Efficient, Failure-Resilient Semantic Web Service Planning

Florian Wagner[1]
(Advisors: Fuyuki Ishikawa[2] and Shinichi Honiden[1,2])

[1] The University of Tokyo, Japan
[2] National Institute of Informatics, Tokyo, Japan
{florian,f-ishikawa,honiden}@nii.ac.jp

Abstract. Over the past years service-oriented architectures have been widely adopted by stakeholders from research and industry. Since the number of services increases rapidly, effective methods are required to automatically discover and compose services according to user requirements. For this purpose, machine-understandable semantic annotations have to be applied in order to enable logical reasoning on the functional aspects of services. However, current approaches are not capable of composing workflows in reasonable time, except for planning tools that require domain-dependent heuristics or constrain the expressiveness of the description language. In addition to that, these tools neglect alternative plans, concealing the danger of creating a workflow having insufficient reliability. Therefore, we propose an approach to efficiently pre-cluster similar services according to their parameters. This way the search space is limited and vulnerable intermediate steps in the workflow can be effectively avoided.

1 Introduction

The service-oriented paradigm envisions the discovery and composition of reusable services by using widely accepted standards such as XML and HTTP. This way loosely coupled components can be dynamically integrated into an infrastructure of e.g. a company, independent of the given hardware and software.

In order to further enhance the automatic selection and composition of services, machine-processable descriptions based on service ontologies such as WSMO and OWL-S and the annotation language SAWSDL have been proposed. Web services that provide semantic descriptions are called *semantic web services*.

Semantic annotations are mandatory for AI planning, that can be incorporated in order to create service workflows automatically. Instead of offering a restricted set of predefined composite services, the user can request customized composite services by providing a set of input and output parameters plus preconditions and the intended effects. Service planning is an active research field that poses one central challenge in service computing research [4].

In this paper we tackle the problems of insufficient performance and lacking reliability of current service planning algorithms. Many registries contain services that have the same intended purpose. Therefore, we will show how the

P.P. Maglio et al. (Eds.): ICSOC 2010, LNCS 6470, pp. 686–689, 2010.

performance and reliability issues can be addressed by identifying and clustering these services beforehand.

2 Problem Statement

In this section we present the two main phases in service planning, the composition of abstract workflows and the subsequent service grouping for each task.

In the composition phase, a planning tool attempts to compute a composite service from a set of given services. For this purpose, the user provides a set of input and output parameters including the guaranteed preconditions and the intended effects to the planer. Since the search space is exponential, the runtime performance and memory usage are central issues in current planning tools and therefore AI algorithms cannot be applied blindfolded. In order to prune the search space, heuristics can be employed but these are in most cases tailored for a particular domain and therefore are only of limited reusability.

After computing an appropriate workflow, for each task all applicable services are selected and grouped in so-called service classes. If in the execution phase one service crashes, then any other service from the same class can be used to compensate the erroneous services (failure resilience). Moreover, QoS-(Quality-of-Service) aware service selection algorithms [7] can be applied to choose a specific service for each class in order to optimize the utility function of the user. However, current planning tools do not take the reliability of the service classes into account but instead attempt to find the shortest possible plan in order to simplify the planning task. If a service class contains only very few services, then these class might comprise the whole workflow, in other words pose a so-called *single point of failure*.

If a service fault occurs, re-planning can be instantiated [2], invoking additional services in order to obtain the desired output parameters and effects. This might include services that try to revert some recent world state changes caused by previous services, entailing additional costs for the user.

3 Proposed Approach

Our approach towards service planning is divided into two steps. First, functional similar services are pre-clustered and arranged in a hierarchical structure beforehand. Second, in the composition stage a planning tool is incorporated, taking only the representatives of each cluster into consideration. This way the search space is limited efficiently and heuristics that take the reliability into consideration can be integrated easily.

3.1 Pre-clustering of Services

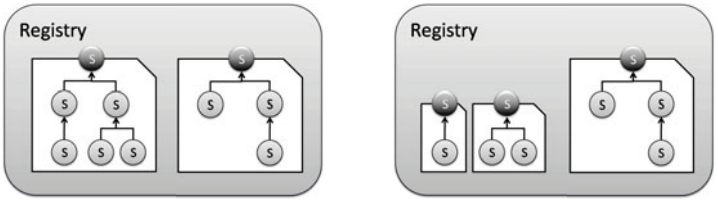
Registries often contain services that have the same intended purpose, e.g. booking of a hotel room or requesting stock market information. By clustering these services automatically, we can avoid redundant computations in the planning

phase and moreover evaluate the reliability of every type of service. In order to identify such clusters we propose to use the partial order relation that is given by *Exact* and *Plug-in* matches and was first introduced as a clustering method in pervasive service computing [3]. Given a specific service s, all services that have an *Exact* or *Plug-in* match with this service can be used as a replacement.

In the end, services are grouped in a hierarchical structure, turning the registry into a set of trees. Roots are called *representatives* since their functionality can be considered as a "least common denominator" of its corresponding subtree.

3.2 Planning Algorithm Outline

In the planning stage only the computed representatives are taken into account. In case no feasible plan could be computed, large service classes are split up by removing the root and adding the subtrees into the registry (cf. Figure 1).



(a) Original service classes

(b) After splitting the biggest class

Fig. 1. A split operation on the service classes

Afterwards, the planning algorithm iteratively attempts to compute a plan with the modified classes until a plan was found. In the end, the workflow containing the corresponding classes is returned.

3.3 Expected Outcome

We expect to shrink the search space to a size that can be solved by planning tools in reasonable time. In contrast to related search algorithms, that employ a domain-specific heuristic, our approach is domain-independent.

Single points of failure can be avoided effectively since each service in the service class can be used as a replacement service in case of a failure.

Selection algorithms, mentioned in Section 2, benefit from our approach as well. Since we attempt to extend the number of services per task, these algorithms have more services to choose from. Using our approach, these algorithms can be reused and therefore bridge the gap between automatic composition and service selection research.

4 Related Work

AI planning has a long history in computer science and various approaches have been introduced to the service domain [5].

The approach described in [6] uses the HTN planner SHOP2, requiring a plan library that contains decomposition rules for dedicated domains, thus this approach cannot be applied to arbitrary domains.

In [1] the HTN planner Xplan is employed, combining forward and HTN planning. It uses a graph plan algorithm if no decomposition rules can be applied. Thereby, it will always find a plan, but the search space is also exponential.

Apart from that, several researchers focus on the problem of crashing services. For instance, in [2] a re-planning scope is defined and iteratively extended until all faulty services have been successfully compensated. Our approach attempts to avoid re-planning, since it entails additional costs for the user.

Our approach is based on [3] that originally introduced the hierarchical structure implied by *Exact* and *Plug-in* matches for the service matchmaking domain.

5 Conclusion and Future Work

In this paper we have presented our approach towards semantic web service planning. Thereby, we addressed performance and reliability issues by pre-clustering functional similar services and determining representatives for the planning stage.

As a next step we intend to further develop the current pre-clustering and planning algorithm. Subsequently, we will examine how to integrate both into given planning tools and evaluate our approach.

References

1. Klusch, M., Gerber, A.: Semantic web service composition planning with OWLS-XPlan. In: Proceedings of the 1st Int. AAAI Fall Symposium on Agents and the Semantic Web, pp. 55–62 (2005)
2. Lin, K.J., Zhang, J., Zhai, Y.: An efficient approach for service process reconfiguration in SOA with End-to-End QoS constraints. In: Hofreiter, B., Werthner, H. (eds.) CEC, pp. 146–153. IEEE Computer Society, Los Alamitos (2009)
3. Mokhtar, S.B., Preuveneers, D., Georgantas, N., Issarny, V., Berbers, Y.: EASY: Efficient semantic service discovery in pervasive computing environments with QoS and context support. Journal of Systems and Software 81(5), 785–808 (2008)
4. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. IEEE Computer 40(11) (2007)
5. Peer, J.: Web service composition as AI planning - a survey. Tech. rep., University of St. Gallen, Switzerland (2005)
6. Wu, D., Parsia, B., Sirin, E., Hendler, J.A., Nau, D.S.: Automating DAML-S web services composition using SHOP2. In: Fensel, D., Sycara, K., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 195–210. Springer, Heidelberg (2003)
7. Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J., Sheng, Q.Z.: Quality driven web services composition. In: WWW 2003: Proceedings of the 12th International Conference on World Wide Web, pp. 411–421. ACM, New York (2003)