BPEL scenario execution: QoS-based dynamic adaptation and exception resolution

Christos Kareliotis[1]

National and Kapodistrian University of Athens
Department of Informatics and Telecommunication
ckar@di.uoa.gr

Abstract. BPEL/WSBPEL is the main approach for combining individual web services into integrated business processes. A BPEL/ WSBPEL scenario allows for specifying which services will be invoked, their sequence, the control flow and how data will be exchanged between them. BPEL however does not include mechanisms for considering the invoked services' Quality of Service (QoS) parameters and thus BPEL scenarios cannot customize their execution to the individual user's needs or adapt to the highly dynamic environment of the WEB, where new services may be deployed, old ones withdrawn or existing ones change their QoS parameters. Moreover, infrastructure failures in the distributed environment of the web introduce an additional source of failures that must be considered in the context of QoS-aware service execution. In this thesis, it is proposed a framework for addressing the issues identified above; the framework allows the users to specify the QoS parameters that they require and it undertakes the task of locating and invoking suitable services. In this dissertation two strategies for selecting the most suitable service are considered: (a) a greedy strategy and (b) a partner link-level strategy. The proposed framework intercepts and resolves faults occurring during service invocation, respecting the QoS restrictions specified by the consumer. The latter also intercepts and resolves faults occurring during service invocation, respecting the QoS restrictions specified by the consumer. Finally, methods for tackling with syntactic differences between functionally equivalent services, broadening thus the pool of available services for each adaptation are considered. Finally, performance metrics for the proposed framework are presented, which validate its applicability to operational environments and present performance metrics for the proposed framework.

1 Introduction

Web services have emerged as a new standard, having as main focus to allow applications over the Internet to communicate with each other, which are independent of execution platform, programming language and implementation details. The web service paradigm has been adopted by research community and industry alike, however a number of challenges still lie ahead for fully covering the needs of both service

[1] Dissertation Advisor: Panayiotis Georgiadis, Professor

providers and consumers. [1] identifies a number of open issues in the current SOA state-of-the-art, spanning across four major categories namely service foundations (service oriented middleware backbone that realizes the runtime SOA infrastructure), service composition, service management and monitoring as well as service design and development. For service governance, in particular, [1] lists “service governance” as a major research challenge, stating that the potential composition of services into business processes across organizational boundaries can function properly and efficiently only if the services are effectively governed for compliance with QoS and policy requirements. Services must meet the functional and QoS objectives within the context of the business unit and the enterprises within which they operate.

In this context, development procedures as well as composition and execution mechanisms need to take into account the QoS dimension of web services in order to formulate successful business processes that will satisfy users’ (either business or individuals) expectations. Regarding service composition into business processes, the predominant approach used nowadays is the formulation of BPEL/WSBPEL scenarios [2], in which the BPEL designer specifies the business process logic; this includes invocation of selected web services, control flow constructs and data flow arrangements in the form of result gathering and parameter passing, while provisions for exception handling (such as service unavailability or business logic faults) also exist.

BPEL scenarios, however, do not include facilities either for specifying QoS parameters for services, or for dynamically selecting the web service to be called at runtime, therefore the BPEL scenario designer must select the concrete service implementation to be invoked in the context of the business process while creating the scenario, by examining the QoS parameters of functionally-equivalent services. This alternative, however, is not a viable one since (a) the same BPEL scenario may be used by different users with diverging or even contradictory requirements and (b) even if the “best choice” is made at some time point there is no guarantee that this choice will continue to be optimal in the future. Moreover, in the presence of failures, it would be desirable for the system to be able to locate and use “second best” choices automatically, provided that they deliver the required functionality and satisfy QoS restrictions.

2 Summary

2.1 Motivation and Challenges

The main objective of web service technology and related research [3] 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) [4], [5]. Trying to model real world business processes with BPEL scenarios a series of challenging issues may emerge. Specifically:

1. processes may be long-running, in the order of hours, days or even longer. Such issues commonly arise in cases where human intervention is required for the completion of all or some of the services that comprise the process.
2. BPEL scenarios may try to model stable and established processes that remain relatively unchanged. Examples of such processes are those that represent interactions with Government-based services, spanning the range of G2x and x2G acronyms
3. as the complexity of the process and the number of cooperating services needed increase, so does the volatility of these services. New services implementing the same process may appear, existing ones may be decommissioned or the BPEL designer may not be aware of all the services that can be utilized at the time of the designing phase
4. quality requirements for the process may change during the lifetime of the BPEL scenario. This may be due to different needs of end-users (a real-world counterpart of this case is one person sending a package using courier mail to minimize delivery time, whereas another person may use ordinary surface mail to pay less), or alterations in organizational policy.

2.2 Problem Identification and Objective

In cases such as the above, the static nature of BPEL scenarios and their handling of BPEL engines fail to accommodate for the dynamic nature of real world processes. To cope with these situations, the BPEL scenario would have to be redesigned and re-deployed possibly forcing existing transactions to fail or be re-started. For accommodating different needs of end-users, the alternative approach of maintaining different versions of the BPEL scenarios could be also taken, with each version being targeted to a specific user category (e.g. “express delivery” vs. “economic delivery”); this arrangement, however, would increase development and maintenance costs and would weaken the overall system manageability.

To tackle these issues, this dissertation proposes an approach that is relying on *dynamic service selection mechanism based on functional and non-functional (quality) criteria* for selecting the most suitable service *per scenario invocation*. Furthermore, this mechanism provides for non-existent or invalidated services allowing them to be replaced with existent and valid ones, choosing the optimal candidate per service invocation based on current criteria. The criteria can be different on each run and can provide for diverse needs depending on the invoker.

So, the basic features and innovations this dissertation introduced were:

- the concept of replacement candidate for web services was formalized considering criteria related to the specific BPEL scenario execution, instead of the generic functionality or behavior of the service. Replacement candidates are used for hot-swapping failed services within a BPEL transaction, allowing thus the BPEL scenario to complete its execution. The formalization introduced allows including more services in the “replacement candidate” pool and therefore formulating execution paths with better qualitative characteristics.

- the notion of service selection affinity was introduced, which allowed for maintaining the transactional characteristics of BPEL scenarios in the presence of adaptation
- an approach to bridging the syntactic differences between functionally equivalent services was proposed, which greatly enhances the maintainability of the equivalent services repository, trading off a degradation in performance, which has been quantified to be quite small.
- a method for distinguishing between system faults and business logic faults was proposed; this distinction is important since faults in the former category can be resolved by automatically invoking a replacement candidate for the failed service, while this is not possible for faults in the second category.
- a framework that enables the automatic resolution of system faults and the dynamic adaptation of BPEL scenario execution according to QoS criteria was proposed. The framework is independent of the particular BPEL execution engine used, and methods have been proposed for setting the QoS criteria granularity (for all scenarios executing in the system; for the scenario as a whole; for each individual service within a scenario). This framework includes provisions for maintaining the transactional characteristics of BPEL scenario execution, making use of the *service selection affinity* notion.
- the feasibility of the above was proved through a complete system implementation and quantification of its performance.
- the issue of BPEL scenario adaptation in the context of secure web services invocation was identified, and a system architecture for a system that supports such an adaptation was drafted.

2.3 Related Work

In this section some related work is adduced in the following research directions:
QoS management in web services composition:

In [6] a framework is presented named AgFlow [7] as middleware platform that enables the quality-driven composition of Web services. In AgFlow, the QoS of Web services is evaluated by means of an extensible multidimensional QoS model. It presents two selection policies: the local optimization of individual tasks and a global planning. The first is similar to the one proposed in this thesis and it uses the Simple Additive Weighting [8] technique to select the optimal service for a given task. The proposed approach differentiate from this since we deal with already defined composition scenario and doesn't propose a re-planning solution method in order to change the task execution order, or replace a set of task with another set. It uses a proxy-like service that is invoked for each individual task in the business scenario in order to discover the optimal services for each one of them based on a specific consumer's quality policy at execution time.

In [9] a web service proxy is introduced in order to perform a dynamic binding of related web services under specified user's constraints. The selection of equivalent services is not only filtered by constraints but also it is measured the quality score for each equivalent service depending on a quality vector and a set of quality weights. In [10] the importance of qualify-able QoS aspect related to the issue of web services

composition and monitoring is illustrated. It describes an algorithm capable of capturing and reflecting the state of web services involved in the integration process.

In exception management web services composition:

In this research work [11] a policy-driven approach is introduced to exception management. An exception handling policy language is designed, which defines deviation situations and the associated exception handlers. The proposed approach complements the above solution by discovering an optimal alternate service task to perform the alternative action mentioned. A remarkable research in this area has been and the one introduced in [12]. It's presenting a component called BPBot (Business Process roBOT). A business process is executed by a collection of BPBots that are dynamically organized as a hierarchical structure. The proposed solution is not re-planning an execution path, but it discovers functionally and qualitatively equivalent services to perform the determined business tasks without changing the task execution sequence. Moreover, during this dissertation the author published relative papers ([20], [21], [22], [23], [24]) considering service BPEL scenario adaptation in the context of exception resolution and security issues in exception handling in [25].

Semantic Web Services:

In the past few years, the issue of exception resolution in composite web services has drawn the researchers' attention. A noteworthy approach to exception handling is the one undertaken by METEOR-S project [13], [14] in cooperation with WSMX (Web Services Execution Environment) [15]. WSMX contains the discovery component, which undertakes the role of locating the services that fulfill a specific user request. This task is based on the WSMO conceptual framework for discovery [16]. WSMO includes a Selection component that applies 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 requestor. Both in the METEOR-S and other approaches, functional and non-functional properties are represented using shared ontologies, typically expressed using DAML+OIL [17] and the latter OWL-S. Such annotations enable the semantically based discovery of relevant web services and can contribute towards the goal of locating services with "same skills" [18] in order to replace a failed service in the process flow. The main difference of the research illustrated 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 (performed through semantic matching) and qualitative replaceability (considering non-functional attributes). 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.

2.4 Brief Description

Service Quality Vectors

In order to enable the selection of the "most suitable" operation according to some QoS specification, the QoS attributes of the operations should be represented in an unambiguous and system-processable format, while additionally means for expressing QoS-related operation selection criteria should be afforded. For brevity, in the

following we will consider only the QoS parameters cost, security, performance, response time and availability, adopting the definitions in [19]. For each such source, mappings between the domains employed by the source and numeric values are used.

Table 1. Mapping of QoS values

	QoS provider 1	QoS provider 2	Value
Cost	10 €	11 €	1
Security	6 (out of 10)	62 (out of 100)	3
Performance	High throughput	99%	5
Response time	0.0001 ms	Real-time	1
Availability	High	> 95%	4

In the approach illustrated here three vectors that define the QoS criteria for process invocation are considered; in other words it is defined a QoS specification as a triple (MAX, MIN, W), where MAX, MIN and W are quality vectors (defined below). The quality vector for the QoS attributes considered in this work can be defined as:

Table 2. Quality Vector

MAX	=	(costmax, secmax, perfmax, respmax, availmax)
MIN	=	(costmin, secmin, perfmin, respmin, availmin)
W	=	(costw, secw, perfw, respw, availw)

ASOB-Framework

Figure 2 illustrates the overall architecture of our approach to dynamic policy-driven execution of a business scenario QoS-aware and policy-adhering exception management techniques. The component undertaking this responsibility is the Alternative Service Operation Bind (ASOB).

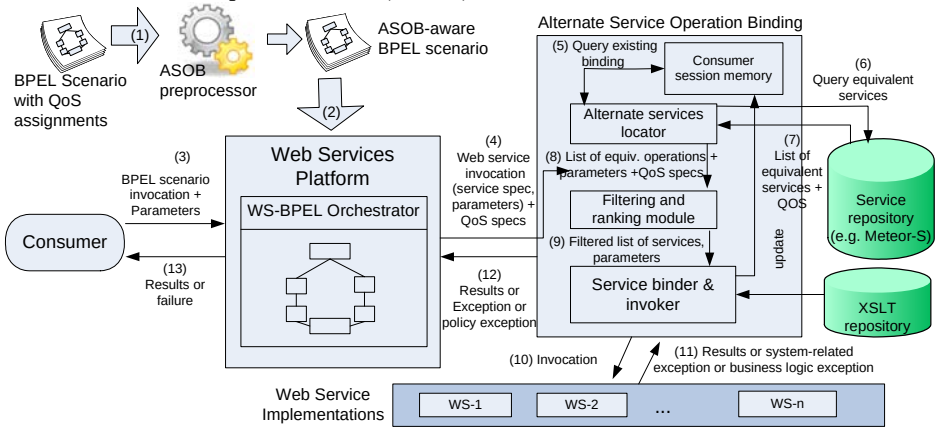

Fig. 1. Overall System Architecture

The BPEL scenario (SC) as crafted by the BPEL designer is processed by the *ASOB preprocessor*, which produces an *ASOB-aware BPEL scenario* (SC_{ASOB}) as output, so that for each service, the ASOB middleware calculates an overall score which takes into account all the operations of the service that are listed in the BPEL scenario and the respective QoS weights that the client has specified at the pre-procession phase.

$$Sc_{WS} = \sum_{op \in Ops} \sum_{atttr \in \{\cos t, \sec, ...\}} attr_{WS,op} * QoS_w(op)_{attr} \quad (1)$$

Depending on the score of each service, in case of a failure, ASOB replaces the failed one with the service that owns the highest score Sc. The interested reader will find in more depth the main processing of the ASOB framework at the main dissertation text.

3 Results and Discussion

The contribution of the ASOB framework to the field is as follows:

1. it allows the BPEL scenario designer to specify the desired QoS parameters for each service. These parameters are specified through standard BPEL variables, thus the designer may examine scenario input parameters for setting them, tuning thus the adaptation of the particular BPEL scenario execution to the desires and needs of the scenario consumer.
2. it does not require any modification to the BPEL syntax or semantics.
3. it takes the execution flow specified by the designer as granted, and optimizes service selection within this flow, contrary to service composition approaches which define this flow dynamically. This is an important aspect in cases where execution flow is carefully crafted by the designer to reflect particularities of the business process, specialized exception handlers are used, etc.
4. it incorporates exception handling as an integral part of the adaptation process, allowing for switching to the “next best” solution when the originally selected candidate is unavailable.
5. it does not use pre-determined alternative paths, but selects services dynamically from a suitable registry.
6. It employs XSLT transformations through which the middleware bridges the syntactic differences between the service originally specified in the BPEL scenario and other services that are semantically equivalent but syntactically different. This arrangement offers to the middleware a wider range of choices, for the stage of deciding which service provider best matches the QoS specifications given in the BPEL scenario.
7. it considers service selection affinity, enabling the conducting of multi-operation transactions with providers.

8. it introduces the notion of the service replacement candidate, which relaxes the requirements for service equivalence. Service replacement candidates are computed for the context of a particular BPEL scenario and takes into account only the operations used in the scenario and not all operations offered by the services. This arrangement enables the middleware to avoid cases where some operation that is not used in a scenario breaks the equivalence of two services, and thus disallows the consideration of some alternates.
9. it elaborates on the management of consumer session memory, which supports the maintenance of service selection affinity.
10. it provides full details for the algorithms used by the middleware to process web service invocations.
11. it includes a partner link-level strategy for deciding which is the service provider that best matches the QoS profile specified in the BPEL scenario; the partner link-level strategy can significantly improve the service provider selection when a BPEL scenario uses multiple operations from the same service provider, while it may also prevent some cases where the greedy strategy is unable to find any appropriate execution path for servicing the scenario.

Algorithms in pseudo-code can be found in the main text of this dissertation.

3.1 Performance Evaluation and Results

Figure 2 illustrates the ASOB internal process time for single web service operation invocations, against the overall service repository (SR) size and the number of equivalent services present in the repository. The overhead increment, on the other hand, when the number of alternate services increases is considerable, mainly affecting the sorting of the candidate operation list (typically of complexity O(n * log(n)).

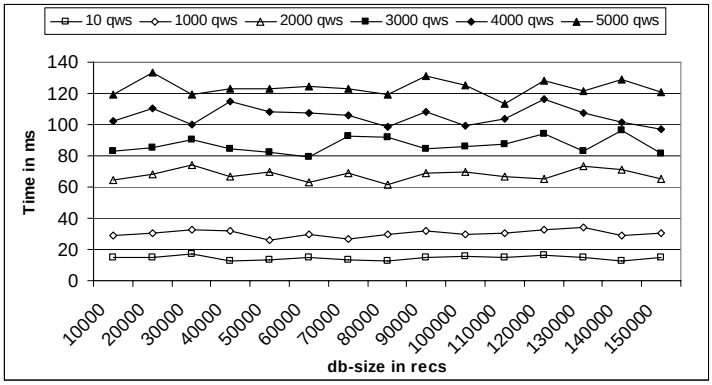

Fig. 2. ASOB internal process time

Table 3. XSLT transformation overhead

concurrent ASOB invocations	20	40	60	80	100
time in msecs (average per transformation)	17.8	18.5	34.5	46.2	61.7

Table 3 shows the overhead incurred by applying XSLT transforms on request and response SOAP messages, to resolve syntactical differences between operations that are *semantically* but not *syntactically* equivalent

Figure 3 illustrates the number of operation invocations that can be served in a unit of time against the number of concurrent invocations when (a) services are directly invoked and (b) when invocations are made through the ASOB middleware.

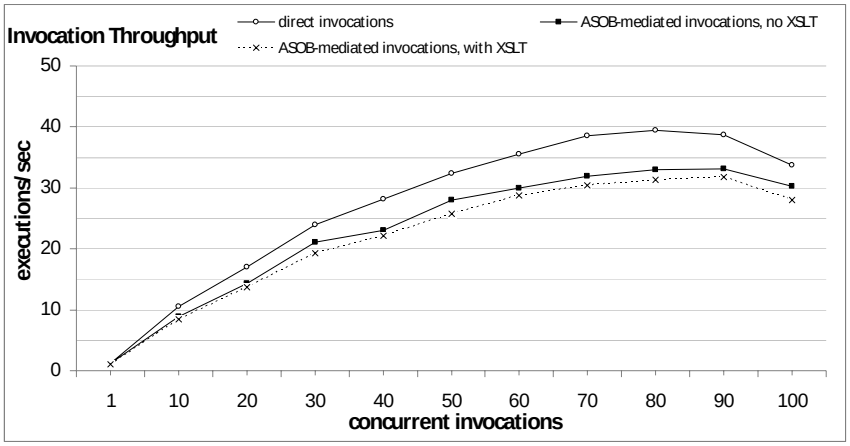

Fig. 3. Invocation throughput

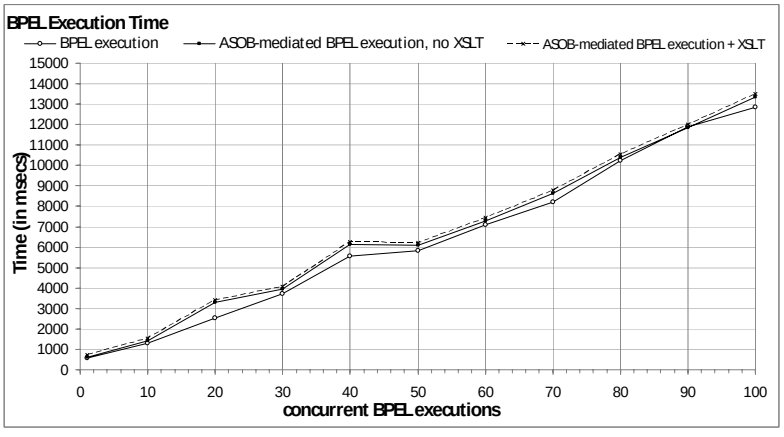

Fig. 4. BPEL scenario execution time

Figure 4 illustrates the BPEL execution time of a BPEL scenario containing two web service invocations against the number of concurrent executions. The increment is very small (4%-9% without XSLT transformations, 8-16% with XSLT transformations).

Figure 5 depicts the BPEL scenario execution throughput against the number of concurrent executions. The behavior is consistent with the previous diagrams.

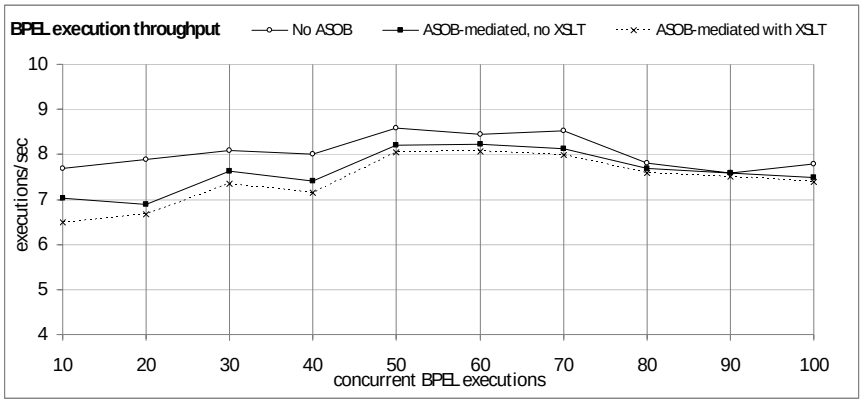

Fig. 5. ASOB-mediated vs. direct invocation BPEL scenario execution throughput

4 Conclusions

Building processes that are able to cope with the dynamics of real world requirements has always been a challenging endeavor. The adoption of BPEL in the design and execution phases of business processes has already obtained gains in speed and reliability, but has not been able insofar to successfully address issues arising form the dynamic nature of the processes themselves, the diversity in user requirements and the inherent instability of distributed environments, which leads to a number of system faults.

The framework presented in this dissertation addresses these shortcomings by employing a dynamic service selection mechanism based on QoS criteria for a BPEL process; these criteria are defined by the BPEL scenario designer and can be set to reflect the end-user requirements. Service attributes are stored in a repository that stores the services' functional and non-functional (qualitative) characteristics. Updating the repository suffices to reflect changes in the real world (service introductions or withdrawals, changing of services' QoS aspects etc). An exception resolution mechanism for faults owing to systemic reasons is also included, easing thus the work of the BPEL designer.

The strategy employed by the presented framework for binding a partner link to a specific service provider can follow either (a) a *greedy strategy*, according to which the QoS aspects of only the first operation invoked for a particular partner link are examined to determine the binding or (b) a p*artner link-level strategy*, which reviews all invocations collectively, avoiding suboptimal bindings and cases where the greedy strategy leads to inability to successfully conclude the BPEL scenario.

Open issues in this field includes a detailed evaluation of the partner link-level strategy regarding (a) its performance, i.e. the time needed to determine the optimal binding for a partner link and (b) the quality of the execution plans it produces. Execution plan quality is a twofold aspect involving (i) the degree to which the bindings performed by the middleware correspond to the QoS specifications listed in the BPEL scenario and (ii) the number of cases where the partner link-level strategy bindings lead to successful execution of the BPEL scenario, contrary to the bindings of the greedy algorithm. Moreover, it could be investigates the collection and exploitation of statistics regarding the number of invocations for each particular operation in the context of a specific BPEL scenario, so as to use a more elaborate weight assignment in the phase of calculating the suitability scores of different bindings.

References

1. M. P. Papazoglou, P. Traverso, Leymann, Service-Oriented Computing: State of the Art and Research Challenges. IEEE Computer (40) 11, Nov. 2007, pp. 38-45.
2. M. Juric, Business Process Execution Language for Web Services BPEL and BPEL4WS (2nd Edition), Packt Publishing, 2006, ISBN-10: 1904811817.
3. Newcomer, E., Lomow, G.: Understanding SOA with Web Services, Addison-Wesley, (2005)
4. F. Leymann, D. Roller, and M. T. Schmidt, Web services and business process management, Available at: http://www.research.ib m.com/journal/sj/412/leymann.html)
5. Martin Hepp, Frank Leymann, John Domingue, Alexander Wahler, and Dieter Fensel Semantic Business Process Management: A Vision Towards Using Semantic Web Services for Business Process Management, IEEE International Conference on e-Business Engineering, 2005, p:535-540
6. Liangzhao Zeng, Boualem Benatallah, Anne H.H. Ngu, Marlon Dumas,Jayant Kalagnanam, and Henry Chang. Qos-aware middleware for web services composition. IEEE Trans. Softw. Eng., 30(5):311–327, 2004.
7. L. Zeng, Dynamic Web Services Composition, PhD thesis, Univ. of New South Wales, 2003.
8. H.C.L and, K. Yoon, Multiple Criteria Decision Making," Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, 1981.
9. K. Verma, R. Akkiraju, R. Goodwin, P. Doshi, J. Lee, On Accommodating Inter Service Dependencies in Web Process Flow Composi-tion, AAAI Spring Symposium PP: 37-43 on Semantic Web Ser-vices.
10. Hassan Issa, Chadi Assi, Mourad Debbabi, QoS-Aware Middleware for Web Services Composition - A Qualitative Approach, in Proceedings of the 11th IEEE Symposium on Computers and Communications, 2006
11. Liangzhao Zeng; Hui Lei; Junjang Jeng; Jen-Yao Chung; Benatallah, B. *Policy-driven exception-management for composite Web services, E-Commerce Technology*, 2005. CEC 2005. 7th IEEE International Conference on Volume , Issue , 19-22 July 2005, 355 – 363
12. Liangzhao Zeng, JunJan Jeng, Santhosh Kumaran and Jayant Kalagnanam, Reliable Execution Planning and Exception Handling for Business Process, LNCS, Springer, Technologies for E-Services, 2003. p.119-130
13. Kochut, K. J.: METEOR Model version 3. Athens, GA, Large Scale Distributed Information Systems Lab, Department of Computer Science, University of Georgia (1999)

14. K. Verma, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller, METEOR-S WSDI: A Scalable Infrastructure of Registries for Semantic Publication andDiscovery of Web services. Journal of Information Technology and Management, Special Issue on Universal Global Integration, Vol. 6, No. 1 (2005) 17-39
15. 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/
16. D. Roman, D2v1.2 Web Service Modeling Ontology (WSMO). WSMO Final Draft April 13, 2005. Available at: http://www.wsmo.org/TR/d2/v1.2/20050413/
17. DAML+OIL, Available at: http://www.daml.org/ 2001/03/daml+oil-index.html
18. Dellarocas, C. and M. Klein, A knowledge-based approach for handling exceptions in business processes, Information Technology and Management 2000.
19. O'Sullivan J., Edmond D., and A. Ter Hofstede (2002), What is a Service?: Towards Accurate Description of Non-Functional Properties, Distributed and Parallel Databases, 12.
20. Kareliotis C., Vassilakis C., Rouvas E., Georgiadis P. (2008), Exception Resolution for BPEL Processes: a Middleware-based Framework and Performance Evaluation. Procs of iiWAS 2008, Linz, Austria.
21. Kareliotis C., Vassilakis C., Georgiadis P. (2007), Enhancing BPEL scenarios with Dynamic Relevance-Based Exception Handling, Proceedings of the ICWS 2007, pp.751-758.
22. Kareliotis C., Vassilakis C., Rouvas E., Georgiadis P. (2009), QoS-Driven Adaptation of BPEL Scenario Execution. Procs of ICWS 2009, July 6-10, 2009, Los Angeles, CA, USA.
23. Christos Kareliotis, Costas Vassilakis, Stathis Rouvas, Panagiotis Georgiadis. QoS-Aware BPEL Scenario Execution and Adaptation: A Middleware-Oriented Framework. Extended version invited from ICWS09 in International Journal on Web Services Research. JWSR. 2009.
24. Kareliotis Christos, Vassilakis Costas, Georgiadis Panagiotis: Towards Dynamic, Relevance-Driven Exception Resolution in Composite Web Services. Proceedings of International Conference on Object Oriented Programming, Systems, Languages and Applications 2006.
25. Costas Vassilakis, Kareliotis Christos: A framework for adaptation in secure web services. 4th Mediterranean Conference on Information Systems 2009. MCIS, Athens, Greece