadeus environment. *Concurrency and Computation: Practice and Experience*, 20(4):331–345, 2008.

[45] M. Bruno, G. Canfora, M. Di Penta, G. Esposito, and V. Mazza. Using Test Cases as Contract to Ensure Service Compliance Across Releases. In *Proceedings of the 3rd International Conference on Service-Oriented Computing (ICSOC)*, page pp. 87 100. Springer, 2005.

[46] R. C. Bryce, Y. Chen, and C. J. Colbourn. Biased covering arrays for progressive ranking and composition of Web Services. *International Journal of Simulation and Process Modelling*, 3(1-2):80–87, 2007.

[47] Gerardo Canfora and Massimiliano di Penta. SOA: Testing and Self-checking. In *Proceedings of International Workshop on Web Services - Modeling and Testing - WS-MaTE*, pages 3 – 12, 2006.

[48] Gerardo Canfora and Massimiliano di Penta. Testing Services and Service-Centric Systems: Challenges and Opportunities. *IT Professional*, 8(2):10 – 17, 2006.

[49] J. Cardoso. Complexity analysis of BPEL web processes. *Software Process: Improvement and Practice*, 12(1):35–49, 2007.

[50] Malu Castellanos, Fabio Casati, Ming-Chien Shan, and Umesh Dayal. iBOM: A platform for intelligent business operation management. In *ICDE '05: Proceedings of the 21st International Conference on Data Engineering*, pages 1084–1095, 2005.

[51] W.K. Chan, S.C. Cheung, and K.R.P.H. Leung. A metamorphic testing approach for online testing of service-oriented software applications. *International Journal of Web Services Research*, 4(2):61–81, 2007.

[52] M.B. Chhetri, J. Lin, S. Goh, J. Yan, J. Y. Zhang, and R. Kowalczyk. A coordinated architecture for the Agent-based Service Level agreement Negotiation of Web service composition. In *In Proc. 2006 Australian Software Engineering Conference, ASWEC'06*, 2006.

[53] Mohan Baruwal Chhetri, Jian Lin, SukKeong Goh, Jian Ying Zhang, Ryszard Kowalczyk, and Jun Yan. A coordinated architecture for the agent-based service level agreement negotiation ofweb service composition. In *ASWEC '06: Proceedings of the Australian Software Engineering Conference*, pages 90–99, Washington, DC, USA, 2006. IEEE Computer Society.

[54] D.K.W. Chiu, S.C. Cheung, Patrick C.K. Hung, and Ho fung Leung. Facilitating e-negotiation processes with semantic web technologies. In *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 1*, page 36.1, Washington, DC, USA, 2005. IEEE Computer Society.

[55] P. Chowdhary, K. Bhaskaran, N. S. Caswell, H. Chang, T. Chao, S.-K. Chen, M. Dikun, H. Lei, J.-J. Jeng, S. Kapoor, C. A. Lang, G. Mihaila, I. Stanoi, and L. Zeng. Model driven development for business performance management. *IBM Syst. J.*, 45(3):587–605, 2006.

[56] L. Chung, B. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic, 2000.

[57] M. Comuzzi and B. Pernici. An architecture for flexible Web service QoS negotiation. In *Proceedings of the 9th IEEE Enterprise Computing Conference*, Enschede, The Netherlands, 2005.

[58] Antonio Ruiz Cortés, Octavio Martín-Díaz, Amador Durán Toro, and Miguel Toro. Improving the automatic procurement of web services using constraint programming. *Int. J. Cooperative Inf. Syst.*, 14(4):439–468, 2005.

[59] F. Curbera. Components contracts in Service-Oriented architectures. *IEEE Computer*, 11:74–80, 2007.

[60] Francisco Curbera, Matthew J. Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana. Colombo: Lightweight middleware for service-oriented computing. *IBM Systems Journal*, 44(4):799–820, 2005.

[61] K. Czajkowski, I.T. Foster, C. Kesselman, V. Sander, and S. Tuecke. SNAP: A protocol for negotiating service level agreements and coordinating resource management in distributed systems. In *In 8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2002), LNCS Vol. 2537*, pages 153–183, 2002.

[62] Karl Czajkowski, Steven Fitzgerald, Ian Foster, and Carl Kesselman. Grid information services for distributed resource sharing. In *10th IEEE International Symposium on High Performance Distributed Computing (HPDC*, pages 181–194, 2001.

[63] G. Dai, X. Bai, Y. Wang, and F. Dai. Contract-based testing for web services. In *31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, volume 1, pages 517–524, 2007.

[64] H. Davulcu, M. Kifer, and I. V. Ramakrishnan. CTR-S: A Logic for Specifying Contracts in Semantic Web Services. In *Proceeding of the International Conference on World Wide Web (WWW)*, pages 144–153, 2004.

[65] Lourival F. Júnior de Almeida and Silvia Regina Vergilio. Exploring Perturbation Based Testing for Web Services. In *IEEE International Conference on Web Services (ICWS)*, pages 717 – 726, 2006.

[66] Gero Decker, Johannes Maria Zaha, and Marlon Dumas. Execution semantics for service choreographies. In *WS-FM*, pages 163–177, 2006.

[67] Seema Degwekar, Stanley Y. W. Su, and Herman Lam. Constraint specification and processing in web services publication and discovery. In *ICWS*, pages 210–217. IEEE Computer Society, 2004.

[68] Nelly Delgado, Ann Q. Gates, and Steve Roach. A taxonomy and catalog of runtime software-fault monitoring tools. *IEEE Trans. Software Eng.*, 30(12):859–872, 2004.

[69] Vikas Deora, Jianhua Shao, W. A. Gray, and N. J. Fiddian. A quality of service management framework based on user expectations. In Maria E. Orlowska, Sanjiva Weerawarana, Mike P. Papazoglou, and Jian Yang, editors, *ICSOC*, volume 2910 of *Lecture Notes in Computer Science*, pages 104–114, Trento, Italy, 2003. Springer.

[70] A. Deutsch, L. Sui, V. Vianu, and D. Zhou. Verification of Communicating Data-driven Web Services. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 90–99, 2006.

[71] Didier Devaurs, Kreshnik Musaraj, Fabien De Marchi, and Mohand-Said Hacid. Timed Transition Discovery from Web Service conversation Logs. In *20th International Conference on Advanced Information Systems Engineering (CAISE'08)*, 2008.

[72] E. Di Nitto, M. Di Penta, A. Gambi, G. Ripa, and M.L. Villani. Negotiation of Service Level Agreements: An architecture and a search-based approach. In *Proc. ICSOC'07*, pages 295–306, 2007.

[73] Massimiliano Di Penta, Gerardo Canfora, Gianpiero Esposito, Valentina Mazza, and Marcello Bruno. Search-based Testing of Service Level Agreements. In *Proceedings of the Conference on Genetic and Evolutionary Computation GECCO*, pages 1090 – 1097. ACM Press, 2007.

[74] Glen Dobson, Russell Lock, and Ian Sommerville. QoSOnt: a QoS ontology for service-centric systems. In *EUROMICRO '05: Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 80–87, Porto, Portugal, 2005. IEEE Computer Society.

[75] Wen-Li Dong, Hang Yu, and Yu-Bing Zhang. Testing BPEL-based Web Service Composition Using High-level Petri Nets. In *EDOC '06: Proceedings of the 10th IEEE International Enterprise Distributed Object Computing Conference*, pages 441–444. IEEE Computer Society, 2006.

[76] N. Dragoni and F. Massacci. Security-by-contract for web services. In *Proceedings of the 2007 ACM workshop on Secure web services (SWS '07)*, pages 90–98, New York, NY, USA, 2007. ACM.

[77] Z. Duan, A. J. Bernstein, P. M. Lewis, and S. Lu. Semantics Based Verification and Synthesis of BPEL4WS Abstract Processes. In *Proceeding of the International Conference on Web Services (ICWS)*, pages 734–737, 2004.

[78] S. Dustdar and S. Haslinger. Testing of service-oriented architectures - a practical approach. In *5th Annual International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World*, volume 3263 of *Lecture Notes in Comput. Sci.*, pages 97–109, 2004.

[79] K. El-Khatib and G. v. Bochmann. Protecting the privacy of user's qos preferences for multimedia applications. In *Proceedings of the 2nd ACM international workshop on Wireless multimedia networking and performance modeling (WMuNeP '06)*, pages 35–42, New York, NY, USA, 2006. ACM.

[80] C. Hankin F. Nielson, H. R. Nielson. *Principles of Program Analysis*. Springer, 2005. Second Ed.

[81] Roozbeh Farahbod, Uwe Glässer, and Mona Vajihollahi. Specification and validation of the business process execution language for web services. In *Abstract State Machines*, pages 78–94, 2004.

[82] P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Int. Journal of Robotics and Autonomous Systems*, 23(3-4):159–182, 1998.

[83] P. Faratin, C. Sierra, and N.R. Jennings. Negotiation decision functions for autonomous agents. *Int. Journal of Robotics and Autonomous Systems*, 24(3-4):159–182, 1998.

[84] A. Farrell, M. Sergot, C. Bartolini, M. Salle, D. Trastour, and A. Christodoulou. Using the Event Calculus for the performance monitoring of service-level agreements for utility computing. In *Proceedings of First IEEE International Workshop on Electronic Contracting (WEC 2004)*, 2004.

[85] David F. Ferraiolo, Ravi Sandhu, Serban Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001.

[86] Gian-Luigi Ferrari, Stefania Gnesi, Ugo Montanari, and Marco Pistore. A model-checking verification environment for mobile processes. *ACM Trans. Softw. Eng. Methodol.*, 12(4):440–473, 2003.

[87] FIPA. FIPA standard status specifications. `http://www.fipa.org/repository/standardspecs.html`.

[88] Document Title Fipa. FIPA Communicative Act Library Specification, 2003.

[89] Steve Fisher. Relational model for information and monitoring. Technical Report GWD-GP-7-1, Global Grid Forum, 2001.

[90] Howard Foster, Sebastian Uchitel, Jeff Magee, and Jeff Kramer. Compatibility verification for web service choreography. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, page 738, Washington, DC, USA, 2004. IEEE Computer Society.

[91] Howard Foster, Sebastian Uchitel, Jeff Magee, and Jeff Kramer. Compatibility verification for web service choreography. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, page 738, Washington, DC, USA, 2004. IEEE Computer Society.

[92] Howard Foster, Sebastian Uchitel, Jeff Magee, and Jeff Kramer. LTSA-WS: a tool for model-based verification of web service compositions and choreography. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 771–774, New York, NY, USA, 2006. ACM.

[93] Ian Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, R. Subramaniam, J. Treadwell, and J. von Reich. The open grid services architecture, version 1.0. Informational Document GFD-I.030, Global Grid Forum, January 2005.

[94] K. Frikken, M. Atallah, and J. Li. Hidden access control policies with hidden credentials. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society (WPES '04)*, pages 27–28, New York, NY, USA, 2004. ACM.

[95] Svend Frølund and Jari Koistinen. Quality of services specification in distributed object systems design. *COOTS'98: Proceedings of the 4th conference on USENIX Conference on Object-Oriented Technologies and Systems (COOTS)*, 5(4):179–202, 1998.

[96] X. Fu, T. Bultan, and J. Su. Analysis of interacting BPEL web services. In *Proceedings of the 13th International World Wide Web Conference (WWW'04)*, 2004.

[97] M.G. Fugini, B. Pernici, and F. Ramoni. Quality analysis of composed services through fault injection. *Information System Frontiers, Special Issue on Collaborative Business Processes*, in press.

[98] Jose Garcia-Fanjul, Claudio de la Riva, and Javier Tuya. Generation of Conformance Test Suites for Compositions of Web Services Using Model Checking. In *TAIC-PART '06: Proceedings of the Testing: Academic & Industrial Conference on Practice And Research Techniques*, pages 127–130. IEEE Computer Society, 2006.

[99] J. Garofalakis, Y. Panagis, E. Sakkopoulos, and A. Tsakalidis. Contemporary Web Service Discovery Mechanisms. *Journal of Web Engineering*, 5(3):265–290, 2006.

[100] R. Gavriloaie, W. Nejdl, D. Olmedilla, K. Seamons, and M. Winslett. No registration needed: How to use declarative policies and negotiation to access sensitive resources on the semantic web, 2004.

[101] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, 1991.

[102] Carlo Ghezzi and Sam Guinea. Run-time monitoring in service-oriented architectures. In Luciano Baresi and Elisabetta Di Nitto, editors, *Test and Analysis of Web Services*, pages 237–264. Springer, 2007.

[103] Ester Giallonardo and Eugenio Zimeo. More semantics in QoS matching. In *International Conference on Service-Oriented Computing and Applications*, pages 163–171, Newport Beach, CA, USA, 2007. IEEE Computer Society.

[104] S. Gilmore, V. Haenel, L. Kloul, and M. Maidl. Choreographing security and performance analysis for web services. In *EPEW/WS-FM*, volume 3670 of *Lecture Notes in Computer Science*, pages 200–214. Springer, 2005.

[105] S. Gilmore and L. Kloul. A unified tool for performance modelling and prediction. *Reliability Engineering and System Safety*, 89:17–32, 2005.

[106] H. Gimpel, H. Ludwig, A. Dan, and R. Kearney. PANDA: Specifying policies for automated negotiations of service contracts. In *Proceedings of the 1st International Conference on Service Oriented Computing*, Trento, Italy, 2003.

[107] Tyrone Grandison and Morris Sloman. A Survey of Trust in Internet Applications. *IEEE Communications Surveys and Tutorials*, 3(4), September 2000. http://www.comsoc.org/livepubs/surveys/public/2000/dec/index.html.

[108] Christian W. Günther and Wil M. P. van der Aalst. Fuzzy mining - adaptive process simplification based on multi-perspective metrics. In *Business Process Management, 5th International Conference, BPM*, pages 328–343, 2007.

[109] S. Hanna and M. Munro. An approach for specification-based test case generation for Web services. In *IEEE/ACS International Conference on Computer Systems and Applications, AICCSA 2007*, pages 16–23, 2007.

[110] Y. He and M. Zhu. A complete and efficient strategy based on Petri Nets in automated trust negotiation. In *Proceedings of the 2nd international conference on Scalable information systems (InfoScale)*, pages 1–7, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[111] Reiko Heckel and Marc Lohmann. Towards Contract-based Testing of Web Services. In *Proceedings of the International Workshop on Test and Analysis of Component Based Systems (TACoS 2004)*, volume 116 of *Electronic Notes in Theoretical Computer Science*, pages 145 – 156. Elsevier B.V., 2005.

[112] Reiko Heckel and Leonardo Mariani. Automatic conformance testing of web services. In *In Proceedings Fundamental Approaches to Software Engineering (FASE 05)*, LNCS, pages 34 – 48, 2005.

[113] Thomas A. Henzinger, Ranjit Jhala, Rupak Majumdar, and Gregoire Sutre. Lazy abstraction. In *POPL '02: Proceedings of the 29th ACM SIGPLAN-SIGACT Annual Symposium on Principles of Programming Languages*, pages 58–70, Portland, Oregon, USA, 2002. ACM.

[114] Jerry R. Hobbs and Feng Pan. An ontology of time for the semantic web. *ACM Trans. Asian Lang. Inf. Process.*, 3(1):66–85, 2004.

[115] Gerard J. Holzmann. *The SPIN Model Checker : Primer and Reference Manual*. Addison-Wesley Professional, September 2003.

[116] J. Huai, H. Sun, C. Hu, Y. Zhu, Y. Liu, and J. Li. Rost: Remote and hot service deployment with trustworthiness in crown grid. *Future Generation Computer Systems*, 23(6):825–835, 2007.

[117] Hai Huang, Wei-Tek Tsai, Raymond Paul, and Yinong Chen. Automated Model Checking and Testing for Composite Web Services. In *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2005)*, pages 300–307. IEEE Computer Society, 2005.

[118] K. Irwin and T. Yu. Preventing attribute information leakage in automated trust negotiation. In *Proceedings of the 12th ACM conference on Computer and communications security (CCS '05)*, pages 36–45, New York, NY, USA, 2005. ACM.

[119] Jun-Jang Jeng, Josef Schiefer, and Henry Chang. An agent-based architecture for analyzing business processes of real-time enterprises. In *EDOC '03: Proceedings of the 7th International Conference on Enterprise Distributed Object Computing*, page 86, 2003.

[120] N.R. Jennings, P. Faratin, A.R. Lomuscio, S. Parsons, M.J. Wooldridge, and C. Sierra. Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation*, 10(2):199–215, 2001.

[121] Kurt Jensen. Coloured Petri Nets – Basic concepts, analysis methods and practical use. In *Monographs in Theoretical Computer Science (2nd Edition)*, volume 1: Basic Concepts. Springer-Verlag, 1997.

[122] M. Karam, H. Safa, and H. Artail. An abstract workflow-based framework for testing composed web services. In *International Conference on Computer Systems and Applications (AICCSA)*, pages 901–908, 2007.

[123] R. Kazhamiakin, P. K. Pandya, and M. Pistore. Representation, Verification, and Computation of Timed Properties in Web Service Compositions. In *Proceeding of the International Conference on Web Services (ICWS)*, pages 497–504, 2006.

[124] R. Kazhamiakin and M. Pistore. Analysis of Realizability Conditions for Web Service Choreographies. In *Proceedings Formal Techniques for Networked and Distributed Systems (FORTE)*, pages 61–76, 2006.

[125] R. Kazhamiakin and M. Pistore. Choreography Conformance Analysis: Asynchronous Communications and Information Alignment. In *Proceedings of the International Workshon on Web Services and Formal Methods (WS-FM)*, pages 227–241, 2006.

[126] R. Kazhamiakin and M. Pistore. Static Verification of Control and Data in Web Service Compositions. In *Proceeding of the International Conference on Web Services (ICWS)*, 2006.

[127] R. Kazhamiakin, M. Pistore, and M. Roveri. Formal Verification of Requirements using SPIN: A Case Study on Web Services. In *Proceedings of the International Conference on Software Engineering and Formal Methods (SEFM)*, pages 406–415, 2004.

[128] R. Kazhamiakin, M. Pistore, and M. Roveri. A Framework for Integrating Business Processes and Business Requirements. In *Proceedings of the International Enterprise Distributed Object Computing Conference (EDOC)*, pages 9–20, 2004.

[129] R. Kazhamiakin, M. Pistore, and L. Santuari. Analysis of Communication Models in Web Service Compositions. In *Proceeding of the International Conference on World Wide Web (WWW)*, 2006.

[130] Raman Kazhamiakin. *Formal Analysis of Web Service Compositions*. PhD thesis, University of Trento, 2007.

[131] Alexander Keller and Heiko Ludwig. The WSLA framework: Specifying and monitoring service level agreements for web services. *Journal of Network and Systems Management*, 11(1):57–81, 2003.

[132] ChangSup Keum, Sungwon Kang, In-Young Ko, Jongmoon Baik, and Young-Il Choi. Generating test cases for Web services using extended finite state machine. In *Proceedings of Testing of Communicating Systems. 18th IFIP/WG6.1 International Conference, TestCom 2006*, Lecture Notes in Computer Science Vol. 3964, pages 103–117, 2006.

[133] Barbara Kitchenham. Guidelines for performing Systematic Literature Reviews in Software Engineering. Technical Report EBSE Technical Report EBSE-2007-001, Software Engineering Group, School of Computer Science and Mathematics, Keele University, 2007.

[134] Christos Kloukinas and George Spanoudakis. A pattern-driven framework for monitoring security and dependability. In *Trust, Privacy and Security in Digital Business, 4th International Conference, TrustBus*, pages 210–218, 2007.

[135] Mariya Koshkina and Franck van Breugel. Modelling and verifying web service orchestration by means of the concurrency workbench. *SIGSOFT Softw. Eng. Notes*, 29(5):1–10, 2004.

[136] Hristo Koshutanski and Fabio Massacci. Interactive credential negotiation for stateful business processes. In *iTrust*, pages 256–272, 2005.

[137] Kyriakos Kritikos and Dimitris Plexousakis. Semantic qos metric matching. In *ECOWS '06: Proceedings of the European Conference on Web Services*, pages 265–274, Zurich, Switzerland, 2006. IEEE Computer Society.

[138] Kyriakos Kritikos and Dimitris Plexousakis. Requirements for QoS-based web service description and discovery. *compsac*, 02:467–472, 2007.

[139] Kyriakos Kritikos and Dimitris Plexousakis. Requirements for QoS-based web service description and discovery. In *COMPSAC '07: Proceedings of the 31st Annual International Computer Software and Applications Conference - Vol. 2- (COMPSAC 2007)*, pages 467–472, Washington, DC, USA, 2007. IEEE Computer Society.

[140] Kyriakos Kritikos and Dimitris Plexousakis. Semantic QoS-based web service discovery algorithms. In *ECOWS '07: Proceedings of the Fifth European Conference on Web Services*, pages 181–190, Halle, Germany, 2007. IEEE Computer Society.

[141] Guoming Lai, Cuihong Li, Katia Sycara, and Joseph Andrew Giampapa. Literature review on multi-attribute negotiations. Technical Report CMU-RI-TR-04-66, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 2004.

[142] S. Lamparter, S. Luckner, and S. Mutschelr. Formal specification of Web service contracts for automated contracting and monitoring. In *Proceedings of the 40th Hawaii International Conference on System Sciences*, pages 63–73, Honolulu, Hawaii, 2007.

[143] Steffen Lamparter, Stefan Luckner, and Sybille Mutschler. Formal specification of web service contracts for automated contracting and monitoring. In *HICSS '07: Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, page 63, Washington, DC, USA, 2007. IEEE Computer Society.

[144] Alexander Lazovik, Marco Aiello, and Mike P. Papazoglou. Associating assertions with business processes and monitoring their execution. In *Service-Oriented Computing - ICSOC 2004, Second International Conference*, pages 94–104, 2004.

[145] A. Lee, M. Winslett, J. Basney, and V. Welch. Traust: a trust negotiation-based authorization service for open systems. In *in SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies. New York*. ACM, 2006.

[146] A. J. Lee, M. Winslett, J. Basney, and V. Welch. The Traust authorization service. *ACM Trans. Inf. Syst. Secur.*, 11(1):1–33, 2008.

[147] J. Li, J. Huai, J. Xu, Y. Zhu, and W. Xue. Tower: Practical trust negotiation framework for grids. *Second IEEE International Conference on e-Science and Grid Computing (e-Science'06)*, 0:26, 2006.

[148] N. Li and J. Mitchell. Rt: A role-based trust-management framework. In *The Third DARPA Information Survivability Conference and Exposition (DISCEX III), April 2003.*, 2003.

[149] Yutu Liu, Anne H. H. Ngu, and Liangzhao Zeng. QoS computation and policing in dynamic web service selection. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *WWW (Alternate Track Papers & Posters)*, pages 66–73, New York, NY, USA, 2004. ACM.

[150] Marc Lohmann, Leonardo Mariani, and Reiko Heckel. *A Model-Driven Approach to Discovery, Testing and Monitoring of Web Services*, pages 173 – 204. Springer, 2007.

[151] Heiko Ludwig, Asit Dan, and Robert Kearney. Cremona: An architecture and library for creation and monitoring of WS-Agreents. In *Service-Oriented Computing - ICSOC 2004, Second International Conference*, pages 65–74, 2004.

[152] Heiko Ludwig, Alexander Keller, Asit Dan, Richard P. King, and Richard Franck. Web Service Level Agreement (WSLA) Language Specification. Technical report, IBM Corporation, 2003.

[153] Daniel Luebke. *Unit Testing BPEL Compositions*, pages 149 – 171. Springer, 2007.

[154] Nan Luo, Junwei Yan, and Min Liu. Towards efficient verification for process composition of semantic web services. In *IEEE SCC* [6], pages 220–227.

[155] Khaled Mahbub and George Spanoudakis. Run-time monitoring of requirements for systems composed of web-services: Initial implementation and evaluation experience. In *2005 IEEE International Conference on Web Services (ICWS 2005)*, pages 257–265, 2005.

[156] Khaled Mahbub and George Spanoudakis. Monitoring S-Agreement: An event calculus-based approach. In Luciano Baresi and Elisabetta Di Nitto, editors, *Test and Analysis of Web Services*, pages 265–306. Springer, 2007.

[157] A. Mani and A. Nagarajan. Understanding quality of service for web services. `http://www-128.ibm.com/developerworks/library/ws-quality.html`, 2002.

[158] Evan Martin, Suranjana Basu, and Tao Xie. Automated Robustness Testing of Web Services. In *Proc. 4th International Workshop on SOA And Web Services Best Practices (SOAWS 2006)*, 2006.

[159] Evan Martin, Suranjana Basu, and Tao Xie. Automated Testing and Response Analysis of Web Services. In *IEEE International Conference on Web Services (ICWS)*, pages 647 – 654, 2007.

[160] Evan Martin, Suranjana Basu, and Tao Xie. WebSob: A tool for robustness testing of web services. In *Companion to the proceedings of the 29th International Conference on Software Engineering (ICSE)*, pages 65–66, 2007.

[161] M. Marzolla and R. Mirandola. Performance prediction of web service workflows. 4880:127–144, 2007.

[162] Claudio Masolo, Stefano Borgo, Aldo Gangemi, Nicola Guarino, Alessandro Oltramari, and Luc Schneider. Wonderweb deliverable d17. the wonderweb library of foundational ontologies and the dolce ontology.

[163] E. Michael Maximilien and Munindar P. Singh. Conceptual model of web service reputation. *SIGMOD Rec.*, 31(4):36–41, 2002.

[164] E. Michael Maximilien and Munindar P. Singh. A framework and ontology for dynamic web services selection. *IEEE Internet Computing*, 8(5):84–93, 2004.

[165] P. Mayer and D. Luebke. Towards a BPEL unit testing framework. In *Proceedings of the 2006 Workshop on Testing, Analysis, and Verification of Web Services and Applications, TAV WEB'06*, volume 2006, pages 33–42, 2006.

[166] U.M. Mbanaso, G.S. Cooper, D.W. Chadwick, and Seth Proctor. Privacy preserving trust authorization framework using XACML. In *2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks(WoWMoM'06)*, pages 673–678. IEEE, 2006.

[167] J.D. McGregor and D.A. Sykes. *A Practical Guide to Testing Object-oriented Software*. Addison-Wesley Professional, 2001.

[168] Massimo Mecella, Francesco Parisi-Presicce, and Barbara Pernici. Modeling e-service orchestration through Petri Nets. In *TES '02: Proceedings of the Third International Workshop on Technologies for E-Services*, pages 38–47, London, UK, 2002. Springer-Verlag.

[169] Hong Mei and Lu Zhang. A Framework for Testing Web Services and Its Supporting Tool. In *SOSE '05: Proceedings of the IEEE International Workshop*, pages 207–214. IEEE Computer Society, 2005.

[170] D. Menascé and V. Dubey. Utility-based QoS brokering in service oriented architectures. In *Proceedings of the 2007 International Conference on Web services*, 2007.

[171] Christof Momm, Robert Malec, and Sebastian Abeck. Towards a model-driven development of monitored processes. *Wirtschaftsinformatik*, 2, 2007.

[172] N. K. Mukhi and P. Plebani. Supporting policy-driven behaviors in Web services: experiences and issues. In *Proceedings of the 2nd International Conference on Service Oriented Computing*, New York, NY, 2004.

[173] G.J. Myers. *The Art of Software Testing*. Wiley, 2004.

[174] Anthony Nadalin, Marc Goodner, Martin Gudgin, Abbie Barbir, and Hans Granqvist. WS-Trust specification, http://www.ibm.com/developerworks/webservices/library/specification/ws-trust/. In *Technical report*. OASIS Working Draft, 2007.

[175] S. Nakajima. Model-checking verification for reliable web service. In *OOPSLA Workshop on Object-Oriented Web Services*, 2002.

[176] Srini Narayanan and Sheila A. McIlraith. Simulation, verification and automated composition of web services. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 77–88, Honolulu, Hawaii, USA, 2002. ACM.

[177] Srinivas Narayanan. Reasoning about actions in narrative understanding. In *IJCAI '99: Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 350–357, Stockholm, Sweden, 1999. Morgan Kaufmann Publishers Inc.

[178] A. Ncho and E. Aimeur. Building a multi-agent system for automated negotiation in Web service applications. In *In. Proc. of AAMAS'04*, 2004.

[179] Harvey B. Newman, I.C. Legrand, Philippe Galvez, and R. Voicu. MonALISA: A distributed monitoring service architecture. In *International Conference on Computing in High Energy Physics (CHEP2003*, 2003.

[180] Hamid R. Motahari Nezhad, Regis Saint-Paul, Boualem Benatallah, and Fabio Casati. Deriving protocol models from imperfect service conversation logs. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2008. to appear.

[181] Jeff Offutt and Wuzhi Xu. Generating Test Cases for Web Services Using Data Perturbation. In *Workshop on Testing, Analysis and Verification of Web Services*, 2004.

[182] N. Oldham, K. Verma, A. Sheth, and F. Hakimpour. Semantic WS-agreement partner selection. In *WWW '06: Proceedings of the 15th International conference on World Wide Web*, pages 697–706, Edinburgh, Scotland, 2006. ACM Press.

[183] Nicole Oldham, Kunal Verma, Amit Sheth, and Farshad Hakimpour. Semantic WS-agreement partner selection. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 697–706, New York, NY, USA, 2006. ACM.

[184] A. D. Oliver-Lalana. Consent as a threat. A critical approach to privacy negotiation in e-commerce practices. In *Trust and Privacy in Digital Business, First International Conference, TrustBus 2004*, pages 110–119. Springer, 2004.

[185] D. Olmedilla, R. Lara, Axel Polleres, and H. Lausen. Trust negotiation for semantic web services. In *Semantic Web Services and Web Process Composition, First International Workshop, (SWSWPC)*, pages 81–95. Springer, 2004.

[186] L. E. Olson, M. J. Rosulek, and M. Winslett. Harvesting credentials in trust negotiation as an honest-but-curious adversary. In *Proceedings of the 2007 ACM workshop on Privacy in electronic society (WPES '07)*, pages 64–67, New York, NY, USA, 2007. ACM.

[187] Lars Olson, Marianne Winslett, Gianluca Tonti, Nathan Seeley, Andrzej Uszok, and Jeffrey Bradshaw. Trust negotiation as an authorization service forweb services. In *ICDEW '06: Proceedings of the 22nd International Conference on Data Engineering Workshops*, page 21, Washington, DC, USA, 2006. IEEE Computer Society.

[188] Leon J. Osterweil. Strategic directions in software quality. *ACM Comput. Surv.*, 28(4):738–750, 1996.

[189] J. O'Sullivan, D. Edmond, and A.H.M. ter Hofstede. Formal description of non-functional service properties. Technical report, Queensland University of Technology, 2005.

[190] C. Ouyang, W.M.P. van der Aalst, S. Breutel, M. Dumas, A.H.M. ter Hofstede, and H.M.W. Verbeek. Formal Semantics and Analysis of Control Flow in WS-BPEL. Technical report, BPMcenter.org, 2005. BPM Center Report BPM-05-15.

[191] C. Ouyang, H.M.W. Verbeek, W.M.P. van der Aalst, S. Breutel, M. Dumas, and A.H.M ter Hofstede. WofBPEL: A Tool for Automated Analysis of BPEL Pprocesses. In *Proceeding of the International Conference on Service-Oriented Computing (ICSOC)*, 2005.

[192] M. Papazoglou and W.-J. van den Heuvel. Service oriented architectures: approaches, technologies and research issues. *VLDB Journal*, 16:389–415, 2007.

[193] Mike Papazoglou and Klaus Pohl. Report on longer term research challenges in software and services. Results from two workshops held at the European Commission premises at 8th of November 2007 and 28th and 29th of January 2008, European Commission, www.cordis.lu, 2008. With contributions from Boniface M, Ceri S, Hermenegildo M, Inverardi P, Leymann F, Maiden N, Metzger A, Priol T.

[194] M.P. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann. Service-Oriented Computing: State of the art and research challenges. *IEEE Computer*, 11:38–45, 2007.

[195] A.M. Paradkar, A. Sinha, C. Williams, R.D. Johnson, S. Outterson, C. Shriver, and C Liang. Automated Functional Conformance Test Generation for Semantic Web Services. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 110–117, 2007.

[196] Michael Parkin, R.M. Badia, and Josep Martrat. A comparison of SLA use in six of the european commissions FP6 projects. Technical report, TR-0129, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, April 2008.

[197] P.Bonatti and P.Samarati. Regulating service access and information release on the web. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 134–143, New York, NY, USA, 2000. ACM.

[198] M. Di Penta, M. Bruno, G. Esposito, V. Mazza, and G. Canfora. *Web Services Regression Testing*, pages 205 – 234. Springer, 2007.

[199] Marco Pistore and Paolo Traverso. Assumption-based composition and monitoring of web services. In Luciano Baresi and Elisabetta Di Nitto, editors, *Test and Analysis of Web Services*, pages 307–335. Springer, 2007.

[200] S. Preibusch. Implementing privacy negotiations in e-commerce. In *Frontiers of WWW Research and Development - APWeb 2006*, pages 604–615. Springer, 2006.

[201] Shaz Qadeer, Sriram K. Rajamani, and Jakob Rehof. Summarizing procedures in concurrent programs. *ACM SIGPLAN Notices*, 39(1):245–255, 2004.

[202] P. Radha Krishna, K. Karlapalem, and D.K.W. Chiu. An ER^{EC} framework for e-contract modeling, enactment, and monitoring. *Data Knowl. Eng.*, 51:31–58, 2004.

[203] P. Ramsokul, A. Sowmya, and S. Ramesh. A test bed for web services protocols. In *Second International Conference on Internet and Web Applications and Services (ICIW)*, 2007.

[204] S. Ran. A model for web services discovery with QoS. *SIGecom Exch.*, 4(1):1–10, 2003.

[205] Shuping Ran. A model for web services discovery with qos. *SIGecom Exch.*, 4(1):1–10, 2003.

[206] Thomas C. Redman. *Data Quality for the Information Age*. Artech House, Inc., Norwood, MA, USA, 1997. Foreword By-A. Blanton Godfrey.

[207] Sidney Rosario, Albert Beneveniste, S. Haar, and Claude Jard. Probablistic QoS and soft contracts for transaction based web services. In *IEEE ICWS*, pages 126–133, 2007.

[208] Heinz Roth, Josef Schiefer, and Alexander Schatten. Probing and monitoring of WSBPEL processes with web services. In *CEC-EEE '06: Proceedings of the The 8th IEEE International Conference on E-Commerce Technology and The 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services*, page 30, 2006.

[209] Mohsen Rouached, Olivier Perrin, and Claude Godart. Towards formal verification of web service composition. In *Business Process Management*, pages 257–273, 2006.

[210] A. Rozinat and Wil M. P. van der Aalst. Conformance testing: Measuring the fit and appropriateness of event logs and process models. In Christoph Bussler and Armin Haller, editors, *Business Process Management Workshops*, pages 163–176, 2006.

[211] Anne Rozinat and Wil M. P. van der Aalst. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.*, 33(1):64–95, 2008.

[212] D. Rud, A. Schmietendorf, and R. Dumke. Performance modeling of WS-BPEL-based web service compositions. *scw*, 0:140–147, 2006.

[213] Sini Ruohomaa and Lea Kutvonen. Trust management survey. In *Proceedings of the iTrust 3rd International Conference on Trust Management*. LNCS 3477, 23-26may 2005.

[214] M. Ruth, S. Oh, A. Loup, B. Horton, O. Gallet, M. Mata, and S. Tu. Towards automatic regression test selection for web services. In *Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007)*, volume 2, pages 729–734, 2007.

[215] M. Ruth and Shengru Tu. A safe regression test selection technique for Web services. In *Second International Conference on Internet and Web Applications and Services (ICIW)*, 2007.

[216] Michael E. Ruth. Concurrency in a decentralized automatic regression test selection framework for web services. In *MG '08: Proceedings of the 15th ACM Mardi Gras conference*, pages 1–8. ACM, 2008.

[217] T. Ryutov, L. Zhou, C. Neuman, T. Leithead, and K. E. Seamons. Adaptive trust negotiation and access control. In *Proceedings of the tenth ACM symposium on Access control models and technologies (SACMAT '05)*, pages 139–146, New York, NY, USA, 2005. ACM.

[218] B. Sabata, S. Chatterjee, M. Davis, J.J. Sydir, and T.F. Lawrence. Taxonomy for QoS Specifications. In *Object-Oriented Real-Time Dependable Systems, 1997. Proceedings., Third International Workshop on*, pages 100–107, 5-7 Feb. 1997.

[219] Akhil Sahai, Anna Durante, and Vijay Machiraju. Towards automated SLA management for web services. Technical Report HPL-2001-310, HP Laboratories, Palo Alto, CA, July 2002.

[220] Akhil Sahai, Vijay Machiraju, Mehmet Sayal, Aad P. A. van Moorsel, and Fabio Casati. Automated SLA monitoring for web services. In *13th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2002*, pages 28–41, 2002.

[221] Rizos Sakellariou and Viktor Yarmolenko. On the flexibility of WS-Agreement for job submission. In *Proceedings of the 3rd International Workshop on Middleware for Grid Computing (MGC'05)*, 2005.

[222] Rizos Sakellariou and Viktor Yarmolenko. *High Performance Computing and Grids in Action*, chapter Job Scheduling on the Grid: Towards SLA-Based Scheduling. March 2008.

[223] Gwen Salaün, Lucas Bordeaux, and Marco Schaerf. Describing and reasoning on web services using process algebra. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, page 43, Washington, DC, USA, 2004. IEEE Computer Society.

[224] I. Schieferdecker, G. Din, and D. Apostolidis. Distributed functional and load tests for Web services. *International Journal on Software Tools for Technology Transfer*, 7(4):351–360, 2005.

[225] Bernd-Holger Schlingloff, Axel Martens, and Karsten Schmidt. Modeling and model checking web services. In *Proceedings of the 2nd International Workshop on Logic and Communication in Multi-Agent Systems*, volume 126. Elsevier, 2005.

[226] K. E. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis. Protecting privacy during on-line trust negotiation. In *Privacy Enhancing Technologies, Second International Workshop (PET' 2002)*, pages 129–143, 2002.

[227] Jan Seidel, Oliver Wäldrich, Wolfgang Ziegler, Philipp Wieder, and Ramin Yahyapour. a survey. Using SLA for resource management and scheduling. Technical report, TR-0096, Institute on Resource Management and Scheduling, CoreGRID - Network of Excellence, August 2007.

[228] Sagar Sen, Benoit Baudry, and Jean-Marie Mottu. On combining mullti-formalism knowledge to select test models for model transformaion testing. In *ACM/IEEE International Conference on Software Testing*, Lillehammer, Norway, April 2008.

[229] Natasha Sharygina and Daniel Krning. Model checking with abstraction for web services. In *Test and Analysis of Web Services*, pages 121–145, 2007.

[230] H. Shen and F.Hong. An attribute-based access control model for web services. In *Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2006)*, pages 74–79, 2006.

[231] R. Siblini and N. Mansour. Testing Web services. In *ACS/IEEE 2005 International Conference on Computer Systems and Applications (AICCSA)*, page 135. IEEE Computer Society, 2005.

[232] A. Sinha and A. Paradkar. Model-based functional conformance testing of Web services operating on persistent data. In *Proceedings of the 2006 workshop on Testing, analysis, and verification of web services and applications*, volume 2006, pages 17–22, 2006.

[233] H. Skogsrud, B. Benatallah, and F. Casati. Trust-Serv: Model-driven lifecycle management of trust negotiation policies for web services. In *Proc. 13th World Wide Web Conf.*, May 2004.

[234] H. M. Sneed and S. Huang. WSDLTest - A tool for testing web services. In *Proceedings of the Eighth IEEE International Symposium on Web Site Evolution (WSE'06)*, pages 14–21, 2006.

[235] Alan Snyder. How to get your paper accepted at OOPSLA. In *Conference on Object Oriented Programming Systems Languages and Applications*, pages 359–363. ACM Press New York, NY, USA, 1991.

[236] I. Sommerville. *Software Engineering. 4th edition*. Addison Wesley, 1992.

[237] George Spanoudakis, Christos Kloukinas, and Kelly Androutsopoulos. Towards security monitoring patterns. In *Proceedings of the 2007 ACM Symposium on Applied Computing (SAC)*, pages 1518–1525, 2007.

[238] George Spanoudakis and Khaled Mahbub. Requirements monitoring for service-based systems: Towards a framework based on Event Calculus. In *19th IEEE International Conference on Automated Software Engineering (ASE 2004), 20-25 September 2004, Linz, Austria*, pages 379–384, 2004.

[239] A. Squicciarini, E. Bertino, E. Ferrari, F. Paci, and B. Thuraisingham. PP-trust-X: A system for privacy preserving trust negotiations. *ACM Transactions on Information and System Security (TISSEC)*, 10(3):12, 2007.

[240] Diane M. Strong, Yang W. Lee, and Richard Y. Wang. 10 potholes in the road to information quality. *Computer*, 30(8):38–46, 1997.

[241] K. Sycara et al. *OWL-S 1.0 Release*. OWL-S Coalition, http://www.daml.org/services/owl-s/1.0/, 2003.

[242] Y. Tang, L. Chen, K. T. He, and N. Jing. SRN: An Extended Petri-Net-Based Workflow Model for Web Service Composition. In *Proceeding of the International Conference on Web Services (ICWS)*, pages 591–599, 2004.

[243] A. Tarhini, H. Fouchal, and N. Mansour. A simple approach for testing Web service based applications. In *Innovative Internet Community Systems. 5th International Workshop*, Lecture Notes in Computer Science Vol.3908, pages p. 134–146, 2006.

[244] The OASIS Group. Quality model for web services. Technical report, The Oasis Group, September 2005.

[245] The OMG Group. Uml^{TM} profile for modeling quality of service and fault tolerance characteristics and mechanisms. Technical Report ptc/2005-05-02, The OMG Group, May 2005.

[246] M. Tian, A. Gramm, M. Nabulsi, H. Ritter, J. Schiller, and T. Voigt. Qos integration in web services. Gesellschaft fur Informatik DWS 2003, Doktorandenworkshop Technologien und Anwendungen von XML, October 2003.

[247] Brian Tierney, Ruth A. Aydt, Dan Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany. A grid monitoring architecture. Informational Document GFD-I.7, Global Grid Forum, January 2002.

[248] Vladimir Tosic, Babak Esfandiari, Bernard Pagurek, and Kruti Patel. On requirements for ontologies in management of web services. In *CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web*, pages 237–247, Toronto, Ontario, Canada, 2002. Springer-Verlag.

[249] Vladimir Tosic, Bernard Pagurek, and Kruti Patel. WSOL - A language for the formal specification of classes of service for web services. In Liang-Jie Zhang, editor, *ICWS*, pages 375–381. CSREA Press, June 2003.

[250] Vladimir Tosic, Kruti Patel, and Bernard Pagurek. WSOL - Web Service Offerings Language. In *CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web*, pages 57–67, London, UK, 2002. Springer-Verlag.

[251] Hong-Linh Truong and Thomas Fahringer. SCALEA-G: a Unified Monitoring and Performance Analysis System for the Grid. *Scientific Programming*, 12(4):225–237, November 2004. AxGrids 2004 Special Issue.

[252] W. . Tsai, Y. Chen, R. Paul, H. Huang, X. Zhou, and X. Wei. Adaptive testing, oracle generation, and test case ranking for web services. In *29th Annual International Computer Software and Applications Conference (COMPSAC)*, volume 1, pages 101–106, 2005.

[253] W. Tsai, X. Wei, Y. Chen, R. Paul, and B. Xiao. Swiss cheese test case generation for web services testing. *IEICE Transactions on Information and Systems*, E88-D(12):2691–2698, 2005.

[254] W. T. Tsai, Y. Chen, Z. Cao, X. Bai, H. Huang, and R. Paul. Testing Web Services Using Progressive Group Testing. In *Advanced Workshop on Content Computing*, pages 314–322, 2004.

[255] W. T. Tsai, R. Paul, L. Yu, A. Saimi, and Z. Cao. Scenario-Based Web Services Testing with Distributed Agents. *IEICE Transaction on Information and System*, E86-D(10):2130 – 2144, 2003.

[256] W. T. Tsai, Ray Paul, Yamin Wang, Chun Fan, and Dong Wang. Extending WSDL to Facilitate Web Services Testing. In *7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02)*, volume 00, page 171. IEEE Computer Society, 2002.

[257] W. T. Tsai, X. Wei, Y. Chen, and R. Paul. A Robust Testing Framework for Verifying Web Services by Completeness and Consistency Analysis. In *SOSE '05: Proceedings of the IEEE International Workshop*, pages 159–166. IEEE Computer Society, 2005.

[258] W. T. Tsai, X. Wei, Y. Chen, B. Xiao, R. Paul, and H. Huang. Developing and assuring trustworthy web services. In *ISADS 2005: Proceedings of the 7th International Symposium on Autonomous Decentralized Systems*, pages 43–50, Chengdu, China, 2005. IEEE Computer Society.

[259] W. T. Tsai, Dawei Zhang, Raymond Paul, and Yinong Chen. Stochastic Voting Algorithms for Web Services Group Testing. In *QSIC '05: Proceedings of the Fifth International Conference on Quality Software*, pages 99–108. IEEE Computer Society, 2005.

[260] Wei-Tek Tsai, Yinong Chen, and Ray Paul. Specification-based verification and validation of web services and service-oriented operating systems. *WORDS*, 0:139–147, 2005.

[261] Wei-Tek Tsai, Raymond A. Paul, Weiwei Song, and Zhibin Cao. Coyote: An XML-Based Framework for Web Services Testing. In *Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE)*, pages 173 – 176, 2002.

[262] W.T. Tsai, X. Bai, Y. Chen, and X. Zhou. Web Service Group Testing with Windowing Mechanisms. In *IEEE International Workshop on Service-Oriented System Engineering (SOSE)*, pages 213 – 218, 2005.

[263] W.T. Tsai, Y. Chen, R. Paul, N. Liao, and H. Huang. Cooperative and Group Testing in Verification of Dynamic Composite Web Services. In *Workshop on Quality Assurance and Testing of Web-Based Applications, in conjunction with COMPSAC*, pages 170 – 173, 2004.

[264] Dimitrios T. Tsesmetzis, Ioanna G. Roussaki, Ioannis V. Papaioannou, and Miltiades E. Anagnostou. Qos awareness support in web-service semantics. In *AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services*, pages 128–134, Guadeloupe, French Caribbean, 2006. IEEE Computer Society.

[265] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. Kaos policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement. In *POLICY '03: Proceedings of the 4th IEEE International Workshop on Policies for Distributed Systems and Networks*, page 93, Washington, DC, USA, 2003. IEEE Computer Society.

[266] Wil M. P. van der Aalst. Workflow verification: Finding control-flow errors using petri-net-based techniques. In *Business Process Management*, pages 161–183, 2000.

[267] Wil M. P. van der Aalst and Ana Karla A. de Medeiros. Process mining and security: Detecting anomalous process executions and checking process conformance. *Electr. Notes Theor. Comput. Sci.*, 121:3–21, 2005.

[268] Boudewijn F. van Dongen, Ana Karla A. de Medeiros, H. M. W. Verbeek, A. J. M. M. Weijters, and Wil M. P. van der Aalst. The ProM framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets 2005, 26th International Conference, ICATPN*, pages 444–454, 2005.

[269] H.M.W. Verbeek and W.M.P. van der Aalst. Analyzing BPEL Processes using Petri Nets. In *Proceedings of the 2nd International Workshop on Applications of Petri Nets to Coordination, Workflow and Business Process Management*, pages 59–78, 2005.

[270] J. Vonk and P. Grefen. Cross-organizational transaction support for E-services in virtual enterprises. *Distrib. Parallel. Dat.*, 14:137–172, 2003.

[271] Xia Wang, Tomas Vitvar, Mick Kerrigan, and Ioan Toma. A qos-aware selection model for semantic web services. In Asit Dan and Winfried Lamersdorf, editors, *ICSOC*, volume 4294 of *Lecture Notes in Computer Science*, pages 390–401. Springer, 2006.

[272] Y. Wang, X. Bai, J. Li, and R. Huang. Ontology-based test case generation for testing web services. In *Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems*, pages 43–50, 2007.

[273] W.H. Winsborough and N. Li. Safety in automated trust negotiation. *ACM Transactions on Information and System Security (TISSEC)*, 9(3):352–390, 2006.

[274] M. Winslett, T. Yu, K.E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. The TrustBuilder architecture for trust negotiation. *IEEE Internet Computing*, 6(6):30–37, 2002.

[275] Marianne Winslett, Charles C. Zhang, and Piero A. Bonatti. Peeraccess: a logic for distributed authorization. In *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, pages 168–179, New York, NY, USA, 2005. ACM.

[276] Andreas Wombacher, Peter Fankhauser, and Erich Neuhold. Transforming BPEL into annotated deterministic finite state automata for service discovery. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, page 316, Washington, DC, USA, 2004. IEEE Computer Society.

[277] WS-AGREEMENT. WS-Agreement Framework. `https://forge.gridforum.org/projects/graap-wg`, September 2003.

[278] Wuzhi Xu, J. Offutt, and Juan Luo. Testing Web services by XML perturbation. In *Proceedings. 16th IEEE International Symposium on Software Reliability Engineering*, page 10, 2006.

[279] J. Yan, J. Y. Zhang, M.B. Chhetri, J. Lin, S. Goh, and R. Kowalczyk. Towards autonomous service level agreement negotiation for adaptvive service composition. In *In Proc. 10th Int. Conf. on Computer Supported Cooperative Work in Design*, 2006.

[280] X. Yi and K. J. Kochut. A CP-nets-based Design and Verification Framework for Web Services Composition. In *Proceeding of the International Conference on Web Services (ICWS)*, page 756, 2004.

[281] T. Yu and M. Winslett. Policy migration for sensitive credentials in trust negotiation. In *Proceedings of the 2003 ACM workshop on Privacy in the electronic society (WPES '03)*, pages 9–20, New York, NY, USA, 2003. ACM.

[282] Ting Yu, Marianne Winslett, and Kent E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Trans. Inf. Syst. Secur.*, 6(1):1–42, 2003.

[283] Carmen Zannier, Grigori Melnik, and Frank Maurer. On the success of empirical studies in the international conference on software engineering. In *ICSE '06: Proceedings of the 28th international conference on Software engineering*, pages 341–350, New York, NY, USA, 2006. ACM.

[284] S. Zhang and F. Makedon. Privacy preserving learning in negotiation. In *Proceedings of the 2005 ACM symposium on Applied computing (SAC '05)*, pages 821–825, New York, NY, USA, 2005. ACM.

[285] Chen Zhou, Liang-Tien Chia, and Bu-Sung Lee. Daml-qos ontology for web services. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services (ICWS'04)*, pages 472–479, San Diego, CA, USA, 2004. IEEE Computer Society.