A Review of Dynamic Web Service Composition Techniques

Demian Antony D'Mello[1], V.S. Ananthanarayana[2], and Supriya Salian[1]

[1] Department of Computer Science and Engineering, St. Joseph Engineering College, Mangalore - 575 028, India
demian.antony@gmail.com, supriyasalian@yahoo.com
[2] Department of Information Technology, National Institute of technology Karnataka, Srinivas Nagar, Mangalore - 575 025, India
anvs@nitk.ac.in

Abstract. The requester's service request sometimes includes multiple related functionalities to be satisfied by the Web service. In many cases the Web service has a limited functionality which is not sufficient to meet the requester's complex functional needs. The discovery mechanism for such complex service request involving multiple tasks (operations) may fail due to unavailability of suitable Web services advertised in the registry. In such a scenario, a need arises to compose the available atomic or composite Web services to satisfy the requester's complex request. Dynamic Web service composition generates and executes the composition plan based on the requester's runtime functional and nonfunctional requirements. This paper provides the review of Web service composition architectures and techniques used to generate new (value added) services.

Keywords: Web Services, Service Registry, Dynamic Composition, Architecture.

1 Introduction

A Web service is defined as an interface which implements the business logic through a set of operations that are accessible through standard Internet protocols. The conceptual Web services architecture [1] is defined based upon the interactions between *three* roles: *service provider*, *service registry* and *service requester*. The requester search for suitable Web services in the registry which satisfy his functional and nonfunctional requirements. The requester's service request sometimes includes multiple related functionalities to be satisfied by the Web service. In many cases the Web service has a limited functionality which is not sufficient to meet the requester's complex functional needs. The UDDI based Web service architecture does not realize complex Web service combinations, hence it provides limited support for service composition. There is a need to identify and compose the available Web services if the complex service request can not be satisfied by a single Web service. To achieve complex business goals in

N. Meghanathan et al. (Eds.): CCSIT 2011, Part III, CCIS 133, pp. 85–97, 2011.
© Springer-Verlag Berlin Heidelberg 2011

real world applications, the execution of multiple Web services should be orchestrated through service composition. The Web service composition can be defined as the creation of new Web service by combining the available services (service operations) that realizes the complex service request. The service composition strategies are broadly classified as *Static* and *Dynamic* composition based on the time when the Web services are composed [2]. Static composition takes place during design time when the architecture and the design of the system is planned. Dynamic composition takes place at run time when the requested service is not provided by the single provider. The effective dynamic Web service composition is a major challenge towards the success of Web services. The following *seven* different issues have a large impact on dynamic Web service composition. They are: Describing Web services and complex service request for effective composition, Generation of composition plan for the complex service request, Modeling (specification) of composition plan (orchestration models), Selection of Web services for the composition, Coordination and Conversation modeling, Execution of composition and Transaction management.

In this paper, the authors provides detailed review of dynamic composition architectures and techniques. Section 2 describes Web service composition architectures and strategies. In section 3 describes various dynamic Web Service Composition plan generation Methods. Section 4 compares the different methods used for dynamic Composition Based on Web Service Signatures. Section 5 draws conclusions and future challenges in dynamic Web service composition.

2 Web Service Composition Strategies and Architectures

Service composition facilitates application reuse where new Web services are created using available Web services which are heterogeneous in nature and spread across organizations. The service composition strategies are classified as *Static Composition*, *Semi-dynamic Composition* and *Dynamic Composition* based on the time of composition plan creation and service binding times [3]. Static composition is also called as design time composition where the application designer manually discovers, binds and assembles the Web services during composite Web services application development. Dynamic Web service composition is a complex and very challenging task in Web services as the composition plan is generated at runtime based on the requester's complex service request [4]. In literature, various architectures have been proposed to facilitate dynamic Web service composition. They are: *Peer-to-Peer (P2P) architectures*, *Agent architectures* and *Hybrid (Multi-role) architectures.*

2.1 P2P Architecture

The Web service composition methods defined on P2P overlay networks [5] works as follows: when a peer wants to share a service, it registers the service to the registration system. Towards service composition, it looks up and gets its successor service from the registration system and then the successor service in turn finds its successor service. This will be repeated until the composition is

accomplished. The *Self-Serv* project uses P2P based orchestration model to support cross organizational compositions [6]. P2P based composition architecture is suitable for facilitating composition of B2B services rather B2C services which are too dynamic in nature.

2.2 Agent Architecture

A software agent can be defined as a computational entity, which acts on behalf of others, is autonomous, proactive and reactive, and exhibits capabilities to learn and cooperate. Agent architectures have been proposed in literature where agents perform specific roles of composition and execution. The various roles include: message processing and composition, service binding and triggering the specification of composite services and monitoring the deployment of specifications [7]. A mobile agent (MA) based multi-platform architecture for Web service composition is proposed [8] where MA is responsible for the execution of composition. The agent based Web service architectures are not suitable for compositions due to performance issues like reliability and security.

2.3 Hybrid (Multi-role) Architecture

The extension of conceptual SOA with new roles and operations towards Web service composition facilitate both static and dynamic composition. In literature, various hybrid architectures have been proposed towards dynamic Web service discovery and composition. A novel business model for dynamic Web service composition involving *two* new roles called *Web Service Composer* and *Web Service Composition Registry* is proposed [9]. Similarly, roles like configuration engine and workflow engine [10], discovery engine, abstract process designer are defined for dynamic composition architectures. The *Eurasia Architecture* which is proposed [11] consists of *three* main components. They are: component description framework, the template composer and data flow based service coordinator. The early initiative towards composition called *e-flow* [12] involves multiple roles including e-flow service broker, e-flow engine and a set of repositories for storing information related to composition. The multi-role architecture is suitable for composition as composition and execution involves many activities which are to be coordinated by the architectural role called *composition manager* or *composition broker*.

2.4 Role of Ontology and Composition Execution

Most of the hybrid architectures contain semantic information component called *Ontology Store* to guide the dynamic composition and execution. In literature, various kinds of ontologies have been designed (Modeled) for composition which include: Domain ontology [10], Parameter ontology [13], Policy ontology [14], Context ontology [15] and Operation mode ontology [16]. In literature, the Web service composition architectures are realized through building Integrated Development Environments (IDE) which facilitate providers and requesters in publishing and integrating Web services according to their needs [17]. Towards the

execution of compositions (composition plan), various techniques have been proposed [18]. They include: *Interleaved composition and execution*, *Monolithic composition and execution*, *Staged composition and execution* and *Template based composition and execution.*

3 Dynamic Web Service Composition Methods

The crux of the dynamic Web service composition is generation of composition plan at runtime according to the needs of the requester. In literature, various strategies have been proposed by numerous researchers towards composition plan generation. Here *eight* categories of composition strategies have been identified based on the nature of information and techniques used to build the composition plan for execution. They are: *Constraint based composition*, *Business rule (Policy) driven composition*, *User Interaction and personalization based composition*, *Planning (AI Planning) based composition*, *Context information based composition*, *Process based composition*, *Model and aspect driven composition* and *Signature (Input and Output) based composition.*

3.1 Constraint and Business Rule (Policy) Driven Composition

Constraint driven Web services composition aims at building a composition plan based on the business constraints which are represented in ontologies. The selection of a specific business constraint from the pool of constraints is dependent on the particular instance of the process [19]. The constraints enforced by the requesters and providers guide the data flow and control flow in Web service composition. The vertical constrains are required to setup the abstract composition and horizontal constraints help to build and manage instance of composite service involving data and control flow [20]. The requester's quality constraints are also used to build concrete service workflow (composition pattern) by binding the activity with services through constraint satisfaction [10]. The authors [21] propose service discovery approach that locates services that conform to independent global constraints. The approach uses a greedy algorithm to identify conforming values and locate composite services. The domain rules are used to derive values that conform to given user constraint and combine services that assign those values to their restricted attributes. From the above discussion, it is observed that the provider's and requester's constraints play a role in generating composition plan involving sequential and parallel execution flow.

The business processes can be dynamically built by composing Web services if they are constructed and governed by business rules (Policies) [22]. The authors [22]propose a rule driven mechanism to govern and guide the process of service composition in terms of *five* broad composition phases spanning abstract definition, scheduling, construction, execution and evolution to support on demand and on the fly business process generation. Thus business rules are used to structure and schedule service composition and to describe service selection and service bindings. Object Constraint Language (OCL) is used to express business rules and to describe the process flow. The composition plan generated must

satisfy the policies (rules) enforced by the providers of the selected Web services for the concrete composition plan [23]. In order to have service compatibility at the level of policies, a policy ontology is used for the selection of policy compatible, candidate Web services for the composition. Thus business rules identified at each phase of service development are used to derive effective service composition process involving inter and intra-organizational processes. Business rule driven composition requires all business rules which are specific to service needs to be documented in a unified way (e.g. ontology) for successful composition.

3.2 Planning Based Composition

Web service composition (WSC) can be seen as a planning problem, where a sequence of services are composed into a business process to reach (satisfy) business goals. In literature, many research efforts tackling Web service composition problem via Artificial Intelligence (AI) planning have been proposed [24]. In such techniques, initial states and the goal states are specified in the requirement by Web service requesters. The composition methods based on AI planning include: *Declarative composition*, *Situation Calculus*, *Rule based composition*, *Theorem proving* and *Graphplan based composition*.

The declarative approach [25], [24] consists of *two* phases: the first phase takes an initial situation and the desired goal as starting point and constructs generic plans to reach the goal. The latter one chooses one generic plan, discovers appropriate services, and builds a workflow out of them. The first phase is realized using PDDL (Planning Domain Definition Language) which provides machine readable semantics and specifies the abstract service behavior. The second phase may be realized by using existing process modeling languages, such as BPEL. In Self-Serv [6], Web services are declaratively composed and then executed in a dynamic peer-to-peer environment. The authors [26] propose a system, which employs goal oriented inferencing from the planning algorithm (TLPlan) to select atomic services that form a composite Web service. To specify service goals, a goal specification language is proposed [29] that allows specification of constraints on functional and nonfunctional properties of Web services. Towards enhancement of planning based WSC, a solution to combine GA with planning is proposed [27] so that, GA helps to navigate the large search space and to build sub-space by querying Web Services.

A logic programming language called *Golog* is built on top of the *situation calculus* [24] for planning based WSC. The Web service composition problem can be addressed through the provision of high level generic procedures and customizing constraints. The general idea of situation calculus is that, software agents could reason about Web services to perform automatic Web service discovery, execution, composition and interoperation. A technique called *Rule based composition* to generate composite services from high level declarative description is described [24]. The method uses composability rules to determine whether two services are composable. The composition technique generates a detailed description of the composite service automatically and presents it to the service requester. The *Theorem proving* approach [24] is based on automated

deduction and program synthesis. In this approach, initially available services and user requirements are described in a first order language related to classical logic, and then constructive proofs are generated with Snark theorem prover. Finally, service composition descriptions are extracted from particular proofs. The *Graphplan* based planning solution involves execution of sequence of steps (graph expansion) on a planning graph to find the goal [28]. The AI planning based composition technique requires the output (goal) of requested service to be specified by the requester in a predefined format for successful compositions.

3.3 User Interaction and Personalization Based Composition

Current approaches to compose function oriented Web services are inappropriate for interactive characteristics of Interactive Web Services (IWS). Towards this, a novel user satisfaction model to evaluate the interactive quality of a composite IWS is proposed [30]. Based on the satisfaction model, an effective satisfaction driven approach has been developed for the service selection in IWS composition, which can meet diverse interactive requirements of users. In order to fulfill the user requirements, researchers have proposed mechanisms for composition, based on requirements of Inputs and Outputs [31]. A multi-tier architecture called *TailorBPEL* is proposed [32] that enables end-users to tailor personalized BPEL based workflow compositions at runtime. The framework provides end-users with capabilities of tailorability through a set of personalization API and tailoring API. The personalization API allows end-users to create a personalized version of a BPEL process which can be tailored later on using the available tailoring API. The modeling of interactive Web services for composition is a challenging task as the interaction pattern varies from service to service.

3.4 Context Information and Process Based Composition

Context provides useful information concerning the environment wherein the composition of Web services occurs [7]. The context information (service context, policy context and process context) is crucial to model, monitor and execute composite processes. The authors [7] propose a context based multi-type policy approach for Web service composition which cater to the necessary information, enables tracking the whole composition process by enabling policies and regulates the interactions between Web services. The authors [15] present composition approach based on context driven Web service modeling. A novel approach named *process context aware matchmaking* is proposed [33] which discovers the suitable services during Web service composition modeling. During matchmaking, the approach utilizes not only semantics of technical process but also the business process of a advertised service, thus further improving the precision of matchmaking.

The finite state machine (FSM) modeling of Web services contribute to the automatic composition of Web services and a few researchers have used FSMs to model service behavior towards composition. The authors [34] model services as FSMs augmented with linear counters to integrate activity processing costs into

the model. The authors further investigate the problem of computing an optimal delegation for a given sequence of service requests. The authors [35] present an automatic e-Service composition which proposes a framework in which the exported behavior of an e-Service is described in terms of its possible executions (execution trees). The framework is also specialized to consider exported behavior (i.e. the execution tree of the e-Service) which is represented by a finite state machine. The modeling of process (behavior) of services using FSM is a complex task for the providers. The context aware Web service composition requires various ontologies to be updated and managed to capture the requester's and provider's context.

3.5 Model and Aspect Driven Composition

The authors [36] [22] introduce the approach of *Model Driven Service Composition*, which is based on dynamic service composition to facilitate the management and development of composite Web services. Unified Modeling Language (UML) is used to provide a high level of abstraction and to enable direct mapping to other standards, such as BPEL4WS. UML based modeling technique can be applied to *Aspect Oriented Programming (AOP)* technologies for Web service compositions. An aspect oriented Web service composition approach is proposed [37] to address modularization and dynamic adaptation problems. The authors design the aspects for *pluggable* behaviors with UML and bind the aspects into the basic model to get the complete composition model. The authors [38] propose the decoupling of core service logic from context related functionality by adopting a Model driven approach based on a modified version of the ContextUML meta model. Core service logic and context handling are treated as separate concerns at the modeling level where AOP encapsulates context dependent behavior. The model driven composition enables the developer to specify composition plan and binding of services for the tasks which happens at runtime based on the requirements.

3.6 Signature (Input and Output) Based Composition Techniques

Operation signature (Input and Output) based composition is the most widely used technique to generate composition plan (sequential or parallel) at runtime based on the requester's demands. The composition plan is generated by matching inputs of service with the outputs of rest services or vice versa. Inputs and outputs are normally specified by URIs and can be extracted directly from the WSDL documents. A brief review of signature based dynamic Web service composition techniques is presented in the next section.

4 Dynamic Composition Based on Web Service Signature

In literature, various techniques have been proposed to exploit Input-Output parameter descriptions and their relationships for the dynamic Web service composition. The service (or operation) signature (or behavior) based composition

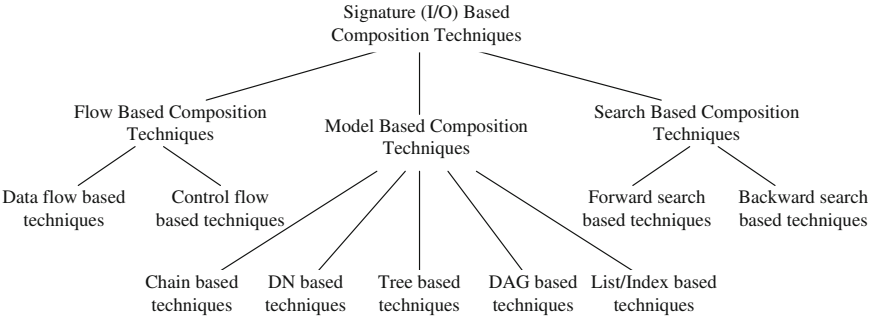

Fig. 1. Taxonomy of QoS aware Web Service Selection Techniques

techniques are classified as *Search based composition*, *Flow based composition* and *Structure (Model) based composition*. Fig. 1 depicts classification of various signature based Web service composition techniques.

4.1 Search Based Composition Techniques

Search based composition is defined based on the nature of generation of composition plan and type of matchmaking involved in the search during plan generation. The search based composition methods are classified as *Forward search based composition* and *Backward search based composition*. In forward search based composition (Goal driven) [28], the composition plan is initialized by the selection of Web services satisfying the requester's input parameters and the plan is expanded by matching inputs of the rest of the services with the outputs of the last identified service. In backward composition method [39], the composition plan is generated by identifying a suitable service which satisfies the requested outputs. The backward search method is more efficient as compared to forward search since backward search does not allow meaningless expansions of the composition plan. The search based composition techniques generate composition plan with redundant execution paths which require further optimization. Moreover, for the matchmaking of inputs with outputs the entire service repository needs to be searched which is a costly activity in terms of execution time.

4.2 Flow Based Composition Techniques

The flow based composition is defined based on the nature of information used to generate composition plan. In literature, *two* types of information items are used to generate the composition plan. They are: data flow (message flow) or data dependency information and control flow (execution sequence or service dependency) information. Thus flow based composition techniques are classified as *Data flow based composition* and *Control flow based composition*. In data flow based composition techniques, the message sequence of individual services [18] or the input-output data dependency among services [40], guide the generation of composition plan at runtime. The control flow (workflow) based composition

techniques exploit the execution order of services to build the plan for Web service composition at runtime [41].

4.3 Structure (Model) Based Composition Techniques

In literature various data models have been proposed to hold the service information (Input or Output) to compose Web services. Based on the type of data structures used for the composition, the model based composition techniques are classified into *five* groups. They are: *Tree based techniques*, *Directed Acyclic Graph (DAG) based techniques*, *Chain (Linked List) based techniques*, *List (index) based techniques* and *Deduced Network (DN) based techniques*.

In tree based approach [42], the service information (especially input-output) is modeled in the form of the tree structure for composition. A *Composition Schema Tree (CST)* is defined in [39] for a service which holds service signature information. A group of such trees of advertised services are used to build the *Complete Composition Schema Tree (CCST)*. For a given service request the CCST involving input nodes, output nodes and Web service nodes is traversed to build composition at runtime. The authors [43] propose the tree structure called *Web Services Composition Tree (WSCT)* which is used to obtain QoS aware composition result for a given service request. The tree based techniques always try to build the tree structures based on the input-output parameters and not on parameter dependency. Moreover parameter ontology needs to be defined to specify service request and service description in a standard form.

The graph (DAG) model [40] for dynamic composition represents the service parameters and their dependencies (constraints). The authors [40] propose a method which constructs a graph model, that represents the functional semantics of Web services as well as the dependency among inputs and outputs. The authors [45] define a dependency graph for composition, where the nodes represent inputs and outputs of Web services and edges represent the associated Web services. The authors [46] present an approach for automatic service composition which discover services based on semantic annotations of service properties, e.g. their inputs, outputs and goals. This approach also uses a graph based search algorithm on composition graph to determine all possible composition candidates and applies certain measures to rank them. Graph based techniques require more search time in the case of large number of advertised Web services involving too many input and output parameters.

In literature the composition techniques are also defined on data structures like chains (linked list) [47], chains with Indexed tables [48], Inverted lists (files) [49] and deduced networks (DN) [50] to represent service information for dynamic Web service composition. The use of lists, chains and deduced networks enhance the composition mechanism in terms of time and number of possible plans for composition. The graph model is more effective only if services are represented with dependency among them instead of representing parameters (input and output) as the nodes. The representation of parameters increase the size of graph thereby the search time of composition. The chains, tables and lists (vectors) are

best data structures to hold input-output parameter information and parameter dependencies.

5 Conclusion

Composition of available Web services based on the requester's functional requirements is a challenging task. In literature, various techniques for dynamic composition of Web services have been proposed. The composition mechanisms proposed in literature build composition plan involving Web services for the various tasks of a complex service request. The Web service is normally a collection of logically related operations and the requester normally requests for a single operation (simple service request) or multiple operations (complex service request). Thus composition must focus on generating composition plan involving abstract operations of available Web services instead of just Web services. The operations of Web service are sometimes dependent on other operations in terms of execution order (flow). Some of the operations of Web service do not implement business logic instead they assist other operations in successful execution. For example, consider a travel scenario where the operation "reserve train ticket" which implements business logic is dependent on one assisting operation called "check train ticket availability". If the requester is interested in operations which implement business logic then all assisting operations which have influence on the requested operations need to be included in the composition plan towards successful composition and execution. Thus, composition plan to be generated at runtime must contain such assisting operations with an order of execution.

References

1. Kreger, H.: Web Services Conceptual Architecture (WSCA 1.0) (2001), http://www.ibm.com/software/solutions/webservices/pdf/wsca.pdf (April 13, 2007)
2. Dustdar, S., Schreiner, W.: A survey on web services composition. International Journal of Web and Grid Services 1(1), 1–30 (2005)
3. Fluegge, M., et al.: Challenges and Techniques on the Road to Dynamically Compose Web Services. In: Proceedings of the ICWE 2006, pp. 40–47. IEEE, Los Alamitos (2006)
4. Sivasubramanian, S.P., Ilavarasan, E., Vadivelou, G.: Dynamic Web Service Composition: Challenges and Techniques. In: Proceedings of the International Conference on Intelligent Agent & Multi-Agent Systems (IAMA 2009). IEEE, Los Alamitos (2009)
5. Lei, W., Jing, S., Xiao-bo, H.: Research on the Clustering and Composition of P2P-based Web Services. In: Proceedings of the 2nd International Conference on Biomedical Engineering and Informatics (BMEI 2009). IEEE, Los Alamitos (2009)
6. Benatallah, B., Dumas, M.: The Self-Serv Environment for Web Services Composition. In: IEEE Internet Computing. LNCS, vol. 4317, pp. 389–402. Springer, Heidelberg (2003)

7. Maamar, Z., Faoui, S.K.M., Yahyaoui, H.: Toward an Agent-Based and Context-Oriented Approach for Web Services Composition. IEEE Transactions On Knowledge And Data Engineering 17(5), 686–697 (2005)
8. Ketel, M.: Mobile Agents Based Infrastructure for Web Services Composition. In: Proceedings of the 2008 IEEE SoutheastCon, part 1. IEEE, Los Alamitos (2008)
9. Karunamurthy, R., Khendek, F., Glitho, R.H.: A Business Model for Dynamic Composition of Telecommunication Web Services. IEEE Telecommunications Magazine, 36–43 (July 2007)
10. Zhao, H., Tong, H.: A Dynamic Service Composition Model Based on Constraints. In: Proceedings of the Sixth International Conference on Grid and Cooperative Computing (GCC 2007). IEEE, Los Alamitos (2007)
11. Ko, J.M., Kim, C.O., Kwon, I.: Quality-of-service oriented web service composition algorithm and planning architecture. The Journal of Systems and Software 81, 2079–2090 (2008)
12. Casati, F., Ilnicki, S., Jin, L., Krishnamoorthy, V., Shan, M.: Adaptive and Dynamic Service Composition in eFlow. White paper, HP Laboratories Palo Alto, HPL-2000-39, Hewlett-Packard Company (March 2000)
13. Liu, J., Fan, C., Gu, N.: Web Services Automatic Composition with Minimal Execution Price. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2005). IEEE, Los Alamitos (2005)
14. Chung, M., Namgoong, H., Kim, K., Jung, S., Cho, H.: Improved Matching Algorithm for Services described by OWL-S. In: Proceedings of the ICACT 2005, pp. 1510–1513 (2005) ISBN 89-5519-129-4
15. Narendra, N.C., Orriens, B.: Modeling Web Service Composition and Execution via a Requirements-Driven Approach. In: Proceedings of the SAC 2007. ACM, New York (2007)
16. Lee, S., Lee, J.: Dynamic Service Composition Model for Ubiquitous Environments. In: Shi, Z.-Z., Sadananda, R. (eds.) PRIMA 2006. LNCS (LNAI), vol. 4088, pp. 742–747. Springer, Heidelberg (2006)
17. Ma, C., He, Y., Xiong, N., Yang, L.T.: VFT: An Ontology-based Tool for Visualization and Formalization of Web Service Composition. In: Proceedings of the 2009 International Conference on Computational Science and Engineering. IEEE, Los Alamitos (2009)
18. Agarwal, V., Chafle, G., Mittal, S., Srivastava, B.: Understanding Approaches for Web Service Composition and Execution. In: Proceedings of the COMPUTE 2008. ACM, New York (2008)
19. Aggarwal, R., Verma, K., Miller, J., Milnor, W.: Constraint Driven Web Service Composition in METEOR-S. In: Proceedings of the 2004 IEEE International Conference on Services Computing (SCC 2004). IEEE, Los Alamitos (2004)
20. Monfroy, E., Perrin, O., Ringeissen, C.: Modelling Web Services Composition with Constraints. In: Revista Avances en Sistemas e-Informatica, Edicion Especial, Medellin, Mayo de, vol. 5(1), pp. 173–179 (2008) ISSN 1657-7663
21. Gooneratne, N., Tari, Z.: Matching Independent Global Constraints for Composite Web Services. In: Proceedings of the WWW 2008, Beijing, China, April 21-25, pp. 765–774 (2008)
22. Orriëns, B., Yang, J., Papazoglou, M.P.: A Framework for Business Rule Driven Service Composition. In: Benatallah, B., Shan, M.-C. (eds.) TES 2003. LNCS, vol. 2819, pp. 14–27. Springer, Heidelberg (2003)

23. Chun, S.A., Atluri, V., Adam, N.R.: Policy-based Web Service Composition. In: Proceedings of the 14th international Workshop on Research Issues on Data Engineering: Web services for E-commerce and E-Government Applications (RIDE 2004). IEEE, Los Alamitos (2004)
24. Rao, J., Su, X.: A survey of automated web service composition methods. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 43–54. Springer, Heidelberg (2005)
25. Zahoor, E., Perrin, O., Godart, C.: Rule-based semi automatic Web services composition. In: Proceedings of the 2009 Congress on Services - I. IEEE, Los Alamitos (2009)
26. Vukovič, M., Kotsovinos, E., Robinson, P.: An architecture for rapid, on-demand service composition. Journal of Service Oriented Computing and Applications - SOCA 1, 197–212 (2007)
27. Yan, Y., Liang, Y.: Using Genetic Algorithms to Navigate Partial Enumerable Problem Space for Web Services Composition. In: Proceedings of the Third International Conference on Natural Computation (ICNC 2007). IEEE, Los Alamitos (2007)
28. Oh, S., Lee, D., Kumara, S.R.T.: A Comparative Illustration of AI Planning-based Web Service Composition. ACM SIGecom Exchanges 5(5), 1–10 (2004)
29. Agarwal, S., Handschuh, S., Staab, S.: Annotation, composition and invocation of semantic web services. Journal of Web Semantics: Science, Services and Agents on the World Wide Web 2, 31–48 (2004)
30. Wan, S., Wei, J., Song, J., Zhong, H.: A Satisfaction Driven Approach for the Composition of Interactive Web Services. In: Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007). IEEE, Los Alamitos (2007)
31. Xiaoming, P., Qiqing, F., Yahui, H., Bingjian, Z.: A User Requirements Oriented Dynamic Web Service Composition Framework. In: Proceedings of the 2009 International Forum on Information Technology and Applications. IEEE, Los Alamitos (2009)
32. El-Gayyar, M.M., Alda, S.J., Cremers, A.B.: Towards a User-Oriented Environment for Web Services Composition. In: Proceedings of the WEUSE IV, pp. 81–85. ACM, New York (2008)
33. Han, W., Shi, X., Chen, R.: Process-context aware matchmaking for web service composition. Journal of Network and Computer Applications 31, 559–576 (2008)
34. Geredea, C.E., Ibarra, O.H., Ravikumar, B., Sua, J.: Minimum-cost delegation in service composition. Theoretical Computer Science 409, 417–431 (2008)
35. Berardi, D., Calvanese, D., Giacomo, G.: Automatic Composition of e-Services. Technical Report (January 10, 2003), http://www.dis.uniroma1.it/mecella/publications/eService/BCDLM-techRport-22-2003.pdf
36. Zhao, C., Duan, Z., Zhang, M.: A Model-Driven Approach for Dynamic Web Service Composition. In: Proceedings of the World Congress on Software Engineering, IEEE, Los Alamitos (2009)
37. Xu, Y., Youwei, X.: Towards Aspect Oriented Web Service Composition with UML. In: Proceedings of the 6th IEEE/ACIS International Conference on Computer and Information Science (ICIS 2007). IEEE, Los Alamitos (2007)
38. Prezerakos, G.N., Tselikas, N.D., Cortese, G.: Model-driven Composition of Context-aware Web Services Using ContextUML and Aspects. In: Proceedings of the 2007 IEEE International Conference on Web Services (ICWS 2007). IEEE, Los Alamitos (2007)

39. Tang, H., Zhong, F., Yang, C.: A Tree-based Method of Web Service Composition. In: Proceedings of the 2008 IEEE International Conference on Web Services, pp. 768–770. IEEE, Los Alamitos (2008)
40. Shin, D., Lee, K.: An Automated Composition of Information Web Services based on Functional Semantics. In: Proceedings of the 2007 IEEE Congress on Services (SERVICES 2007). IEEE, Los Alamitos (2007)
41. Kona, S., Bansal, A., Blake, M.B., Gupta, G.: Towards a General Framework for Web Service Composition. In: Proceedings of the 2008 IEEE International Conference on Services Computing. IEEE, Los Alamitos (2008)
42. Chan, P.P.W., Lyu, M.R.: Dynamic Web Service Composition: A New Approach in Building Reliable Web Service. In: Proceedings of the 22nd International Conference on Advanced Information Networking and Applications. IEEE, Los Alamitos (2008)
43. Chen, Z., Ma, J., Song, L., Lian, L.: An Efficient Approach to Web Services Discovery and Composition when Large Scale Services are Available. In: Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC 2006). IEEE, Los Alamitos (2006)
44. Shen, Z., Su, J.: On Completeness of Web Service Compositions. In: Proceedings of the 2007 IEEE International Conference on Web Services (ICWS 2007). IEEE, Los Alamitos (2007)
45. Hashemian, S.V., Mavaddat, M.: A Graph-Based Approach to Web Services Composition. In: Proceedings of the 2005 Symposium on Applications and the Internet (SAINT 2005). IEEE, Los Alamitos (2005)
46. Shiaa, M.M., Fladmark, J.O., Thiell, B.: An incremental graph-based approach to Automatic Service Composition. In: Proceedings of the 2008 IEEE International Conference on Services Computing. IEEE, Los Alamitos (2008)
47. Xu, B., Li, T., Gu, Z., Wu, G.: SWSDS: Quick Web Service Discovery and Composition in SEWSIP. In: Proceedings of the 8th IEEE International Conference on E-Commerce Technology and the 3rd IEEE International Conference on Enterprise Computing, E-Commerce, and E-Services (CEC/EEE 2006). IEEE, Los Alamitos (2006)
48. Li, L., Jun, M., ZhuMin, C., Ling, S.: An Efficient Algorithm for Web Services Composition with a Chain Data Structure. In: Proceedings of the 2006 IEEE International Conference on Services Computing (APSCC 2006). IEEE, Los Alamitos (2006)
49. Ren, K., Chen, J., Xiao, N., Zhang, W., Song, J.: A QSQL-based Collaboration Framework to Support Automatic Service Composition and Workflow Execution. In: Proceedings of the 3rd International Conference on Grid and Pervasive Computing - Workshops. IEEE, Los Alamitos (2008)
50. Liu, J., Fan, C., Gu, N.: Web Services Automatic Composition with Minimal Execution Price. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2005). IEEE, Los Alamitos (2005)