What's New in WSMX?

Srdjan Komazec and Federico Michele Facca

Semantic Technology Institute (STI) - Innsbruck,
ICT Technologiepark, Technikerstrasse 21a, 6020 Innsbruck, Austria
`firstname.lastname@sti2.at`

Abstract. The Web Service Execution Environment (WSMX) is the most complete implementation of a Semantic Execution Environment to support the automation of the Web service life-cycle. WSMX is a constantly evolving project. The demo will provide insight and justify the value of the newly introduced features: the Complex Event Processing engine, the Notification Broker engine and the Orchestration engine.

1 Introduction

A Semantic Execution Environment (SEE) represents a class of middleware solutions that support the common service-related life-cycle tasks through the exhaustive use of machine-processable service descriptions. The most comprehensive existing SEE implementation to date is the Web Service Execution Environment (WSMX)[1]. WSMX is a reference implementation of Web Service Modeling Ontology (WSMO) [1]. WSMX is adopted as one of the reference architectures of OASIS Semantic Execution Environment (SEE) Technical Committee[2]. In the past edition of the demo [2] at ESWC'09 we demonstrated a new WSMX version implementing some new features: Ranking engine, Grounding engine and Monitoring facilities. In this demo we will focus on the latest WSMX advancements comprised of:

- Complex Event Processing engine that, coupled with the existing WSMX monitoring facilities, enables the next level of the environment agility;
- Notification Broker engine that enables the asynchronous notification of discovery results;
- Orchestration engine that enables the orchestration of Semantic Web services; and
- A set of smaller improvements such as a new format for Web Service discovery results, caching of the discovery results and security facilities.

The paper is structured as follows: Section 2 updates the WSMX architecture; in Section 3 we present the advancements in SEE monitoring with special emphasis on the integrated complex event processing engine; Section 4 presents the asynchronous notification of discovery results; in Section 5 we describe the

[1] `http://www.wsmx.org`
[2] `http://www.oasis-open.org/committees/semantic-ex`

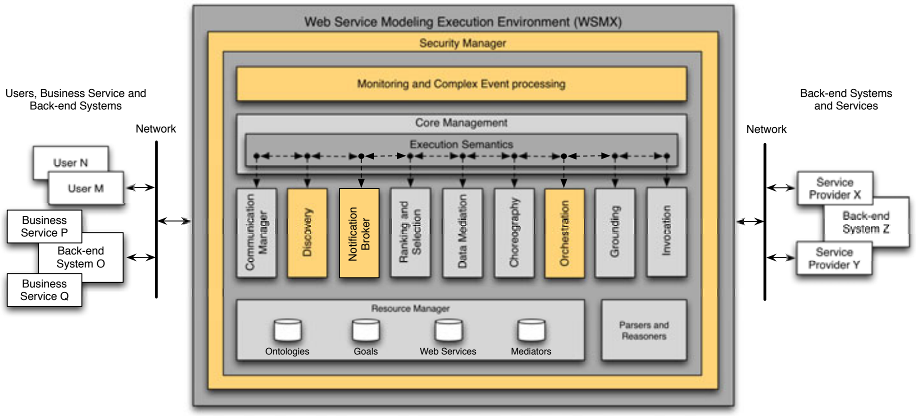

Fig. 1. The advanced WSMX architecture

orchestration engine; Section 6 gives a brief overview of other smaller improvements; finally in Section 7 we draw some conclusions and describe the intended demonstration plan.

2 Updated WSMX Architecture

WSMX adopts a typical multi-layered service-oriented architecture with the goal of minimizing cross-cutting concerns and encapsulate functional responsibilities, as shown in Figure 1. The top layer is devoted to the Core Management facilities, which orchestrate the lower-level broker services by providing data and control flow capabilities in order to enable execution of useful processes such as: Web service discovery, Web service invocation and a combination of the two. The middle layer is represented by the set of broker services offering the basic functionalities (e.g., discovery, ranking, choreography, etc). The lower-level provides a set of fundamental logistical services such as management of WSMO-related artifacts as well as parsing and reasoning functionality. More details about the WSMX broker services can be found in [3]. In contrast to the architecture shown in the previous demo [2], the following differences can be identified: First, the integration of the Complex Event Processing engine as a part of the monitoring facilities of the platform that enables timely responses to complex situations of particular interest (described more in Section 3). Second, the addition of the Notification Broker engine, which enables asynchronous discovery of Web services fulfilling the submitted goals and notification of the goal owners by relying on publish/subscribe mechanisms (described more in Section 4). Third, the addition of the Orchestration engine, which enables the execution of complex Semantic Web service compositions (explained in a bit more detail in Section 5). The discovery and ranking processes have also undergone some smaller improvements. Additionally, a component to manage user authentication and authorization has been introduced. These advancements are described briefly in Section 6.

3 Complex Event Processing

A SEE implementation generates a rich source of events, including events emitted at the level of the overall platform (e.g. at the start of the invokeWebService process), events at the level of single broker services (e.g. tracking of the functionality provided by the Discovery engine) and events at the level of the external services (e.g. a fault during a Web service invocation). Since the events have ontological representations, the complete event history is preserved in an RDF repository for convenience purposes (support for massive storage, inferencing and querying). Before reaching the repository, the generated events must be adequately processed in order to detect and react in a timely manner to the complex situation of interest (e.g. three failed attempts to invoke a particular Web service may trigger another round of Web service discovery to select the second best solution if such exists).

WSMX now integrates an RDF-based Complex Event Processing engine (RDF-CEP) built on top of the existing WSMX monitoring facilities described in [4] and demonstrated in [2]. The primary objective of RDF-CEP is to detect occurrences of RDF triples satisfying expressive RDF patterns. When a complex event is detected, the engine executes the appropriate actions associated with the detected event (e.g. update of the aggregated statistical measures, notification of the administrator about successive failed attempts to access WSMX facilities). The engine can be also used outside of the WSMX environment as a general purpose tool for filtering and processing RDF data streams.

4 Notification Broker

Asynchronous communication within WSMX is enabled through the newly introduced broker service called the ‘Notification engine’. This service is responsible for registering and storing user subscriptions and related results (in order to avoid duplicate notifications) and sending notifications of results related to user subscriptions. For now, WSMX supports subscriptions to goal-based discovery, but the mechanism can easily be extended to other execution semantics.

To support the full process, a new execution semantics was created that is executed as soon as a user subscribes to a goal or a new Web service is registered in the platform. This execution semantics checks for existing (and not expired) user subscriptions, and executes for each subscribed goal the discovery process. Once new Web services have been matched w.r.t. previous notifications, they are notified to the users. The notification occurs according to the mechanism selected by the user. Currently we support email or Web service invocation. In the later case a subscriber must implement a specific WSDL definition. A wider discussion related to the solution implemented in WSMX can be found in [5].

5 Web Service Orchestration

Choreography, as currently used in WSMO/L/X, involves a client communicating with several components to achieve the desired result. In contrast, orchestration involves not just client-component communication, but also communication directly between components. The Orchestration engine introduced in WSMX is conformant with the WSMO dataflow-based orchestration language as defined in [6]. The engine supports dataflow in a manner consistent with WSMO/L, and adopts the explicit concept of *performance*, and a new type of mediator to mediate between performances.

In a nutshell the engine and the underlying language solve the following issues:

– mediation in any connection, including dataflow, between heterogenous components;
– *extraction* and *aggregation* in the consumption and production of messages according to the WSML grounding approach; and
– execution of service discovery and service invocation within composed services.

6 Other Minor Improvements

To keep the architecture up-to-date with results from the OASIS SEE-TC, we refined the discovery and ranking engine so that it now returns, instead of a set of services, a set of Web service-to-goal mediators (wgMediator). In this way we are able to associate the service with the goal, and to attach to this tuple results coming from the discovery or ranking steps, such as type of the match, value of the non-functional properties associated with the ranking results, goal specific quotes, and so on. All this information enriches the knowledge on why and how a certain service has been associated with a certain goal, thus giving better support for the selection of services.

The results of the discovery process are now cached. This enables a reduction in the computational time associated with discovery when a goal is submitted multiple times by relying on previous results. The cache is invalidated each time a new service is (un)registered in the system.

Last but not least, WSMX now introduces a set of security features as the first line of defense against malicious use of the offered functionality. Basically, authentication, authorization, and accounting for both SOAP and Web-based WSMX endpoints has been implemented. Security is based on the developed security ontology following the Role-based Access Control model. While authentication verifies supplied security credentials against the ontology, authorization checks whether the client has sufficient privileges to consume particular WSMX functionality. Integrated with the monitoring RDF store (see Section 3) the accounting facilities provide the possibility to track the consumption of WSMX functionality and resources.

7 Conclusions and Demonstration Plan

After completion of the basic SEE requirements, WSMX is now on track to embrace advanced techniques in order to make it a more controllable and self-aware environment suitable for enterprise-level use. This paper presents some of the techniques, namely the inclusion of a Complex Event Processing engine coupled with the existing Monitoring facilities, a deferred goal Notification mechanism and an Orchestration engine, together with some smaller improvements related to the Web service discovery process and security concerns.

The prospective attendees to the demo will be able to examine and simulate different situations on a live WSMX instance prepared with a number of Semantic Web Service descriptions coming from the Enterprise Interoperability and Enterprise Collaboration domain. In particular, an attendee will be able to submit goals for asynchronous execution and then examine the platform behavior when suitable subsequent Web services are registered and notifications generated. In the context of complex event processing, the attendee will be able to change and adapt the event pattern descriptions. After executing the appropriate WSMX processes, the attendee will observe the behavior of the engine. The Orchestration engine will be demonstrated with an example of a Web service composition showing all the features of the WSMO orchestration language. The platform internals will be exposed through the Web console that will graphically presents the current status of the system and its evolution.

References

1. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services: The Web Service Modeling Ontology. Springer, Heidelberg (2006)
2. Facca, F.M., Komazec, S., Toma, I.: WSMX 1.0: A Further Step toward a Complete Semantic Execution Environment. In: Aroyo, L., Traverso, P., Ciravegna, F., Cimiano, P., Heath, T., Hyvönen, E., Mizoguchi, R., Oren, E., Sabou, M., Simperl, E.P.B. (eds.) ESWC 2009. LNCS, vol. 5554, pp. 826–830. Springer, Heidelberg (2009)
3. Fensel, D., Kerrigan, M., Zaremba, M. (eds.): Implementing Semantic Web Services: The SESA Framework, 1st edn., May 2008. Springer, Heidelberg (2008)
4. Komazec, S., Facca, F.M.: Towards a Reactive Semantic Execution Environment. In: Meersman, R., Herrero, P., Dillon, T. (eds.) OTM 2009 Workshops. LNCS, vol. 5872, pp. 877–887. Springer, Heidelberg (2009)
5. Facca, F.M., Komazec, S., Zaremba, M.: Towards a Semantic Enabled Middleware for Publish/Subscribe Applications. In: ICSC, pp. 498–503. IEEE Computer Society, Los Alamitos (2008)
6. Norton, B., Haselwanter, T.: Dataflow for Orchestration in WSMO. In: Working Draft d15.1, WSMO Working Group (2007), `http://www.wsmo.org/TR/d15/d15.1/v0.1/`