Context-aware Services Composition based on AI Planning

Lirong Qiu[1,2] Zhongzhi Shi[1]

[1] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, P.O. Box 2704-28, Beijing 100080 China
[2] Graduate School of the Chinese Academy of Sciences, Beijing, China
qiulr@ics.ict.ac.cn

Abstract. AI planning technologies has proven to be useful for services composition. By treating service as an action, planners do various sorts of reasoning about how to combine services into a plan. However, planners typically support only limited reasoning capabilities which cannot handle the enormous size of the data involved in the planning process over Web. In parallel, the field of context-aware computing has been focusing on providing information that can be used to characterize the situation of an entity, and thereby using context can filter the inappropriate candidate services and adapt to user's preference. The major technical contributions of this paper are: (1) We propose an OWL-SC model for the context-aware composition of Web services.(2) We propose context-aware plan architecture and thus is more scalability and flexibility for the planning process, and thereby improving the efficiency and precision. (3) We propose a hybrid approach to build a plan corresponding to a context-aware service composition, based on global planning and local optimization, considering both the usability and adoption. solution.
Key Words: Semantic Web Service, AI planning, Context-awareness, OWL-S

1 Introduction

A Web service is a set of related functionalities identified by URIs that can be advertised, located, and triggered by other applications and humans through the Web [1]. Currently, human beings perform Web service composition by reading information provided on service's Web pages. With the ever increasing number of Web services being made available on the Web, it is already beyond the human ability to analysis them and generate the composition process manually. This has triggered an active area of research and development on Web service composition [2].

By describing a Web service as an action, which is specified by its precondition and effects, many researchers propose their composition approach based on AI planning techniques. However planning is a costly computational approach and the size of the data involved in the planning process over Web will be much

bigger than the ones encountered in classical planning problems. The number of component services selected in the composite service maybe large, and the number of Web services being considered from which these component services are selected is likely to be even larger. Therefore, the performance and applicability to this complex composition problem of planners are still being debated.

While context information, which is the important element to be considered during selecting and combining services, can increase the effectiveness and acceptance of composition. A service composition model that can integrate, and make use of context information to derive the optimal component services of the composite service is still an ongoing research problem.

In this perspective, utilizing a means of context aware, AI planning method to services composition is the motivation and central foundation of our work. The remainder of this paper is structured as follows. Section 2 provides extensions to OWL-S as regards support for describing context information. In Section 3 we present our system architecture and we test our approach on a simple, but realistic example. The related work and conclusions will be given in Section 4.

2 Extending OWL-S with Context Elements

OWL-S is a set of OWL ontologies supporting the rich description of Web services, thus facilitating the automation of service composition. OWL-S comprises three interrelated subontologies, known as the profile, process model, and grounding. In short, Service Profile ontology describes capabilities of Web services by specifying the input and output types, preconditions and effects, namely IOPE. The Process Model describes how the service works. The Service Grounding specifies the information necessary for service invocation and execution. The OWL-S has an extensible service parameter mechanism which allows the additional description of non-IOPE attributes of services. Therefore, we can extend the OWL-S with context attributes.

There are various categories of context knowledge that are pertinent to both the user and the service. Due to evolving nature of services, completely formalizing all context information is likely to be an in-surmountable task. To ascertain the quality of the composition, three types of context are defined to track the user, the service, and the environment. Figure 1 shows the OWL-S based context model for U-Context, E-Context and W-Context.

U-Context: A user's context should leverage knowledge about who the user is, where the user is and what the user is doing. Then, we divide the information into two types of contexts: User static context: specified in a user's profile and describes information interests and preferences. User dynamic context: a user's location, current activity and task.

W-Context encompasses a number of non-functional properties of services, such as execution price, execution duration, availability, reliability, and reputation. In Short, W-Context is much like the QoS model. Many of the ideas of QoS modeling proposed to track the service context could be integrated into our approach.

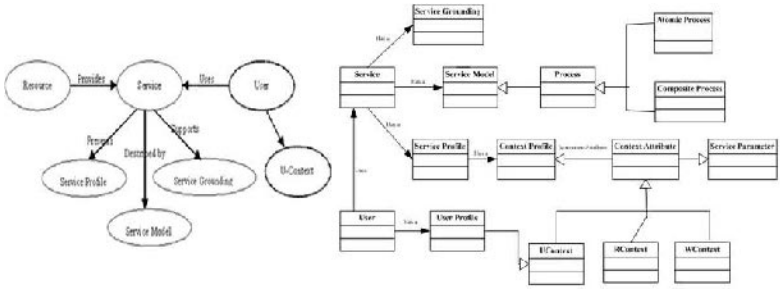


Fig. 1. OWL-SC:Adding Context to Service Description

E-Context comprises the information about where the user is, such as the weather, the date, and some kind of surrounding situation.

3 Context-based AI planning for Service composition

3.1 System Architecture

The diagram shown in Figure 2 gives an overview of the pivotal components of the system. *Context Proxy* retrieves context information and generates context

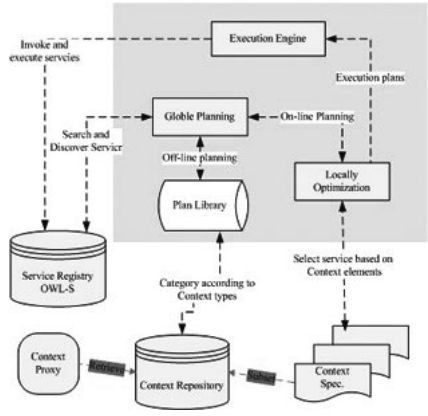


Fig. 2. System Architecture

conditions that must hold for this composition request. *Context Repository* is a component that stores the explicit context representations of users, services and environment based on the definition of OWL-SC ontology. How to sense the current context and deliver it to the application is also a hot research domain. Typical techniques are used in many context-aware applications, which is not

the point of this paper. *Context Spec.* stored predefined rules with associated context conditions, which is the judging rules for filtering actions considering the current states and situation.

The core of the composition framework is the planner, takes user's request, and uses the domain definition, available context information, to identify a sequence of actions that can be executed in order to achieve the desired request. In the *Plan Library*, there stored many classified plan schemas defined off-line by users or can be learned from the execution. The diagram represents the main motivation behind our method: global planning (search *plan library* and find a plan template) and local optimization (generate DAG dynamically and select an execution path).

It worth noting that Web services will be grouped according to different domains classified by context attributes. Thus, we can define different plan scenarios to represent typical domain cases. For each scenario, the pertinent context elements is specified which can deduce actions.

3.2 Global AI Planning Algorithm

A composite service is specified as a collection of generic service tasks described in terms of service ontologies and combined according to DL-based reasoning. In AI domain, researchers exploited AI planning techniques for automatic service composition by treating service composition as a planning problem, given a representation of services based on OWL-S.

Definition 1 (Planning Domain) *A planning domain corresponds to a particular domain to be planned, provided with states, actions, and functions. Each Planning domain D is described as a tuple:*

$$D = (S, A, C, I, \Gamma, \Upsilon)$$

(1) S is the set of states.
(2) A is the set of actions(services).
(3) C is the set of context elements.
(4) I is the set of initial states and $I \subseteq S$;
(5) Γ is the transitional function, $S \times A \rightarrow 2^s$, which associates to each current state $s \in S$ and to each action $a \in A$ the set $\Gamma(s,a) \in S$ of next states;
(6) Υ is the context function, $S \times C \rightarrow A$, which associates to a state $s \in S$ and a context $c \in C$, an action $a \in A$ to be executed, $\Upsilon(s, c) \in A$.

A state can be represented as a conjunction of positive literals. For example, $At(Beijing, Zhang) \wedge Destination(Shanghai)$.

An action $a \in A$ is defined as $\langle Pre, Eff \rangle$, $Pre \in S$ defines the precondition must hold before the action can be executed, and ensue when it execute; effects $Eff \in S$ defines the execution outcome.

There are three kinds of context elements in C defined in the above section. User may define his preference in advance. For example, if the user wants to

find a restaurant to eat, and he emphasizes to save time. Then if current state is $Hungry(Zhang)$ and current context is $At(Yayuncun, Zhang)$, then the action finding a restaurants in the vicinity of $Yayuncun$ is selected.

A plan task in a planning domain D is defined as a three-tuple $T = (S_1, S_n, D)$, where $S_1 \in I$ denotes the initial states of the planner; S_n denotes the goal state the planning system attempted to reach; D is a domain description.

A plan in a planning domain D for a planning problem $T = (S_1, S_n, D)$ is defined as a sequence of actions (or services) that will achieve S_n from S_1 in D. In brief, by taking $T = (S_1, S_n, D)$ as input, the planner will return a plan $P = (a_1, a_2, \ldots, a_n)$.

There are some plan schemas stored in *Plan Library* classified by different domains. Those plan schemas are represented in UML statechart [3], which have many advantages in composite service specification, such as possessing a formal semantics, easily integration into the Unified Modeling Language and suitable for expressing typical control-flow dependencies. To simplify the discussion, we initially assume all the startcharts involved in this paper are acyclic.

Definition 2 (Planning Process in Statechart) *The decomposition of a Planning Task $T = (S_1, S_n, D)$ of a statechart is a sequence of states $[S_1, S_2, \ldots, S_n]$, such that S_1 is the initial state, S_n is the final state, and for every state $S_i (1 < i < n)$:*
(1) S_i is a direct successor of one of the states in $[S_1, \ldots, S_{i-1}]$ and is not a direct successor of any of the states in $[S_{i-1}, \ldots, S_n]$.
(2) $\nexists S_j \in [S_1, \ldots, S_{i-1}]$ where S_j and S_i belong to OR-states of the statechart.
(3) if an initial state of one of the concurrent regions of an AND-state execute, then all the other branches of this AND-state are executed.

Many planning methods already exist and we choose HTN planning method as our global planning algorithm to build statechart. The key to the global planning algorithm is the construction of a *plan library*. This process is a kind of off-line planning, thus can alleviate the issue of performance efficiency.

3.3 Local Optimization based on Context Configuration

If a statechart contains OR-states, it has multiple paths from the initial state to the final state. Each path represents a different plan to complete a compound service execution. Hence, it is possible to execute different path of a statechart by allocating different Web service to the basic states in the path. How to structure and maintain execution paths play a key role in supporting efficient planning.

A directed acyclic graph (DAG) can be used to represent a set of actions, which gives an allowable (total) order for carrying out the basic actions one at a time.

Therefore in this paper, we using Directed Acyclic Graph (DAG) to represent an execution plan. DAG, $G = G(V, E)$, where $V = \{v_1, \cdots, v_n\}$ is a set of actions and E is a set of weighted directed edges (arcs) identify the dependencies. We list the detailed definition of the weighted DAG as follows:

Definition 3 (DAG representation of Task Accomplishment Path) *Given a task decomposition statechart* $[S_1, S_2, \ldots, S_n]$*, the DAG* $G = G(V, E)$*, representation of the plan is a graph based as follows:*
(1)The DAG has at least two nodes Start *and* Finish.
(2)$a_i = (a_{i1}, a_{i2}, \ldots, a_{it})$ *is the action set for state* S_i *in ST and* $Eff(a_i) \supseteq S_i$
(3)The DAG has one node for each action $(a_1, a_2, \ldots, a_n)$.
(4) The DAG contains an edge from action a_i *to action* a_j *iff* S_j *is a direct successor of* S_i.
(5) The edge from a_i *to* a_j *is weighed by the context elements* $c \in C$ *and* $\Upsilon(S_i, c) \rightarrow a_i$.

To note that there are some edges have no weighed attributes in the DAG, which demonstrate that the action a_i can be executed in state S_i no matter what current context is. The empty plan contains just the *Start* and *Finish* actions.

The local optimization process can be guided by following methods: (1) domain control knowledge; (2) Utility function; (3) User input.

3.4 Case Study

To illustrate the application domain and to derive requirements, we describe a particular example happened in Olympic Games 2008 Scenario.

At the Olympics, different user groups such as athletes, coaching teams, organizers, spectators, journalists and other groups of people from all over the world need a broad range of information services and transactional services. These users may have different purposes range form network stored information retrieval to user shopping and communication.

A foreigner, who has never been to China before, decides to attend the Olympic Games in Beijing and to combine it with some sightseeing. He registered in the Digital Olympic Services via his mobile phone and put forward his preference and the personal information demand. Context information such as the actual position of the user, the actual time can be stored using our method.

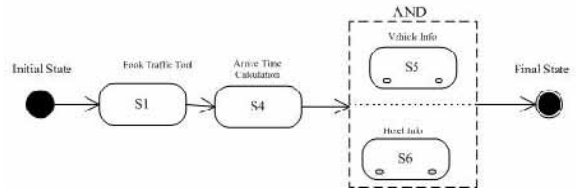
The detailed scenario is as follows: He is in Beijing China and wants to watch a game in Qingdao. He requires a service of "Travel Plan". In his preference, he defined his transportation preference is train if it's rain. The composite request can be achieved using our method presented above.

Firstly, according to users preference, update the *context repository* and *context spec*. A rule $R1 : (S, C) \rightarrow a_i$ is stored in the context function set Υ, where S is BookTraffic, C is the weather information,
$(BookTraffic, Sunny) \rightarrow AirlineBooking$,
$(BookTraffic, Cloudy) \rightarrow TrainBooking$

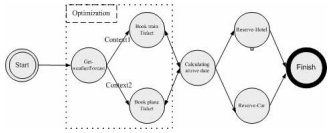
Then Planner will search *Plan Library* to find the task statechart using global planning algorithm. This gives us a consistent plan, represented in UML statechart as shown in Figure 3(a).

If there are four actions listed as follows:
$a_{21} : China - Airline - Booking$; $a_{22} : Japan - Asia - Airline - Booking$
$a_{23} : China - Train - Booking$; $a_{24} : Japan - Asia - Ship - Booking$

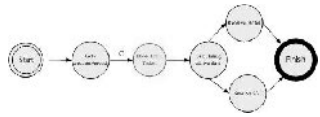
According to the location context, service a_{21}, a_{23} is pick out, and according to the rules in Υ, if acquired current weather is sunny, then a_{21} is selected, else a_{23} is selected.



(a) task statechart for attending a conference in plan library



(b) Optimizing process



(c) Final execution path in DAG

Fig. 3. the Planning process of "Participating Conference" Example

3.5 Implementation

In short, the overall process consists of three steps: Firstly, Gets the user's request and context information. Secondly, the planner searches *Plan Library* to find the statecharts template to determine subgoals. Then, selects the applicable path according to context elements and generates DAG execution path.

We developed a prototype OntoService used to validate the feasibility and benefits of the proposed approaches. Currently, the system OntoService has been implemented as a platform that provides tools for: (1) defining plan scenarios and represent in UML statechart; (2) building service ontologies and checking its consistency; (3)demonstrating DAG execution path graphically;(4) modeling and reasoning context information based on RDQL.

In addition, we are developing a website applied for Olympic Games 2008 based on OntoService. We have developed application scenarios for different user groups, e.g. a visitor or a journalist, to identify typical use cases and to derive requirements on platform functionality and services to be integrated. And according to different user groups, we put forward different user profile template.

For user context modeling we envisage the approach that is still under development. At the time being we restrict ourselves to the use of time and location information that can be automatically gathered and of a manually maintained information about a user's mode of activity.

4 Related Work and Conclusion

Various efforts have been done to apply AI planning to services composition. Nahrstedt [4] proposed global planning algorithms for dynamic composition. Zeng [5] provided a QoS-driven service composition model. In this work, the authors propose statecharts and DAG to represent the execution plans and execution paths. The issues of context-aware service composition have been widely discussed in pervasive and mobile computing domain [6]. Sonia Ben Mokhtar [7] presents an approach for context-aware service composition based on workflow integration in pervasive computing environments.

Despite the relatively large related work in service composition domain and context computing domain, few efforts have specifically addressed the topic of Context-based service composition using AI planning technology.

In this paper, we propose a framework for composing context-aware Web services. The work of this paper is a part of our ongoing research work. Future work includes extending learning plan scenarios during planning process based on user's feedback and considering context modeling and context reasoning based on Description Logics.

Acknowledgements

Our work is supported by the National Science Foundation of China (No.60435010), the national 973 Project (No.2003CB317004) and the Nature Science Foundation of Beijing (No.4052025).

References

1. G. Alonso, F. Casati, H. Kuno, and V.Machiraju. *Web Services*. Springer-Verlag, 2003.
2. A. Sheth and J.Miller. Web services: technical evolution yet practical revolution. *IEEE Intelligent systems*, 18(1):78–80, 2003.
3. D. Harel and A. Naamad. The statemate semantics of statecharts. *ACM transactions on Software Engineering and Methodology*, 5(4):293–333, 1996.
4. K. Nahrstedt, D. Xu, D. Wichadakul, and B.Li. Qos-aware middleware for ubiquitous and heterogeneous environments. *IEEE Communications magazine*, 39(11):2–10, 2001.
5. Zeng Liang-Zhao, Benatallah B., Anne H.H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. Qos-aware middleware for web services composition. *IEEE Tranctions on Software Engineering*, 30(5):311–327, 2003.
6. Ariel Pashtan. *Mobile Web Services*, chapter Ontology of mobile user context. Combridge unversity press, 2005.
7. Sonia Ben Mokhtar, Nikolaos Georgantas, and Valeriae Issarny. Ad hoc composition of user tasks in pervasive computing environments. In *Proceedings of the 4th workshop on software composition*, 2005 LNCS3628.