An AI Planning Based Approach for Automated Web Services Composition

Xingwei Liu Hui Zhao

School of Mathematics and Computer Engineering, Xihua University, Chengdu 610039, P. R. China

Abstract

In a dynamic Web services composition, the most of the planning approaches existed are usually centralized, whereas, the underlying layer services are distributed. Therefore, an approach for automated composition of semantic Web services based on AI planning is proposed. The HA is designed to make the first level planning and responsible to decompose the complex task into series of simple subtasks. Then the SAWU immediately makes the second level planning and deliveries the subtask planning results back to the HA through the agent. Finally, the elementary experimental results indicate that this approach is superior in efficiency.

Keywords: Artificial intelligence, Agent, Semantic Web services composition, Ontology

1. Introduction

Recently, so-called service driven architectures promise a new paradigm providing an extremely flexible approach for building complex information systems [1]. Generally, services can be composed in two modes: static and dynamic mode. In a static composition, the services to be composed are chosen at the time of designing [2]. While in a dynamic composition, the services to be composed are chosen at run-time [3]. Up to now, several leading methods of dynamic Web services composition have been proposed [4]-[5]. Nevertheless, the most of the planning approaches existed are usually centralized [6]-[7], whereas, the underlying layer services are distributed and autonomous. Hence, the integration of a large number of Web services requires more scalable and flexible techniques [8]-[9], and in this paper, we also propose an approach for automated composition of semantic Web services based on a formal, multilevel distributed framework, which is named as AC4SWS.

The remainder of this paper is organized as follows: section 2 gives an overview of the AC4SWS. Section 3 introduces the key technologies of the AC4SWS. In the next section, some metrics are introduced to evaluate the performance of the AC4SWS. Finally, the last section summaries the entire paper and addresses the future work.

2. The architecture of AC4SWS

In AC4SWS, there are two types of components: head agent (HA) and subagent with a UDDI-based registry (SAWU), which is shown in Figure 1. Given that the communication agent receives the message from the requester, it immediately sends the message to the central controller. If the central controller receives the message, the SHOP2 planner is invoked to make the first level planning and is responsible to decompose the complex task into a series of simple subtasks [10]-[11]. Then, after forwarding to the first subtask to the dispatcher, the central controller keeps listening the agent and waiting for the SAWU's response. If the response is correct, the central controller deliveries the next one immediately. Otherwise, the exception handler is invoked. But now it is a pity that the process of the SHOP2 creator is still manual.

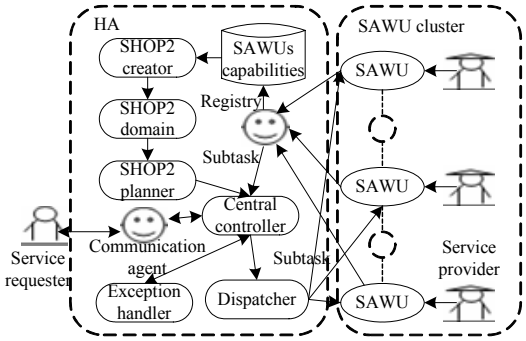

Fig. 1: The architecture of AC4SWS.

In Figure 1, there are many SAWUs, which form the SAWU's cluster and disperse in P2P environment. After the providers have published the semantic descriptor of its Web services to the appropriate SAWU, the abstract service ontology (it represents a subtask or a resolution of one special domain task) can be created, and as the SAWU's capability can be

registered in the HA. At the same time, after the SAWU received a subtask planning request, it immediately makes the second level planning and deliveries the subtask planning results back to the HA through the agent.

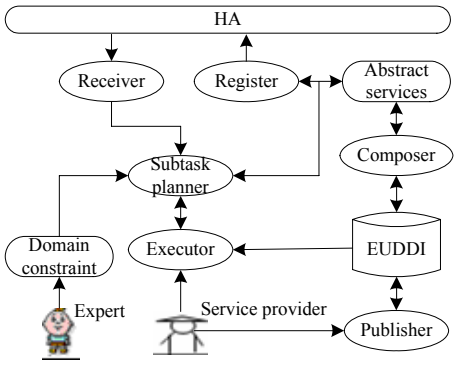

Fig. 2: The structure of the SAWU.

Figure 2 shows the structure of the SAWU, which is composed of six active components and three ontology components. One of the active components is the receiver with responsibility for delivering the subtask planning request came from the HA to the subtask planner of the SAWU. Then the subtask planner immediately selects an appropriate abstract service and domain constraint provided by domain experts, which all are translated into golog language to finally perform the subtask planning [12]-[14]. In the proceeding of the planning, the subtask planner may invoke some Web services through the executor which often needs to acquire the detail information from the EUDDI about the invoked Web services. At the same time, the service providers may automatically publish the Web services to the EUDDI through the publisher. In Figure 2, the EUDDI is a semantics-extended UDDI registry, which supports both of OWL-S and WSDL. And each EUDDI or SAWU can store only one special domain service description information which is used to compose the abstract service ontology. At last, as the capability description of the SAWU, the abstract service ontology is registered in the HA through the agent.

3. The SAWU's capability

The SAWU's capability is an abstract service with the domain constraint, and viewed as one action in HA system, or as an operator in SHOP2 system, which describes the profile of the subtask can be finished of the Web services published in the SAWU. Therefore, it is very import to exactly describe the capability of the SAWU.

The OWL-S supplies Web services providers with a core set of markup language which constructs for describing the properties and capabilities of their Web services [7]. It includes three components: ServiceProfile, ServiceModel and ServiceGrounding. Generally speaking, the ServiceProfile provides the information needed for an agent to discover a service, while the ServiceModel and ServiceGrounding together provide enough information for an agent to bind a service. The OWL-S can be used to describe the abstract service, but cannot perfectly express the domain constraint and distinguish which description of the SAWU's capability [15]. Therefore, in this paper the RuleDC (Rule for Domain Constraint) , as a semantics Web rule markup language, is designed for describing the domain restrictions [12]. Moreover, it is used to extend the standard OWL-S, named as OWL-S4SAWU, and solve the aforementioned problem.

Generally, a domain constraint can be defined as DS=<T, F, R>, here T is an axiom lists defined by OWL, and F is a set of facts. Each fact is expressed as $p(\bar{a})$, here p is a predicate and $\bar{a}$ is a constant tuple. R is a set of rules, and each rule can be expressed as $p1(X1) \wedge \ldots \wedge pn(Xn) \Rightarrow Q(Y)$, here Xi and Y respectively denote two variable or two constant tuple [12]. The Atoms is an atom lists, and each atom has a name and some domain constraints, which includes two properties: the predicate and the variable tuple.

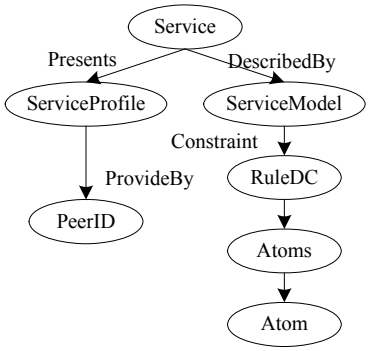

Fig. 3: The top level ontology of OWL-S4SAWU.

In order to describe the SAWU's capability, this paper provides the OWL-S4SAWU ontology, which is shown in Figure 3. Each abstract service belongs to a SAWU. When the HA dispatches the subtask to the SAWU, it needs the peerID. Therefore a new class peerID is extended in the ServiceProfile. The abstract service is always a composite Process, including IOPE and some domain constraints. And the abstract service cannot be executed directly, so the ServiceModel is hold and extended. Whereas, the ServiceGrounding is removed, because it is not used in this paper any more.

4. Experimental results

Generally, the criteria for evaluating composting Web services are: execution price, execution duration, reputation, reliability an availability of Web services [16]. In this paper, we mainly discuss execution duration of this system. The execution duration $Q_{du}(p)$ is equal to the total time of executing a planning in this system, and it is measured as delay between the moment when a request is sent and the moment when the results are received. Supposing the scale of the complex task is N, and the complex task may be divided into M subtasks which scale is n_i. Obviously

$$\sum_{i=1}^{M} n_i = N \tag{1}$$

The subtask´s execution duration is:

$$q_{golog}(n_i) = T_{plan}(n_i) + T_{trans}(n_i) \tag{2}$$

Here, $T_{plan}(n_i)$ is the consuming time of planning and $T_{trans}(n_i)$ is used to transfer information between HA and SAWU. Therefore, the total time is:

$$Q_{du}(p) = q_{jshop2}(M) + \sum_{i=1}^{M}(T_{plan}(n_i) + T_{trans}(n_i)) \tag{3}$$

Here, $q_{du}(M)$ is the consuming time of the HTN planning in HA.

Here, we choose three tasks to experiment, their scales are respectively n1=5, n2=124, n3=1024. The Table 1 shows how much time consumed with different planning algorithm.

n_i	10	100	1000
$q_{jshop2}(n_i)$	0.05	0.181	8.53
$q_{golog}(n_i)$	0.132	0.220	4.423

Table 1: Experiment results with different planning algorithm.

We select the third task, and suppose that it exists three different kinds of partition method: one portion, eight portions and one hundred portions. The detail is shown in Table 2.

M	1	10	100
$q_{jshop2}(n_i)$	0.02	0.05	0.181
$q_{golog}(n_i)$	4.423	0.220	0.132
$Q_{du}(p)$	4.443	2.25	13.381

Table 2: Experiment results with different partition method.

Table 1 and Table 2 show that this system takes on high efficiency for complex, large scale task, and the domain constraints have large advantage for golog planning.

5. Conclusions, related and future work

In this paper, we propose an approach for automated composition of semantic Web services based on AI planning. The HA is designed to make the first level planning and is responsible to decompose the complex task into a series of simple subtasks. After the SAWU has received a subtask planning request, it immediately makes the second level planning and deliveries the subtask planning results back to the HA through the agent. Especially, we design a rule markup language, which is used to extend the OWL-S for describing domain restrictions. Finally, the elementary experimental results show the high efficiency of the approach, and the practical advantage of automated composition at the semantic level.

Different planning approaches based on AI planning have been proposed for the composition of Web services from HTNs to golog [16]-[17]. But how to find out all required Web services before composition in a centralized, massive server, and how to generate appropriate and executable service processes in time in these frameworks are still an open issue. In [4], the HTN planning system SHOP2 provides a sound and complete algorithm to translate OWL-S service description to a SHOP2 domain [7]. OWL-S processes are, like HTN, pre-defined description of actions to be carried out to make a certain task done. A system is also implemented which can plan over a set of OWL-S descriptions using SHOP2 and execute the resulting plans over the Web. Hence, it can deal with very large and complex planning problem effectively. However, it need to provide the planner with a special task explicitly, which is not always available in dynamic environment. In [5], the situation calculus and golog are presented to formalize the task and solution by Sheila McIlraith. Furthermore, an approach to building agent technology based on the notion of generic procedures and customizing user constraint, is proposed to adapt and extend the golog language to enable programs that are generic, customizable and usable in the context of the Web. In [6], it provides a formal, semantically justified account of how to plan with complex actions using operator-based planning techniques, it also has definition, characterization, and computation of precondition and condition effects for complex actions.

In the future, we will aim at a solution to generate the program for HTN and golog automatically. Especially, we want to solute the problem that each SAWU works independently and linearly. Finally, we intend to test our approach over realistic case studies in projects, including more criteria for evaluating composting Web services.

References

[1] S.A. McIlraith, T.C. Son and H. Zeng, Semantic web services, *IEEE Intelligent Systems Sp. Issue on The Semantic Web*, 16(2): 46-53, 2001.

[2] J. Koehler, B. Srivastava, Web service composition: current solutions and open problems, *Workshop on Planning for Web Services (ICAPS'03)*, pp. 28-35, 2003.

[3] D. Skogan, R. Gronmo and I. Solheim, Web service composition in UMLm, *The 8th International IEEE Enterprise Distributed Object Computing Conference (EDOC)*, Monterey, California, pp.32-36, 2004.

[4] E. Sirin, B. Parsia, D. Wu, J. Hendler and D. Nau, HTN planning for web service composition using SHOP2, *Journal of Web Semantics*, 1(4): 377-396, 2004.

[5] S. McIlraith, T.C. Son, Adapting Golog for programming the semantic web, *International Conference on Knowledge Representation and Reasoning*, pp.482-493, 2002,.

[6] S. Mcllraith, R. Fadel, Planning with complex actions, *The 9th International Workshop on Non-Monotonic Reasoning*, pp. 356-364, 2002,.

[7] The OWL Services Coalition, OWL-S: Semantic markup for web services, *Technical White paper (OWL-S version 1.0)*, 2003.

[8] Q.Z. Sheng, B. Benatallah and M. Dumas, SELF-SERV: A platform for rapid composition of web services in a peer-to-peer environment, *Proc of the 28th VLDB Conference (Demonstration Session)*, pp. 1051-1054, 2002,.

[9] R. Aggarwal, K. Verma, Dynamic web service composition in METEOR-S, *Services Computing, IEEE International Conference on (SCC'04)*, 23-30, 2004.

[10] J. Cao, M.L. Li, Composing web services based on agent and workflow, *GCC Part I*, *LNCS* 3032 pp. 948-955, 2003,.

[11] O. IIghami, Documentation for JSHOP2, *Techical Report CS-TR-4694*, 2005.

[12] S. Liang, Design and implementation of a semantic web rule markup language OWLRule+, *Jouranl of Computer Research and Development*, 41(7):1088–1096, 2004.

[13] R. Reiter, Knowledge in action: logical foundations for specifying and implementing dynamical systems, *The MIT Press*, 2001.

[14] H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin and R.B. Scherl, GOLOG: A logic programming language for dynamic domains, *Journal of Logic Programming*, 31(1-3): 59-83, 1997.

[15] M. Helmert, Complexity results for standard benchmark domains in planning, *Artificial Intelligence*, 143(2): 219-262, 2003.

[16] L. Zeng, B. Benatallah, M. Dumas, J. Kalagnanam and Q.Z. Sheng, Quality driven web service composition, *Proc. of the Twelfth International World Wide Web Conference (WWW'03)*, pp. 411-421, 2003.

[17] P. Traverso and M. Pistore, Automated composition of semantic web services into executable processes, *Proc. of International Semantic Web Conference*, pp. 7-11, 2004.