Service Composition with Pareto-Optimality of Time-Dependent QoS Attributes

Benjamin Klöpper*, Fuyuki Ishikawa, and Shinichi Honiden

National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan

Abstract. Quality of Services (QoS) plays an essential role in realizing user tasks by service composition. Most QoS-aware service composition approaches have ignored the fact that QoS values can depend on the time of execution. Common QoS attributes such as response time may depend for instance on daytime, due to access tendency or conditional Service of Level Agreements. Application-specific QoS attributes often have tight relationships with the current state of resources, such as availability of hotel rooms. In response to these problems, this paper proposes an integrated multi-objective approach to QoS-aware service composition and selection.

Keywords: Service Composition, Multi-objective Optimization.

1 Introduction

Both, functional and non-functional properties - the latter expressed by Quality of Service (QoS) Attributes - have to be taking into account when identifying a suitable composition of a user task. Current approaches consider these QoS attributes to be static, so they do not change over time. Hence, these approaches cannot properly reflect certain business models or the interaction with scarce resources. For instance, service providers maybe use flexible price models in order to level the utilization of their computational resources. Furthermore, the availability and hence the price of goods and non-computational services are also time-dependent, e.g. flight tickets or hotel rooms get less overtime. Hence, composition approaches should be able to consider the time-dependency of QoS attributes. Furthermore, this time dependency makes the decision problem more complex and hardly comprehensible for human users. A composition process is desirable that presents a set of relevant solutions to the user.

2 The Timed Service Composition Problem

To realize a user task by a composed service, two problems must be solved. First, a workflow must be identified that implements the user task. Often, alternative workflows can be found. Second, specific service instances must be selected to implement the workflow. If these problems are solved subsequently, only local optima can be found. Hence, we introduce an integrated solution approach.

* Benjamin Klöpper is a visiting researcher and scholarship holder of the (DAAD).

2.1 Workflow Model

The service model used throughout the paper follows basically a model introduced by Kalasapur et al. [3]. A user service s is described by a workflow schema, a directed acyclic graph (DAG) $w = (ST, D, Source, Sink)$ where the vertex set ST represents the services required to implement the user service and the edge set D describes input-output relation or dependencies between two services. If there is a $d = (st_a, st_b) \in D \; and \; st_a, st_b \in ST$, the service st_a has to be accomplished before st_b can start. To include alternative workflow schemes for the implementation of user services, we extend the model from [3]. Two different types of *services* are considered:

1. Atomic services (S_A): services offered by a service provider
2. Complex services (S_C): services with alternative workflow schemes

A complex service $cs \in S_C$ is a set of alternative workflow schemes W_{cs}. The composition of a complex service cs consists of the implementation of all services s_t in one of the workflows $w_{cs} \in W_{cs}$. An implementation of an atomic service is defined by the selection of a service instance offered by a service provider and a corresponding start time. A specific implementation of a service s is referred to as i_s.

2.2 Solution of the Timed-Composition Problem

A timed composition plan CP of a user service or complex service cs is the set of all service instances $s_o, ..., s_k$ from a workflow schema $w_{cs} \in W_{cs}$ with start time $start_i$ and finishing time $finish_i$ of each service instance s_i in the list. A feasible composition plan contains no service instances s_i and s_j such that there is a directed edge from s_j to s_i in the workflow schema w_{cs} and s_i starts before s_j ends. A number of QoS attributes qa [1..n-1] and the response time of the user service i_{cs} establish a n-dimensional vector PV, describing the performance of the composition plan. Given the n-dimensional vector, the pareto-optimality of composition plans can be defined. A composition plan with performance vector PV is pareto-optimal or non-dominated if there is no other composition plan with performance vector PV' such that:

$$PV'_i \leq PV_i \; \forall i, 1 \leq i \leq n+1 \text{ and } PV'_i < PV_i \; \exists i, 1 \leq i \leq n+1 \tag{1}$$

If FS is the set of feasible solutions, a pareto-optimal composition process returns a set PO with all performance vectors that satisfy two conditions:

$$\forall pv_{po} \in PO : \nexists s \in FS : pv_s \prec pv_{po} \tag{2}$$

$$\forall pv_{po} \in PO : \exists s \in FS : pv_s = pv_{po} \tag{3}$$

2.3 Monotone Quality of Service Attributes

If QoS attributes depend in execution time and response time is a relevant QoS attribute, a potential trade-off occurs: It is possible to achieve a gain in some QoS attributes by delaying the execution of a service. To catch this trade-off properly in the composition problem, we will assume that the QoS attributes of service instances are monotonically decreasing. Any service instance without this property will be split into several logical service instances with an earliest and latest start time. These logical services are encapsulated in a complex service, where each alternative workflow consists of exactly one atomic service with the corresponding earliest and latest start time.

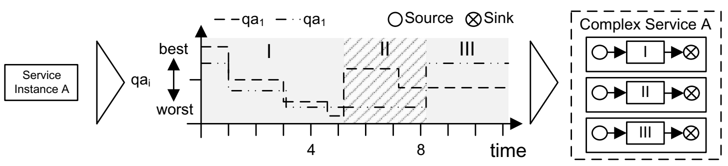


Fig. 1. Service Instance split into Logical Instances I, II, III with Monotony Property

3 Partial Composition Plan Search Tree

This section maps the timed composition problem defined above to a search problem. Each node in the search tree represents a partial composition plan of the complex service to be composed, selecting a number of specific services for composition. Starting from an empty composition (no services are selected) the partial composition plans subsequently are extended until a complete composition is found.

A partial composition plan is a tuple $(S_{pc}, A_{pc}, N_{pc}, ST_{pc}, FT_{pc})$, where S_{pc} is the set of services selected for the composition plan (complex services and atomic service), A_{pc} is the set of services available for expansion, and N_{pc} contains all services that are not yet available. A service is available for expansion, if all its predecessors from the corresponding service graph are already included in S_{pc}. ST_{pc} and FT_{pc} are two functions mapping the service S_{pc} to their start and finishing times. The start time of any service is determined by the latest finishing time of any of its predecessors in the workflow schema and the earliest start time of the service. The root of the search tree is the partial composition plan which contains the services from the user specified service and S_{pc} is empty, A_{pc} contains only the source of the user service workflow and N_{pc} contains all remaining services.

Partial composition plans are extended towards a complete composition plan by two repeated steps. In the first step, a service s from A_{pc} is selected for expansion. In the second step, all offspring pc' regarding parent pc and selected service s are generated. Regarding the generation of offspring, two cases have to be distinguished. In the *first case* a service instance as from the set A_{pc} is added to the partial composition plan. The result is one offspring for each service instance. In each of these new partial composition plans, as is added to the set of selected services S_{pc}. The earliest start time depends on the latest finishing time of any predecessor service, the finish time as well as the QoS attributes q_a depend on the performance offered by the service provider at start time. In this way, the QoS attributes of a service instance depends of the selected start time of its predecessor. After adding as to the set C_{pc}, all services from N_{pc} that are now available are moved to A_{pc}.

In the *second case*, a complex service cs from the set A_{pc} is added to the composition plan. The result is one offspring for each decomposition d of the complex service. In each new partial composition plan, the source node of the decomposition d is added to the set of selected services S_{pc}. The remaining services from the decomposition are added to A_{pc} if the source is their only predecessor or to N_{pc} otherwise. The complex service cs will be finally moved to A_{pc}, when the sink node of d is moved to A_{pc}. Hence, all successors of cs can become available, when the decomposition d is entirely included in the partial composition plan. Given these two expansion operations, a search tree is defined. The leaves of the tree are complete composition plans of the user specified service. A composition plan is complete, when the two sets A_{pc} and N_{pc} are empty. We refer to the set of leaves reachable from partial composition pc as CP_{pc}.

Given an admissible heuristic function that systematically underestimates the distance towards a complete schedule; heuristic search algorithms can be used to identify optimal compositions. The extension to the multi-objective case is done by not optimizing a scalar value (e.g. weighted sum over all QoS attributes), but identifying all non-dominated compositions regarding the n-dimensional objective vector. In [4] we introduced a suitable heuristic function. Hence, multi-objective heuristic search algorithms (for instance, cf. [6,5]) can be used to identify the set of non-dominated compositions.

3.1 Use Case with Multi-objective Search Based Composition

Figure 2 shows an example user task with two alternative workflows. As well, the complex task E_1 encompasses two alternative workflows, the first one consisting of a single task and the second one consisting of two subsequent tasks.

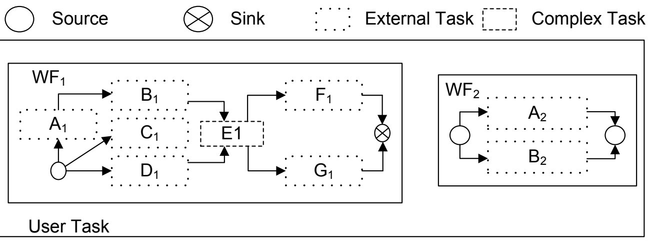
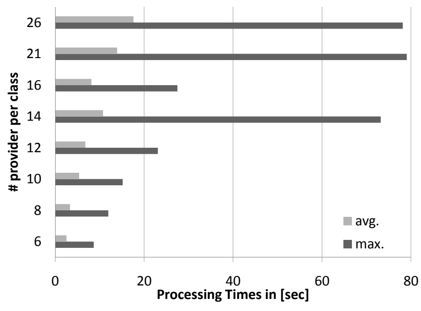


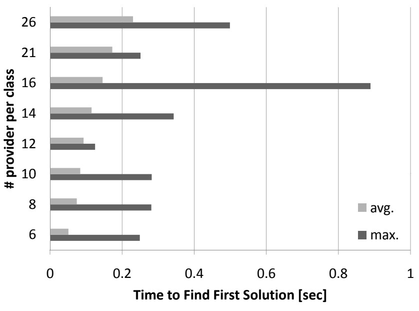
Fig. 2. User Task

For the evaluation regarding the use case, the time-dependent QoS attributes and the execution time were randomly chosen. The QoS attributes for each service instance were generated for a time horizon with 100 time points. The attributes were generated as monotone increasing on time intervals between 3 and 7 time steps. Hence, each service instance is split into 15 up to 34 logical service instances.

Figure 3 shows results from experiments[1] with different of service instances. Each experiment consisted in 50 randomly generated problem instances. Figure 3(a) shows that the run-times do not solely depend on the size of the problem, but also on the relation between the objective functions. The maximum run-times for bigger instances are significantly too long for interactive systems. Figure 3(b) on the other hand shows that a feasible solution can be found rather fast (below one second). This is a strong hint, that anytime algorithms could be developed for interactive systems.



(a) Processing Times



(b) Time to find a first Composition

Fig. 3. Experimental Results from the Example Problem

4 Related Work

The introduced service model and solution procedure are in between *staged composition and execution* and *template-based composition* [1]. The composition is not as static as in template based approaches, but the structure of complex services resembles predefined workflow scheme. The hierarchical structure and the subsequent refinement of complex tasks resembles Hierarchical Task Network Planning (HTN) that is quite popular in the service community [2]. HTN based approaches do not consider time-dependent attributes and cannot find the set of non-dominated compositions regarding a set of QoS attributes. Multi-objective approaches for service composition were until now limited to template-based approaches [7,8].

[1] Performed on a PC with an Intel Core 2 Quad CPU (2.99Ghz), 8GB RAM, Windows 7 64 Bit, JVM 1.6 using a single thread implementation.

5 Conclusion

In this paper we introduced the *timed service composition problem* that considers time-dependent QoS attributes. Furthermore, we mapped the problem to a search problem and showed first results of the implementation of a composition algorithms based on multi-objective heuristic search. In our opinion, the *timed service composition problem* is an important step towards the proper consideration of complex business models of service providers as well as scarce physical resources and services, such as hotel rooms or flight tickets. The employment of the multi-objective composition algorithm is more straightforward then classical single objective approaches because a priori definition of an objective function is not necessary. To use the approach in practical application, approximation algorithms have to be developed and implemented.

References

1. Argarwal, V., Chafle, G., Mittal, S., Srivastava, B.: Understanding Approaches for Web Service Composition and Execution. In: Proceedings of the 1st Bangalore Annual Compute Conference, ACN (2008)
2. Chen, K., Xu, J., Reiff-Marganiec, S.: Markov-HTN Planning Approach to Enhance Flexibility of Automatic Web Service Composition. In: IEEE International Conference on Web Services, pp. 9–16 (2009)
3. Kalasupur, S., Kumar, M., Behrooz, A.: Dynamic Service Composition in Pervasive Computing. IEEE Transactions on Parallel and Distributed System 18, 907–917 (2007)
4. Klöpper, B.: First Steps Towards Distributed Multiobjective Scheduling for Self-Optimizing Manufacturing Systems. In: 10th IFAC Workshop on Intelligent Manufacturing Systems (2010)
5. Mandow, L., Pérez de la Cruz, J. L.: A new approach to multiobjective A* Search. In: Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI 2005). pp. 218–223. Edinburgh, Scotland (2005)
6. Stewart, B.S., White, C.C.: Multiobjective A*. Journal of the Association for Computing Machinery 38(4), 775–814 (1991)
7. Wang, J., Hou, Y.: Optimal Web Service Selection based on Multi-Objective Genetic Algorithm. In: International Symposium on Computational Intelligence and Design (2008)
8. Zeng, L., Benatallah, B., Ngu., A., Dumas, M., Kalagnanam, Chang, H.: QoS-Aware Middleware for Web Services Composition. IEEE Transactions on Software Engineering 30, 311–327 (2004)