2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology

A framework to guarantee time-bounded composed services

Elena del Val, Martí Navarro, Vicente Julián and Miguel Rebollo
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022, Valencia, Spain
Email:{edelval, mnavarro, vinglada, mrebollo}@dsic.upv.es

Abstract

Time is an important non-functional parameter to consider in service compositions, especially in environments where a service must be provided before a deadline. This paper presents a framework that deals with service compositions taking into account the service execution time. To enhance this composition it is important to provide service execution times with reliability, taking into account the workload and availability of the service.

1. Introduction

Service Oriented Architectures (SOA) raises many challenging research issues, one of the most prominent being web service composition. Web service composition addresses the situation in which a client request cannot be satisfied by an available service, but by suitably combining "parts of" available services. With automatic service composition, a repository of available services is searched for, and compositions of services that meet some user defined criteria are found. Often the criteria indicate some input and output requeriments but there are proposals which also consider QoS parameters in order to get a more suitable composition.

One of the most important parameters related to QoS is service execution time. It is important to bear this in mind because some composed services are completely useless if they are not provided on time, mainly in enviroment with temporal bounded processes. Currently, the majority of the proposals [1][2][3][4] take time properties and constraints into account, none of them provide mechanisms to guarantee compliance with temporal contraints at execution time in advance.

Thus, compositions of services must be carried out taking temporal restrictions established by the client into account. But, this is not enough to guarantee that these compositions will not be fulfilled by the estimated time. It is necessary to consider the service provider workload at the moment when the client makes a request. So, in order to evaluate if an initially suitable service composition is feasible, it is necessary for each service provider to report about its availability to deal with the request. To do it, the SAES (Search And Execution Services) framework has been developed.

The sections of the paper are structured as follows: In Section 2, a general description of the SAES framework is presented. In Section 3, a module in charge of making the composition Next, in Section 4, the module in charge of consulting the execution time of the services is described. Section 5 presents the Real-Time Service Provider.Section 6 shows tests, results and final remarks.

2. SAES Framework

SAES (Search And Execution Services) framework has been developed in order to offer a mechanism for searching for complex services that fullfil the client's goal, bearing in mind that the goal must be achieved before a maximum time limit (deadline) established by the client. Therefore, the SAES framework contacts service providers with the intention of reserving the required services to achieve the goal and ensure that the services are executed correctly and on time. To provide these functions, the SAES is composed of two modules (Figure 1):

- **Service Composer Module (SC)**: It is in charge of searches for service compositions which achieve client's goal. These compositions are based on theoretical information contained in service descriptions.This information does not consider the current workload in the system.
- **Commitment Manager Module (CM)**: This module analyzes the service composition and the current situation of services that compose it. Moreover, this module monitors service execution in order to take it into account service performances for similar future situations.

Furthermore, in our system, the concept of Real-Time Service Provider (RTSP) is introduced as a provider with mechanisms that allow the execution and control of real-time services (RT-services). The following sections describe the modules that form the proposed SAES framework and the Real-Time Service Provider in more detail.

978-0-7695-3801-3/09 $26.00 © 2009 IEEE
DOI 10.1109/WI-IAT.2009.75

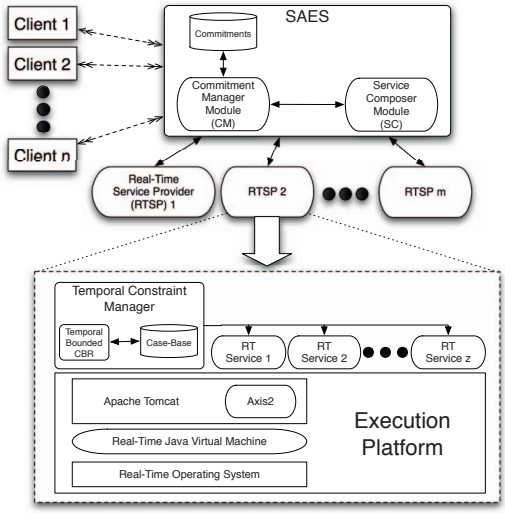

Figure 1. SAES Framework

3. Service Composer Module

When a client request cannot be solved with a single service, a possible solution is to automatically compose several services. However, dynamic service composition is a complex problem and it is not entirely clear which techniques are the best. There are several proposals that use techniques such as hypergraphs [5], modelchecking [6] or planning [7] to deal with this problem.

Service composition describes a service as a process in terms of inputs, outputs, preconditions and effects. In *Artificial Intelligence Planning* a service is described with the metaphor of an action and composition can be viewed as a planning problem.

In SAES, the SC provides a service composition (or a set of service compositions) which satisfies the goal and the client's deadline. To do this, the SC considers the execution time of services and employs AI planning techniques to automate this process. Basically, the idea is to translate real-time service descriptions to a PDDL durative actions in order to generate a plan. The CSM has three components: (i) *Service Translator* which is responsible for translating real-time service descriptions into PDDL 2.1 durative actions[8] (ii) *Problem Translator* responsible for translating the client query into a PDDL 2.1 problem and (iii) *Composer* takes as input a PDDL 2.1 problem and provides a set of plans which represent service compositions.

Before explaining how the SC works, the description of a real-time service is presented.

3.1. Real-Time Service Description

The concept of real-time sometimes generates confusion. In some situations, real-time applications are interpreted as on-line applications, but in our context, a real-time application means an action that responds to an external event in a timely and predictable manner. Therefore, a real-time service (RT-service) is a service which has its service execution time bounded. The service description of a RT-service, apart from the usual information related to Inputs, Outputs, Preconditions and Effects (IOPEs), should contain an explicit representation of time as a non-functional parameter. This parameter facilitates the selection task between a set of services that offer the same functionality.

In order to offer this information, an OWL-S service description, extended with non-functional parameters, has been used (see Figure 2). This description introduces a non-functional parameter which represents service execution time. The value of the service execution time is represented by an average execution time. This execution time is obtained by measuring the cost of the service off-line. Besides that, preconditions and effects are time-stamped annotated.

```
<Duration_param:hasLocal>
  <duration:Duration-Expression rdf:ID="PDDXML-Duration">
   <expr:expressionBody    rdf:datatype="http://...#string">
    <and><equals>
      <variable><var type="object">?duration</var></variable>
      <constant><const type="int">8</const></constant>
    </equals></and>
   </expr:expressionBody>
  </duration:Duration-Expression>
 </Duration_param:hasLocal>
...
<process:hasPrecondition>
 <pddxml:PDDXML-Condition rdf:ID="PDDXML-Precondition">
  <expr:expressionBody   rdf:datatype="http://...#string">
    <and><atStart><not>
       <pred name="agentHasKnowledgeAbout">
        <param>?http://.../Packing/GetItems.owl#FinishEvent</param>
       </pred>
    </not></atStart></and>
   </expr:expressionBody>
  </pddxml:PDDXML-Condition>
</process:hasPrecondition>
```

Figure 2. Non-functional parameter duration and temporal precondition

3.2. Composing Temporal Services

Basically, the SC works as follows. When a RT-service is registered in the system, the SC takes its description and sends it to the *Service Translator*. The *Service Translator* is responsible for translating RT-service descriptions into PDDL 2.1 durative actions. Once the services are modeled as actions, it generates a PDDL domain file which contains the definition of the actions structured as a planning problem. This process is an extension of a converter presented by Klusch and Gerber [9] which is limited to dealing with services without time-stamped annotations. When a client query arrives the composition process starts. The client query is an OWL document which contains the information related to the inputs of the desired service and the goals (outputs)

to be achieved. This file is sent to the *Problem Translator* which translates the OWL file into an equivalent one in PDDL 2.1 language.

Once the service composition problem has been translated into a planning problem, the domain and problem files in PDDL 2.1 are sent to the *Composition Service*. This service contains a planner which deals with PDDL 2.1 language and considers time as a parameter to optimize the plans. Any planner which deals with PDDL 2.1 can be used. In this proposal the planner is only used as a tool for obtaining service compositions. It is not a goal of this work to study in-depth the use of planners for service composition. The task of the planner is to obtain a plan which represents a service composition sequence that satisfies the client's goal before a deadline.

The *Composition Service* continues searching for plans until the CM sends the SC a message to finish the search process because a previous plan has been accepted. Besides this, the *Composition Service* obviously stops if no more plans are found.

4. Commitment Manager Module

The CM has two main functions: (i) to check if the set of services offered as a solution by the SC will be available to fulfill the client request; and (ii) when the client selects a service composition, the CM must establish a commitment relationship with the RTSPs that the selected RT-services provide.

To fulfill the first function, the CM must communicate with all of the RTSPs that offer services involved in the composition. Each RTSP analyzes if the required RT-service can be executed on time considering the current workload and the success probability associated to it. The result of this analyses is returned to CM. The result consists of a tuple $< T_{start}, T_{duration}, SP >$ where T_{start} indicates the moment when the service can start its execution, $T_{duration}$ indicates the necessary time to complete the service and *SP* is the probability of a successful execution. Moreover, a pre-commitment between the RTSP and the CM is established. This pre-commitment forces the RTSP to reserve the required time and other necessary resources to execute the service until the client decides if the service composition is to be executed. Of course, the clients have a maximum time to take this decision. If this reservation is not made, the required time or some of the resources could be used to deal with another client request, and therefore the guaranteed execution time of the composed service by the CM could be compromised.

When all RTSPs have responded to the CM, considering the information that the RTSPs provide, checks that the service composition can be executed before the client deadline. Then, the CM must calculate the success probability associated to the service composition. To do this, the CM uses the success probability sent for all RTSPs weighted with the information from previous executions of similar services. The service composition success probability is calculated as follows:

$$SP_{composition} = \prod_{i=0}^{N} SP_i * \omega_i \quad (1)$$

Where $\omega_i \in [0, 1]$ is the weight associated to the service i. This weight is related to the previously fulfilled commitments; A RTSP which has many unfulfilled commitments will have a low weight.

Once the CM calculates the service composition success probability, it sends the client the composed service and its probability ($SP_{composition}$). The client analyzes if it is a suitable composition. If the client agrees with the service composition, the client communicates to CM that the RT-service executions can start. When this is the case, the pre-commitments established with the RTSPs are confirmed by the CM and the RTSPs execute the corresponding RT-services. If the client does not agree with the service composition, the CM breaks the pre-commitments, freeing the time slack reserved by the RTSPs.

The CM is also in charge of ensuring that the acquired commitments are fulfilled. In cases where a commitment cannot be fulfilled, the CM penalizes the RTSP which provides the service. This penalty is captured through the weights applied when the CM calculates the service composition success probability in future situations.

So far, the main SAES framework components have been described. In the next section the funcionality of the RTSP is introduced.

5. Real-Time Service Provider

As previously pointed out, the RTSP is in charge of executing the RT-services. Besides this, the RTSP analyzes when a RT-service can be executed considering the RTSP workload. In order to develop and execute RT-services a Real-Time Operating System (RTOS) is necessary (see Figure 2). More specifically, the RTOS used in this architecture is Suse Linux Enterprise Real-Time 10. To develop the RTSP and the RT-services, the Real-Time Java Language has been used as a suitable language for expressing temporal features is necessary. The RT-services are running on the Apache-Tomcat server with Axis2.

Otherwise, prior to performing the analysis, it is necessary to make an estimation of the temporal cost of the RT-service execution. To make this estimation, each RTSP incorporates a module, called *Temporal Constraint Manager*. This module is able to make an accurate prediction of the service execution time using information from previous service executions. For more details about the *Temporal Constraint Manager* module you can consult [10].

6. Tests and Results

Several simulation experiments have been carried out to monitor the behavior of the system and evaluate the incorporation of the CM into the SAES framework in order to provide the clients feasible plans. The experiments basically consist of launching a set of client requests to the SAES, including the CM or not.

The points to be analyzed from the results obtained after executing the tests are: (i) the number of client requests that can be attended with a positive answer (service composition) (ii) once the SAES provides a suitable service composition, to check if the composition provided has been carried out successfully.

In Figure 3 the behavior of the SAES, including the CM or not, is compared. On this graph it can be seen that SAES with the CM provides a lower number of answers (service compositions) than the SAES without the CM. The reason is that the SAES with CM rejects some of the compositions provided by the SC because this configuration takes into account not only the composition suitability but also the service provider availability.

In Figure 4 the percentage of successfully accepted plans, in both SAES configurations(with CM and without it), is shown. This graph reflects that once the client has accepted a service composition provided by SAES with CM, the probability of service composition success is higher than the SAES without the CM. Obviously, this is because the CM first checks the providers current availability.

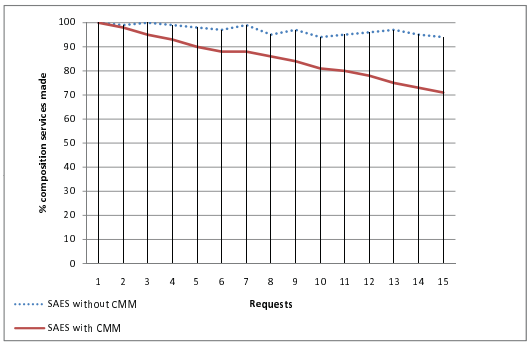

Figure 3. Number of service compositions provided by SAES including the CMM or not.

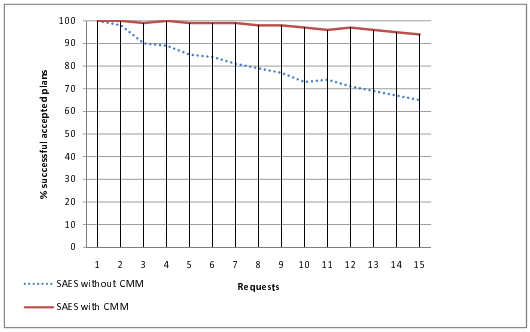

Figure 4. Execution success of the service compositions provided by SAES including the CMM or not.

Acknowledgment

This work is supported by TIN2005-03395 and TIN2006-14630-C03-01 projects of the Spanish government, FEDER funds and CONSOLIDER-INGENIO 2010 under grant CSD2007-00022.

References

[1] Pan, F.: Temporal aggregates for web services on the semantic web. Web Services, IEEE International Conference on **0** (2005) 831–832

[2] Martín-Díaz, O., Cortés, A.R., Durán, A., Müller, C.: An approach to temporal-aware procurement of web services. In: ICSOC. (2005) 170–184

[3] Naseri, M., Towhidi, A.: Qos-aware automatic composition of web services using ai planners. In: ICIW '07: Proceedings of the Second International Conference on Internet and Web Applications and Services, Washington, DC, USA, IEEE Computer Society (2007) 29

[4] Fernández-Olivares, J., Garzón, T., Castillo, L., García-Pérez, O., Palao, F.: A middle-ware for the automated composition and invocation of semantic web services based on temporal htn planning techniques. (2007) 70–79

[5] Benatallah, B., Hacid, M.S., Rey, C., Toumani, F.: Request rewriting-based web service discovery. In: International Semantic Web Conference. (2003) 242–257

[6] Gao, C., Liu, R., Song, Y., Chen, H.: A model checking tool embedded into services composition environment. In: GCC '06: Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC'06), Washington, DC, USA, IEEE Computer Society (2006) 355–362

[7] Carman, M., Serafini, L., PaoloTraverso: Web service composition as planning. In: CAPS'03 Workshop on Planning for Web Services. (2003)

[8] Fox, M., Long, D.: Pddl2.1: An extension to pddl for expressing temporal planning domains. J. Artif. Intell. Res. (JAIR) **20** (2003) 61–124

[9] Klusch, M., Gerber, A.: Semantic web service composition planning with owls-xplan. In: In Proceedings of the 1st Int. AAAI Fall Symposium on Agents and the Semantic Web. (2005) 55–62

[10] Navarro, M., amb M. Rebollo, E.D.V., Julian, V.: Composing and ensuring time-bounded agent services. In: IWANN'09. (2009) 553–560