WEB SERVICE COMPOSITION IN DYNAMIC ENVIRONMENT: A COMPARATIVE STUDY

Aram AlSedrani[1] and Ameur Touir[2]

[1]Department of Computer Science, King Saud University, Riyadh, KSA
Aram_sedrani@ccis.imamu.edu.sa
[2]Department of Computer Science, King Saud University, Riyadh, KSA
touir@ksu.edu.sa

ABSTRACT

Web service composition development is a complex and dynamic process. It is one of the challenges in distributed dynamic environments. Although, SOA (Service Oriented Architecture) facilitates service composition process through standard protocols in searching and binding with web services. Yet composition in SOA paradigm faces many challenges. One of the main challenges is the environment in which composition services are developed. Nowadays the environment becomes more dynamic due to the increase in the number of web services that are frequently changing. Therefore, the need for self-adapted composition methods that acts according to environment changes is advocated. In this paper, we will study the existed researches that address the web service composition in a dynamic environment to state the art in this area and assist future research.

KEYWORDS

Service composition, web services, dynamic environment.

1. INTRODUCTION

The fast growth in the utilization of SOA led to the growth of the number of web services published on the net. Web services offer huge potentials through a large number of simple services developed by several service providers in various servers [1]. However, today's world has become not only more complex but also more dynamic. The single service offers simple and basic functionality that became inadequate to satisfy the needs for future requirement. Therefore combining multiple services to provide complex composite service is of a high demand nowadays.

Web service composition under dynamic environment gained researchers attention recently since around 2004 compared to the emergence of SOA and web services since the late 1990s [2]. This recent interest in this issue arises because of the significant number of available web services and the diversity of web service providers. Due to those reasons, the environment of web services becomes more dynamic with a great probability of web service added, deleted or even moved.

Therefore, service composition that depends on dynamic entities must adapt to any changes occurred during its phases.

In this paper, we aim at investigating the existed researches that deal with the problem of web service composition in a dynamic environment in order to identify the research gap of this field and improve future research. The remainder of the paper is organized as follow: in section two, an introduction of web service composition is provided. Section three will highlight the main challenges that web service composition faces in a dynamic environment. We conduct our study in section four, and the dissection of study results is provided in section five. In the last section, we present the conclusions of our study.

2. WEB SERVICE COMPOSITION

The services composition process consists of several phases. The first is the *composition planning* where specifying service request and decompose it into an abstract set of tasks. During the next phase, *service discovery,* a search for services that match the functionality and non-functionality requirements for each task in the composition is performed. Next, from the multiple services discovered in the previous phase, *service selection* is about selecting the most appropriate service for each task in the composition to satisfy user requirement. The last phase is the *service execution* where the individual task in the composition is invoked and executed to come up with the final service.

3. SERVICE COMPOSITION CHALLENGES IN DYNAMIC ENVIRONMENT

Due to the success of SOA, the development of web services gains much attention from the software developer. This success leads to the growth in the number of web services available over the internet and thus increases the dynamic nature of web services environment. Therefore, service composition that depends on the dynamic environment of web services demands dynamic approaches to ensure the reliability of the composition process.

To illustrate the dynamism of composition environment we need to understand the possible changes that could occur in such environment. The dynamism is due to two types of changes; one is caused by the web service, and the other is caused by the environment itself. For example, some of the changes in web services that could affect the composition are as follow: First, one or more of the participant services fail to accomplish its task, or they become unavailable. Second, participant services could not provide its expected QoS or updated their values. Third, new candidate services advertise for better QoS than participant service. On the other hand, some major faults that could affect the execution of web service composition caused by the environment like, network or connection failure during service invocation or unexpected server crash.

Therefore, service composition under dynamic environment needs new requirements. According to [3] we can set the new composition requirement as follow: self-configuring, self-optimizing, self-healing and self-adapting. Self-configuring composition indicates that the composition able to discover and select services automatically. Self-optimizing means that the services within the composition are chosen according to QoS constraint. The self-healing composition is the composition that is capable of detecting constraints violation and react accordingly. The last requirement is the self-adapting composition that is capable of altering its process to adapt to changes occurs with minimum human intervention.

4. THE COMPARATIVE STUDY

Dynamic web service composition had researchers attention in recent years. Many solutions are proposed to handle the dynamic environment changes during composition process. In this study, we will investigate some recent researches in this field and conduct a comparative study about them. Later we will review our observations and recommendations.

4.1. Overview of Major Researches:

1.. (Chang et.al 2004) [4] is a middleware platform for service composition. The main goal of this work is to provide selection algorithms to maximize the user satisfaction through QoS. The platform considers the changing in QoS attribute values during Execution and proposed a re-planning technique for contingencies. The plan, in the case of QoS changes, is partitioned into three regions based on the criteria if the task has executed, currently executing, or will be executed. Then, based on a particular set of constraints the planner will identify the task (T) affected and perform a re-planning starting of the region that contains (T).

2.. (Canfora et. al 2005) [5] is a triggering algorithm to control the workflow of service composition in the case of QoS values changes during execution. The algorithm in this work basically will estimate the global QoS value of the workflow whenever an update occurs. However, the planning in this algorithm is static designed by the user.

3.. (Nariai et. al 2005) [6] the proposal in this research uses the situation calculus (SC)theory, in the form of first-order logic language, to enable instant reaction of user preferences changes during composition and planning. The authors incorporate the SC theory in an intelligent infrastructure to analyze the pattern of exceptions experienced by a particular machine to manage the user requirements. In fact, they discussed five exception patterns such as unavailability exception and un-satisfy user constraints exception. However, the work considers mainly the user preferences as the dynamic change and doesn't take into account the QoS changes during planning and composition. Moreover, the SC theory does not support the constructing of complex composed service.

4.. (Sapena et. al 2008) [7] this research proposes an on-line planning algorithm based on overlapping the planning and execution phases. The algorithm mainly developed for the systems under time constraints. Therefore, a temporary initial plan is constructed first and then whenever there is available time an improvement of this plan is performed. The algorithm follows a state space search method, specifically, a depth-first search to improve the initial plan. However, a conflict repairing mechanism is used in case if some action fails to complete the plan. This mechanism based on repairing the preconditions of affected tasks to override the failed action and find alternative tasks.

5.. (Yi-an et. al 2009) [8] this research proposes the NVDSCM (iNcomplete Visible and Dynamic Service Composition Method) algorithm to compose services in the partial-known and vibrant environment. The main concept of NVDSCM algorithm is to monitor the cost changes between neighbor services, and if changes happen to this cost, an adjustment to the following costs will be performed until the end of the composition path. The algorithm utilizes the D* search for finding composition path.

6.. (Ying et. al 2009) [9] the research utilizes the ECA rules in multimedia conference systems to manages web service composition in the case of updating user requirement. The proper event will trigger when the business process request is changed and allow service rescheduling. However, the proposed composition is semi-automated and focuses on the changes made by the user for updating requirement. Moreover, the work does not consider the changes in QoS of participant services.

7.. (Dai et.al 2009) [10] the authors in this research present a self-healing service composition framework based on performance prediction. The framework contains a QoS monitor that collects the quality values for participating services. At runtime, when a service is predicted to be failed according to its quality values, a reselection algorithm will be triggered to replace the failed service before it invoked. The aim of the proposed framework is to shorten the delay time from service failure recovery by early predicting the failures.

8.. (Wiesner et.al 2009) [11] in this research a recovery mechanisms are proposed for OWL-S semantic services in case of service failure. These mechanisms are based on OWL-S file annotation. By extracting the semantic information of services, the research proposes a set of operations to find alternative services in order to compensate the failed ones.

9.. (Friedrich et.al 2010) [12] The aim of this research is to provide a model-based approach and tools to analyze the reparability of service-based processes. The approach uses the forward reparability by creating repair plans to reach a correct execution state in case of fault or failure processes. In fact, the approach is divided into two steps. First, at design time, the heuristic-based analysis will check whether the plan is applicable for repairable executions of its processes. Second, at runtime, the approach will generate the repair plan based on the information from the previous step and predefined actions.

10.. (Bartalos et.al 2011) [13] the algorithm proposed in this research starts first by pruning the search space of web services by defining them as useable and Unusable. Then, the algorithm searches for the composition path using Fast Forward search. After adding a single web service to the composition plan, the algorithm entered a waiting mode to check if any changes occurred to web services. If so, it will react to the change according to three situations; if service deleted, the algorithm will delete the service, and its connections from the repository; if services added, the algorithm will classify it as useable or Unusable. If QoS values are updated, the algorithm will change accordingly. However, this research focuses on preparing and classifying the web services before composition to fasten the search process.

11.. (Kuzu et.al 2012) [14] the research proposes an algorithm for anytime planning that is based on overlapping between planning and execution. The tasks in anytime planning are executed ones it entered the plan. The execution outcomes of a single task are used to enhance the remaining planning process and provide information about services that could improve planning in non-deterministic situations. This approach is motivated by the fact that a web service repository at planning time could not be the same at execution time, so an overlapping between planning an execution of tasks are proposed. However in the case of contingency situation at some point during planning and effect previous executed task the re-planning will be costly since the approach needs to plan and execute from the start point. Another problem in anytime planning is the risk of planning and executing of a single task one by one without considering a situation where the plan could be unachievable.

12.. (Barakat et.al 2012) [15] the work proposes an algorithm that reacts to changes occurred in the participating web services during the selection phase. The algorithm modeled the problem as a graph, and each node will store the optimal path from started node. Then when a change occurred the algorithm will re-track the stored information in each node in the path and reselect a new service. The proposed algorithm will adapt to a dynamic environment. However, with the increase in the number of nodes, the storage capacity will increase.

13.. (Saboohi et.al 2013) [16] the SRPFR algorithm proposed in this work is an automatic sub-digraph renovation plan for failure recovery. It consists of two steps: an offline and an online process. First, the offline process begins when composite services are stored in a repository; the algorithm calculates all possible sub-digraph of a composite web service digraph. Then, it finds an appropriate replacement sub-digraph based on ranking mechanism. On the other hand, the online process starts when a service failure occurred in the execution time. In this case, the executor exchanges the sub-digraph that contains the failed service with a pre-calculated sub-digraph from the previous step.

14.. (Markou et.al 2014) [17] the MAPPPA algorithm proposed in this research is an alternative plan generation and merging algorithm. It aims basically at generating multiple plans for the same problem and merges these plans in a single decision tree. The algorithm is designed mainly for non-deterministic problems and could be applied to unexpected situations. The algorithm used A* search over the decision tree and guided by probability values for successful execution of specific branches.

15.. (Gupta et.al 2014) [18] in this research the authors propose a mechanism for fault recovery during service composition execution. The mechanism is performed by a broker to generate all possible subsets of the composition sets if some service in the set is failed. Then, it is the responsibility of the broker to rank the subsets according to QoS values and replace the fault service.

16.. (Wang et.al 2016) [19] this research presents a framework for service composition based on multi-agent re-enforcement learning. The composition goal, in this framework, is decomposed among the agents to reduce the computational cost. Then, each agent will conduct a learning process to search for a solution to its goal. Furthermore, the research introduces a sharing algorithm between agents in order to reduce the time of convergence. From the experiences of agents, the framework able to adapt to changes occurred in a dynamic environment. However, the framework does not consider the quality of web services as a dynamic factor in the environment.

4.2. Study Evaluation Factors:

The next table illustrates the analytical study performed on these selected researches. In this study, we will overview the main technique used by the research to adapt to environment changes in order to evaluate the similarity of those approaches. Moreover, the main factor of this study is the composition phase that the research considers in detecting and reacting to dynamic changes. Moreover, with the importance of QoS in determining the quality of composite service, we will analyze the main quality attributes that affect the composition and whether the researches consider them as a primary factor of composition or not. Then the main research outcomes will be reviewed to highlight the benefits of those researches. The last factor in our analysis is whether the research presents an experiment to proof the correctness of its technique or not.

Table 1: Comparative Table

Number	Research	Technique used	Dynamic Considered phase	QoS Considered attributes*	Research outcomes	Correctness proof
1	[4] Chang et.al 2004	Constraints set	Execution	EC, ET, REL, AV, REP	AgFlow platform	NO
2	[5] Canfora et. al 2005	Triggering Algorithm	Execution	ET, AV	Simulation	YES
3	[6] Nariai et. al 2005	Situation calculus	Execution	NON	Infrastructure	NO
4	[7] Sapena et..al 2008	Conflict-repairing	Planning Execution	NON	Algorithm	YES
5	[8] Yi-an et. al 2009	D* Search	Planning	EC	Algorithm	YES
6	[9] Ying et. al 2009	ECA Rules	Execution	NON	Algorithm	NO
7	[10] Dai et.al 2009	Performance prediction	Execution	EC, ET	Framework	YES
8	[11] Wiesner et.al 2009	Set of operations	Execution	NON	Algorithm	NO
9	[12] Friedrich et.al 2010	Model-based	Execution	NON	WS-Diamond Architecture	YES
10	[13] Bartalos et.al 2011	FF Search	Planning	NM	Algorithm	YES
11	[14] Kuzu et.al 2012	Triggering Signals	Execution	NON	Simplanner Framework	NO
12	[15] Barakat et.al 2012	Graph-based	Selection	EC, ET	Algorithm	YES
13	[16] Saboohi et.al 2013	Subdigraph replacement	Execution	EC, ET, REL	Algorithm	YES
14	[17] Markou et.al 2014	Alternative Plan	Planning	NON	Algorithm	YES
15	[18] Gupta et.al 2014	Subset replacement	Execution	NM	Broker	YES
16	[19] Wang et.al 2016	Agent Re-enforcement Learning	Execution	NON	Simulation	YES

*EC: Execution Cost, ET: Execution Time, REL: Reliability, AV: Availability, REP: Reputation, NM: Not-Mentioned.

5. DISCUSSION

We can see from the number of researches on web service under a dynamic environment that this area is a new research direction, and is not mature enough to satisfy composition process requirements. The need for new techniques is highly recommended; especially with the fast grow of the web, which is the main environment where web service composition is operated. By conducting our study we identify several issues in this area.

As we can see from the next chart, which demonstrates the composition phase considered by subjected researches, that most existed service composition methods (75%) handles the dynamism of the environment in the execution phase and treats the environment as static in early phases. In such situations, any change will be detected and handled after their occurrence. This mechanism causes undesirable problems at execution time that leads to decrease the composition performance with re-planning or re-selecting overhead. However, some researches consider planning mechanisms to handle changes in web services. These approaches detect changes at early phases which build a more reliable composition plan. Thus, prevent re-planning overhead in case of changes during later phases. Most of the planning mechanisms proposed in subjected researches are focused on searching for alternative paths to overcome the changes of web services in original plan path.

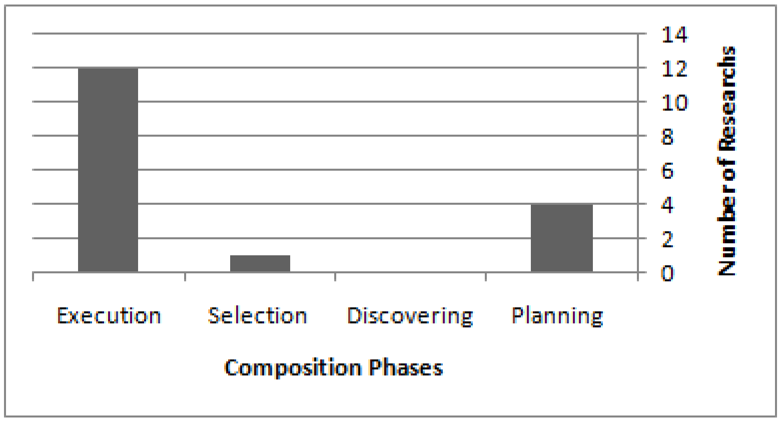

Figure 1: Composition Phases Chart

We can see also from the chart in figure 1 that almost none of the researches except [15] consider proposing a solution to dynamic changes in discovery and selection phases. Those two phases, in fact, deal directly with available web services while planning is considered as ordering tasks to achieve a particular goal. In the case of the discovery phase, the composition process will search for all possible web services to do a specific task in the plan. However, with dynamic web environment, the discovery phase outcomes must be a set of active, and reliable services that afterward the composition process can select from. Therefore, discovery process must depend on algorithms that can react to changes occurred on services, either when deleting inactive discovered services incidence or in the case of some newly exist equivalent service that must be included in the discovered services set.

The same concept is applied to the selection phase. First of all, the set of candidate web services to be select from must be reliable and active. Then, during selection, the changes on web services must be considered in order to improve the selection. As an example, if new service advertises for QoS values that could improve the quality of overall composite service, then the selection algorithm must consider this service and add it to the candidate services set.

In another hand, 50% of the subjected researches, as shown in figure 2, consider the QoS attributes as the primary factor in the dynamic environment. QoS attributes are an effective factor in the dynamic environment. Increasing or decreasing the quality values of individual web service in the composition will affect the overall quality of composite service. We can observe from Table 1, that the most used QoS attributes in these researches are the execution cost (EC) and the execution time (ET). We can refer to the reason for these choices, is that most of the researches aim at enhancing the performance of the composition by reducing the time and cost. Since the composition process is a long process and dealing with multiple providers, the execution time and cost could be high in some cases.

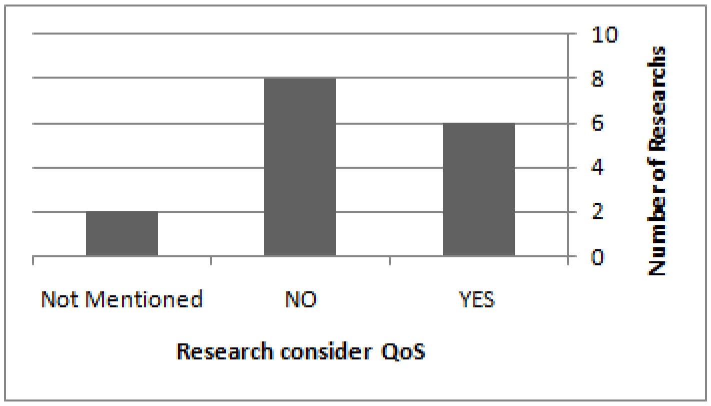

Figure 2: QoS chart

Another observation from the study is that there are two main methods to handle the changes in the composition. First, a pre-processing method in which all possible situations are predefined before composition process. As an example, Saboohi et al. proposed a method to identify all the potential sub-graphs and their alternative to replace a service failure before the execution of the composite web service. However, identifying all the sub-graphs and their alternatives are time-consuming, even if failure does not occur. In another hand, a re-planning and re-selection process are performed in the case of service failure during execution as in [9]. However, this method is inefficient due to the re-processing overhead. To be in the middle, we can recommend for methods that prepare for any changes could be occurred while selecting and planning at the first time, so in the case of service failure, the re-selection and re-planning will be in linear time. As an example, when selecting a service, we can identify some possible alternative services that equal in functional and non-functional properties. Therefore, in the case of service failed during selection or execution prepared solutions are replaced directly, according to its similarity ranking, into the composition.

The last comment on our study findings is that some researches conduct an experiment on their algorithms to proof the enhancement on execution time instead of proofing that the composition service is impeccable, and the dependencies between services after altering the composition are resolved.

6. CONCLUSION

In this paper, a comparative study is conducted that targets the web service composition methods in a dynamic environment. To the best of our knowledge, the selected researches are the only ones that have been found in this area. From our study we can summarize our observations and recommendations in the field of web service composition under dynamic environment as follow:

1. The need for composition methods that is capable of monitoring and reacting to any environment changes during all the composition phases.

2. An enhancement of QoS-aware web service composition methods is recommended due to the importance of quality values changes in the dynamic environment.

3. During each composition phase, preparation mechanisms that define alternative solutions to replace any service failure in linear time are important.

4. Any proposed solution must guarantee the correctness of composition after altering it.

ACKNOWLEDGMENTS

This research has been supported by the Computer Science Dept. of the College of Computer and Information Sciences at King Saud University.

REFERENCES

[1] A. Jula, E. Sundararajan, and Z. Othman, “Cloud computing service composition: A systematic literature review,” Expert Syst. Appl., vol. 41, no. 8, pp. 3809–3824, Jun. 2014.

[2] T. Erl, Service-Oriented Architecture: Concepts, Technology, and Design, 7th Ed. Indiana: Prentice Hall, 2008.

[3] Q. Z. Sheng, X. Qiao, A. V Vasilakos, C. Szabo, S. Bourne, and X. Xu, “Web services composition : A decade ’ s overview,” Inf. Sci., vol. 280, pp. 218–238, 2014.

[4] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, “QoS-aware middleware for Web services composition,” IEEE Trans. Softw. Eng., vol. 30, no. 5, pp. 311–327, May 2004.

[5] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani, “QoS-Aware Replanning of Composite Web Services,” in IEEE International Conference on Web Services (ICWS’05), 2005, pp. 121–129.

[6] K. Nariai, I. Paik, and M. Shinozawa, “Planning and composition of Web services with dynamic constraints using situation calculus,” in Proceedings of The Fifth International Conference on Computer and Information Technology (CIT’05), 2005, pp. 1009–1013.

[7] O. Sapena and E. Onaindía, “Planning in highly dynamic environments: An anytime approach for planning under time constraints,” Appl. Intell., vol. 29, no. 1, pp. 90–109, 2008.

[8] Z. Yi-an and L. Wan-bao, “The Services Composition Algorithm in Incomplete Visible and Dynamic Environment,” in WCSE ’09. Second International Workshop on Computer Science and Engineering, 2009, vol. 1, pp. 169–171.

[9] Z. Ying, C. Junliang, C. Bo, and Z. Yang, “Using ECA rules to manage web service composition for multimedia conference system,” in 2nd IEEE International Conference on Broadband Network & Multimedia Technology, 2009, pp. 545–549.

[10] Y. Dai, L. Yang, and B. Zhang, “QoS-Driven Self-Healing Web Service Composition Based on ã,” J. Comput. Sci. Technol., vol. 24, no. 2, pp. 250–261, 2009.

[11] K. Wiesner, R. Vacul, M. Kollingbaum, and K. Sycara, “Recovery Mechanisms for Semantic Web Services,” in International Conference on Distributed applications and interoperable systems (DAIS), 2009, pp. 100–105.

[12] G. Friedrich, M. Fugini, E. Mussi, B. Pernici, and G. Tagni, “Exception Handling for Repair in Service-Based Processes,” in IEEE Transactions On Software Engineering, 2010, vol. 36, no. 2, pp. 198–216.

[13] P. Bartalos and M. Bieliková, “Effective QoSAwareWeb Service Composition in Dynamic Environment,” in Information Systems Development, 1st ed., W. W. Song, S. Xu, C. Wan, Y. Zhong, W. Wojtkowski, G. Wojtkowski, and H. Linger, Eds. Springer New York, 2011, pp. 101–113.

[14] M. Kuzu and N. K. Cicekli, “Dynamic planning approach to automated web service composition,” Appl. Intell., vol. 36, no. 1, pp. 1–28, 2012.

[15] L. Barakat, S. Miles, and M. Luck, “Reactive Service Selection in Dynamic Service Environments,” in First European Conference in Service-Oriented and Cloud Computing, 2012, pp. 17–31.

[16] H. Saboohi and S. Abdul Kareem, “An automatic subdigraph renovation plan for failure recovery of composite semantic Web services,” Front. Comput. Sci., vol. 7, no. 6, pp. 894–913, 2013.

[17] G. Markou and I. Refanidis, “Anytime Planning for Web Service Composition via Alternative Plan Merging,” in IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI), 2014, pp. 91 – 98.

[18] S. Gupta and P. Bhanodia, “A Flexible and Dynamic Failure Recovery Mechanism for Composite Web Services Using Subset Replacement,” Int. J. Sci. Res., vol. 3, no. 12, pp. 1886–1890, 2014.

[19] H. Wang, X. Wang, X. Zhang, Q. Yu, and X. Hu, “Effective service composition using multi-agent reinforcement learning,” Knowledge-Based Syst., vol. 92, pp. 151–168, 2016.