An Approach for Mining Web Service Composition Patterns from Execution Logs

Ran Tang and Ying Zou
Department of Electrical and Computer Engineering
Queen's University
Kingston, Canada
{ran.tang, ying.zou}@queensu.ca

***Abstract* — A service-oriented application is composed of several web services to provide complex functionality that a single web service cannot provide. A set of services along with their control flows can be frequently used in multiple applications. Such services form a service composition pattern which is well tested in the numerous adoptions. Reusing service composition patterns in service composition provides an efficient way to improve the quality of new applications. To facilitate the documentation of service composition patterns, we propose an approach to automatically recognize service composition patterns from various applications. We identify service composition patterns by locating a set of associated services commonly used by different applications and recovering the control flows among the set of associated services.**

Keywords — web service; composition; pattern mining; log

I. INTRODUCTION

A service-oriented application is composed of multiple web services to provide complex functionality that a single web service cannot provide [1]. Traditionally, applications are individually deployed on the service providers' infrastructure. With the increasing uses of "Platform as a service" paradigm, which provides a centralized runtime execution environment, more and more service-oriented applications are deployed on centralized runtime execution environments, such as Microsoft Azure Services Platform [7]. In comparison with the traditional deployment model, "Platform as a service" paradigm can reduce the cost and complexity of managing the underlying hardware and software. Such a paradigm facilitates the monitoring of services-oriented applications to obtain the execution logs.

In the SOA application domain, different service-oriented applications can be composed to fulfill the similar functional requirements from various organizations. For example, in the travel agency domain, a travel reservation application may contain a web service to reserve tickets for sports events in the destination. Another travel reservation application may offer a web service to search for local restaurants. Variations exist in the two applications. However, most of travel reservation applications deliver common functionality, such as booking transportation and accommodation. The set of web services that deliver such common functionality along with the control flows among them can be identified as a service composition pattern and be reused to compose applications.

In general, a service composition pattern consists of a set of web services and their control flows (e.g., sequential order, parallel order). The set of web services contained in a pattern is frequently executed together by service-oriented applications. The service composition patterns are well tested by large amount of adoptions to reflect the best practices. Therefore, an application composed by reusing patterns potentially has better quality. Moreover, the patterns can be used to optimize the allocation of maintenance personnel and resources. Web services involved in a pattern are more heavily used than those that are not. Thus the service providers can allocate more resources to maintain and improve the services used in the patterns.

To facilitate the documentation and reuse of service composition patterns, a great research effort has been devoted to identifying service composition patterns using top-down and bottom-up approaches. In the top-down approach [8], the business process management architect reviews the business processes from different organizations to identify patterns. Although the identified pattern indicates commonalities between business processes, the pattern may not have been frequently used in practices. In the bottom-up approaches [2][3][4][5][6], the execution logs of applications are analyzed to mine business processes. However, the existing research focuses on recovering the control flows of a single business process without extracting patterns shared among multiple business processes and applications.

To identify patterns frequently used in practices, we present an approach that extends existing bottom-up approaches to mine service composition patterns from execution logs of service-oriented applications. Instead of recovering a single business process from the logs, we identify the frequently executed pattern used by multiple service-oriented applications.

The remainder of this paper is organized as follows. Section II describes our approach of identifying service composition patterns. Section III concludes the paper and explores the future work.

II. OVERVIEW OF OUR APPROACH

Figure 1 gives an overview of our approach which is broken down into three major steps: (1) collect and pre-process execution logs; (2) identify a set of associated web services that are frequently executed together; and (3) recover the control flows among these services.

978-0-7695-4128-0/10 $26.00 © 2010 IEEE
DOI 10.1109/ICWS.2010.35

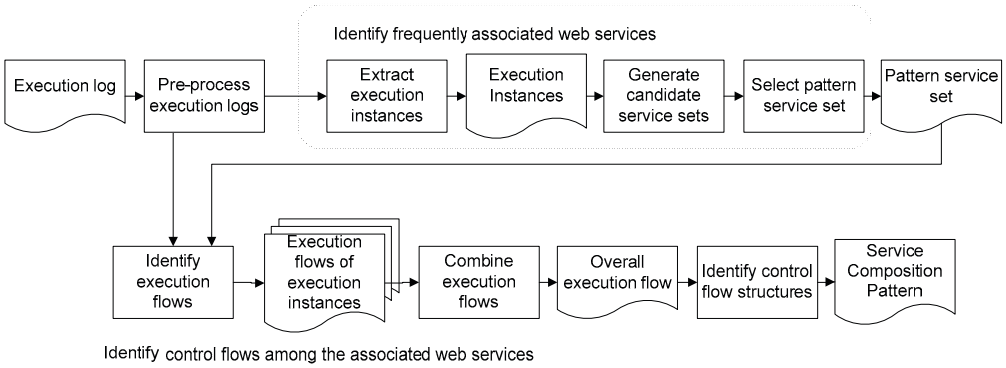

Figure 1 Overview of Our Approach

Pre-processing execution logs: Once a service-oriented application is deployed in a runtime execution environment, the application can be executed in many execution instances. Each execution instance is uniquely identified with an identifier (i.e., id). Execution instances from different applications may co-exist. In each execution instance, events can be triggered. We record the triggered events in the log. An execution log contains different types of events, such as resource adapter events which record the interaction between the service-oriented application and a legacy system, business rule events that track the runtime status of business rules, and service invocation events which indicate the timeline of a web service execution. We are interested in service invocation events. In particular, an ENTRY event is triggered when a service is invoked. An EXIT event occurs when a service completes the computation and returns results. Each event is recorded with the time of triggering, the name of the service which triggers the event, and the id of the execution instance. We process the logs to filter out events of other types.

Identifying frequently associated web services: Each execution instance invokes a set of services. To detect a service composition pattern, we find a set of associated services that frequently appear together in many execution instances. Such frequently associated services form the service set for a service composition pattern (i.e., pattern service set). The number of services in a service composition pattern is important. A service composition pattern contains at least four frequently associated services. A service composition pattern with very few services (e.g., two services) may provide incomplete functionality with minimal value for reuse. We generate candidate service set and select the one with highest frequency as the pattern service set.

Recovering the control flow among the services identified as a pattern: The control flows among services would make the patterns more structured to ease the reuse of a pattern as an independent component in service composition. To recover the control flow of a pattern, we use the execution instances that contain the pattern. Such execution instances may also contain services not in the pattern. We disregard such services. Hence, for each of these execution instances, we identify the execution flow among the services in the pattern. We combine execution flows from all execution instances to obtain the overall execution flows of the service composition pattern. We further infer the control flow structures (e.g., sequence structure, parallel structure) from the overall execution flows.

III. Conclusion and Future Work

In this paper, we present an approach for identifying service composition patterns from execution logs. To identify a pattern, we find a set of web services frequently executed together along with their control flows. Our approach facilitates the documentation and reuse of service composition patterns. Such patterns can improve the productivity of SOA developers. In the future, we plan to verify our approach using a case study which identify pattern from the execution logs.

Acknowledgment

We would like to thank Ms. Joanna Ng, Mr. Leho Nigul and Ms. Janette Wong at IBM Canada for their constructive feedback on this work.

References

[1] F. Curbera, et al. “Business process execution language for web servi ces”, http://www.ibm.com/developerworks/webservices/library/ws-bp el, last accessed on May 4, 2010

[2] R. Agrawal, D. Gunopulos, and F. Leymann, “Mining Process Models from Workflow Logs”, Sixth International Conference on Extending Database Technology, 1998, pp. 469–483

[3] W. M. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. Weijters, “Workflow mining: a survey of issues and approaches”, Data Knowl. Eng. 47, 2 Nov. 2003, pp. 237-267

[4] J.E. Cook, and A.L. Wolf, “Event-based detection of concurrency”, Proceedings of the Sixth International Symposium on the Foundations of Software Engineering , 1998, pp. 35-45.

[5] G. Schimm, “Generic linear business process modeling”, Proceedings of the ER 2000 Workshop on Conceptual Approaches for E-Business and The World Wide Web and Conceptual Modeling

[6] A.J.M.M. Weijters, W.M.P. van der Aalst, “Rediscovering workflow models from event-based data”, Proceedings of the 11th Dutch-Belgian Conference on Machine Learning Benelearn 2001, pp. 93–100.

[7] Microsoft Azure Services , http://www.microsoft.com/windowsazure, last accessed on May 4, 2010

[8] R. Dijkman, M. Dumas, and L. Garc´ıa-Ba˜nuelos, “Graph matching algorithms for business process model similarity search”, In Proc. of BPM 2009, Ulm, Germany, Sept 2009