A Model for Automated Service Composition System in SOA Environment

Paweł Stelmach, Adam Grzech, and Krzysztof Juszczyszyn

Wrocław University of Technology, Institute of Computer Science, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
{Pawel.Stelmach,Adam.Grzech,Krzysztof.Juszczyszyn}@pwr.wroc.pl

Abstract. In this paper a holistic approach to automated service composition in Service Oriented Architecture is presented. The model described allows to specify both functional and non-functional requirements for the complex service. In order to do so a decomposition of the complex service composition process into three stages is proposed, in which a structure, a scenario and an execution plan of a complex service is built. It is believed that service composition tool based on the proposed model will be able to create complex services satisfying efficiently both functional and nonfunctional requirements.

Keywords: service oriented architecture, service composition, SOA.

1 Introduction

The current trend in business is increasingly leading to the use of external services. The number of such services is growing and will grow even faster in the future. Soon delegated services will not be limited to simple tasks performed by individual providers, but they will be complex services performing complex business processes.

Business process designers are rarely experts in web-services technology. More often they can specify the business process in the form of functional requirements that are abstract to the underlying software infrastructure. As a consequence a service composition system should focus on formalization and interpretation of the functional and non-functional requirements in order to find services that fulfill them at a given time and are not hardcoded into the business logic.

The composition of Web services makes that a reality by building complex workflows and applications on the top of the SOA model [5, 8, 3, 13]. However, literature shows a wide range of composition approaches [1, 2]. In [4] it is mentioned that, rather than starting with a complete business process definition, the composition system could start with a basic set of requirements and in the first step build the whole process, whereas many approaches (i.e. [6]) require a well-defined business process to compose a complex service.

Current work often raises the topic of semantic analysis of user requirements, service discovery (meeting the functional requirements) and the selection of specific services against non-functional requirements (i.e. execution time, cost, security). Review of the literature indicates, however, that these methods have not yet been

Luis M. Camarinha-Matos (Ed.): DoCEIS 2011, IFIP AICT 349, pp. 75–82, 2011.

successfully combined to jointly and comprehensively solve the problem of composition of complex services that satisfy both functional and non-functional requirements. In many cases only one aspect is considered. For example [9] focuses on services selection based only on one functional requirement at a time. [6, 14, 15, 16, 17] show that non-functional requirements are considered to be of a key importance, however many approaches ignore the aspect of building a proper structure of a complex service which is key to optimization of i.e. execution time. Many AI Planning-based approaches ([7]) focus on functionalities of the complex service but leave no place for the required non-functionalities satisfaction.

The following sections will present an approach to service composition which models functional and non-functional requirements (in section 3) and in section 3 the decomposition of the service composition task that fulfills both functional and non-functional requirements. Section 5 shows an example to visualize the composition process.

2 Contribution to Sustainability

Service Oriented Architecture (SOA) is an approach which prolongs the life of applications, increasing their reusability by publishing them as web services which can be accessed independent of the system (i.e. its programming language) that wants to use them. They are accessible through their web-enabled interfaces by the means of XML message system. In the case of SOA legacy applications or their parts can be searched and executed whenever functionalities are needed but those functionalities have to be properly modeled in order to conduct a successful search. In this context automatic service composition allows for sustainability of the system through the re-use of various web-services in a form of complex web-services.

3 Complex Service Composition Problem Definition

3.1 Service Level Agreement

The Service Level Agreement (SLA) is a contract that specifies user requirements:

$$SLA_l = (SLA_{lf}, SLA_{lnf}) \tag{1}$$

where SLA_{lf} are the functional requirements, SLA_{lnf} are the non-functional requirements and l stands for a l-th complex service request.

The functional requirements have to be fulfilled by functionalities offered by a set of atomic services. Those services composed in a specific structure form a complex service fulfilling both functional and non-functional requirements (like availability, performance etc.).

Functional requirements consist of request of functionalities and some time dependencies among them:

$$SLA_{lf} = (\Phi_l, R_l) = (\{\varphi_s, \varphi_{l1}, \varphi_{l2}, \dots, \varphi_{lk}, \dots, \varphi_k\}, R_l) \tag{2}$$

where:

- Φ_l is a set of functionalities φ_{li}, in which φ_s and φ_k are starting and ending functionalities
- R_l is a set of time dependencies among functionalities such that:

$r_{ijl} = \begin{cases} 1 & when\ \varphi_{li} \prec \varphi_{lj} \\ 0 & otherwise \end{cases}$ and $r_{ijl} \in R_l$, where $i, j \in \{1,2,\dots,n_l\}$

- n_l is a number of required functionalities in SLA_{lf}

Non-functional requirements are represented by a set of constraints:

$$SLA_{lnf} = \{\psi_{l1}, \psi_{l2}, \dots \psi_{lk}, \dots \psi_{lm_l}\} \tag{3}$$

where ψ_{li} is a requirement of a single non-functionality – a constraint put on the composed complex service (i.e. required execution time of the whole service) and m_l is a number of required non-functionalities in SLA_{lnf}.

3.2 Service Composition Task

The service composition task is formulated as follows:

for a given

- request of a complex service SLA_l
- repository of services AS: as_k is a k-th atomic service,
- family of execution graphs $\{G(V,E)\}$ representing all possible execution plans of the requested complex service in which vertices contain services ($as_k \in V$) and edges from E define the succession of services in a complex service,

find an optimal execution plan G(V,E) of a complex service, fulfilling the functional and non-functional requirements by minimizing the quality criterion:

$$G(\{as_1^*, as_2^*, \dots, as_k^*\}, E^*) \leftarrow \min_{as_1, as_2, \dots, as_k, E} Q(G(\{as_1, as_2, \dots, as_k\}, E), SLA_l) \tag{4}$$

where:

- $as_1^*, as_2^*, \dots, as_k^*$ are the selected optimal services that belong to corresponding k-th vertex of graph G (each vertex contains one service that fulfils one functional requirement), where optimal means that all services fulfill functional requirements and no other set of services gives better non-functional properties of the complex service (given a set of edges E)
- E* is an optimal set of edges that determines the succession of services $as_1^*, as_2^*, \dots, as_k^*$. Optimal means here that no other set of edges gives better non-functional properties of the complex service and that the set of edges is minimal, meaning no edge could be rejected without the loss of consistency of the graph.

4 Problem Decomposition

Service composition process can be described as a series of graph transformations beginning with the transformation of users requirements (SLA_l) to the graph form and

ending with an execution graph representing the complex service fulfilling the functional and non-functional requirements of the SLA_l. The service composition problem can be decomposed into three stages:

- **structure**: SLA_l is given a graph form, where all functionalities are embedded in vertices and time dependencies are expressed in the form of edges (preferably in the minimal number of edges possible without loss of information),
- **scenario**: the set of edges of the graph is extended so that the graph is consistent (complete information about order of execution of all functionalities is known) and service repository is searched in order to find services capable of providing requested functionalities,
- **execution plan**: for each vertex only one service is chosen (from candidates gathered in the previous step), so that all the services in the structure fulfill the non-functional requirements.

4.1 Complex Service Structure Stage

At this stage users requirements are transformed into the form of the graph. Each vertex contains one functionality required by the user and each edge represents the order dependency between two requirements (services that fulfill them). In the resulting graph there can be less edges that would appear from the set of time dependencies R_l. This is because a simpler structure is preferred if the simplification does not change the correct order defined in R_l. Fig. 1 depicts the situation where in the left side there is a graph with dependencies defined by the user while graph on the right after simplification has a serial structure (with edge (c) omitted). All the dependencies are still valid.

Fig. 1. Reduction of complexity of the structure

The task is formulated as follows:

for a given $SLA_{lf} = (\{\varphi_s, \varphi_{l1}, \varphi_{l2}, \dots, \varphi_{lk}, \dots, \varphi_k\}, R_l)$

find:

$$GB_l(VB_l, EB_l = EB_l^*):\ EB_l^* = \arg\min_{EB_{lk}} \|EB_{lk}\| \tag{5}$$

where:

- $VB_l = \{v_s, vb_{l1}, vb_{l2}, \dots, vb_{lk}, \dots, vb_{ln_l}, v_k\}$, $v_s = \varphi_s$, $vb_{li} = \varphi_{li}$, $v_k = \varphi_k$
- $EB_l \ni eb_{ijl}$, such that $\forall_{r_{ijl} \in R_l}$ if $r_{ijl} = 1$, then exists a path $d(vb_i, vb_j)$

4.2 Complex Service Scenario Stage

Adding missing edges. The graph obtained at the previous stage rarely is consistent. However, there cannot be any functionalities that have no connection with the rest of

the graph, because this would result in them not being executed at all. The task is formulated as follows:

for a given $GB_l(VB_l, EB_l)$ **find** a consistent graph $GC_l(VC_l, EC_l = EC_l^*)$ such that:

$$EC_l^* = \arg\min_{EC_{ls}} d_{max}(GC_l(VC_l, EC_{ls}), vb_s, vb_k) \tag{6}$$

where:

- $d_{max}(GC_l(VC_l, EC_{ls}), vb_s, vb_k)$ – length of the longest path from vertex vb_s to vertex vb_k in graph $GC_l(VC_l, EC_{ls})$
- $VC_l = VB_l$
- $EB_l \subset EC_{ls} = EB_l + EA_{ls}$

The criterion in (6) ensures that structures with a higher degree of parallelization are preferred. This will directly influence the non-functional properties of the complex service i.e. execution time.

Services selection. Based on the users functional requirements atomic services are selected from the services repository. Services are found after semantic and structural comparison of their descriptions and functionalities from graph GC_l. The problem of services selection could not approached in depth in this paper. For more information please refer to [10] and [11]. The task of services selection is formulated as follows:

for a given

- consistent graph $GC_l(VC_l, EC_l)$ (in each vertex containing functional requirements)
- service repository AS

find:

- graph $GS_l\left(VS_l = \left\{AS_{l1}, AS_{l2}, \dots, AS_{lk}, \dots, AS_{ln_l}\right\}, ES_l\right)$ of valid realizations (services fulfilling the functional requirements) such that:

$$\exists_{j\in\{1,2,\dots,J\}} as_j = as_{lk_u} \in AS_{lk} \text{ if } \boldsymbol{\varphi\left(as_{lk_u}\right) \supset \varphi_{lk}} \text{ for each } u = \{1,2,\dots,u_{lk}\} \tag{7}$$

where:

- as_{lk_u} is a service that fulfills requirement φ_{lk} and u_{lk} is a **number** of services that fulfill that requirement (and will replace it in appropriate vertex AS_{lk} in graph GS_l)
- $\varphi\left(as_{lk_u}\right)$ are functionalities offered by the service and $\varphi\left(as_{lk_u}\right) \supset \varphi_{lk}$ means that service as_{lk_u} offers more or equal functionalities than required
- $AS_{lk} = \{as_{lk_1}, as_{lk_2}, \dots, as_{lk_{u_{lk}}}\}$, $AS_{lk} \subset AS$, $AS_{lk} \in VS_l$

4.3 Complex Service Execution Graph

After the scenario construction stage, each vertex of graph GS contains a number of services that satisfy the corresponding functional requirements. To obtain the execution graph G_l in each node of graph GS_l one service has to be selected, such that – with regard to the edges that create the structure of the graph – the complex service consisting of selected services fulfills the non-functional requirements. The task is formulated as follows:

for given

- graph $GS_l\left(VS_l = \{AS_{l1}, AS_{l2}, \ldots, AS_{lk}, \ldots, AS_{ln_l}\}, ES_l\right)$,
- non-functional requirements $SLA_{lnf} = \{\psi_{l1}, \psi_{l2}, \ldots \psi_{lk}, \ldots \psi_{lm_l}\}$,
- weight w_{lk} of k-th non-functionality of SLA_{lnf},

find

- an optimal execution graph $G_l(V_l, E_l)$ such that:

$$G_l(V_l, E_l) \leftarrow \min_{G_{lp}(V_{lp},E_l)} \sum_{k=1}^{m_l} w_{lk}\left[\psi_{lk} - f_{\psi_k}\left(G_{lp}(V_{lp}, E_l)\right)\right] \tag{8}$$

where:

- $G_{lp}\left(V_{lp}, E_l\right) \in FG_l(V_l, E_l)$ – family of valid execution graphs (each vertex contains one service selected from AS_{lk} : $V_{lp} \ni v_{lpk} = \{as_j \in AS_{lk}\}$ and the set of edges stays the same as in GS_l graph: $E_l = ES_l$)
- m_l – length of vector of non-functional requirements Ψ_l of l-th service request
- $f_{\psi_k}\left(G_{lp}(V_{lp}, E_l)\right)$ is a function aggregating non-functionalities (corresponding to ψ_k) of atomic services embedded in the execution graph $G_{lp}(V_{lp}, E_l)$

with subject to the following constraints:

$$\forall_{k=1}^{m}: f_{\psi_k}\left(G_{lp}(V_{lp}, E_l)\right) \leq \psi_{lk} \tag{9}$$

5 Example

For given:

$\boldsymbol{SLA_{lf}} = (\{\varphi_1, \varphi_2, \varphi_3\}, R_l)$, where $r_{ijl} \in R_l$: $r_{ijl} = \begin{cases} 1 & for\ i = 1\ and\ j = 2 \\ 0 & otherwise \end{cases}$

(for simplicity φ_s and φ_k are omitted in this example)

$\boldsymbol{SLA_{lnf}} = \Psi_l = [7,4]$; $w_{l1} = 1, w_{l2} = 2$, ψ_{l1} – cost, ψ_{l2} – time,

Service repository $AS = \{as_j\},\ j = 1,2, \ldots J$, where:

$\varphi(as_1) = \varphi_1, \varphi(as_2) = \varphi_2, \varphi(as_3) = \varphi_2, \varphi(as_4) = \varphi_3, \varphi(as_5) = \varphi_3, \ldots$

$\psi(as_1) = [2,1], \psi(as_2) = [3,2], \psi(as_3) = [2,2], \psi(as_4) = [1,4], \psi(as_5) = [3,1]$

Find optimal complex service execution graph $G_l(V_l,E_l)$ that fulfills the SLA.

Solution:

In the **structure** stage from the given SLA_{lf} only one possible can be obtained:

$\boldsymbol{GB_l = GB_l(\{\varphi_1, \varphi_2, \varphi_3\}, \{(\varphi_1, \varphi_2)\})}$

In the **scenario** stage there are numerous possible consistent graphs (Fig. 3). As (6) ensures the more parallel structure is preferred (here: $d_{max}(GC_l^*, vb_s, vb_k) = 1$) and the optimal graph GC_l^* is obtained:

$\boldsymbol{GC_l^* = GC_{ln}(\{\varphi_1, \varphi_2, \varphi_3\}, \{(\varphi_1, \varphi_2), (\varphi_1, \varphi_3)\})}$

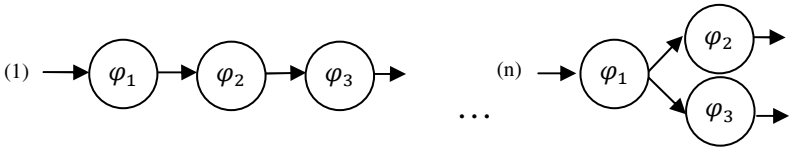

Fig. 2. Possible scenarios of a complex service

In step two of scenario stage for each functional requirement services that fulfill it are selected from the repository AS and so graph GS_l is as follows:

$$\boldsymbol{GS_l}(\{\boldsymbol{AS_1} = \{\boldsymbol{as_1}\}, \boldsymbol{AS_2} = \{\boldsymbol{as_2}, \boldsymbol{as_3}\}, \boldsymbol{AS_3} = \{\boldsymbol{as_4}, \boldsymbol{as_5}\}\}, \{(\boldsymbol{AS_1}, \boldsymbol{AS_2}), (\boldsymbol{AS_1}, \boldsymbol{AS_3})\})$$

In the **execution graph** stage based on the non-functional parameters of selected service an optimal execution graph can be found:

$$\boldsymbol{G_l}(\boldsymbol{V_l}^*, \boldsymbol{E_l}) = \boldsymbol{G_l}(\boldsymbol{v_{l1}^*} = \{\boldsymbol{as_1}\}, \boldsymbol{v_{l2}^*} = \{\boldsymbol{as_3}\}, \boldsymbol{v_{l3}^*} = \{\boldsymbol{as_5}\}, \{(\boldsymbol{v_{l1}^*}, \boldsymbol{v_{l2}^*}), (\boldsymbol{v_{l1}^*}, \boldsymbol{v_{l3}^*})\})$$

For the above execution graph G the value of quality criterion (8) is minimal:

$$\sum_{k=1}^{2} w_{lk}\left[\psi_{lk} - f_{\psi_k}\left(G(V^*, E_l)\right)\right] = 1 * (7 - 7) + 2(4 - 3) = 2$$

6 Conclusions

The model for service composition presented in this paper allows for specification of both functional and non-functional requirements. The decomposition of the task allows for specification of three problems: transformation of user requirements into the structure of the graph, creating a scenario for the complex service by making the graph consistent and selecting services and at last finding the execution graph for the service that fulfills all requirements. The proposed approach takes into consideration the fact that constructing an appropriate structure for the complex service and selection of services with various non-functionalities are crucial to the problem of fulfilling non-functional requirements and should not be solved separately.

During the next stage of development the security assessment of atomic services will be taken into account, according to formal approach recently presented in [12]. This will allow to address security as an important non-functional property of complex services, barely covered in recent research and practical applications.

A composition framework based on the presented model is developed and will be made available in the form of package of tools for service composition.

The incoming stage of the project also assumes the application and evaluation of the framework in an industrial scenario.

Acknowledgments

The research presented in this paper has been partially supported by the European Union within the European Regional Development Fund program no. POIG.01.03.01-00-008/08.

References

1. Jinghai, R., Xiaomeng, S.: 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)
2. Milanovic, N., Malek, M.: Current Solutions for Web Service Composition. IEEE Internet Computing 8(6), 51–59 (2004)
3. Charif, Y., Sabouret, N.: An Overview of Semantic Web Services Composition Approaches. Electronic Notes in Theoretical Computer Science, 146, 33–41 (2006)
4. 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, pp. 23–30 (2004)
5. Blanco, E., Cardinale, Y., Vidal, M., Graterol, J.: Techniques to Produce Optimal Web Service Compositions. IEEE Congress on Services, 553–558 (2008)
6. Ko, J.M., Kim, C.O., Kwon, I.-H.: Quality-of-service oriented web service composition algorithm and planning architecture. The Journal of Systems and Software 81, 2079–2090 (2008)
7. McIlraith, S., Son, T.: Adapting Golog for composition of semantic web services. In: Proceedings of 8th Intl. Conf. of Knowledge Representation and Reasoning (2002)
8. Ponnekanti, S.R., Fox, A.: SWORD: A developer toolkit for Web service composition. In: Proceedings of the 11th World Wide Web Conference, Honolulu, HI, USA (2002)
9. Klusch, M., Fries, B., Sycara, K.: OWLS-MX: A hybrid Semantic Web service matchmaker for OWL-S services. In: Web Semantics: Science, Services and Agents on the World Wide Web, vol. 7, pp. 121–133 (2009)
10. Stelmach, P., Prusiewicz, A.: An improved method for services selection. In: XVII International Conference on Systems Science, Wrocław (2010)
11. Stelmach, P., Juszczyszyn, K., Grzelak, T.: The scalable architecture for the composition of soku services. In: 31th International Conference on Information Systems, Architecture, and Technology, Szklarska Poręba (2010)
12. Kolaczek, G., Juszczyszyn, K.: Smart Security Assessment of Composed Web Services. Cybernetics and Systems 41(1), 46–61 (2010)
13. Agarwal, V., Chafle, G., Dasgupta, K., Karnik, N., Kumar, A., Mittal, S., Srivastava, B.: Synthy: A system for end to end composition of web services. In: Web Semantics: Science, Services and Agents on the World Wide Web. World Wide Web Conference 2005, Semantic Web Track, vol. 3(4), pp. 311–339 (2005)
14. Zeng, L., Kalagnanam, J.: Quality Driven Web Services Composition. In: 12th International Conference on the World Wide Web, pp. 411–421 (2003)
15. Huang, A.F.M., Lan, C., Yang, S.J.H.: An Optimal QoS-based Web Service Selection Scheme
16. Karakoc, E., Senkul, P.: Composing semantic Web services under constraints. Expert Systems with Applications 36(8), 11021–11029 (2009)
17. Ko, J.M., Kim, C.O., Kwon, I.-H.: Quality-of-service oriented web service composition algorithm and planning architecture. Journal of Systems and Software 81(11), 2079–2090 (2008)