Scalable Computing: Practice and Experience
Volume 12, Number 4, pp. 377–384. http://www.scpe.org
ISSN 1895-1767

TOWARDS A TAXONOMY OF WEB SERVICE COMPOSITION APPROACHES

DESSISLAVA PETROVA-ANTONOVA* AND ALEKSANDAR DIMOV†

Abstract. Service-Oriented Architecture (SOA) is a well known paradigm for development of flexible and loose coupled software applications using services that are available in a network. The latter provide business functionality through well-defined interfaces that can be dynamically discovered. Services can be aggregated into more complex ones called composite services. Currently, there exist a lot of composition approaches that serve different goals. In order to be able to comprehensively study the web-service composition process, different approaches should be analyzed and organized into appropriate taxonomy framework. This paper presents an overview of current approaches for service composition and further analyzes them toward various aspects of the composition model.

Key words: Composition model, Service-Oriented Architecture, Web services

1. Introduction. Latest trend in software engineering is presented by the so-called Service-Oriented Architecture (*SOA*). It makes possible development of software intensive systems via loosely coupled services, which provide business functionality through contractually specified interfaces. The promise of services is for increased maintainability and scalability of systems, achieved at predictable time with less efforts and decreased cost of development.

The preferred way to realize SOA is based on web services. The basic web service infrastructure founded on standards like WSDL, SOAP and UDDI provides simple interaction between clients and web services. Functionality of the latter is often combined into composite ones that satisfy specific business goals. In such case, additional composition models, languages, as well as development and run-time environments should be elaborated. Although a lot of efforts for development of web service composition methods are available, most of them aim toward satisfying particular needs. Some composition approaches focus mainly on QoS aspects of the composition [2], [8], [9], while the goal of others is to provide web service orchestration [4], [6], [7], [12], [14]. In such context this paper aims to study the fundamental characteristics and essential aspects of different approaches. For this purpose we extend the number of the dimensions of an existing composition model and provide comparative analysis of different web service composition approaches according to these dimensions.

Some currently existing literature surveys on web service composition approaches directly relate to our work. For instance, in [21], authors present in details a large number of approaches, but only few of them are compared according to the preliminary defined criteria. Rao and SU discuss in [22] only the approaches based on AI Planning. The approaches using Petri nets, statecharts and other techniques for orchestration modeling are not considered. In this paper several categories of orchestration modeling techniques are presented and used as a comparison criterion of presented approaches. Evaluation of QoS Based web service selection techniques for service composition is given in [23]. In this paper, other dimensions of the composition process such as data modeling, transaction modeling and exception handling, are not mentioned. In our work we extend the above mentioned research, by establishing a more sound classification and comparison scheme, which could serve as a basis for a taxonomy framework of the web service composition process.

The rest of the paper abstract is organized as follows: Section 2 introduces the current web service composition approaches, Section 3 analyzes the approaches with respect to aforesaid dimensions and Section 4 concludes the paper.

2. Overview of Web Service Composition Approaches. This section introduces the key aspects of the current web service composition approaches.

Chang and Lee [2] propose a quality driven web service composition methodology, which evaluates the quality of web services in three dimensions: QoS (quality of services), QoC (quality of contents), and QoD (quality of devices). QoS is related to the quality properties of web services such as performance, interoperability, security, and so on. QoC considers the quality of the context in which the web services work. QoD addresses the quality of devices and the physical environment in which the web services operate. The web service composition methodology uses Multi-Criteria Decision Making solution, called Preference Ranking Organization METHod

*Sofia University, Faculty of Mathematics and Informatics, Department of Software Engineering, 1164 Sofia, Bulgaria (d.petrova@fmi.uni-sofia.bg).

†Sofia University, Faculty of Mathematics and Informatics, Department of Software Engineering, 1164 Sofia, Bulgaria (aldi@fmi.uni-sofia.bg).

for Enrichment Evaluations (PROMETHEE). The proposed method for web service composition consists of six steps. First, web service composition is described as abstract workflow using Business Process Modeling Notation (BPMN). Next, quality factors from the three dimensions are selected and their quality weights are determined. Then, constraints of quality factors are defined and preference indexes for all quality factors are specified. The preference index expresses the preference of one service over another considering all quality factors. Finally, outranking flows between web service candidates are determined according to the preference indexes. The proposed methodology is implemented in a tool, named Event-driven web service composer, and is evaluated through an example process with five tasks. Twenty five web services and ten event services are developed as candidates for the execution of the process's tasks.

Qiao et al. [3] a broker based architecture for dynamic QoS monitoring and adaptation for composite web services. The web service compositions are described with Business Process Execution Language (BPEL), which is extended with a new construct, called flexPath, allowing definition of multiple execution paths of the BPEL process. The proposed architecture consists of two main components: QoS broker and BPEL compiler. The QoS broker is responsible to monitor and adapt the QoS properties. It keeps track of the execution of the BPEL process based on the schema of the running instance and the current execution point. The values of QoS properties are measured at run-time using a probing technique. The key role of the QoS broker is to decide whether the adaptation has to be triggered and if so, to find a better execution path in order to improve the QoS. The BPEL compiler instruments the BPEL process with a new logic in order to communicate with the QoS broker. The proposed components of the architecture are implemented in a prototype tool and verified through a BPEL process that invokes a number of dummy partner Web services. A disadvantage of the architecture is that it handles only Response time as QoS property. The measured Response time in the broker is not the same as those obtained from user experiences. This drawback is due to the usage a third party monitor.

Zheng and Yan [4] model web service composition problem as a Planning Graph (PG). Each web service in a composition is mapped to an action of a PG. The input parameters of the web service are mapped to the action's preconditions, and its output parameters are mapped to the action's effects. The input parameters of the composition request are mapped to the initial state of a planning problem. The output parameters of the composition request are mapped to goal propositions. The proposed algorithm takes as an input a set of actions, an initial state, and a set of goal propositions. Firstly, it assigns an initial state to a proposition set in level 0 of a simplified PG. Then, the algorithm expands PG iteratively until it reaches a level where a proposition set contains goal propositions or the fixed point level of PG. If the former happens first, then the algorithm outputs a solution. Otherwise, the solution is not produced. The performance of the proposed approach is studied through a sample repository, containing 143 web services. It is compared with the performance of Service Composition Algorithm used in Georgetown Java Software. A drawback of the proposed approach is that it sometimes produces redundant web services.

Yu and Reiff-Marganiec [5] propose a technique for web service composition using Planning as Model Checking (PMC). They translate a message based paradigm to a state oriented one by allowing a single operation in the web service to imply a state, which essentially encapsulates the change after execution of the operation. Each operation is modeled in four aspects, namely quality, domain, purpose and communication. The quality aspect captures non-functional requirements of the web service. The domain aspect describes the area of interest of the operation. The purpose aspect refers to the aim of the operation. The communication aspect defines a message exchange protocol that is used for interaction with the clients and other web services. The model of the web service operation is applied in the design of a composition framework, which has four phases. The first phase specifies the planning goal and describes initial knowledge about the client as input for the next phase. On the second phase, a plan search space is built. The third phase runs the proposed algorithm to search for plans. On the fourth phase, the clients choose the best plan. Then, the composition framework generates an executable plan that is described in BPEL. It monitors the execution of the web service composition and in case of failures automatically revises the executable plan using the alternative ones, obtained in the third phase. A disadvantage of the composition framework is that the clients need to select the best plan instead of the composition framework itself. This can be done automatically based on the non-functional properties of web services.

The AI planning is also applied to the problem of web service composition by Klusch and Gerber [6]. They propose a semantic web service composition planner, called OWLS-XPlan. OWLS-XPlan takes OWL-S web service descriptions, OWL domain description and a planning goal as input and generates a planning sequence of web services, which corresponds to the planning goal. The web service descriptions and domain descriptions

are transformed in Planning Domain Definition Language (PDDL). They are used to create a PDDL plan that solves a given problem in the actual domain. OWLS-XPlan is evaluated according to completeness of planning, the average plan length in relation to the complexity of the problem definition and the average plan quality in terms of the average distance of individual plans from the optimal solution of a given problem. Bartalos and Bielikova [7] also propose a solution of semantic web service composition problem. They propose an approach, which creates web services workflows, satisfying a given goal. The goal is described as a pair: concept type and constraint. The concept type is a concept from the ontology used for semantic annotation of web services. The constraint is defined as a first order logic formula. The proposed approach generates a plan containing all possible alternative branches that lead to the goal. The condition determining which branch will be taken is evaluated during execution. The composition process continues until there is no web service which has not specified source for its input data. During the composition, the value restrictions defined in the goal are back propagated based on the pre- and post-conditions of chained services. The proposed approach is evaluated with respect of performance and usefulness.

Cardellini et al. [8] present a web service composition approach, where web service selection is carried out per flows of requests rather than per request. The web service selection problem is scaled to Linear Programming Optimization problem and takes into account QoS properties of the candidate web services. Its solution is used in the design of a service broker, which performs several tasks. First, it defines a business process in BPEL for the requested web service composition and discovers the candidate web services. Next, the service broker negotiates Service Level Agreements (SLAs) with the providers of the candidate web services. During negotiation it establishes the values of QoS properties of each web service in correspondence with a mean volume of requests generated by the broker for that service. Then, the negotiation continues with the requestor establishing the offered QoS level of the composition in correspondence with a mean volume of requests generated by the requestor to the broker. Finally, the selection of concrete web services is realized by solving the optimization problem. The service broker also collects information about web service composition usage. The collected data is used to find out whether a new solution of the optimization problem is required.

Chakhar et al. [9] describe a framework for composite web services selection based on multicriteria evaluation. The proposed solution extends the current web service architecture by adding a Multicriteria Evaluation Component (MEC) in the UDDI registry. MEC takes as input a set of composite web services and a set of evaluation criteria. Its output is a set of recommended composite web services. The set of QoS evaluation criteria is extracted from the SOAP message sent by the client to the UDDI registry in order to find a given web service. They are transformed into quantitative ones by assigning values to the qualitative data. Potential web service compositions are constructed as graphs and then evaluated according to preliminary defined rules. The proposed framework is under development, but its feasibility is shown through an illustrative example.

Aiello et al. [10] propose an algorithm that creates a web service composition, which satisfies the functional requirements of the client. The algorithm is based on an extended Breadth First Search (BFS) algorithm that uses a priority queue with cost based on Response time and Throughput. It is integrated in a system, called RuGQoS, which consist of three components: XML parser, Composition engine and BPEL code generator. The XML parser translates the WSDL, WSLA and OWL descriptions into a set of indexes where each operation name is associated with a web service and its corresponding Response time and Throughput. The Composition engine implements the BFS algorithm. The BPEL code generator converts the output from the Composition engine into business process described in BPEL.

Lecue [11] proposes an approach for web service composition based on the semantic similarities between input and output web service parameters. The semantic similarities are evaluated using semantic links that are specified by statecharts (S). The semantic links are valued with two quality criteria, namely Common description rate and Matching quality. The common description rate provides one possible measure for the degree of similarity between an output parameter of one web service and an input parameter of another one. The matching quality is a value in the range of [0, 1] that can be 1 (Exact), 3/4 (PlugIn), 1/2 (Subsume) or 1/4 (Intersection). The semantic links are also augmented with two QoS properties of web services: Execution price and Response time. Thus, the proposed approach of web service computation optimizes both QoS and the quality of semantic fit.

Sohrabi and McIlraith [12] present a template based web service composition system using Hierarchical Task Networks (HTNs). The system assumes that both web services and composition templates, specified by the clients, are described in OWL-S. OWL-S service profiles and process models are translated into HTNs. The client preferences are described with extension of Planning Domain Definition Language (PDDL3), which allows specification of preferences over how tasks are decomposed as well as over QoS properties of web services. The

web service compositions generated by the proposed system adhere to policies and regulations expressed as a subset of Linear Temporal Logic (LTL). Thus, the system guarantees that the synthesized composition preserves certain properties of the world.

Pistore et al. [13] address the web service composition problem by developing a planning technique based on the Planning as Model checking approach. The planning process uses a BPEL4WS description of the external protocols and client requirements for the composition. The external protocols are represented by means of finite state machines due to their nondeterminism and partial observability. The clients requirements are expressed in a goal language, called EaGLe, and are used to navigate the planning. As a result, an executable BPEL4WS process and a monitor are generated. The monitor checks the actual interactions of the BPEL4WS process with the external web services and detects incorrect ones. The proposed planning technique is implemented in a BPM planner and is evaluated on a sample case study.

Farhan et al. [14] propose a framework for web service composition that consists of four main components: Translator, Evaluator, Composer, Execution Engine and Matching Engine. The Translator converts the client request into a form used by the framework. The Matching Engine checks for the requested web services in a web service database. If the requested web services are not found, then it starts to search in UDDI registries. The results of searching are sent to the Evaluator, which evaluates the web services using interface and functionality based rules. After that it sends selected services to the Composer in order to generate a composition. The result composition is executed by the Execution Engine. Finally, the results are sent to the client through the Translator.

Zeng et al. [15] propose a middleware platform for QoS-driven web service composition, called AgFlow. The QoS properties of web services are presented with a multi-dimensional QoS model. The model considers only five QoS properties, namely Execution price, Execution duration, Reputation, Reliability, and Availability, which are also used to evaluate the quality of the final web service composition. The composition is specified as a collection of generic service tasks described in terms of service ontologies and combined according to a set of control-flow and data-flow dependencies. These dependencies are presented as statcharts. AgFlow implements two composition approaches: local optimization and global planning. The local optimization approach performs optimal web service selection for each individual task without considering QoS properties of web services. The global planning approach is based on the preferences specified for the whole composition as well as QoS properties of web services. It uses integer programming in order to compute optimal plan that correspond to the composition. The proposed platform provides an adaptive execution engine, which re-plans the execution of the composition in response to changing QoS. It is implemented as a prototype system and is evaluated over a travel planning application.

Mili et al. [16] address web service composition problem as a function cover problem. The web service composition process starts an algorithm with the strictest interpretation of type equivalence, and then invokes looser versions only if the current ones fail to return appropriate results. The proposed solution is inspired by the programming languages, which handle type equivalence using one of two strategies: name equivalence and structural equivalence.

Che et al. [17] use a XML nets to model control flows and data flows of web service compositions as well as to discover and select web services for that compositions. Control flows can be easily modeled with XML nets due to their graphical nature. The data flow modeling is more complex due to problems that can occur during message passing. One of them is incompatibility between an output message of one web service and an input message of another one. XML nets adjust output and input messages by adding a web service chaining transition as mediator between two web service invocation transitions. The mediating transitions are applicable to message aggregation or disassembly. Web service discovery and selection process is aligned to definition and (re)configuration of XML net process rules. Transition inscriptions are used to define constraints for web service selection.

Lin et al. [18] propose a web service composition technique, in which the web services are described in OWL-S and the client preferences are described in PDDL3. The planning of web service composition combines HTNs with best-first search using a heuristic selection mechanism based on ontological reasoning. Thus, the proposed technique provides web service compositions with a minimal cost of violations of the user preferences.

Ge et al. [19] solve the web service composition problem by using OWL ontology. The candidate web services are semantically matched and composed based on their OWL descriptions. Four cases for check similarity of an output and input parameter from the same ontology are defined. First, if the input and output parameters are the same, then the similarity is maximal. Second, if the output parameter of one service is subsumed by

Table 3.1: Comparison of web service composition approaches.

Ref.	OM	CM	QM	T	DDAM	EH	SSM	TS
2	BPMN	WSDL	Yes	No	Partial	Yes	Yes	Yes
3	BPEL	WSDL	Partial	No	No	No	Yes	Yes
4	PG	WSDL	No	No	No	No	No	Yes
5	PMC	WSDL	No	No	No	Yes	Yes	Yes
6	AIP	OWL–S,OWL	No	No	No	No	No	Yes
7	OWL–S	SWRL	No	No	Yes	No	No	Yes
8	WS–BPEL	WSDL,SLA	No	Yes	Partial	No	No	No
9	GM	WSDL,UDDI	Yes	Partial	No	Partial	No	Yes
10	GM	WSDL,WSLA,OWL	Partial	No	No	No	No	Yes
11	S	OWL-S,SAWSDL	Partial	No	Yes	No	No	Yes
12	HTN	OWL-S	Yes	No	No	No	Yes	Yes
13	PMC	WSDL	No	No	Yes	Yes	Yes	Yes
14	AIP	WSDL	Yes	No	No	No	Yes	Yes
15	S	WSDL,SLA	Partial	No	No	Yes	Yes	Yes
16	SM	WSDL,UDDI	No	No	No	No	No	Yes
17	XML Nets	OWL–QoS	Yes	No	Yes	No	Yes	No
18	HTN	OWL–S	No	No	No	No	No	Yes
19	OWL	WSDL	No	No	Yes	No	No	Yes

the input parameter of another one, then the similarity value depends on their distance in the ontology. Third, the output parameter of one service subsumes the input parameter of another one and the properties of the parameters are partially satisfied. Fourth, if two parameters do not have subsumption relation or they are come from different ontology, then the similarity value can be obtained by Tversky's feature-based similarity model. The final composition is presented as direct graph and is converted to executable business process described in BPEL4WS.

3. Analysis of Web Service Composition Approaches. This section presents a comparative analysis of web service composition approaches. It is based on a composition model proposed by Alonso et al. [1]. According to this model, web service composition is characterized with six different dimensions: Component Model (CM), Orchestration Model (OM), Data and Data Access Model (DDAM), Service Selection Model (SSM), Transactions (T), and Exception Handling (EH). The CM considers the type of components that participate in a composition as well as assumptions about them. The OM deals with the way in which web services are composed into more complex services. DDAM is responsible for data exchange among components. The SSM defines whether a particular web service is selected as a component dynamically or statically. Transactions define transactional semantics associated to the composition. Finally, the EH specifies the mechanisms for handling of exceptional situations that is possible to occur during composition invocation.
Evaluation of the composition quality is an important aspect of the composition process. When several candidate web services have the same functionality, their QoS properties are examined in order to select the best one. Thus we identify QoS support as an additional dimension of the composition model, called Quality Model (QM).
Table 3.1 summarizes the comparison of the web service composition approaches. The first column refers to composition approaches. The last column, named Tool Support (TS), shows which of the approaches are implemented as software tools or platforms. They are marked with filled circle. The rest of the columns correspond to the dimensions of composition model presented above.

OM column in Table 3.1 shows the abstractions and languages used for modeling of control flow of the compositions in terms of sequence of operation execution. Here, variety of paradigms exists. Most common are techniques based on AI Planning such as HTNs, PGs, and Semantic Markup for Web Services (OWL-S) and Web Ontology Language (OWL) [4], [6], [7], [12], [14], [18], [19]. PMC is another planning technique that is also applicable to OM [5], [13]. The languages for description of business processes like BPEL and BPMN are also proposed for web service orchestration [2], [3], [8]. Statecharts are abstractions that extend state machines with possibility to define activities while moving between states [11], [15]. HPNs have the ability to model concurrency of the systems, analyze concurrent behavior, and express the dynamically changing software. Here, they are presented by XML nets, which provide advantages in the description of process objects and inter-organizational exchange of XML structured data [17]. Graph Models (GM) are another way for design of OMs,

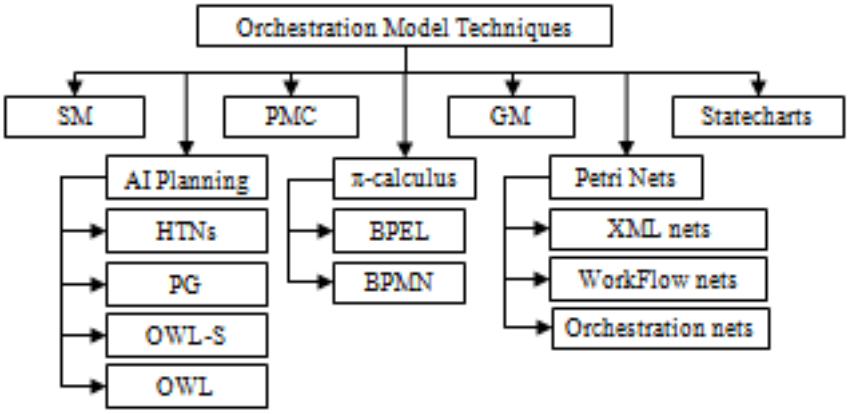

Fig. 3.1: Classification scheme of web service

where web services are presented with graph nodes and the sequence of their execution is shown by the graph edges. They are applied in [9] and [10]. Signature Matching (SM) is a technique used in [16] in order to compose web services. It is instance of more general problem, called function realization problem [20]. A classification scheme that summarizes web service orchestration techniques is shown on Figure 3.1.

The CM column shows the standards and languages used for description of composition components. Currently, most of the composition approaches (along with the analyzed ones here) assume that components are WSDL services based on standards such as HTTP, SOAP and WS-Transaction. As shown in Table 3.1, some approaches rely on additional standards like SLA and OWL-S in order to enrich the composition model with QoS and semantic data. They take into account the QoS properties of candidate web services in order to produce composition that aggregates web services with the maximum quality. This aspect of the composition process is presented in column QM of the table. Here, Ÿesm̈eans that the proposed QM is capable to covers all QoS properties. The P̈artialv̈alue shows that given approach supports QM with limited number of QoS properties. We call such approaches QM limited. For example, the QM proposed in [3] concerns only Response time.
QMs can be further classified according to the formalism that is applied to the composition when QoS properties are taken into account. For example the QM proposed in [2] is based on Multi Criteria Decision Making. It considers quality in three aspects: QoS that is related to web service execution, Quality of Contents (QoC) referring to the quality of the content with which the web services work, and Quality of Devices (QoD), concerning physical environment. The T column shows whether given approach consider transactional behavior of web service composition. Since the underlying middleware is often responsible for providing transactional capabilities, approaches presented in Table 3.1 do not consider this dimension of the composition model. For example, the compensation logic for roll back of web service activities may be handled by engine that implements WS-Transaction specifications.
The DDAM column shows which approaches compose web services in terms of data types and data transfer. According to [1] the data of given web service composition can be divided into application-specific data and control flow data. Data exchanged by web service messages is called application-specific data. Data that is used for evaluation of branching conditions is called control flow data. Data transfer method can follow blackboard approach or explicit data flow approach. The first one relies on a blackboard, which is a collection of variables defining the output and input of each web service activity. The explicit data flow allows developers to specify that the input data of an activity should be taken from the output data of previously executed activities. For instance, the DDAM presented in [17] uses data type definitions of XML Schemas. The proposed approach solves a problem with message passing, where an output message of a web service may be incompatible with the required input message of another WS. The approach uses XML nets to create mediating transition between two web service invocation transitions. In [19] the similarity of an output and input parameter of web service activities are checked according to preliminary defined rules using OWL.
The EH column indicates approaches that take into account unexpected behavior of web service compositions. A possible solution is to use conditional branch that checks the result from invocation of an activity for failures

or timeout according to which an activity will be terminated if the timeout expires. Another solution is to associate exception handling logic to an activity or group of activities. Rule-based languages are also applicable to the exception handling problem. For example, the approach presented in [13] uses Computation Tree Logic (CTL) to model web service composition failures. A global planning approach is used in [15] to reconfigure composition execution in case of one or several failures during web service invocations.
The SSM column show which approaches support dynamic binding of web services in a composition. Dynamic binding have advantage on dealing with web services that change their URIs.

4. Conclusion. Composition of web services is appealing tactic to enable even more powerful business processes and enrich software applications. A lot of web service composition approaches exist in the literature differing in the applied techniques and strategies for web service orchestration, data modeling, transaction support, QoS awareness and exception handling. In this paper we analyze a number of such approaches and distinguish them according to their essential characteristics.
Directions for future work include further developing the proposed classification into ontological framework for web service composition process and also a composition meta-model. This will help software architects to choose a particular approach best suited for a given application domain.

Acknowledgments. The work presented in this paper was partially supported by the FP7 Specific Programme 'Capacity' - Research potential under Grant No. 205030 and project BG 051PO001-3.3.04/13 of the HR Development OP of the European Social Fund 2007-2013.

REFERENCES

[1] G. Alonso, F. Casati, H. Kuno and V. Machiraju, *Web services, Concepts Architectures and Applications*, Springer-Verlag Berlin Heidelberg (2004).
[2] H. Chang and K. Lee, *Quality-Driven Web Service Composition Methodology for Ubiquitous Services*, Journal of Information Science and Engineering 26(6) (2010), pp. 1957–1971.
[3] M. Qiao, F. Khendek, A. Serhani, R. Dsouli and G. Roch, *An Architecture for Automatic QoS Adaptation for Composite Web Services*, Journal of Web Services Practices, Vol. 4, No.1 (2009), pp. 18–27.
[4] X. Zheng and Y. Yan, *An Efficient Syntactic Web Service Composition Algorithm Based on the Planning Graph Model*, 8th IEEE Int. Conf. on Web Services (2008), pp. 691–699.
[5] H. Q. Yu and S. Reiff-Marganiec, *Semantic Web Services Composition via Planning as Model Checking*, Technical report CS-06-003, University of Leicester (2006).
[6] M. Klusch and A. Gerber, *Evaluation of service composition planning with QWLS-XPlan*, Int. Conf. on Web Intelligence and Intelligent Agent Technology (2006), pp. 117–120.
[7] P. Bartalos and M. Bielikova, *Fast and Scalable Semantic Web Service Composition Approach Considering Complex Pre/Postconditions*, Int. Workshop on Web Service Composition and Adaptation (2009), pp. 414–421.
[8] V. Cardellini, E. Casalicchio, V. Grassi and F. Lo Presti, *Flow-Based Service Selection for Web Service Composition Supporting Multiple QoS Classes*, IEEE Int. Conf. on Web Services (2007), pp. 743–750.
[9] S. Chakhar, S. Youcef, V. Mousseau, L. Mokdad and S. Haddad, *Multicriteria Evaluation-Based Conceptual Framework for Composite Web Service Selection*, http://www.lipn.univ-paris13.fr/ youcef/BookQoS.
[10] M. Aiello, E. Khoury, A. Lazovik, P. Ratelband, *Optimal QoS-Aware Web Service Composition*, IEEE Conf. on Commerce and Enterprise Computing (2009), pp. 491–494.
[11] F. Lecue, *Optimizing QoS-Aware Semantic Web Service Composition*, 8th Int. Semantic Web Conference, LNCS Vol. 4273, (1989), pp 375–391.
[12] S. Sohrabi, S. A. McIlraith, *Optimizing Web Service Composition While Enforcing Regulations*, 8th Int. Semantic Web Conference, USA (2009), pp. 601–617.
[13] M. Pistore, F. Barbon, P. Bertoli, D. Shaparau and P. Traverso, *Planning and Monitoring Web Service Composition*, Lecture Notes in Computer Science Vol. 3192 (2004), pp. 106–115.
[14] H. K. Farhan, M. Y. Javed, B. Saba and H. K. Sikandar, *QoS Based Dynamic Web Services Composition and Execution*, Int. Journal of Computer Science and Information Security, Vol. 7, Issue 2, USA (2010), pp. 147–152.
[15] L. Zeng, B. Benatallah, A.Ngu, M. Dumas, J. Kalagnanam and H. Chang, *QoS-aware Middleware for Web Services Composition* IEEE Transactions on Software Engineering, Vol. 30 Issue5 (2004), pp. 311–327.
[16] H. Mili, G. Tremblay, A. Caillot and R. B. Tamrout, *Web service composition as a function cover problem*, MCeTech Montreal Conference on eTechnologies, Canada (2005), pp. 73–85.
[17] H. Che, Y. Li, A. Oberweis and W. Stucky, *Web Service Composition Based on XML Nets*, Hawaii Int. Conf. on Systems (2009), pp. 1–10.
[18] N. Lin, U. Kuter and E. Sirin, *Web service composition with user preferences*, 5th European semantic web conference on semantic web: research and applications, LNCS Vol. 5021, Spain (2008), pp. 629–643.
[19] J. Ge, Y.Qiu and S. Yin, *Web Services Composition Method Based on OWL*, Int. Conf. on Computer Science and Software Engineering, China (2008), pp. 74–77.
[20] H. Mili, O. Marcotte and A. Kabbaj, *Intelligent Component Retrieval for Software Reuse*, Third Maghrebian Conference on Artificial Intelligence and Software Engineering, Rabat, Morocco (1994), pp. 101–114.

[21] S. Dustdar and W. Schreiner, *A survey on web services composition*,International Journal of Web and Grid Services, Vol. 1, No. 1 (2005), pp. 1–30.

[22] J. Rao and X. Su, *A Survey of Automated Web Service Composition Methods*, 1st Int. Workshop on Semantic Web Services and Web Process Composition, USA (2004), pp. 43–54.

[23] M. Sathya, M. Swarnamugi, P. Dhavachelvan and G. Sureshkumar, *Evaluation of QoS Based Web- Service Selection Techniques for Service Composition*, Int. Journal of Software Engineering, Vol. 1 Issue 5 (2011), pp. 73–90.

Edited by: Dana Petcu and Jose Luis Vazquez-Poletti
Received: November 1, 2011
Accepted: November 30, 2011