Preface

Computational Intelligence (CI) has emerged as a rapid growing field over the last decade. CI techniques, e.g., artificial neural networks, genetic algorithms, fuzzy theory, evolution computing, machine learning, and many others, are powerful tools for intelligent information processing, decision making and knowledge management in organizations ranging from manufacturing firms to medical and healthcare institutions. Some of the recent examples of applying CI techniques include case-based reasoning in medical diagnosis and clinical decision support, customer relationship management through data mining, fault detection and quality improvement using artificial neural networks, design rationale assessment through semantic modeling of design documents, and so on.

The editors of this book have been working on data and text mining, machine learning, information retrieval, digital libraries and related research fields. Some of them have conducted CI research to address problems in industrial application scenarios, e.g., product design and manufacturing process control, for a few years. They are fully aware of the demand for advanced tools to support daily routine tasks like process monitoring as well as to support decision making in strategic planning. Therefore, it is timely to have a book that presents the latest advances in CI topics with a strong focus on industrial applications. This book aims to report and demonstrate how industrial organizations can gain a competitive advantage by applying different CI techniques. The industrial organizations here refer to a wide range of enterprises, bodies from design studios and manufacturing firms to hospitals and non-profit organizations.

The book includes 17 chapters selected from more than 30 submissions based on a thorough and strict peer review process. The 17 chapters accepted were contributed by international researchers from 14 countries and regions. They present a snapshot of the cutting-edge research in CI and the state-of-the-art industrial CI applications. These 17 chapters are broadly grouped into two main sections, namely, Section I: Theory and Foundation, and Section II: Industrial Applications.

In the Theory and Foundation section, Das et al. give a comprehensive overview of particle swarm optimization (PSO) and differential evolution (DE) algorithms. This chapter describes several schemes to control the convergence behaviors of PSO and DE by carefully selecting and tuning their parameters. Furthermore, the authors have explored the possibility of a hybrid approach by combining PSO and DE with other soft computing algorithms which leads to a powerful global search algorithm. Its application in various engineering domains has been highlighted.

The chapter by Maamar et al. covers Web services and their integration with context and policies, a topic not discussed much in CI. In their framework, a three-level approach to compose Web services has been proposed, i.e., component, composite and resource level. In their work, context is defined as the information about the environment within each level, i.e., context of Web service, context of composite Web service and context of resource, and jumping from one level to another requires policy activation, either engagement policies or deployment policies.

Qiu focuses on privacy protection when delegating data mining tasks in industrial systems, one of the prevailing topics in the area. Qiu introduces a Bloom filter-based approach for privacy preserving in association rule mining. His findings show that the privacy security level and analysis precision are positively correlated using the Bloom filter. To increase the privacy security level, more data storage is needed. The additional storage required is in a linear relation with respect to the privacy level or the analysis precision. Further research on other data mining tasks and testing in a distributed database environment are suggested.

Tsivtsivadze et al. present a comprehensive overview of various kernels for text analysis, including bag-of-words kernel, string kernel, gappy string kernel, convolution kernel, and graph kernel. What intrigues us is its unique feature of incorporating prior domain knowledge into kernels for text analysis. An evaluation using different features, e.g., grammatical bigram, word and part-of-speech tag, link type, link length, and random walk features from graph feature representation, shows the importance and contribution of grammatical bigram and link bigram and their merits in embedding prior domain knowledge in feature representation.

Wang et al. report their work on the discovery of frequent patterns from long sequences which may contain many events. Different from the existing approaches in sequential pattern mining that simply count the occurrence of patterns in disjoint sequences, Wang et al.'s work adopts the concept of maximum cardinality to describe the number of pattern occurrences in an individual sequence. The advantages of their approach include not requiring sliding window sizes and efficient implementation with sound precision.

Lin's work on gauging image and video quality in industrial applications highlights an important area which receives little attention in CI. Based on a unified framework and its computational modules, three perceptual visual quality metrics and an embedded just-noticeable difference estimator have

been proposed. More applications of perceptual models in industrial scenarios are expected to further stress the importance of Lin's study.

Finally, after a nice review on model construction, Stein presents an interesting methodology regarding model construction for knowledge intensive engineering tasks. Based on the observation that domain experts and knowledge engineers often develop a new and appropriate model by modifying some existing ones, Stein proposes a means of horizontal modeling construction. A few real-world case studies are provided to demonstrate its practical value.

In Sect. II, we emphasize the state-of-the-art application of CI techniques in various industrial scenarios. We start with a few chapters focusing on medical and healthcare areas. Almeida et al. adopt case based reasoning and genetic algorithm in the implementation of a virtual medical office for medical diagnosis and knowledge transfer and management purpose. In their case study, promising results are achieved.

Chan et al. address the important issue of using data mining approach in clinical decision support. They highlight the complicated reality that a clinical decision is often drawn by synthesizing knowledge from multiple disciplines while information needed is presented in a number of different formats, e.g., magnetic resonance images, somatosensory evoked potentials and radiographs. Hence, Chan et al. propose the DICOM model to form a synergy platform which aims to extract and present relevant features from multiple disciplinary knowledge sources for medical knowledge discovery and clinical decision support purposes. An example is given subsequently.

Caldwell et al. tackle the problem of promoter prediction using neural network for promoter prediction (NNPP) algorithm. They have raised a point that DNA sequence information, which is released through DNA sequence quantitative measurements instead of DNA sequence pattern information, can significantly improve the precision of computational promoter prediction. Their extended algorithms based on three coding region lengths, e.g., TLS–TSC length, TSS–TLS length, and TSC–TTS length, have shown their merits. Future research directions have been outlined.

Das et al. investigate the anomaly detection problem using two successful negative selection algorithms, i.e., the self-organizing RNS algorithm and the V-detector algorithm. For the latter, both a single-stage and a multi-stage proliferating V-detector algorithm are studied to create such vectors.

Braga et al. report their study using a number of pattern classification algorithms, e.g., multi-layer perceptron, RBF neural network, support vector machines, and decision tree, to automatically monitor the dressing procedure in the grinding process of an industrial CNC machine. They also examine the robustness and effectiveness of different algorithms when the controlled noise is purposely introduced in simulation.

Clifton et al. provide a nice survey of the existing methods for detecting novelty or something abnormal in the industrial environment. Using an SVM core, novelty classifiers are set up in a way that their output is calibrated into probabilities, resulting in a probabilistic novelty threshold being specified

automatically. This is in contrast to the existing distance-based methods where heuristics are needed and the models are manually constructed. Their validation and experiments are conducted based on real-world jet-engine data.

Khosravi et al. explore another important problem – predicting the origin of power sags in distributed substations using a multiway principal component analysis (MPCA), and an extension of the classic principal component analysis (PCA). In their study, MPCA classifiers are able to achieve better performance in differentiating high-voltage and medium-voltage sags compared to other possibilities, e.g., multi-layer perceptron, RBF neural network, and decision tree.

Cheong and Tan report their study of using a multi-objective multi-colony ant algorithm (MOMCAA) in solving the berth allocation problem, an important topic in logistics and supply chain management. In their work, berth allocation problem has been taken as an optimization of complete shipping schedules with minimum service time and delay for the departure of ships, subject to a number of temporal and spatial constraints, and hence a multi-objective and multi-modal combinational optimization problem. The proposed MOMCAA framework possesses several features, such as using ant groups in searching for each candidate group, concurrent optimization, flexibility and expandability of incorporating other meta-heuristics, e.g., genetic algorithm and simulated annealing, etc.

Yu's chapter focuses on the application of neural network on dynamic system identification and adaptive control. Two examples are given, i.e., rotorcraft acoustic data modeling and DC voltage controller.

Hammiche et al. describe their work of query rewriting for semantic multimedia data retrieval, a domain knowledge based approach to retrieve multimedia data formatted according to MPEG-7 standard. The basic idea is to add a semantic layer on top of the MPEG-7 metadata layer. To fulfill the retrieval task, users' queries are pre-processed, rewritten and translated into the XQuery queries using the mapping rules and a set of XQuery functions defined over the MPEG-7 descriptions.

We are very grateful to all our reviewers: Danilo Avola, Peter Bannister, Andre Ponce de Leon F. de Carvalho, Kit Yan Chan, Wing Chi Chan, Tan Kay Chen, David Clifton, Sanjoy Das, Chi Keong Goh, Jason J. Jung, Dariusz Krol, Wenyuan Li, Weisi Lin, Yan-Xia Lin, Ying Liu, Kok-Leong Ong, Ling Qiu, Benno Stein, Evgeni Tsivtsivadze, Changzhou Wang, Kevin Wang, Jianshu Weng, Xiao-Hua (Helen) Yu, Yun-Shuai Yu, and Zhiwen Yu. Their backgrounds include data and text mining, information retrieval, digital library, evolution algorithm and agents, image and video processing, machine learning, neural network and genetic algorithms, applied statistics, Web services and ontology engineering, control engineering, optimization and operation research, and knowledge management. Reviewers' diversified expertise has greatly strengthened our confidence in reviewing the state-of-the-art of CI applications in various industrial systems. Finally, we express our special thanks to Springer, in particular, Dr. Thomas Ditzinger – the series editor,

and Heather King, for giving us the opportunity to gather the latest research findings and for publishing this book. Throughout the year, the editors have been working very closely with the two Springer editors on this book. Their quick endorsement on the book proposal and assistance are deeply appreciated by the book's editors.

China   *Ying Liu*
Singapore   *Aixin Sun*
2007   *Han Tong Loh*
                                                                      Wen Feng Lu
                                                                      Ee-Peng Lim

Web Services, Policies, and Context: Concepts and Solutions

Zakaria Maamar[1], Quan Z. Sheng[2], Djamal Benslimane[3],
and Philippe Thiran[4]

[1] Zayed University, U.A.E., `zakaria.maamar@zu.ac.ae`
[2] The University of Adelaide, Australia, `qsheng@cs.adelaide.edu.au`
[3] Claude Bernard Lyon 1 University, France, `djamal.benslimane@liris.cnrs.fr`
[4] Louvain School of Management & University of Namur, Belgium,
`pthiran@fundp.ac.be`

1 Introduction

Despite the extensive adoption of Web services by IT system developers, they still lack the capabilities that could enable them to match and eventually surpass the acceptance level of traditional integration middleware (e.g., CORBA, Java RMI). This lack of capabilities is to a certain extent due to the *trigger-response* interaction pattern that frames the exchanges of Web services with third parties. Adhering to this interaction pattern means that a Web service only performs the requests it receives without considering its internal execution state, or even questioning if it would be rewarded for performing these requests (e.g., to be favored over similar Web services during selection). There exist, however, several situations that insist on Web services self-management so that *scalability*, *flexibility*, and *stability* requirements are satisfied.

The objective of this chapter is to discuss the value-added of integrating *context* and *policies* into a *Web services* composition approach.

Web services offer new opportunities to deploy B2B applications, which tend to crosscut companies' boundaries. Web services are independent from specific platforms and computing paradigms, and have the capacity to form high-level business processes referred to as composite Web services [3].

Policies are considered as external, dynamically modifiable rules and parameters that are used as input to a system [13]. This permits to the system to adjust to administrative decisions and changes in the execution environment. In the field of Web services, policies are intended to specify different aspects of the behavior of a Web service, so this one can align its capabilities to users' requirements and resources' constraints.

Context "... is not simply the state of a predefined environment with a fixed set of interaction resources. It is part of a process of interacting with an ever-changing environment composed of reconfigurable, migratory, distributed,

and multiscale resources" [4]. In the field of Web services, context is used to facilitate the development and deployment of flexible Web services. Flexibility refers to a Web service that selects appropriate operations based on the requirements of the business scenario that this Web service implements.

While context and policies are separately used for different needs of Web services, this chapter discusses their role in framing the composition process of Web services. We propose a three-level approach to compose Web services. This approach does not only make Web services bind to each other, but emphasizes the cornerstone of refining this binding at the following levels: *component* level ($\mathcal{W}$-level) to deal with Web services' definitions and capabilities, *composite* level ($\mathcal{C}$-level) to address how Web services are discovered and combined (semantic mediation is discarded), and finally, *resource* level ($\mathcal{R}$-level) to focus on the performance of Web services.

The role of policies and context to support the composition of Web services is depicted as follows:

– Policies manage the transitions between the three levels. Going from one level to another direct level requires policy activation. Two types of policies are put forward: *engagement* policies manage the participations of Web services in compositions, and *deployment* policies manage the interactions of Web services with computing resources.
– Context provides information on the environment wherein the composition of Web services occurs. Because of the three levels in the approach, three types of context are defined, namely $\mathcal{W}/\mathcal{C}/\mathcal{R}$-context standing for context of $\mathcal{W}$eb service, context of $\mathcal{C}$omposite Web service, and context of $\mathcal{R}$esource.

The rest of this chapter is organized as follows. Section 2 presents the approach to compose Web services. Section 3 discusses the impact of policies on Web services and specifies the policies for the behavior of Web services. Section 4 is about exception handling. Section 5 reviews some related works. Finally, Sect. 5 concludes the chapter.

2 The Proposed Composition Approach

2.1 Presentation

Figure 1 illustrates our approach to compose Web services. The figure reads as follows: bottom-up during normal progress of composition and top-down during exception in composition. The underlying idea is that context coupled to policies handle the specification and execution of a composite Web service. Three levels form this approach. The component level is concerned with the definition of Web services and their announcements to composite Web services that are located in the composition level. This level is concerned with the way component Web services are discovered (based on their capabilities)
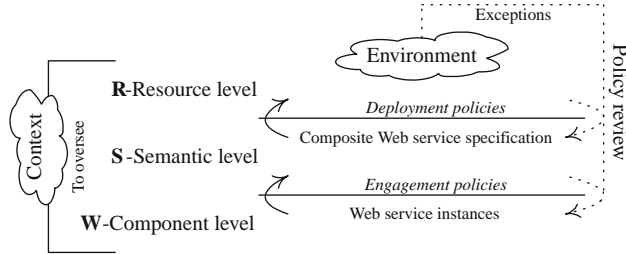
Fig. 1. Approach to compose Web services

and assembled (according to particular collaboration patterns). Finally, the resource level focuses on the performance of a composite Web service. This performance is subject to the availability of resources and the commitments of these resources to other concurrent composite Web services.

In Fig. 1, policies support level transitions. Transiting from one level to a higher direct-level is subject to policy execution. This execution uses the information that context provides on different elements like number of Web services that are currently running, composite Web services that are under preparation, etc. Two types of policies are shown in Fig. 1: engagement and deployment. While policies support level transitions (i.e., how a composite Web service specification progresses), context oversees first, the actions that occur within and across these levels and second, the elements like Web service, composite Web service, or resource, that are involved by these actions. Prior to triggering any policy and executing any action, the information that context provides is validated.

Any deviation from the specification of a composite Web service like execution delays, raises exceptions. This leads to reviewing the different actions of the policies that were carried out within and across the levels. The review process occurs in a descending way commencing with resource, composite, and then, component levels (Fig. 1). Transiting to a lower level means that the current level is not faulty and extra investigations of the exception are deemed appropriate at this lower level. To keep track of the actions of policies that happened, are happening, and might happen, hence corrective measures (e.g., compensation policies) are scheduled and undertaken as part of the review process, context monitors the whole composition of Web services. Additional details on exception handling are discussed in Sect. 4.

2.2 Description of the Three Levels

The W-Component level shows the Web services that participate in a composite Web service. This participation is upon approval and happens in accordance with the Web services *instantiation* principle that was introduced in [15]. This principle emphasizes the simultaneous participation of a Web service in several compositions.
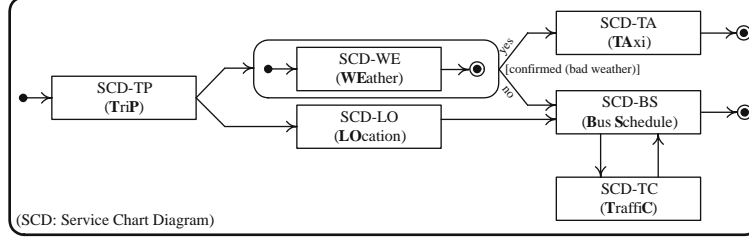
Fig. 2. Sample of a composite Web service specification

Before a Web service accepts the invitation to participate in a composite Web service, it consults its $\mathcal{W}$-context so that it can verify the number of current participations vs. the maximum number of participations, the expected completion time of its other participations, and the features of the newly received invitation of participation like when the new Web service instance is required. It happens that a Web service refuses an invitation of participation for various reasons like risk of overload.

The $\mathcal{C}$-Composite level is about the specification of a composite Web service in term of business logic. Figure 2 shows a simple specification that combines service chart diagrams and state chart diagrams [12]. The component Web services of this specification are: trip (`TP`), weather (`WE`), location (`LO`), taxi (`TA`), bus schedule (`BS`), and traffic (`TC`). The respective service chart diagrams of these components are connected through transitions. Some of these transitions have constraints to satisfy like [confirmed(bad weather)]. A composite Web service consults its $\mathcal{C}$-context so that it can follow up the execution progress of its specification. This enables, for example, identifying the next Web services to be subject to invitation of participation and the status of each component Web service of the composite Web service (e.g., initiated, running, suspended, resumed). Figure 3 illustrates the editor we developed for specifying composite Web services. The editor provides means for directly manipulating service chart diagrams, states, and transitions using drag and drop actions.

The $\mathcal{R}$-Resource level represents the computing facilities upon which Web services operate. Scheduling the execution requests of Web services is prioritized when sufficient resources are unavailable to satisfy all these requests at once. A Web service requires resources for different operations like self-assessment of current state before accepting/rejecting participations in compositions, and satisfying users' needs upon request. Before a resource accepts supporting the execution of an additional Web service, it consults its $\mathcal{R}$-context so that it can verify the number of Web services currently executed vs. the maximum number of Web services under execution, the approximate completion time of the ongoing executions of Web services, and the features of the newly received request like requested time of execution. Like with Web
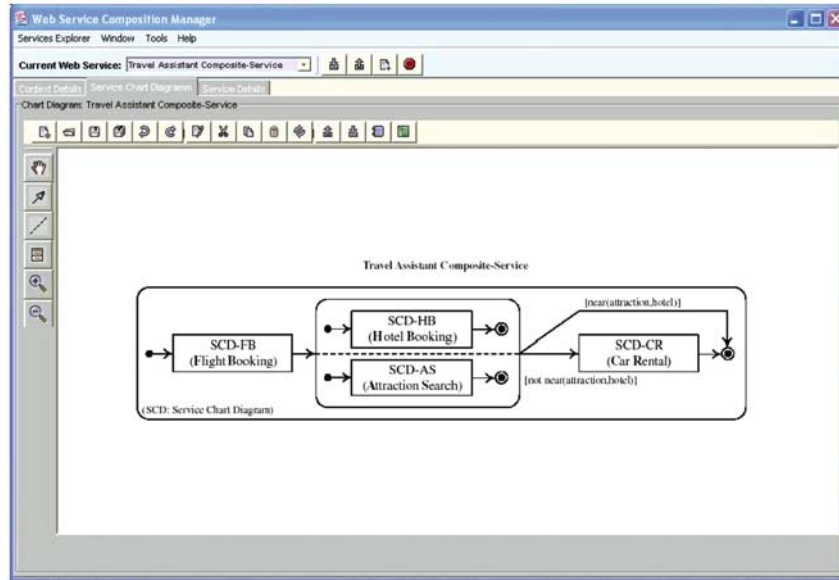
Fig. 3. Editor for composite Web service specification

services, similar considerations apply to resources when it comes to turning down execution requests of Web services.

2.3 Description of the Three Contexts

To comply with the three levels of the composition approach, three types of context are defined: $\mathcal{W}$-context, $\mathcal{C}$-context, and $\mathcal{R}$-context (Fig. 1).

The *$\mathcal{W}$-context* of a Web service returns information on the participations of the Web service in different compositions. These participations are made possible because of the Web services instantiation principle. $\mathcal{W}$-context has the following arguments (Table 1): label, maximum number of participations, number of active participations, next possibility of participation, resource&state per active participation, local ontology per active participation, previous Web services per active participation, current Web services per active participation, next Web services per active participation, regular actions, reasons of failure per active participation, corrective actions per failure type and per active participation, and date. Figure 4a, b shows how context arguments are instantiated at run-time.

The *$\mathcal{C}$-context* of a composite Web service is built upon the $\mathcal{W}$-contexts of its component Web services and permits overseeing the progress of a composition.

The *$\mathcal{R}$-context* of a resource oversees the execution of the Web services that operate on top of this resource prior it accepts additional Web services for execution. A resource has computation capabilities that continuously change

Table 1. Arguments of $\mathcal{W}$-context

Label: corresponds to the identifier of the Web service

Maximum number of participations: corresponds to the maximum number of compositions in which the Web service can participate at a time

Number of active participations: corresponds to the number of active compositions in which the Web service is now participating

Next possibility of participation: indicates when the Web service can participate in a new composition. This is subject to the successful termination of the current active participations

Resource&State per active participation: corresponds to the identifier of the selected resource and the state of the Web service in each active composition. State can be of types in-progress, suspended, aborted, or completed, and will be obtained out of the state argument of $\mathcal{R}$-context of this resource

Ontology of interpretation per active participation: refers to the ontology that the Web service binds to per active participation for interpreting the values of its input and output arguments when conversion functions are triggered

Previous Web services per active participation: indicates the Web services that were successfully completed before the Web service per active composition (null if there are no predecessors)

Current Web services per active participation: indicates the Web services that are concurrently being performed with the Web service per active composition (null if there is no concurrent processing)

Next Web services per active participation: indicates the Web services that will be executed after the Web service successfully completes its execution per active composition (null if there are no successors)

Regular actions: illustrates the actions that the Web service normally performs

Reasons of failure per active participation: informs about the reasons that are behind the failure of the execution of the Web service per active composition

Corrective actions per failure type and per active participation: illustrates the actions that the Web service has performed due to execution failure per active composition

Date: identifies the time of updating the arguments above

---

depending on the number[1] of Web services that are now under execution and the execution duration per Web service. Due to lack of space, the structures of $\mathcal{C}$-context and $\mathcal{R}$-context are not presented.

---

[1] Number is used for illustration purposes. Additional criteria could be execution load of Web services.

W-Context Parameters	Values
Label	WEather
MaximumNumberOfParticipations	2
NumberOfActiveParticipations	0
NextPossibilityOfParticipation	Possible
RegularActions	Providing Weather Details
PrevisousWebServicesPerActiveParticipation	Nil
CurrentWebServicesPerActiveParticipation	Nil
NextWebServicePerActiveParticipation	Nil
ResourceAndStatePerActiveParticipation	DBResource/Active
ReasonsOfFailurePerActiveParticipation	Nil
CorrectiveActionsPerFailureType	Nil
Daate	Mar 2, 2006 11:47:51 AM

(a). W-Context

R-Context Parameters	Values
Label	DBResource
MaximumNumbrOfComponentWebServices	2
NumberOfActiveComponentWebServices	0
PreviousComponentWebService	Nil
CurrentComponentWebService	Nil
NextComponentWebService	Nil
NextAcceptenceOfComponentWebSrvices	Possible
Daate	Mar 11, 2006 1:47:45 PM

(b). R-Context

Permission Policy Parameters	Values
CurrentNumberOfParticipations	0
MaximumNumberOfParticipations	2
Avilability	true

(c). Permission Policy

DispensationPermission Policy Parameters	Values
PeakTime	Mar 25, 2006 5:02:47 PM
RequestTime	Nil
Avilability	false

(d). Dispensation Permission Policy

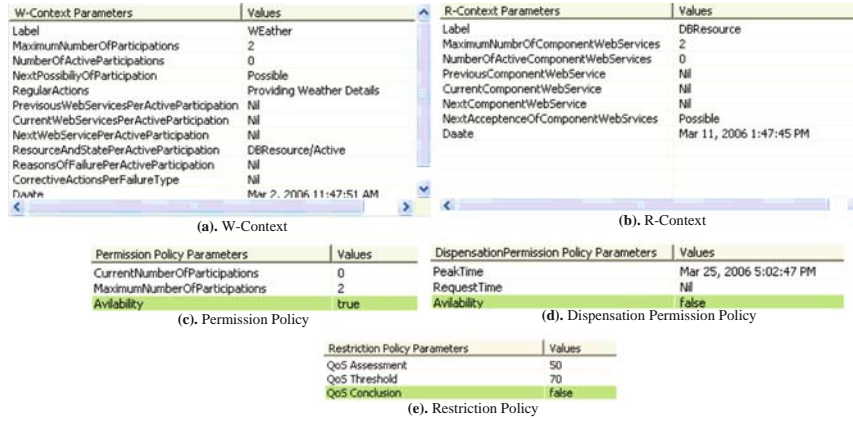| Restriction Policy Parameters	Values
QoS Assessment	50
QoS Threshold	70
QoS Conclusion	false

(e). Restriction Policy

Fig. 4. Illustration of context and policy specification

2.4 Description of the Two Policies

Policies permit during a normal progress scenario to move a composition from one level to a higher-direct level (Fig. 1). Policies' outcomes are also subject to review in case of exceptions, i.e., abnormal progress.

Engagement policies frame the participation of component Web services in composite Web services. This highlights the opportunity of a Web service to take part in several compositions at a time. Since Web services are context-aware, they assess their participations and use the engagement policies to back their either acceptance or denial of participation in additional compositions.

Deployment policies frame the interactions between component Web services and computing resources. We earlier argued that resources have limited computation capabilities, and scheduling execution requests of component Web services is deemed appropriate. Since resources are context-aware, they assess their commitments and use the deployment policies to back their decisions of either accepting or denying a component Web service's execution request.

3 Role of Policies

3.1 Behavioral Web Services

In the composition approach of Fig. 1, policies have an impact on first, the behavior that Web services expose towards composite Web services and second, the behavior that composite Web services (through their component Web services) expose towards computing resources. We identify this behavior with the following attributes: permission and restriction.

Engagement policies associate two types of behavior with a Web service: permission and restriction. Engagement policies satisfy the needs of Web services' providers who have interests and/or obligations in strengthening or restricting the participation of their Web services in some compositions.

– *Permission.* A Web service is authorized to accept the invitation of participation in a composite Web service once this Web service verifies its current status.
– *Restriction.* A Web service cannot connect other peers of a composite Web service because of non-compliance of some peers with this Web service's requirements.

Deployment policies associate two types of behavior with a Web service: permission and restriction. Deployment policies satisfy the needs of resources' providers who have interests and/or obligations in strengthening or restricting the execution of Web services on these resources.

– *Permission.* A Web service receives an authorization of execution from a resource once this resource checks its current state.
– *Restriction.* A Web service is not accepted for execution on a resource because of non-compliance of this Web service with the resource's requirements.

3.2 Specification of Policies

In what follows, the policies that define the behavior of Web services are specified. To this end, a policy definition language is required. In this chapter, the Web Services Policy Language (WSPL) is used [1]. The syntax of WSPL is strictly based on the OASIS eXtensible Access Control Markup Language (XACML) standard[2]. Figure 4c–e shows how policies are instantiated at run-time.

Engagement Policy

Figure 5 illustrates the way the different behaviors of a Web service are connected together based on the execution outcome of the engagement policy per type of behavior. For instance, a positive permission (i.e., yes) is not confirmed until no restrictions are identified (i.e., no). More details on this figure are given in the description of each type of behavior. The dashed lines in Fig. 5 represent potential cases of exceptions to be discussed in Sect. 4.

1. An engagement policy for permission consists of authorizing a Web service to be part of a composite Web service. The authorization is based

---

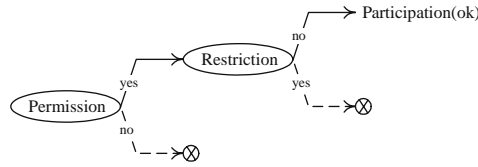[2] www.oasis-open.org/committees/download.php/2406/oasis-xacml-1.0.pdf.

Fig. 5. Behaviors of a Web service in an engagement scenario

on the status of the Web service that is known through $\mathcal{W}$-context. The following illustrates an engagement policy for permission in WSPL. It states that a Web service participates in a composition subject to evaluating `<Condition>` to true. This latter refers to some arguments like the number of current active participations of the Web service in compositions and the next possibility of participation of the Web service in additional compositions. These arguments are known as vocabulary items in WSPL. In the policy, `<TrueConclusion>` shows the permission of participation, whereas `<FalseConclusion>` shows the contrary. In case of positive permission, `yes-permission-participation` procedure is executed, which results in updating the following arguments in $\mathcal{W}$-context of the Web service: *number of active participations*, *previous Web services*, *current Web services*, and *next Web services*. Figure 6 shows the outcome of executing an engagement policy of type permission. This policy grants a Web service the permission to take part in a composite Web service.

```
Policy (Aspect="PermissionParticipation") {
<Rule xmlns="urn:oasis:names:tc:xacml:3.0:permission:policy:schema:wd:01"
xmlns:proc="permission-participation" RuleId="PermissionParticipationWS">
  <Condition>
    <Apply FunctionId="and">
    <Apply FunctionId="integer-less-than" DataType="boolean">
    <SubjectAttributeDesignator AttributeId="CurrentNumberOfParticipations"
    DataType="integer"/>
    <SubjectAttributeDesignator AttributeId="MaximumNumberOfParticipations"
    DataType="integer"/>
    </Apply>
    <SubjectAttributeDesignator AttributeId="Availability" DataType="boolean"/>
    </Apply>
  </Condition>
  <Conclusions>
    <TrueConclusion>
    <proc:do> yes-permission-participation </proc:do>
    </TrueConclusion>
    <FalseConclusion>
    <proc:do> no-permission-participation </proc:do>
    </FalseConclusion>
  </Conclusions>
</Rule>}
```

2. An engagement policy for restriction consists of preventing a Web service from taking part in a composite Web service. A restriction does not implement a negative permission of participation. However, it follows a positive permission of participation (Fig. 5). Restrictions could be geared towards

Fig. 6. Outcome of policy execution

the quality issue (e.g., time of response, reputation) of the component Web services with whom a Web service will interact. For example, a Web service's provider has only interest in the Web services that have a "good" QoS record [16]. The following illustrates a restriction policy in WSPL. It states that a Web service can be restricted from participation subject to evaluating `<Condition>` to true. This latter checks that a positive permission of participation exists and the assessment level of the QoS of the Web services is low. These Web services are identified using *previous Web services per active participation* argument of $\mathcal{W}$-context of the Web service. In this policy, `QoSAssessment` is an integer value that is the result of evaluating the QoS of a Web service, and `QoSThreshold` is the minimum QoS assessment value that is acceptable for composition.

```
Policy (Aspect="RestrictionPermission") {
    <Rule xmlns="urn:oasis:names:tc:xacml:3.0:generalization:policy:schema:wd:01"
    RuleId="RestrictionParticipationWS">
      <Condition>
        <Apply FunctionId="and">
        <SubjectAttributeDesignator AttributeId="YesPermissionParticipation">
        DataType="boolean"/>
        <Apply FunctionId="integer-great-than-or-equal">
        <SubjectAttributeDesignator AttributeId="QoSAssessment" DataType="integer"/>
        <SubjectAttributeDesignator AttributeId="QoSThreshold" DataType="integer"/>
        </Apply>
        </Apply>
      </Condition>
      <Conclusions>
        <TrueConclusion RestrictionParticipation = "No"/>
        <FalseConclusion RestrictionParticipation = "Yes"/>
      </Conclusions>
</Rule>}
```

Deployment Policy

Figure 7 illustrates the way the different behaviors of a Web service are connected together based on the execution outcome of the deployment policy per type of behavior. For instance, a positive permission for execution (i.e., yes) is confirmed if there are no restrictions that could deny this permission at run time. More details about this figure are given in the description of each type
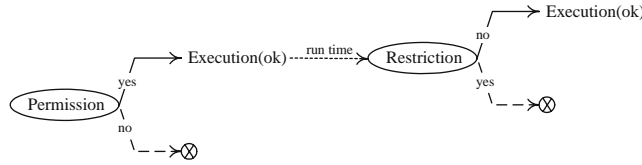
Fig. 7. Behaviors of a Web service in a deployment scenario

of behavior. The dashed lines in Fig. 7 represent potential cases of exceptions and are discussed in Sect. 4.

1. A deployment policy for permission is about a Web service that receives the necessary execution authorizations from a resource. These authorizations are based on the state of the resource, which is reflected through $\mathcal{R}$-context. The following illustrates a deployment policy for permission in WSPL. It states that a resource accepts the execution request of a Web service subject to evaluating `<Condition>` to true. This latter refers to some arguments like the number of active component Web services that the resource supports their execution and the next acceptance of the resource for additional component Web services. In the policy, `<TrueConclusion>` shows the permission of execution, whereas `<FalseConclusion>` shows the contrary. In case of positive permission of execution, `yes-permission-deployment` procedure is executed, which results in updating the following arguments: *resource&state per active participation* of $\mathcal{W}$-context of the Web service and *number of active component Web services* of $\mathcal{R}$-context of the resource.

```
Policy (Aspect="PermissionDeployment") {
<Rule xmlns="urn:oasis:names:tc:xacml:3.0:permission:policy:schema:wd:01"
xmlns:proc="permission-deployment" RuleId="PermissionDeploymentWS">
  <Condition>
    <Apply FunctionId="and">
    <Apply FunctionId="integer-less-than" DataType="boolean">
    <SubjectAttributeDesignator AttributeId="NumberofActiveComponentWebServices"
    DataType="integer"/>
    <SubjectAttributeDesignator AttributeId="MaximumNumberofComponentWebServices"
    DataType="integer"/>
    </Apply>
    <SubjectAttributeDesignator AttributeId="NextAcceptanceofComponentWebServices"
    DataType="boolean"/>
    </Apply>
  </Condition>
  <Conclusions>
    <TrueConclusion>
    <proc:do> yes-permission-deployment </proc:do>
    </TrueConclusion>
    <FalseConclusion>
    <proc:do> no-permission-deployment </proc:do>
    </FalseConclusion>
  </Conclusions>
</Rule>}
```

2. A deployment policy for restriction consists of preventing a Web service form being engaged in an execution over a resource. A restriction does not

implement a negative permission of deployment, rather it follows a positive permission of deployment (Fig. 5). Besides the example of resource failure, restrictions could be geared towards the reinforcement of the execution features that were agreed on between a Web service and a resource. For example a Web service binds a resource for execution prior to the scheduled time. The following illustrates a deployment policy for restriction in WSPL. It states that a Web service can be restricted from execution subject to evaluating `<Condition>` to true. This latter checks that a positive permission of execution has been issued and the agreed execution time is valid. The execution time of a Web service is identified using *next component Web services per active participation* argument of $\mathcal{R}$-context of the resource.

```
Policy (Aspect="RestrictionDeployment") {
<Rule xmlns="urn:oasis:names:tc:xacml:3.0:generalization:policy:schema:wd:01"
RuleId="RestrictionDeploymentWS">
  <Condition>
    <Apply FunctionId="and">
    <SubjectAttributeDesignator AttributeId="YesPermissionDeployment"
    DataType="boolean"/>
    <Apply FunctionId="equal" DataType="boolean">
    <SubjectAttributeDesignator AttributeId="ExecutionTime" DataType="String"/>
    </Apply>
    </Apply>
  </Condition>
  <Conclusions>
    <TrueConclusion RestrictionDeployment = "No"/>
    <FalseConclusion RestrictionDeployment = "Yes"/>
  </Conclusions>
</Rule>}
```

4 Exception Handling

4.1 Rationale

In [22], Yu et al. adopt the example of a programmer who writes a BPEL business process. The complexity of some business domains results sometimes in processes with errors. When a process is put into operation, exception handling is to occur otherwise, angry customers and loss of revenues arise. In [19], Russell et al. looked into the range of issues that may result in exceptions during workflow execution and the various ways these issues can be addressed. This work classifies exceptions using patterns. Some of the exception types that are inline with what is handled in the proposed composition approach include work item failure, deadline expiry, and resource unavailability. In the composition approach, exception handling means reviewing policies' outcomes and taking appropriate actions in a top-down manner by consulting the three levels shown in Fig. 1. An exception could take place at one of the following levels:

– *Resource level.* There is no guarantee that a particular resource is up at the execution time of a Web service. A resource could be down due to power failure.
– *Composite level.* There is no guarantee that the specification of a composite Web service is error-free. Conflicting actions like concurrent accept and reject and deadlocks may occur during this specification execution.
– *Component level.* There is no guarantee that a particular Web service is still available at execution-time. A provider could withdraw its Web service from a composition without prior notice.

4.2 Exception Types per Policy Type

Deployment policy. Permission and restriction show the behaviors of a Web service following deployment-policy triggering. Potential origins of exception are as follows. First, an execution permission for a Web service over a resource was not issued but there was tentative of execution over this resource. Second, an execution restriction on a Web service was detected but that was not properly reported.

Exceptions in case of permission could arise because of the inaccurate assessment of some arguments in $\mathcal{R}$-context of a resource. These arguments are as follows:
– *Number of active component Web services* argument. A resource did not correctly assess the exact number of Web services that currently run on top of it. This number should not exceed the maximum number of authorized Web services for execution.
– *Next acceptance of component Web services* argument. The execution of a Web service got disrupted, which has negatively affected the execution scheduling of the forthcoming Web services over a resource.

Exceptions in case of restriction could arise because of the inaccurate assessment of some arguments in a resource's $\mathcal{R}$-context like *next component Web services per active participation.* The execution of a Web service should not occur prior to the time that was agreed upon between the resource and the Web service.

Engagement policy. Permission and restriction show the behaviors of a Web service following the triggering of an engagement policy. The dashed lines in Fig. 5 represent the potential origins of exceptions. First, a participation permission for a Web service was not issued, but there was tentative of participation. Second, a participation restriction on a Web service was detected, but the restriction has not been taken effectively.

Exceptions in case of permission could arise because of the inaccurate assessment of some arguments in a Web service's $\mathcal{W}$-context:
– *Number of active participations* argument. The Web service did not exactly assess its current participations in compositions. This number should not exceed the maximum number of authorized compositions.

In the engagement policy for permission given in Sect. 3.2, the condition of participation is shown with the following statements.

```
<Apply FunctionId="integer-less-than" ... >
<SubjectAttributeDesignatorAttributeId="CurrentNumberOfParticipations" .../>
<SubjectAttributeDesignator AttributeId="MaximumNumberOfParticipations" .../>
</Apply>
```

Since there is no permission of participation the invocation of the Web service fails due to unavailability. A policy needs to be triggered so, actions are taken such as reviewing the current number of active participations of the Web service and notifying the relevant composite Web service about the disengagement of this Web service.

– *Next possibility of participation* argument. The execution of a Web service got delayed, which has negatively affected the participation scheduling of the Web service in forthcoming compositions.

Exceptions in case of restriction could arise because of the inaccurate assessment of some arguments in a Web service's $\mathcal{W}$-context like *next Web services per active participation*. A Web service cannot connect other peers if they are not listed in this argument.

5 Related Work

This related work is presented from two perspectives: context for Web services and policies for Web services.

In literature the first time "context-aware" appeared was in [21]. Schilit and Theimer describe context as location, identities of nearby people, objects and changes in those objects. Dey defines context as the user's emotional state, focus of attention, location and orientation, date and time, as well as objects and people in the user's environment [7]. These kinds of definitions are often too wide. One of the most accurate definitions is given by Dey and Abowd [6]. They refer to context as "*any information that can be used to characterize the situation of entities (i.e., whether a person, place or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves*".

Making Web services context-aware is not straightforward; many issues are to be addressed (adapted from [20]): how is context structured, how does a Web service bind context, how are changes detected and assessed for context update purposes, and what is the overload on a Web service of taking context into account? Responses to some of these questions have guided Maamar et al. to organize a Web service's context along three interconnected perspectives [14]. The participation perspective is about overseeing the multiple composition scenarios in which a Web service concurrently takes part. The execution perspective is about looking for the computing resources upon which a Web service operates, and monitoring the capabilities of these

computing resources so that the Web service's requirements are constantly satisfied. Finally, the preference perspective is about ensuring that user preferences (e.g., execution time at 2 pm) are integrated into the specification of a composite Web service.

Some research projects have focused on how to facilitate the development and deployment of context-aware and adaptable Web services. Standard Web services descriptions are augmented with context information (e.g., location, time, user profile) and new frameworks are developed to support this. The approach proposed in [17] is intended to provide an enhancement of WSDL language with context aware features. The proposed Context-based Web Service Description Language (CWSDL) adds to WSDL a new part called Context Function which is used in order to select the best service among several ones. This function represents the sensitivity of the service to context information. Another interesting approach was proposed in [11] to deal with context in Web services. The approach consists of two parts: a context infrastructure and a context type set. The context infrastructure allows context information to be transmitted as a SOAP header-block within the SOAP messages.

Kagal et al. argue that policies should be part of the representation of Web services, and in particular of semantic Web services [10]. Policies provide the specification of who can use a service under what conditions, how information should be provided to the service, and how provided information will be used later. In [2], Baresi et al. proposed a policy-based approach to monitor Web services' functional (e.g., constraints on exchanged data) and non-functional (e.g., security, reliability) requirements. In this approach Baresi et al. report on the different types of policies that can be defined along the life cycle of a Web service [18]. These types are service policies, server policies, supported policies, and requested policies.

Desai et al. have recognized the importance of separation of concerns from a software engineering perspective [5]. Indeed, they split a business process into two parts: protocol and policy. The protocol part is a modular, public specification of an interaction among different roles that collaborate towards achieving a desired goal. The policy part is a private description of a participant's business logic that controls how this participant takes part in a protocol.

Policies have also been used to access and use resources in other fields. In grid computing, Dumitrescu et al. describe two strategies that a virtual organization must deploy to ensure grid-wide resource policy enforcement in a decentralized manner [8]. In the semantic Web, Gavriloaie et al. present an approach to manage access to sensitive resources using PeerTrust, a language for expressing access control policies [9]. PeerTrust is based on guarded distributed logic programs and has been demonstrated for automated trust negotiation.

6 Conclusion

In this chapter we proposed an approach to compose Web services. The role of policies and context in this composition has been depicted as follows.

– Policies manage the transitions between component, composite, and resource levels. Transiting from one level to another requires activating policies. Two types of policies were developed: engagement policies assess a Web service participation in a given composition, and deployment policies manage the interactions between component Web services and computing resources.
– Context provides useful information concerning the environment wherein the composition of Web services occurs. Context caters the necessary information that enables tracking the whole composition process by enabling for instance to trigger the appropriate policies and to regulate the interactions between Web services according to the current status of the environment. Three types of context were defined, namely $\mathcal{W}/\mathcal{C}/\mathcal{R}$-context standing for context of Web service, context of composite Web service, and context of resource.

References

1. A. H. Anderson. Predicates for Boolean Web Service Policy Language (SWPL). In *Proceedings of the International Workshop on Policy Management for the Web (PM4W'2005) Held in Conjunction with The Fourteenth International World Wide Web Conference (WWW'2005)*, Chiba, Japan, 2005.
2. L. Baresi, S. Guinea, and P. Plebani. WS-Policy for Service Monitoring. In *Proceedings of the Sixth VLDB Workshop on Technologies for E-Services (TES'2005) Held in Conjunction with the 31st International Conference on Very Large Data Bases (VLDB'2005)*, Trondheim, Norway, 2005.
3. I. Budak Arpinar, B. Aleman-Meza, R. Zhang, and A. Maduko. Ontology-driven Web services composition platform. In *Proceedings of the IEEE International Conference on E-Commerce Technology (CEC'2004)*, San-Diego, USA, 2004.
4. J. Coutaz, J. L. Crowley, S. Dobson, and D. Garlan. Context is Key. *Communications of the ACM*, 48(3), March 2005.
5. N. Desai, A. U. Mallya, A. K. Chopra, and M. P. Singh. Processes = Protocols + Policies: A Methodology for Business Process Development. In *Proceedings of the 14th International World Wide Web Conference (WWW'2005)*, Chiba, Japan, 2005.
6. A. K. Dey and G. D. Abowd. Towards a Better Understanding of Context and Context-Awareness. In *Proceedings of the Workshop on the What, Who, Where, When, and How of Context-Awareness Held in Conjuction with CHI'2000*, The Hague, The Netherlands, 2000.
7. Anind K. Dey. Context-Aware Computing: The CyberDesk Project. In *Proceedings of the AAAI'98 Spring Symposium on Intelligent Environments (AAAI'1998)*, Menlo Park, CA, USA, 1998.

8. C. Dumitrescu, M. Wilde, and I. T. Foster. A model for usage policy-based resource allocation in grids. In *Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'05)*, New-York, USA, 2005.
9. 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. In *Proceedings of The First European Semantic Web Symposium on The Semantic Web: Research and Applications (ESWS'2004)*, Heraklion, Crete, Greece, 2004.
10. L. Kagal, M. Paolucci, N. Srinivasan, G. Denker, and K. Finin, T. Sycara. Authorization and privacy for semantic Web services. *IEEE Intelligent Systems*, 19(4), July/August 2004.
11. M. Keidl and A. Kemper. A framework for context-aware adaptable web services. In *EDBT*, pages 826–829, 2004.
12. Z. Maamar, B. Benatallah, and W. Mansoor. Service chart diagrams – description & application. In *Proceedings of the Alternate Tracks of The 12th International World Wide Web Conference (WWW'2003)*, Budapest, Hungary, 2003.
13. Z. Maamar, D. Benslimane, and A. Anderson. Using policies to manage Composite Web Services? *IEEE IT Professional*, 8(5), Sept./Oct. 2006.
14. Z. Maamar, D. Benslimane, and Nanjangud C. Narendra. What Can Context do for Web Services? *Communications of the ACM*, Volume 49(12), December 2006.
15. Z. Maamar, S. Kouadri Mostéfaoui, and H. Yahyaoui. Towards an agent-based and context-oriented approach for Web services composition. *IEEE Transactions on Knowledge and Data Engineering*, 17(5), May 2005.
16. D. A. Menascé. QoS Issues in Web Services. *IEEE Internet Computing*, 6(6), November/December 2002.
17. S. Kouadri Mostéfaoui and B. Hirsbrunner. Towards a context-based service composition framework. In *ICWS*, pages 42–45, 2003.
18. N. Mukhi and P. Plebani. Supporting policy-driven behaviors in web services: experiences and issues. In *Proceedings of the Second International Conference on Service-Oriented Computing (ICSOC'2004)*, New York City, NY, USA, 2004.
19. N. Russell, W. M. P. van der Aalst, and A. H. M. ter Hofstede. Exception handling patterns in process-aware information systems. Technical Report, BPM Center Report BPM-06-04, BPMcenter.org, 2006.
20. M. Satyanarayanan. Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8(4), August 2001.
21. B. Schilit and M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5), 1994.
22. X. Yu, L. Zhang, Y. Li, and Y. Chen. WSCE: A flexible web service composition environment. In *Proceedings of The IEEE International Conference on Web Services (ICWS'2004)*, San-Diego, California, USA, 2004.