ICAPS'04 Demo: Planning and Monitoring Web Service Composition

M. Pistore
University of Trento - ITALY
pistore@dit.unitn.it

F. Barbon, P. Bertoli, D. Shaparau, P. Traverso
ITC-Irst - Trento - ITALY
[barbonfab,bertoli,shaparau,traverso]@irst.itc.it

Abstract

Web services are rapidly emerging as the reference paradigm for the interaction and coordination of distributed business processes. This demo shows how advanced automated planning techniques, implemented in the MBP system, can be exploited to automatically compose web services, and to synthesize monitoring components that control their execution.

Introduction

Web services are rapidly emerging as the reference paradigm for the interaction and coordination of distributed business processes. The ability to automatically plan the composition of web services, and to monitor their execution is therefore an essential step toward the real usage of web services.

In this demo, we show how automated planning techniques based on the "Planning via Model Checking" paradigm can effectively support these features, taking as input realistic modelings of web services, inspired by the BPEL4WS standard language for specifying web services interfaces and execution.

Automated web service composition starts from the description of a number of protocols defining available external services (e.g., expressed as BPEL4WS specifications), and a "business requirement" for a new composed process (i.e., the goal this new service should satisfy, expressed in a proper goal language). Given this, the planner must synthesize automatically the code that implements the internal process that achieves the business goal, exploiting the services of the external partners. This code can be expressed in some process execution language like the executable part of BPEL4WS.

Notice that this kind of planning problem requires dealing with non-determinism (since the behavior of external services cannot be foreseen a priori), partial observability (since their status is opaque to the composed service), and extended goals (since realistic business requirements specify complex expected behaviors rather than just final states). By tackling this problem, we show the capabilities of the MBP system (Bertoli, Cimatti, Dal Lago, Pistore, Traverso 2003) in realizing such a complex planning task.

Moreover, MBP is also used to generate process monitors, i.e., pieces of code that detect and signal whether the external partners behave consistently with the specified protocols. This is vital to detect unpredictable run-time misbehaviors, i.e. such as those that may originate by dynamic modifications of the partners' protocols. Thanks to these features, the MBP system will be the engine providing web services composition capabilities within the context of the ASTRO project, which aims at supporting the design process and execution of distributed web services.

This demo, whose concrete nature is defined in the following section, targets researchers interested in realistic advanced applications of planning, and will especially suit people interested in the rapidly emerging area of web and grid services.

The example

Our reference example is taken from (Pistore, Bertoli, Traverso 2004), and consists in providing a furniture purchase & delivery service. We do so by combining two separate, independent existing services: a furniture producer, and a delivery service.

The furniture producer becomes active upon a request for a given article. In case the article is not available, this is signaled to the request applicant, and the protocol terminates with failure. Otherwise, the applicant is notified with information about the product (e.g., its size), and the protocol stops waiting for either a positive or negative acknowledgment, upon which it either continues, or stops failing. Should the applicant decide that the product is acceptable, the service provides him with the cost and production time; once more, the protocol waits for a positive or negative acknowledgment, then terminating (with success or failure respectively).

The protocol provided by the delivery service starts upon a request for transporting an object of a given size to a given location. This might not be possible, in which case the applicant is notified, and the protocol terminates failing. Otherwise, a cost and delivery time are computed and signaled to the applicant; the protocol suspends for either a positive or negative acknowledgment, terminating (with success or failure resp.) upon its reception.

The expected protocol the user will execute when interacting with our composed service goes as follows. The user sends a request to get a given article at a given location, and expects either a signal that this is not possible (in which case the protocol terminates, failing), or an offer indicating the price and cost of the service. At this time, the user may either accept or refuse the offer, terminating its interaction in both cases.

Of course several interaction sequences are possible with these services; e.g., in a *nominal* scenario, none of the services answers negatively to a request; in non-nominal scenarios, unavailability of the article, user refusals or shipping service unavailability may make it impossible to reach an agreement for the purchase and delivery. Taking this into account, the business requirement for the composed service is composed of two subgoals. The "nominal" subgoal consists in reaching the

agreement to purchase and delivery the article. This includes enforcing that the data communicated to the various processes is consistent with their mutual availabilities; e.g., the total service time communicated to the user cannot be less than the sum of production and delivery times. The "recovery" subgoal consists in insuring that every partner has rolled back from previous pending requests, and is only pursued when the nominal subgoal cannot be pursued anymore. The specification and combination of these subgoals is expressed in the EaGLe language (Dal Lago, Pistore, & Traverso 2002).

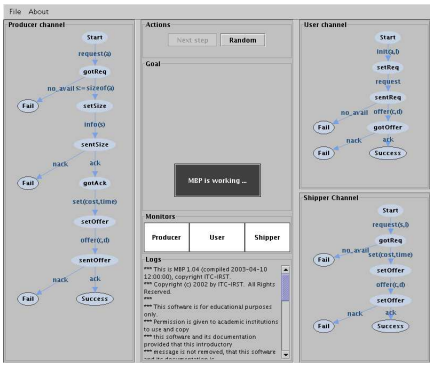

Figure 1: The web service composition phase.

Demo

The demo first shows the way MBP automatically obtains a composed service that interacts with the user, shipper and producer to achieve the above business requirement, and also produces monitors to control that protocols behave in a way compliant with their specification. Figure 1 shows this composition phase. The external protocols are depicted, in the form of finite state machines, in the left and right parts of the figure.

After the composition is completed, a high-level representation of the synthesized service is shown in the central part; its states are associated to the current *intention* of the service (i.e., to which subgoal is currently pursued). Monitors are simply represented by "protocol violation flags" that may raised at run-time (central part, low); below monitor flags, a log window describes the current and past interaction steps.

Several scenarios can be selected and run, each describing a protocol interaction; at each step of a run, the current states of the external protocols and of the synthesized protocols are highlighted, and monitor flags are updated. Different interaction modes are possible to determine the evolution of a scenario during the demo. Figure 2 shows an intermediate step of a *nominal* scenario, where the three component protocols answer positively to the requests, thus allowing the composed service to eventually achieve the preferred subgoal.

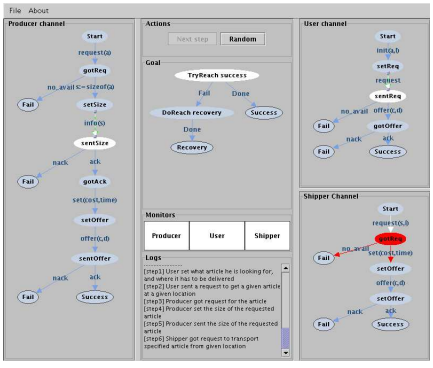

Figure 2: Inside a nominal scenario.

In Figure 3, a different (non-nominal) scenario shows how the unavailability of one of the services in providing its service causes the main subgoal to fail. In this case the composed service will perform 'unrolling' to achieve the recovery subgoal.

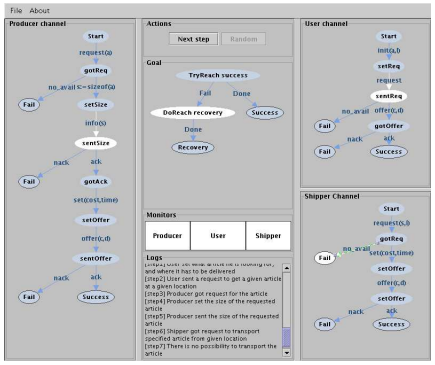

Figure 3: Inside a non-nominal scenario.

Finally, Figure 4 shows a different scenario, where one of the services (namely, the producer, on the left) has changed its protocol. Monitoring is able to detect this, since messages appear out of the expected order, and is thus able to prevent runtime misbehaviours.

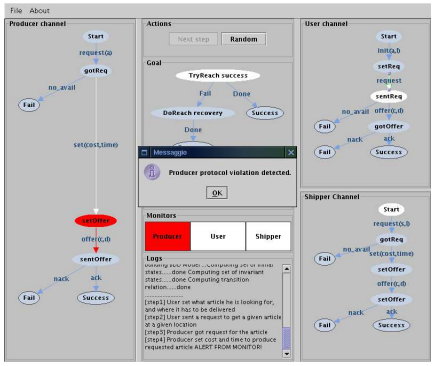

Figure 4: A protocol violation scenario.

References

Pistore, M.; Bertoli, P.; and Traverso, P. 2004. Planning and Monitoring Web Service Composition. In *ICAPS'04 Workshop on Planning and Scheduling for Web and Grid Services.*

Bertoli, P.; Cimatti, C.; Dal Lago, U.; Pistore, M.; and Traverso, P. 2003. MBP: Model Based Planner. In *ICAPS'03 System Demos.*

Dal Lago, U.; Pistore, M.; and Traverso, P. 2002. Planning with a Language for Extended Goals. In *Proc. AAAI'02.*