A Framework for Dependency Based Automatic Service Composition

Abrehet M. Omer and Alexander Schill

Chair for Computer Networks, TU Dresden,01062 Dresden, Germany
omer@rn.inf.tu-dresden.de, alexander.schill@tu-dresden.de

Abstract. Developing service based complex applications (service composition) has become an important area of research in SOA. In spite of this, there has been little effort to understand and to manage the different forms of dependencies that occur in applications that are built from services. Moreover, doing composition (semi-) automatically is a crucial aspect in overcoming problems arising due to dynamic nature of a runtime environment. In this paper, we argue that automation of process model creation is one of the critical tasks to achieve dynamic service composition. We propose and present a framework for performing automatic service composition by exploring and managing dependency between services making use of semantic web service descriptions.

Keywords: Service composition, Service dependency.

1 Introduction

In Service Oriented Architecture (SOA) the task of creating composite services from component services brings dependencies between the component services. Primarily these services are created by same or different providers and they are meant to be accessed and work independently. However, establishment of composite services necessitates interaction, communication, cooperation and coordination of services. The process of combining available component services to create a composite service is called service composition. A composite service can be regarded as a combination of services invoked in a predefined order and executed as a whole and that has more functionality than its components.

1.1 Overview of Composition Techniques

The service composition process comprises different sub activities. In this paper three major sub-activities are considered: (1) Creation of process model, which is a model that simplifies the representations of activities and their enactment. (2) Service binding. (3) Execution (invocation) of composite service. Service composition could be done statically, semi-dynamically or dynamically. In static composition the process model is created manually and service binding is done at design time. Whereas, in dynamic composition the process model is created automatically and service binding is done at runtime. All methods between these two extremes are categorized as semi-dynamic [1]. Due to shortcomings of static

D. Ardagna et al. (Eds.): BPM 2008 Workshops, LNBIP 17, pp. 517–523, 2009.

composition techniques, nowadays, there is a growing tendency for shifting to (semi-) dynamic service composition techniques. This requires not only run time service binding, which has been achieved by many researchers, but also (semi-) automatic process model creation. The automatic process creating part is not thoroughly tackled by researches. For example, in the work by [2], and [3] process model generation is done by automated chaining of services or graph representation of all discovered services. The main limitation with such kind of methods is scalability. Specially chaining methods may insert degree of uncertainty regarding semantic correctness & the search space is very large [4].

WSDL [5] describe only requirements and capabilities of web services. It provides only a comprehensive technical description of a service. But a dynamic Web services composition needs semantic descriptions of services. Approaches like OWL-S [6], SA-WSDL [7] and WSDL-S [8] are being discussed in semantic Web service community. WSDL-S extends WSDL by defining new elements and annotations for already existing elements. It connects WDSL and OWL. OWL[9] is a W3C standard based on RDF(S) and it has been designed to meet the need for a web ontology language. OWL-S is an ontology represented in OWL which contains a bunch of classes and property definitions. SA-WSDL is based on WSDL-S and provides semantic characterization to Input and Outputs of web services by defining a small set of WSDL extension attributes. Such semantic descriptions could help in enhancing existing service composition techniques and in developing new automatic service composition mechanism that involve the usage of semantic knowledge in the composition process.

In addition, during service composition, beside automatic creation of process model, its modification or re-generation might be required due to runtime failure of a composite service. Such failures can occur due change in service landscape, or service failure. Researchers considered different adaptability techniques to overcome limitations of their developed service composition techniques. For example the work by [10] and [11] can be mentioned. Late-binding and re-binding are the main solution strategies considered by researchers. But these strategies fail in circumstance of service failure or removal and when it is not possible to get a replaceable service. In such cases process model re-generation is suggested as solution strategy in the work by [11].

Thus, it can be seen that the complexity of creating the process model automatically is one of the main bottle necks towards achieving dynamic service composition. Consequently, its generation, modification or re-generation during service composition is a critical task that needs research focus.

1.2 Overview of Dependencies

The concept of dependency is principally explored in component based systems, particularly for component based systems management [12]. However, there are some research works that recognize its importance in SOA, specifically in service composition. For example,[13] looks for service dependencies from a composite service management point of view. Their approach shows that dependencies could be tracked from log files which normally are available in SOA audit files.

Another work by [14] discusses the possibility of deploying and reusing composite services based on service dependency. In their research, the composite service is described in terms of elementary service dependency extracted from a pre-existing process model. And the invocation of the composite services is done by managing these dependencies. At a composite service level, dependencies between services arise from the connection among component services and constraints on their interaction, such as input/output, temporal and functional dependencies. These relationships involve data and control flow. Determining the data and control flow from dependencies is actually equivalent to the creation of process model. A thorough investigation of works for process model creation in service composition shows that all methods try to extract implicit or explicit dependencies. For example, in graph-based and chaining algorithms for service composition input/output relationships between services are explicitly searched. In workflow-based techniques the programmer identifies implicit and explicit dependencies manually. In case of AI-based methods input/output, temporal and some other logical relationships are considered by using domain knowledge.

Therefore, in this paper, we try to establish an automatic process model creation method based on web service dependency and with the support of statically available semantic knowledge about services and semantic description of request, with intent of reaching dynamic composition. Moreover, the proposed method takes into account the establishment of runtime adaptability techniques after deployment based on service dependency.

1.3 Problem Description

As it is described above there are many issues and knowledge gaps that hinder the transition towards achieving dynamic composition. Specifically, lack of runtime process model creation techniques complicates the intent for (semi)dynamic service composition. In earlier researches the usage of service dependency in service composition was not significant. Therefore, there is a clear research need to get more refined means to understand, identify, represent, analyze, and use service dependency for automatic process model creation. Consequently, developing a framework for automatic process model creation will be the main research problem. Moreover, we consider developing runtime process adaptability techniques, which could be process model modification or re-generation, based on kinds of failure during service binding.

2 Proposed Approach

Composite service structure description holds information about service components and how they interconnect. Both structural and behavioral relationships among component services of a composite service can be determined by extracting their dependency. The dependency extraction would be supported by the semantic description of services. This section describes the overview of the proposed architecture followed by detailed mechanism description using example scenario. A travel scenario is used to illustrate the proposed methodology. Let

us say an imaginary user wants to travel to city A, stay there for 3 days and rent a car during the stay. Moreover, for one free afternoon he wants either to take a boat tour or watch a movie depending on weather condition. Figure 1 shows the proposed two layered architecture. In the architecture, it is assumed that a formal user request description is available.

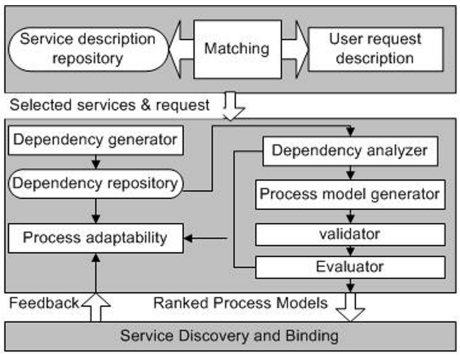

Fig. 1. Architecture

Although a user-request is in the form of natural language there are natural language processing techniques that parse a request and convert it into a formal description. This formal user request can be formulated as web service with semantic description. The semantic description can be found from the user request constraints or as an additional component a query facility can be provided to receive more input from the user.

This description contains (IOPE) inputs, outputs, preconditions, effects, goals, and constraints as it is defined in OWL [9]. But in the future we plan to investigate a mechanism to incorporate an additional annotation that describe association attributes between different goals which can be found from constraints or query facility. Such descriptions can be used in service discovery, matching and extraction of service dependencies and help in limiting the search space during match making process. Therefore, the IOPE matching technique that will be used during dependency extraction will be supported by the additional semantic annotations incorporated inside the user request description. In the architecture the first layer consists of two data repositories and a service matching module. The first data repository contains a formal user request description as it is explained above. The second data repository contains a statically available list of semantic service descriptions. Although service descriptions have the same format as user request, for dynamic composition that we wish to achieve, in service descriptions semantic annotation of service dependencies are not expected to be included. This is because one, it requires prior knowledge about composition requirements of services at design time and this limits flexibility. Two, a service could have a varying dependencies for different composition requests. This means achieving a comprehensive prior knowledge is unlikely. Therefore, incorporating semantic dependency description in service description is unpractical.

The matching module outputs a list of service descriptions that are required for the composite service. This module is based on semantically enabled matchmaking techniques that have the capability to find services required for composition from service descriptions and a formal user request description. For the above travel scenario these 7 required services could be discovered: WS1(Flight booking), WS2(weather forecast), WS3 (boat tour booking), WS4(Movie ticket

purchase),WS5(Hotel reservation), WS6(Car rental), WS7(Payment). The second layer consists of one data repository and six modules which are responsible for creating the process model using the output of the first layer. The dependency repository contains services dependencies occurring during composition. The role of each module is described as follows:

1. Service dependency generator: this module extracts service dependencies upon receiving a formal description of a user request and a list of semantic service descriptions required for the composition. The dependency extraction is based on two inputs received from layer 1; these are semantic description of services and user request with additional annotation that enables the dependency generator to extract different dependencies. Specifically, we believe the semantic description of requests with additional annotation minimizes the search space of dependency generator. Then, the extracted dependencies will be represented in an appropriate data structure to be stored in the dependency repository and will be ready for further use or re-use when needed. For the travel scenario the booking boat tour or buy movie ticket services are dependent on weather forecast, which can be identified from constraints specified by user and part of user request description. And other dependencies among services and between services and user request can be found by using IOPE matching with support of semantic user description.
2. Dependency analyzer: this module takes the service dependencies stored by the dependency generator as input and it analyses the dependencies to put them in understandable and interpretable format. The analyzer has a key role in processing and converting raw dependency data into data that is more applicable. Anticipated application areas are alternative process model generation, development of process adaptability or composite service management. For example one way of analysis can be counting the number of services dependent on it or the dependency between a service and user request which possibly provide the priority level of a service. For the travel scenario WS1 and WS5 have higher execution priority because they take input directly from user request. Moreover, other services are dependent on them.
3. Process Model generator: in this architecture the dependency analyzer is assumed to work interactively and iteratively when it is necessary to provide enough processed data for the algorithm running in the process generator. There can be sequential, alternative, concurrent or iterative coordination mechanisms to form the process model. Thus, by taking the analyzer output the process model generator will further interpret and associate it with any of the coordination mechanisms. By doing so the process model generator creates all possible process models for the intended composite service. For the travel scenario this sequential process model can be reached by simply sorting based on number of services a particular service dependent on.WS1⇒WS5⇒WS6⇒WS2⇒WS3⇒WS4
4. Validator: this is responsible for checking the correctness of the generated process model(s) based on extent of user request satisfaction. When the generated process model does not fully or partially satisfy the user request then it should be excluded. And another process model can be generated if necessary.

5. Evaluator: by taking valid process models from the validator this module evaluates and rank them using selected non-functional properties.
6. Corrector (Process adaptability): when there is a failure this module receives feedback from the service discovery and binding phase. Here process model regeneration or modification, based on the stored dependencies, will take place. It may be necessary to update the dependency matrix when a described service lost which may intern be needed to update the service description repository.

3 Conclusions

Dependencies reflect the potential for one service to affect or be affected by the elements of other services that compose the application. Analysis and tracking of dependencies is important in SOA management. However, there is limited work done towards the usage of service dependency in automatic service composition and for development of process adaptability techniques. In this paper we argue that semantic description of web services and user request enables detection of dependencies between services. And this allows automatic creation of composite services. We believe that the proposed architecture will allow seeing service composition as service dependency identification and analysis problem. This opens ways for developing more flexible and scalable applications from smaller and semantically described services. Currently we are working on implementation of the architecture making use of case studies to test its applicability. For future work, it is necessary to implement all components of the architecture to validate and prove the proposed concepts.

References

1. Matthias, F., Ivo, J.G., Neil, P.T., Edmundo, R.M.: Challenges and Tech.on the Road to Dyn. Compose Web Services. In: ICWE, pp. 40–47. ACM, California (2006)
2. Ramasamy, V.: Syntactical & Semantical Web Services Discovery & Composition. In: Proc. of the 8th IEEE Int. Con. On E-Commerce Tech. & 3rd IEEE Int. Con. on Enterprise Comp. E-Commerce and E-Services, p. 68. IEEE Press, California (2006)
3. Ponnekanti, S.R., Fox, A.: SWORD: A developer toolkit for web service composition. In: Proc. of the 11th Int. WWW Conf., Hawaii (2002)
4. Ulrich, K., Micro, S., Birgitta, K.: A Classification of Issues and Approaches in Automatic Service Composition. Int. Work. on Engineering Service Compositions (2005)
5. Web Services Description Language (WSDL), `http://www.w3.org/TR/wsdl`
6. OWL-S: Semantic Markup for Web Services, `http://www.w3.org/submission/owl-s`
7. Semantic Annotation for WSDL&XML, `http://www.w3.org/TR/sawsdl\&WSDL-S`
8. Web Service Semantics-WSDL-S, `http://www.w3.org/submission/WSDL-S`
9. OWL Web Ontology Language Guide, `http://www.w3.org/TR/owl-guide/`
10. Verma, K., Gomadam, K., Sheth, A.P., Miller, J.A., Wu, Z.: The meteor-s approach for conf. &executing dyn. web processes. Technical report, LSDIS LAB (2005)

11. Meyer, H., Kuropka, D., Tröger, P.: ASG - Techniques of Adaptivity. In: Proceedings of the Dagstuhl Seminar on Autonomous and Adaptive Web Services (2007)
12. Bixin, L.: Managing Dependencies in CBS Based on Matrix Model. In: Proc. of Net Object Days (2003)
13. Basu, S., Casati, F., Daniel, F.: Web Service Dependency Discovery Tool for SOA Management. In: IEEE Intern. Conf. on Services Computing, Utah (2007)
14. Zhou, J., Pakkala, D., Perala, J., Niemel, E., Riekki, J., Ylianttila, M.: Dependency-aware SOA & Service Composition. In: IEEE Int. Conf. on Web Services, Utah (2007)