Towards Dynamic, Relevance-Driven Exception Resolution in Composite Web Services

Kareliotis Christos[1], Vassilakis Costas[2], Georgiadis Panayiotis[1]

[1] Department of Informatics and Telecommunications, University of Athens
{ckar,georgiad}@di.uoa.gr
[2] Department of Computer Science and Technology, University of Peloponnese
costas@uop.gr

Abstract. Web services have become the leading technology for application-to-application (A2A) communication over distributed and heterogeneous environments. Both academia and industry have strived to enable useful service collaborations among distributed systems without any human intervention. Web service composition can be used to this end, to achieve business automation within one company or realize business-to-business (B2B) integration of heterogeneous software and cross-organizational computing systems. Service composition pro-vides added value, when a web service composition itself becomes a higher level composite web service. However, as business processes are long-lasting transactions, exceptions may often occur, necessitating the replacement of a service component which has been made unavailable, hindering the completion of some business process. In this paper we present an exception resolving approach based on discovering replacement components that are functionally equivalent, taking also into account criteria for qualitative substitutability. The proposed solution introduces the Service Relevance and Replacement Framework (SRRF) which undertakes exception handling.

1 Introduction

Web services are unanimously supported by major software vendors of middleware technology [1]. The main objective of web service technology and related research [2] is to provide the means for enterprises to do business with each other and provide joint services to their customers under specified Quality of Service (QoS) levels. The collaboration of web services, possibly provided by different companies, in order to create composite and potentially highly complex business process, elevates the need of a Business Process Management (BPM). BPM addresses how organizations can identify, model, develop, deploy, and manage their business process, including processes that involve IT systems and human interaction. An important aspect of business processes is the definition of a binding agreement or contract between the two suppliers and customers, specifying QoS items such as deadlines, quality of products, and cost of services.

Business processes can be composed from sub-processes that act as atomic processes in the execution scenario. Composite services include two or more distinct

ser-vices and are frequently specified using the Web Services Business Process Execution Language (WS-BPEL) and executed by a Web Services Orchestration (WSO) plat-form. If multiple, possibly cross-organization, business processes need to collaborate, their interaction can be modeled using the Web Services Choreography Description Language (WS-CDL).

Due to their distributed, heterogeneous and highly volatile nature, services-based systems are inherently vulnerable to exceptions: software, machine or communication link failures may render certain sub-process of composite services unavailable, precluding thus the successful execution of the business process. In addition to these transient failures, certain web services may be permanently withdrawn and/or alternatives to some services may be offered by different providers. In these cases, a replacement component should be identified and substituted for the failed one. The replacement component should have the "same skills" with the failed one i.e. to have same functionality and QoS [3]. We name this component a "task". In this context, we consider a task as an application delivered by a web service. Adapting the relevant definition from [4], a task execution may be either an elementary web service, i.e., a service which does not transparently rely on other web services. Note that this definition includes composite services, which do not expose any details of their composition to their invokers, but appear as atomic services. Note that the dynamic and volatile nature of the execution environment implies that it is infeasible to list all possible alternatives in the BPEL execution scenario; instead, a dynamic approach should be adopted, where service selection is undertaken by a distinct module, which has access to a registry listing all pertinent functional and qualitative characteristics of available services, as illustrated in Figure 1. Component replacement should respect Service Level Agreement signed between the consumer and the provider; moreover, when a service selection succeeds a hot-swapper takes over in order to replace at run-time the corrupted service with the working one.

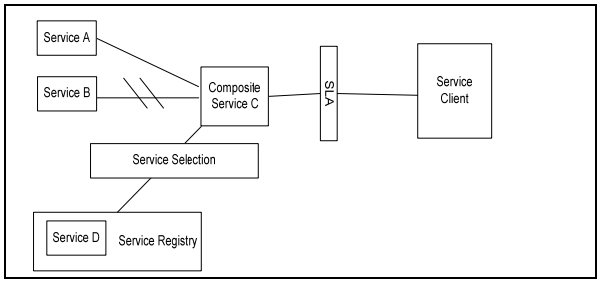

Fig. 1. Replacing a failed component with one having "same skills"

Hot-swapping for a failed service/component with a new same-skilled one may be technically complicated in some cases. For example, if a task within a composite web service stops functioning when having partially updated a local database, it must be ascertained that these changes will be undone to avoid inconsistent states. A solution to this could be the concept of transaction, as used in databases, which guarantees that in case of failure, the partial updates of a service execution are rolled back. BPEL adopts an alternative to transactions based on compensation. The execution of actions in a

context may fail or be cancelled for a variety of business and technical reasons (e.g., communication failure). In this case, a *compensation code*, explicitly specified by the provider is executed. For dealing with partially completed tasks, the proposed framework may employ either transaction-based or compensation-code based recovery, depending on the technology adopted by the service provider. For a more thorough discussion on transaction-based or compensation-code based recovery, the interested reader is referred to [5].

In this paper we define what "same skills" means and we present an architecture that can support dynamic selection and hot swapping of a failed task. The dynamic selection procedure takes into account both functional and non-functional aspects of tasks.

2. Related Work

With the advent of web services technology, discovering similar/relevant web services became a key challenge in the context of modeling complex business processes that consist of distributed applications. To achieve this, the development of semantic web services have been introduced, which are annotated based on shared ontologies, typically through DAML+OIL and the latter OWL-S. Such annotations enable the semantically-based discovery of relevant web services [6] and can contribute towards the goal of locating services with "same skills" [3] in order to replace a corrupted service in the process flow. Research effort has been expended on Automatic Service Discovery promoted by the need of business automation. A remarkable research on this area is Aspect-Oriented Programming (AOP) [7]. AOP has been adopted throughout the development of the Web Services Management Layer (WSML) [8]. It offers dynamic identification, com-position, invocation and management of web service(s) independently from the client. In [9], the WSML is proposed as an adaptive middleware layer, placed in between the application and the world of web services. The WSML allows dynamic selection and integration of services into an application, client-side service management, and support for service criteria based on non-functional properties that govern the selection, integration and composition. AOP satisfies requirements for the WSML such as Hot Swapping, Just-in-time integration, Pro-active selection. Note that both WSML and AOP are abstract mechanisms for supporting the basic operations of exception handling; for a fully featured exception handling approach, these operations need to be complemented with semantic elements for service description and rules for "equivalent" service selection, while the basic operations provided by WSML and AOP need to be appropriately invoked. The Service Relevance and Replacement Framework proposed in this paper adopts WSML and AOP as tools for implementing exception handling.

A noteworthy approach to exception handling is the one undertaken by METEOR-S project in cooperation with WSMX (Web Services Execution Environment) [10]. WSMX contains the Discovery component has the role of locating the services that fulfill a specific user request. This task is based on the WSMO conceptual framework for discovery [11] which envisions three main steps in this process: Goal Discovery, Web service Discovery, and Service Discovery. It also includes a Selection component

that apply different techniques ranging from simple "always the first" to multi-criteria selection of variants (e.g., Web services non-functional properties as reliability, security, etc.) and interactions with the service requester. The METEOR (Managing End-To-End OpeRations) project at addressed the issues related to workflow process management for large-scale, complex workflow process applications in real-world multi-enterprise heterogeneous computing environments [12]. The METEOR-S project endeavors to define and support the complete lifecycle of Semantic Web processes [13] using four kinds of semantics – data, functional, non-functional and execution semantics. Data semantics describe the data (inputs/outputs) of the web services, while functional semantics describe the functionality of the web service (what it does), the non-functional semantics describes the non-functional aspects like quality of service and business rules and the execution semantics models the behavior of Web services and processes.

The main difference of our research with the one referenced above is that selection of replacements to services that have failed within an execution plan is made dynamically, instead of using pre-determined exception resolution scenarios. Replacement service selection is based on both functional equivalence (per-formed through semantic matching) and qualitative replaceability (considering non-functional attributes such as response time, security or cost). Furthermore, qualitative replaceability criteria may be defined by the composite service invoker, to more accurately specify which replacement service is the .most suitable one in the context of the current execution. Finally, hot swapping and/or just-in-time integration techniques are employed as supporting technologies to integrate the chosen component into the business process.

3. Exception Handling within Business Processes

Exceptions can arise from changes in resources, organizational structure, company policy, task requirements or task unavailability. Business process systems are currently ill-suited for dealing with exceptions effectively. In this paper we will focus on the automated resolving exception step, and take a closer look to finding new task in order to replace the one that failed in the composite process flow [3]. In workflows systems once an exception has been detected and at least tentatively diagnosed, one is ready to define a prescription that resolves the exception and returns the workflow to a viable state. This can be achieved, by selecting and instantiating one of the generic exception resolution strategies that are associated with the hypothesized diagnosis. These strategies, being processes like any other, are captured in a portion of the process taxonomy, and are annotated with attributes defining the preconditions (expressed using the appropriate query language) that must be satisfied for that strategy to be applicable. We have accumulated many such strategies, including for example:

- IF a sub-process fails, THEN try a different process for achieving the same goal
- IF a highly serial process is operating too slowly to meet an impending dead-line, THEN pipeline (i.e. releasing partial results to allow later tasks to start

earlier provided that this capability is available) or parallelization to in-crease concurrency.

- IF a service component in a serial production line fails to perform a task, THEN re-allocate the task to an appropriately skilled task further down the line.

In this paper we examine the exception resolving with "find new task with the same skills" resolving exception choice. We consider that the phrase "same skills" can be adopted in finding replacement tasks that will not affect the orchestration scenario on a business process flow, as shown in figure 2. We elaborate on the semantics of "same skills" and suggest a framework to overcome this problem.

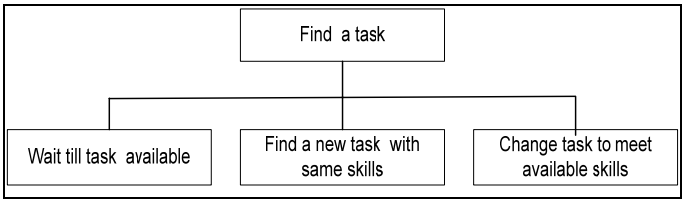

Fig. 2. "Find replacement task" alternatives.

We claim that if a BPEL execution scenario fails due to a single task failure and it is necessary to replace it at run-time with a new one, that task must have "same skills" in order not to violate the SLA between the service provider/broker and the client. If we achieve to define the rules that make two tasks to have "same skills", then we are capable of resolving exceptions mentioned with the most effective way. These rules help us compare tasks, according to a specific policy defined by the BPEL orchestrator.

Tasks have attributes, where are separated to functional and qualitative. For example, we consider a task that receives as input a date of birth, a nationality specification and a social security number, and produces a birth certificate as out-put data. Inputs and outputs are functional attributes of the task. Note here that descriptions of inputs and outputs go beyond the specifications employed in a WSDL specification, since the latter are machine-oriented types, whereas the former include higher-level semantics. To formally model these semantics, domain ontologies or do-main taxonomies can be employed; for example, in the e-government domain, the ontology presented in [14]. Adopting high-level semantics is indispensable, since if ma-chine-oriented types are employed, comparison of functional attributes will be imprecise. For instance, a task modeling an application for a green card might accept as input an application date, a nationality specification and a social security number and produce a green card certificate as output. At machine-type level, the birth certificate and the green card task are indistinguishable since both the input types (date, string, number) and the output type (byte array) are identical; at a higher level of semantics though, it can be easily determined that the tasks are not functionally equivalent.

Task response time, availability, reliability, cost, encryption, reputation and authentication are the qualitative attributes of tasks, complementing the functional attributes. Domain ontologies-taxonomies (for high-level type semantics) along with

the task attributes constitute the task ontology which is used in the process of selecting “same skilled” tasks.

Task ontology

```
<rdf:RDF xmlns:rdf="&rdf;" xmlns:a="&a;"
    xmlns:kb="&kb;" xmlns:rdfs="&rd;">
<rdfs:Class rdf:about="&kb;DataType"
rdfs:label="DataType">
    <rdfs:subClassOf rdf:resource="&rd;Resource"/>
</rdfs:Class>
<rdf:Property rdf:about="&kb;FP_InputParameters"
    rdfs:label="FP_InputParameters">
    <rdfs:range rdf:resource="&kb;DataType"/>
    <rdfs:domain rdf:resource="&kb;Task"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;FP_OutputParameters"
     rdfs:label="FP_OutputParameters">
    <rdfs:range rdf:resource="&kb;DataType"/>
    <rdfs:domain rdf:resource="&kb;Task"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;QP_Authenication"
     a:maxCardinality="1" rdfs:label="QP_Authenication">
    <rdfs:domain rdf:resource="&kb;Task"/>
    <rdfs:range rdf:resource="&rd;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;QP_Availability"
    a:range="float" a:maxCardinality="1"
    rdfs:label="QP_Availability">
    <rdfs:domain rdf:resource="&kb;Task"/>
    <rdfs:range rdf:resource="&rd;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;QP_Cost" a:range="float"
     a:maxCardinality="1" rdfs:label="QP_Cost">
    <rdfs:domain rdf:resource="&kb;Task"/>
    <rdfs:range rdf:resource="&rd;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;QP_Encryption"
    a:maxCardinality="1"rdfs:label="QP_Encryption">
    <rdfs:domain rdf:resource="&kb;Task"/>
    <rdfs:range rdf:resource="&rd;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;QP_Reliability"
    a:range="float" a:maxCardinality="1"
    rdfs:label="QP_Reliability">
    <rdfs:domain rdf:resource="&kb;Task"/>
    <rdfs:range rdf:resource="&rd;Literal"/>
</rdf:Property>
```

```
<rdf:Property rdf:about="&kb;QP_Reputation"
    a:range="integer" a:maxCardinality="1"
    rdfs:label="QP_Reputation">
    <rdfs:domain rdf:resource="&kb;Task"/>
    <rdfs:range rdf:resource="&rd;Literal"/>
</rdf:Property>
<rdf:Property rdf:about="&kb;UDDI_Reference"
    a:range="UDDI_Ref" a:maxCardinality="1"
    rdfs:label="QP_ Reference ">
    <rdfs:domain rdf:resource="&kb;Task"/>
    <rdfs:range rdf:resource="&rd;Literal"/>
</rdf:Property>
<rdfs:Class rdf:about="&kb;Task"
    rdfs:label="Task">
    <rdfs:subClassOf rdf:resource="&rd;Resource"/>
</rdfs:Class>
```

Domain ontologies-taxonomies are placed underneath the "DataType" entity, while for each individual task, a separate instance of the "Task" entity is created. Note that the "Task" entity includes a reference to the UDDI registry, from where additional data (e.g. description and WSDL URI) can be retrieved. In this sense, the task ontology complements the standard UDDI registry instead of replacing it, allowing non-ontology aware orchestrators to keep functioning by querying the UDDI registry only, while ontology aware orchestrators may query both the ontology and the UDDI registry to make use of the additional features.

Having available the task ontology, we can define the "same skills" term:

***Definition*:** We say that two or more tasks have "same skills" when the tasks have identical functional and equivalent qualitative attributes.

The latter attributes include a set of metrics that measure the Quality of Service. Examples of qualitative attributes include response time, availability, security, cost, authentication, reliability, reputation. Two providers of some task supporting the same generic functionally equivalent operation may have different values for their qualitative attributes. These tasks are not identical however in some cases they may be considered as equivalent [15]. Equivalence is defined by a policy. Policy is defined from business process broker in corporation with service requestor. For example, consider two tasks that are functionally equivalent, i.e. they have the same functional attributes. Also, say Task1, already functioning in BPEL execution, has response time RS1, availability A1, cost C1 then a policy of replacement could be: "if exception occurs, replace Task1 with a Task2 when the following condition is true":

$$\text{(same functional attributes) AND } (RS1 \geq RS2) \text{ AND } (C1 \geq C2) \qquad (1)$$

Notice that in this case, the availability attribute is not included in the condition parameters. In other situations, availability or other qualitative attributes may take place in the policy. Figure 3 illustrates the proposed framework for establishing equivalence between tasks.

In the suggested solution we present a Service Relevance and Replacement Framework that consists of 5 modules: Exception Handling Module (EHM), Dynamic

Discovery Module (DDM), Task Selection Module (TSM), Relevance Module (RM) and a Hot-Swapping Module (HSM).

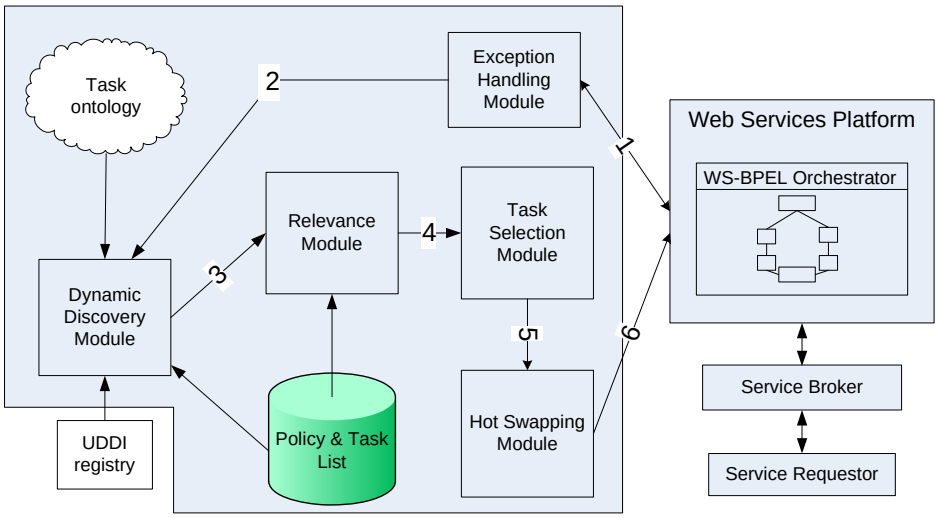

Fig. 3. . Service Relevance and Replacement Framework

The sequence of tasks performed after an exception occurs and the BPEL orchestrator decides to resolve it by finding and replacing the task that stopped functioning presented below step by step. In step1 BPEL orchestrator sends an exception signal/message tagged by the task id and the replacement policy to SRRF. The replacement policy can be any of "default" (all qualitative attribute values of the replacement task need to be "better" than the corresponding attributes of the failed task), "minimize-cost", "limit-cost" (specifying an upper threshold), "minimize-response-time", "maximize-availability" and so forth. This data is pipelined to EHM, which keeps all exception calls in a priority exception queue. In step 2, EHM sends the data DDM needs for discovering functional relevant task using searching semantic in framework's ontology schema and a replacement policy. EHM in step 3 stores the resulting list task ids of qualifying tasks in a local task repository. At the same time EHM ends a list of ids of discovered tasks to RM. Then RM according to equivalence policy selects the tasks that passed and the qualitative conditions and sends, in step 4, to TSM the list of qualifying tasks. Then in step 5, TSM sends the dominant task to HSM, that is responsible of the executing the task replacement at run-time. The algorithm TSM uses to selects the dominant task is out of the scope of this paper. Further analysis of SRRF will include all the execution details. In step 6, HSM replaces the non-functioning task with the selected one and sends the data needed to BPEL orchestrator.

4. Conclusion and Future Work

In this paper we deal with aspects that involve dynamic discovery, selection, and replacement of tasks in a BPEL execution scenario. We introduce the SRRF framework whose main concept is resolving exceptions by finding relevant web tasks, exploiting qualitative and functional metadata semantics. AOP and METEOR-S comes close to efficient service discovery and selection solution, but the tightness with UDDI registries hinders the full resolution of the problem, since the UDDI registry does not encompass adequate semantics. The proposed framework makes use of ontologies and Semantic Web Services to address the problem.

Our future work is to rigorously define the notion of *task equivalence*, elaborate replacement policy infrastructure, and define framework interoperability and communication interfaces between SRRF and software platforms.

5. References

1. Leymann, F., Roller, D., Schmidt, M.-T.: Web services and business process management.IBM Systems Journal, Vol 41, 198 No2 (2002)
2. Newcomer, E., Lomow, G.: Understanding SOA with Web Services, Addison-Wesley, (2005)
3. Dellarocas, C. Klein, M.: A knowledge-based approach for handling exceptions in business processes.Information Technology and Management (2000).
4. Liangzhao Zeng, et al.: Policy-Driven Exception-Management for CompositeWeb Services.Proceedings of the Seventh IEEE International Conference on E-Commerce Technology (2005) 355-363.
5. Korth, H.F., Levy, E.: A Formal Approach to Recovery by Compensating Transactions. Proc. of the 16th Int. Conference on Very Large Data Bases, Brisbane, Australia, September (1990), 95-106
6. Sivashanmugam, K., Verma, K., Sheth, A., Miller, J.: Adding semantics to web services standards. Proceedings of the 1st International Conference on Web Services (2003)
7. Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J.M., Irwin, J.: Aspect-oriented programming. In Aksit, M., Matsuoka, S. (eds.): 11th European Conf. Object-Oriented Programming. LNCS Vol. 1241,Springer Verlag (1997)
8. Charfi A., Mezini, M.: Aspect-Oriented Web Service Composition with AO4BPEL. Proceedings of the European Conference on Web Services (2004)
9. Verheecke, B., Cibrán, M. A., Vanderperren, W., Suvee, D., Jonckers, V.: AOP for Dy-namic Configuration and Management of Web services in Client-Applications. International Journal on Web Services Research (JWSR), Volume 1, Issue 3, (2004)
10. Cimpian, E., Moran, M., Oren, E., Vitvar, T., Zaremba, M.: Overview and Scope of WSMX. Technical report, WSMX Working Draft, http://www.wsmo.org/TR/d13/d13.0/v0.2/
11. Feier, C., Roman, D., Polleres, A., Domingue, J., Stollberg, M., Fensel, D.: Towards Intelligent Web Services: Web Service Modeling Ontology, In Proc. of the International Conf on Intelligent Computing (2005)
12. Kochut, K. J.: METEOR Model version 3. Athens, GA, Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia (1999)
13. Verma K., Sivashanmugam K., Sheth A., Patil A., Oundhakar S., and Miller J., METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Dis-covery of Web services. Journal of Information Technology and Management, Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) 17-39

14. DIP consortium: Data, Information and Process Integration with Semantic Web Services, WP 9: Case Study eGovernment, D9.3,e-Government ontology, with Annex1 eGovernment Ontology (OCML), Annex2 SeamlessUK taxonomy (XML)
15. Medjahed, B.: Semantic Web Enabled Composition of Web Services. PhD dissertation, Faculty of the Virginia Polytechnic Institute and State University (2004)