INTENTION DRIVEN SERVICE COMPOSITION WITH PATTERNS

Emna Fki, Chantal Soulé Dupuy
Université de Toulouse, IRIT-UMR 5505, UT1, 2 rue du Doyen-Gabriel-Marty 31042 Toulouse Cedex 9, France

Saïd Tazi
CNRS, LAAS, 7 avenue du colonel Roche, F-31077 Toulouse, France
Université de Toulouse, UT1, UPS, INSA, INP, ISAE, LAAS, F-31077 Toulouse, France

Mohamed Jmaiel
Uiversity of Sfax, ReDCAD Research Unit, B.P. 1173, 3038 Sfax, Tunisia

Keywords: SOA, User intentions, Service patterns, Service composition.

Abstract: Service-oriented architecture (SOA) is an emerging approach for building systems based on interacting services. Services need to be discovered and composed in order to meet user needs. Most of the time, these needs correspond to some kind of intentions. Therefore these needs are not expressed in a technical way but in an intentional way. We note that available service descriptions and those of their composition are rather technical. Matching user needs to these services is not a simple issue. This paper defines an intention driven service description model. It introduces a service composition mechanism which provides the intention achievement of a user within a given context. We consider service patterns and we investigate how these patterns can be used in representing reusable generic services and in generating specific composite services.

1 INTRODUCTION

The service-oriented architecture (SOA) is a logical way of creating loosely coupled, interoperable services, via published and discoverable interfaces (Weerawarana et al., 2005). Services are described, published, discovered and can be assembled to create complex service based systems and applications (Papazoglou, 2008). The most used technology for SOA implementation is web services. These offer open and standardized interfaces, allowing the encapsulation and componentization of software and applications, thereby enabling relatively easy configuration or reconfiguration of software applications. The first generation of service specification standards such as WSDL inevitably focused on the description of the procedures that a service offers. A WSDL service specification is essentially a description of the signatures of the procedures offered by the service including the types of their parameters and return values. Thus, this kind of descriptions is low level and technical. At the same time, there is a need to map user's (high level) objectives to the objectives that services may help to provide. The user's objectives expression is intentional, but services objectives are expressed in operational manner. As a consequence, it seems necessary to bridge the gap between existing service definition and user's needs objective expression. Furthermore, services have to be composed in order to provide more complex and more useful solutions that meet user's objectives. In order to reduce the complexity of service composition, various approaches have been developed for automatic service composition. However, these approaches focused on composition aspects and have not taken into account user intention and related context.

In this paper, we propose an intention based composition mechanism to provide the composition schema that achieves the user's intention within a given context. We based our definition of the intention on the one proposed by (Kanso et al., 2008). So, the intention corresponds to a structured expression of user needs in terms of goals, means and reasons. We introduce service patterns as generic services de-

Fki E., Soulé Dupuy C., Tazi S. and Jmaiel M. (2010).
INTENTION DRIVEN SERVICE COMPOSITION WITH PATTERNS.
In *Proceedings of the 12th International Conference on Enterprise Information Systems - Information Systems Analysis and Specification*, pages 236-241
DOI: 10.5220/0002904102360241

scribed with goals. This allows users to express their intentions as an abstract high-level goal, and then let the system automatically satisfy this intention by assembling services, on-the-fly. Service pattern is created at design time and is permanently stored in the system. The instantiation of service patterns at composition run time enables personalization of the processes based on user intentions.

This paper is organized as follows: In Section 2, we present an overview of related work. Section 3 introduces the service pattern concept and we present our specification of service patterns. In Section 4, we propose a service composition mechanism for building tailored processes and we illustrate it by an example. Finally, we conclude by an assessment and future work.

2 RELATED WORK

Generally speaking, research on service description and composition is relevant for our work. It represents a very active research and development area. In this section, we focus on semantic web related works that present an interesting contribution in services description. We focus also on SOA and middleware approaches.

In order to cope with limitation of basic service descriptions, approaches relying on Semantic Web technology emerge to support service discovery and composition. OWL-S (Web Ontology Language for Web Services) (Martin et al., 2004) takes up the challenge of representing the functionalities of Web services. Most of the Web services specified with OWL-S are related to concrete Web services, and hence, the composition reasoning process has to deal with very large search spaces. In WSMO (Web Service Modeling Ontology) (De Bruijn et al., 2005), functional descriptions are called capabilities that are defined in terms of preconditions, assumptions, postconditions, and effects. WSDL-S (Akkiraju et al., 2005) only partially realizes the SWS (Semantic Web Services) approach, therewith it can be considered as a light-weight framework. However, it follows the tradition of extending existing technologies standardized by the W3C.

Most approaches address service discovery by semantic matchmaking of formally described requested and provided functionalities, i.e. OWL-S service profiles or WSMO capabilities as explained above. This allows clients to determine whether a Web service can solve the given request with respect to the preconditions and effects. However, these approaches lack methods of increasing the level of abstraction.

The authors in (Rolland et al., 2007) propose a move towards a description of services in terms of the designer intentions. They refer to these services as intentional services and present a model for intentional service modeling. They propose a methodology to determine intentional services that meet business goals. This work focuses on capturing service needs from exploring business goals. In our work, emphasis has been placed on service composition by adding rules and the use of context information in our composition approach. Indeed, from our point of view, intentions are specific to individual users. Hence we consider service patterns as generic and reusable components which are utilized to compose services based on specific user intentions.

Regarding service composition approaches, they can be categorized according to the research area they base their methods on: AI Planning introduces the principles of Artificial Intelligence into service composition like Situation calculus in, Hierarchical task networksand Finite state machines.

We can find another composition approach based on patterns. Patterns can be defined as sets of generic rules which can be used to make or generate a solution to a problem. The use of patterns in service composition is introduced in (Tut and Edmond, 2002). Analysis of service usage patterns can identify sets of services which are good candidates for the creation of new composite services tailored to user requirements. Authors do not give a formal description of the pattern, and do not show how instantiation is done. The use of patterns for the coordination of services from different participants from a workflow-based viewpoint is proposed in (Zirpins et al., 2004). We use service patterns in our work to facilitate the matching between user intention and existing services.

The creation of generic service patterns at design-time and their instantiation dynamically at run-time seem to be a promising approach for service composition activities. Instantiation of service patterns based on user needs leads to the creation of a specific composite service that in the end is transformed into an executable composite service (generally called executable process).

3 SERVICE PATTERNS

Within our work, we propose a model of service pattern. According to our point of view, a service pattern is a generic service which has generic and reusable descriptions.

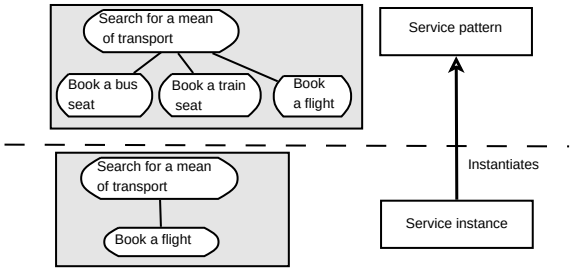

Figure 1: Service pattern and service instance.

3.1 Service Pattern Presentation

Service patterns are stored permanently in the system. Their creation and management reside mainly at design time.

A service instance is created by the instantiation of a service pattern with inputs representing contextual information (or user preferences). User preferences are a set of parameters that correspond to the choice made by the user in relation to the activity he will do. For example, in the case of traveling service, the user can give priority order to means of transport that can be used (plane, car, train, boat), or in relation to activities that he would make (riding, golf , visits).

Figure 1 illustrates the basic idea of service instantiation of the service "Plan a travel". The service pattern defines three means to accomplish the goal of "Searching a mean of transport". Based on user request and contextual information, this service is instantiated to generate the service "Book a flight".

The proposed service pattern is motivated by the following principal aspects:

- The need to facilitate the matching between user intentions and available services.
- The need of flexibility in composing services to satisfy particular needs.
- The importance of reusability of services.

3.2 Service Pattern Model

The diagram depicted in Figure 2 shows the model of the service pattern. In the following, we explain the key concepts that constitute the service pattern. The service pattern is composed of a goal, one or more activities, and some rules. A service pattern is associated with a service type that can be atomic, composite or conditional.

Goal. Our approach for designing service patterns is based on a "user" perspective. Users think also in terms of the problem in addition to the result that will be delivered by a solution. Hence, we have chosen to

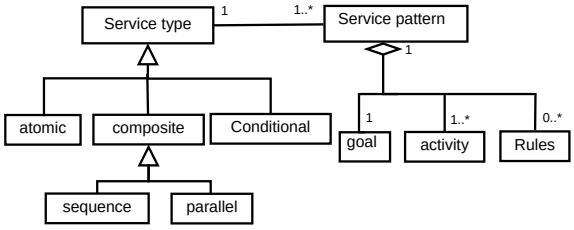

Figure 2: Service pattern model.

Goal : search for a mean of transport
Activities :
- Book a bus seat
- Book a train seat
- Book a flight

Service type: Conditional
Rules:

if user preference. mean of transport = train
then chosen activity= Book a train seat
else if distance (origin, destination) > 500 km
then activity= book a flight
Default activity= Book a bus seat

Figure 3: Service pattern "*search for a mean of transport*".

use the goal in the service pattern description. It emphasizes the problem that the service pattern solves. It corresponds to the service interface which will be used during discovery and selection of services to establish the correspondence between the intention expressed by the request and potential services allowing to realize it.

Rules. Rules are needed to drive the selection of alternative activities in the service pattern. Flexibility comes from the fact that the process of service composition is governed and guided by the rule mechanism (see Figure 3). Rules are set by a domain expert. They can be adapted after feedback and learning from service usage.

Activity. This part of the service pattern characterizes the manner to solve the problem. It specifies possible activities which can participate in the request satisfaction. The manner in which a service pattern is formed of activities is in relation with the service type.

Service Type. We distinguish three types of services: atomic, composite, and conditional. The atomic service realizes an elementary goal. It is not decomposable into sub goals. It can be directly implemented by a concrete Web service. A composite service corresponds to a complex goal. It is necessarily completed by a composition of several ser-

vices. In this case, the service pattern is constituted of more than one activity. Execution of these activities may be in sequence or in parallel. Each activity corresponds to another service pattern. And finally, a conditional service proposes several alternative activities to achieve a goal. Introduction of variability in service modeling is justified by the need to introduce flexibility in goal achievement and adaptability in service composition process. At composition time, one or several activities will be selected. Context information and rules in the service pattern are used to drive the selection of alternative activities.

4 COMPOSITION APPROACH

Figure 4 shows the proposed service composition approach. It is composed of two major steps: the composition schema generation and the execution plan generation. The composition schema is a high level service process model. It defines a flow of activities and their control flows without identifying the concrete services to be invoked. The execution plan is a realized composition schema where concrete, outsourced Web services are assigned to activities. This latter step will not be discussed in this paper.

The generation of the composition schema consists in composing a set of service patterns in order to satisfy a request. Service patterns are considered as process fragments, so they can be composed to build complex processes. Service patterns are stored in the service patterns base. They are pre-defined by a domain expert. His knowledge of the domain is primordial for the design of service patterns. This knowledge is structured in a domain ontology. This latter provides the definitions for important concepts and their relationships in a dedicated domain.

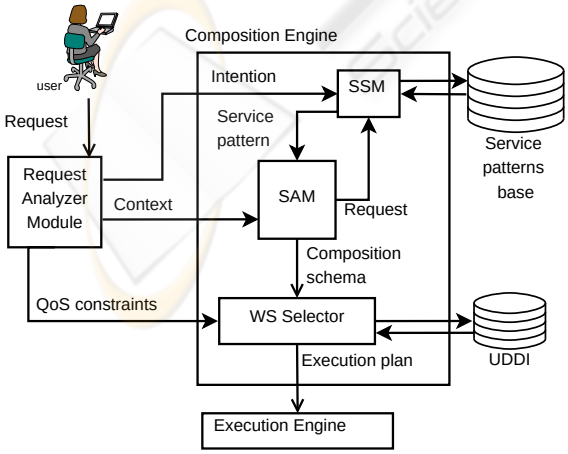

Figure 4: Composition mechanism.

4.1 Composition Approach Architecture

In what follows, we present our composition approach architecture and an algorithm (see Table1) that handles the generation of the composition schema.

Table 1: Generic algorithm for composition schema generation.

```
1 CS : Composition Schema
2 SP : service pattern
3 Algorithm :
4 CS=empty
5 search SP that corresponds to user goal
6 if SP is atomic then CS= activity of SP
7 else{
8 {
9  if SP is composite then CS=the set of
10    activities in SP
11  if SP is conditional then
12  {
13    use rules of SP to instantiate it
14    CS=selected activity
15  }
16  For each untreated activity in CS
17  {
18    consider activity as a goal
19    search SP that corresponds to this goal
20    if SP is composite then CS= CS with
21       activity substituted by the
22       set of activities in SP
23    if SP is conditional then
24    {
25      use rules of SP to instantiate it
26      CS=CS with activity substituted by
27      selected activity
28    }
29    if SP is atomic then mark activity as treated
30  }
31 }
```

The objective of service composition is to create specific processes by the combination of processes through existing service patterns.

The input of the composition schema process is constituted by two important aspects: (a) the intention that the user seeks to realize, and (b) the context information. We define a service user intention by these elements: the user's goal, means to achieve the goal and constraints. They will be used by the composition engine to create the composition schema that fulfills the requirements of the user as well as possible.

4.2 Composition Engine Description

The architecture depicted in Figure 4 shows a Request Analyzer Module that deals with the recognition and the extraction of the intention expressed by the request, and the related contextual information. In addition to the functional requirements, a service requester may have non-functional ones: QoS constraints. QoS are related to execution constraints (for example: execution time of a service). They can be also related to security. For example, a service requiring authentication may be requested by the user. These QoS constraints can be deduced by the request analyzer or by observing user behavior. User intention, context, and QoS requirements will serve as inputs to the composition engine. This latter contains three modules:

1. Service Search Module (SSM): this module deals with the search for service patterns in the service base. Seeking for a service is establishing coincidence between the goal the user intends to reach, and the goal expressed by the service pattern.
2. Service Assembler Module (SAM): this module assembles service patterns and builds the process that satisfies user intention. In case of conditional services, this module uses rules of the service pattern to decide which activity to select. It uses also means and constraints parts of the user intention to select appropriate activities.
3. The WS (Web Service) selector takes the composition schema, and user QoS constraints as inputs and generates an execution plan that satisfies the user's QoS requirements as the output. It assigns to every activity of the composition schema a concrete Web service from the UDDI registry.

Given a user request, the RAM extracts the intention expressed by the request, then it sends it to the SSM. It extracts also, from the request, eventual information that could help the SAM in instantiating services. So, this information constitutes an input of the SAM. The SSM deals with the search for service pattern in the base. It establishes a comparison between the goal expressed by the intention and the goal of the service pattern. The selected service will be returned to the SAM. This module will eventually use information provided by the RAM: means and constraints expressed by the intention. The SAM creates the service instance which will provide the process answering to the request. It builds a composition schema by substituting activities by corresponding service patterns. Three cases can arise:

i) The service pattern is composed of only one activity corresponding to an atomic service; in this case the SAM provides the process directly.

ii) The service pattern represents a composite service. All its activities must be analyzed in their turn. If one of these activities corresponds to a complex goal, the service patterns base is explored again in order to find the corresponding service pattern. To do this, the SAM calls again the SSM.

iii) The service pattern represents a conditional service; this type of service implies a choice between the proposed activities. This choice is carried out at composition run time based on rules provided by the service pattern and provided information. This allows adaptation to user intention. Services corresponding to selected activities which can be atomic, composite, or conditional, are treated by applying rules defined in i),ii) or iii).

It should be noted from the above description of the composition process that the composition mechanism is not treated as a one-shot activity. It is based on partial and recurrent instantiations. The process of composition is stopped when all activities obtained correspond to atomic services. The generated process corresponds to the composition of the activities of low level to satisfy the high level intention.

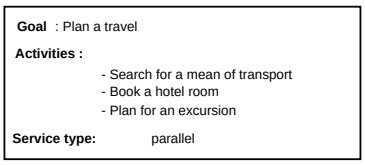

Figure 5: "Plan a travel" service.

4.3 Illustration

This section is dedicated to an illustration example that details the execution of the generic algorithm (see Table 1). We focus particularly on the two modules SSM and SAM (see Figure 4).

We consider the request of a user based in Toulouse (France) that plans to spend the next holidays in Andorra. This request is treated by the composition engine. The RAM extracts user intention and contextual information. The intention structure of the user is as follows:

- Goal: travel to Andorra
- Mean 1: take the train.
- Mean 2: lodge in a hotel.
- Constraint1: arrive as quickly as possible.
- Constraint2: do not spend much money.

The SSM contacts the service base to find a service pattern matching the user goal (see Table 1 line 5). It returns the service pattern named "Plan a

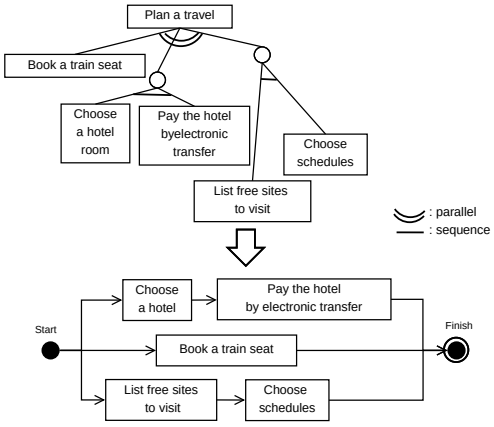

Figure 6: Generation of the composition schema.

travel". This service pattern is composed of three activities shown in Figure 5.

The SAM asks for research of service patterns corresponding to "Search for a mean of transport" and "Book a hotel", and so on (see Table 1 line 18 and 19) until all activities correspond to atomic patterns.

For the service pattern "Search for a mean of transport" (see Figure 3), the SAM needs to use the rule part because this pattern is conditional (see Table 1 lines 23, 24, 25, 26, 27 and 28). The result here is the service pattern "Book a train seat". The service patterns "Book a hotel room" and "Plan for an excursion" are composite service patterns. The SAM executes on them the instructions in lines 20, 21 and 22 of the algorithm in Table 1.

All obtained service patterns are atomic except "Pay the hotel" and "List the sites to visit" which are conditional service patterns. The SAM executes the instructions in lines 23, 24, 25, 26, 27 and 28.

Based on Constraint2 of the user intention, the SAM chooses the activity "List free sites" of the conditional service pattern "List the sites to visit" using the corresponding rules.

The SAM generates the composition schema, in which, activities correspond to obtained atomic service patterns (see Figure 6). In this process we have three branches in parallel. This is due to the organization of the activities in the composite service pattern "Plan a travel". Indeed, the control flow of the generated process is deduced from service patterns.

5 CONCLUSIONS

This paper proposed the use of service patterns for representing generic services. The proposed specification of these services is important for their reusability. In addition, the proposed specification facilitates the matching between user intentions and available services. This helps in efficiently finding relevant services. Besides, service patterns can be conditional, which allows the instantiation of generic processes in different situations according to users intention and context. We introduced a composition engine which generates on the fly a process allowing to achieve the user intention based on contextual information and user preferences.

Future research will focus on further refinements of the composition engine: how to acquire contextual information, and how can rules of service patterns evolve dynamically to adapt to different complex user intentions and contexts? We intend to study the correspondence between high level services studied in this work and the concrete Web services which allow physical implementation of user intention.

REFERENCES

Akkiraju, R., Farrell, J., Miller, J., Nagarajan, M., Schmidt, M.-T., Sheth, A., and Verma, K. (2005). Web service semantics - wsdl-s. W3C Member Submission.

De Bruijn, J., Bussler, C., Domingue, J., Fensel, D., Hepp, M., Keller, U., Kifer, M., König-Ries, B., Kopecky, J., Lara, R., Lausen, H., Oren, E., Polleres, A., Roman, D., Scicluna, J., and Stollberg, M. (2005). Web service modeling ontology (wsmo). W3C Member Submission.

Kanso, H., Elhore, A., Soul-Dupuy, C., and Tazi, S. (2008). Semi-Automatic Analyzer to Detect Authorial Intentions in Scientific Documents. *Journal of Computer Science*, 3(1):19–24. ISSN 1306-4428.

Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., et al. (2004). OWL-S: Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S/.

Papazoglou, M. P. (2008). *Web Services: Principles and Technology*. Pearson, Prentice Hall.

Rolland, C., Kaabi, R. S., and Kraïem, N. (2007). On isoa intentional services oriented architecture. In *CAiSE*, pages 158–172.

Tut, M. T. and Edmond, D. (2002). The use of patterns in service composition. In *CAiSE '02/ WES '02: Revised Papers from the International Workshop on Web Services, E-Business, and the Semantic Web*, pages 28–40, London, UK. Springer-Verlag.

Weerawarana, S., Curbera, F., Leymann, F., Storey, T., and Ferguson, D. F. (2005). *Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging and More*. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Zirpins, C., Lamersdorf, W., and Baier, T. (2004). Flexible coordination of service interaction patterns. In *ICSOC '04: Proceedings of the 2nd international conference on Service oriented computing*, pages 49–56, New York, NY, USA. ACM Press.