Improving Planning Techniques for Web Services

Francisco Carlos Palao Reinés
Dept. of Computer Science and Artificial Intelligence
University of Granada, SPAIN
palao@decsai.ugr.es

Abstract

The new trend on software development is oriented to web services running in collaborative environments. Intelligent planning techniques are very useful to compose complex calls to these web services. However, there are still some issues that need to be improved to use planning and scheduling techniques in dynamic and collaborative contexts like the web service environment. This paper proposes some extensions to our planning system for using it to compose web service calls.

Introduction

This research extends the SIADEX environment, which is a planning system oriented to assist the command technical staff for decision support in forest fire fighting operations. The system is composed of different components communicating each other and working together through the Internet. These components are implemented as web services: they are pieces of software that make themselves available over the Internet and use standard XML messaging system. The World Wide Web is turning into a new paradigm called the Collaborative Web (Pallot, Prinz, & Schaffers 2005) where not only documents are connected through the network but collaborative services as well. The SIADEX project has some different web services working together for a common goal (Figure 1). One web service to store the knowledge of the problem; another to make the planning process; and a third monitors the plan execution. The planning web service uses the SIADEX planner that has been developed by us and is a forward state-based HTN temporal planner (Castillo *et al.* 2006). Moreover there is a user interface where technical staff can introduce the problem to solve it, see the plan generated by the planner and follow the execution.

To achieve a correct system operation we need to call the different web services in the correct order and time. In order to do that we have developed a central server that synchronizes all the component of the architecture. This central server has been called the InfoCenter. The InfoCenter is based on a publish/subscribe architecture (Carzaniga, Rosenblum, & Wolf 2001) (PSA) that works as follows. Each web service or user interface can publish information in the central server (InfoCenter) and also it may subscribe to the information (published by others web services or clients) that they want. For instance, when the technical staff publishes the problem, the InfoCenter sends it to the planner web service that is subscribed to it.

At the present, our architecture is very simple because only a web service of each kind is available and no external web services can connect to the InfoCenter to make the system more complete. Therefore the InfoCenter can easily compose calls to the web services to achieve the requests, the execution and the monitoring of the plans.

Now, we want to extend the system not only to assist the technical staff for decision support in forest fire fighting but also for e-business, e-tourism and workflow applications among others. Therefore we need to extend the system with new web services, some of them different than the current ones and another with similar capabilities. However, the current PSA presents some lacks that impede the extension of the system. For instance, the InfoCenter can not choose what web service is the correct one (or optimal one) when there are more than one web service that offer the same functionality. Furthermore, the PSA is purely reactive because it only make web service calls upon receptio nof a publication and it is not able to compose sequences of web services with a longer time horizon.

In order to extend the system architecture we are going to use planning techniques into the InfoCenter, as well some frameworks to use planning techniques for web services composition have been purposed (Madhusudan & Uttamsingh 2006; Mithum, desJardins, & Finin 2003) but there are still a lot of issues to solve to fulfill our InfoCenter requirements. We are thinking of using our own planner, SIADEX, to use it inside the InfoCenter. However, we need to improve it for some reasons. Firstly, the web services environment is a very dynamic context. Therefore, the state of web services (the domain) and the requests of the users (the goal) can change during the execution of calls to web services. And our planner can not check domain and state changes during planning time. Secondly, there could be a large number of web services with similar capabilities and the InfoCenter has to evaluate them to decide which one is the best to achieve its goals. And thirdly, we are thinking of an architecture oriented to the Collaborative Web, so our intelligent InfoCenter need to communicate with others intelligent servers in order to access trough them to resources that are not directly connected to it. So, the planner needs to be extended with distributed planning skills and to be able to understand web services standard languages.

In this work we present our system SIADEX as a framework that will be extended with novel ideas to give solutions to all these problems.

The SIADEX architecture

SIADEX is a system being developed under a research contract with the Andalusian Regional Ministry of Environment. Its objective is to assist the command technical staff in the design, dispatching and progress of forest fire fighting plans. It is composed of different, domain independent web services (Figure 1), that offers different services, that are distributed and communicate with each other using XML-RPC standard protocols.

- SIADEX Planner: Is a planning web service that can be called by XML-RPC protocol. SIADEX is a forward state-based HTN temporal planner (Castillo *et al.* 2006). It uses its own hierarchical extension of PDDL 2.2 level 3 language, that makes it very expressive. It also has the capability to include embedded Python scripts in the domain definition, that allows us to implement external calls at planning time.
- BACAREX: Is an ontology web service that stores the knowledge related to the planning domain. In our case its stores information about the forest fire fighting domain in Andalusia (Spain). BACAREX is also capable of generating domain and problem files that are processed by our planning web service.
- Monitor service: This web service splits the plan into several pieces and sends every piece to the person in charge of executing it. These parts of the plan will be presented to the user using any portable electronic device. The monitor controls the plan executions attending the dependences between tasks and their possible delays (Castillo *et al.* 2006).
- User interfaces: We have provided GUI capabilities to the planning system for the expert. The GUI is built on top of the ArcView GIS tool (ESRI). This GUI is totally domain dependent and oriented toward the interaction with the forest fire technical staff. We have also developed a web interface to monitor the execution of the plan with any available web browser.
- InfoCenter: It is the central component of our architecture. All the aforementioned web services are connected to the InfoCenter and collaborate each other by passing messages through it. The InfoCenter has been developed as a publish/subscribe architecture (PSA) in which the others web services can subscribe to the information that they want and publish the information that they have to share with others web services. The PSA works correctly in small environments like this with a few web services but it is purely reactive. And we want to extend this architecture to larger and more dynamic environments where we would need a deliberative server able to compound sequences of calls to web services. To achieve this we are thinking on extending the InfoCenter with planning techniques that we need to develop and are explained below.

Composing Web Services and SIADEX

There is an interchange of information between all the web services described above during a planning episode to assist the technical staff for decision support. In this section we show how this interchange of information is done at the present and how would be done in the future supporting larger environments of web services.

Present operation between web services

The InfoCenter is the Broker Server in our PSA . Each web service or user interface can publish information in the central server (InfoCenter) and also can subscribe to the information (published by others web services or clients) that they want. The basic cycle of the present architecture is:

1. The user interface publishes the goal of the planning problem defined by the technical staff and the InfoCenter sends it to the ontology web service that is subscribed to all new information about the world state and the new goals.
2. The ontology web service BACAREX publishes the domain and the problem translated into PDDL from the ontology knowledge and the InfoCenter sends it to the SIADEX Planner that is subscribed to all the domains and problems in PDDL generated.
3. The SIADEX Planner publishes the plan generated and the broker server sends it to the Monitor because it is subscribed to new plans.
4. The Monitor publishes the actions that have to be executed at each time and the InfoCenter sends it to the technical staff in charge of doing it.
5. Until the plan is completely executed, the technical staff send (public) confirmations about actions completed to the InfoCenter and the Monitor, that is subscribed to new events of the actions, publishes new actions. BACAREX is also subscribed to the new events of the actions in order to update the world state in the ontology.

The cycle shown above can be carried out with the present PSA that implements the InfoCenter. However, if we had more web services connected and some of them offer the same or similar functionalities, we would need to make more complex compositions of web services that we can not make now. At the moment, the InfoCenter can not choose what web service is the correct one (or optimal one) when there are more web services that offer the same functionality. Furthermore, the PSA is purely reactive because it only make web service calls when receive a publication and it is not able to compose sequences of web services with a longer time horizon. Note that it is not only a selection problem to pick the best web service, we need to make a sequence of calls to web services to know if the goal can be achieved with the available web services. We want to keep the easy

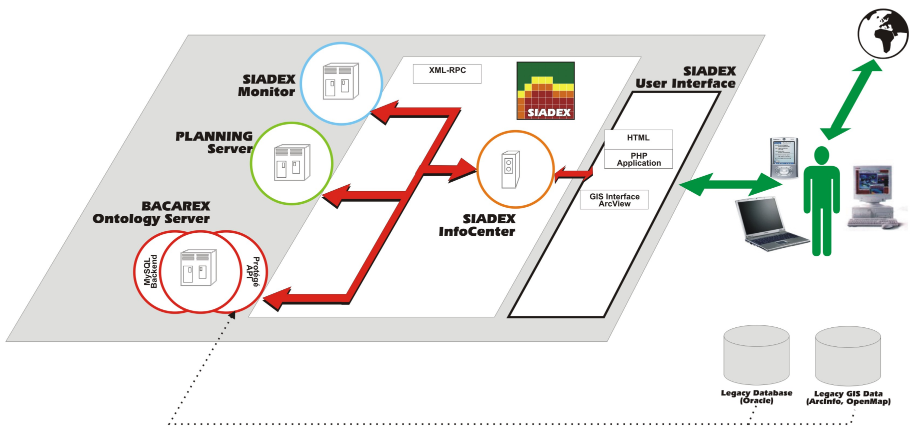

Figure 1: General overview of SIADEX Architecture.

connectivity and scalability of our current PSA but in a deliberative way. In order to do that, we will extend the InfoCenter or Broker Server with intelligent planning techniques like we describe below.

Future operation between web services

In the last section we have seen that we need to compose complex calls to web services in dynamic and larger environments. In order to do that we need to improve some features of our own SIADEX planner to use it in the InfoCenter. The features with which we need to extend our planner are shown in this section.

Continuous revision of the state and the goal. A great advantage of our PSA is the high response capabilities to the environment changes. So, it has to be supported by the planner. The set of services available could be constantly changing as online or offline status while we are executing the sequence of calls to web services. In addition the user could change his goals at execution time. Therefore the planning process needs to be continually checking for changes in the domain, in the state or in the goals (Giunchiglia & Traverso 1999; Madhusudan & Uttamsingh 2006) during the execution of web services. We can achieve this using the embedded Python scripts in the domain definition to implement external calls at planning time. There are two kinds of calls to extern web services. Firstly, calls to ensure a complete solution by checking that the web service is available. Secondly, calls to ensure a sound solution by checking that the web service behavior is the correct one.

Automated generation of domains and heuristics. As new web services are connected or disconnected to the InfoCenter the domain knowledge changes and the planner need to know it to compose the sequences of calls to web services (plans). We need to make an ontology inside the InfoCenter that stores the web service information (the domain) and define an automatic process that translates the ontology knowledge into the PDDL readable by the planner (Sirin *et al.* 2004). In addition, this web service has to be evaluated to make optimal plans. We need to implement automated heuristics generations (Zimmerman & Kambhampati 2003). To judge the optimality of the plans we need to evaluate the web services by using metrics about the network behavior (response time, transfer rating) and the final user preferences (it could be the price of a product, the duration of a trip, etc).

Distributed planning capabilities. As we have said in the Introduction section, Internet is turning into a new paradigm named the ”Collaborative Web” where services collaborate between them. Therefore the InfoCenter has to be able to share with other intelligent servers its plans (or part of them) or to ask others intelligent servers for plans (or part of them). In order to do that we need to consider all the work done in collaborative planning environments (desJardins & Wolverton 1999). Furthermore, the InfoCenter has to be able to understand standard web service description languages such as WSDL (Christensen *et al.* 2001) and to generate plan outputs as web service flows with standard specifications such as BEPEL4WS (Curbera 2002) or OWL-S (Coalition 2003).

Concluding remarks

We have described a new approach in the SIADEX planning system architecture in order to prepare it for the new trends on web services environments. The main challenges faced at planning time are in dynamic conditions and the need to collaborate with others web services architectures. We sketch an extension of our planner to check the completeness and soundness of the solutions in these environments and to be able to communicate with others web services architectures. All the changes proposed are about the central component of the architecture: the InfoCenter. That is the one in charge to compose the sequences of calls to web services with the new planning techniques that will be developed. These sequences of calls to web services have to be formulated in standards specifications such as BPEL4WS or OWL-S.

Acknowledgements

This research was supported by the contract NET033957 with the Andalusian Regional Ministry of Environment for

the assisted design of forest fighting plans.

References

Carzaniga, A.; Rosenblum, D.; and Wolf, A. 2001. Design and evaluation of a widearea event notification service. *ACM Transactions on Computer Systems* 19:332–383.

Castillo, L.; Fdez-Olivares, J.; Garcia-Perez, O.; and Palao, F. 2006. Efficiently handling temporal knowledge in an htn planner. In *International Conference on Automated Planning & Scheduling*.

Christensen, E.; Curbera, F.; Meredith, G.; and Weerawarana, S. 2001. The web services descriptio nlanguage wsdl. http://www-4.ibm.com/software/solutions/webservices/resources.html.

Coalition, O. S. 2003. Owl-s: Semantic markup for web services. OWL-S White Paper http://www.daml.org/services/owl-s/0.9/owl-s.pdf.

Curbera, F. e. a. 2002. Business process execution language for web services. http://www-106.ibm.com/developerworks/webservices/library/ws-bpel.

desJardins, M., and Wolverton, M. 1999. Coordinating planning activity and information flow in a distributed planning system. *AI Magazine* 20:45–53.

ESRI. http://www.esri.com.

Giunchiglia, F., and Traverso, P. 1999. Planning as model checking. In *In Proc. 5th European Conference on Planning*.

Madhusudan, T., and Uttamsingh, N. 2006. A declarative approach to composing web services in dynamic environments. In *Decision Support System 41 (2) 325-357*.

Mithum, S.; desJardins, M.; and Finin, T. 2003. A planner for composing services described in daml-s. In *ICAPS2003 Workshop on Planning for Web Services*.

Pallot, M.; Prinz, W.; and Schaffers, H. 2005. Future workplaces, towards the 'collaborative web'. In *1st AMI@WORK Communities Forum Day*.

Sirin, E.; Parsia, B.; Wu, D.; Hendler, J.; and Nau, D. 2004. A declarative approach to composing web services in dynamic environments. In *J. Web Sem 1(4): 377-396*.

Zimmerman, T., and Kambhampati, S. 2003. Learning-assisted automated planning: Looking back, taking stock, going forward. *AAI Magazine* 24:(2) 7396.