Towards a Federation of Semantic Execution Environments

Davide Cerri and Srdjan Komazec

Semantic Technology Institute (STI) Innsbruck, Universität Innsbruck, Austria
`{davide.cerri, srdjan.komazec}@sti2.at`

Abstract. The most comprehensive implementation of a Semantic Execution Environment (SEE) is currently represented by WSMX, which can however support only single-instance deployments. In this paper we discuss the motivations behind distributed scenarios and outline the issues to be addressed in order for WSMX to support them.

1 Introduction

A *Semantic Execution Environment* (SEE) represents a class of middleware solutions that support the common tasks related to service life-cycle through the usage of machine-processable service descriptions. As such, SEE middleware constitutes the core of a Semantically-Enabled Service Oriented Architecture (SESA) [6]. At present, the most comprehensive existing implementation of SEE is the Web Service Execution Environment *WSMX*[1], which is a reference implementation of the Web Service Modeling Ontology (WSMO) [1].

A single instance of SEE can fulfil the requirements of scenarios characterised by limited scale (in which a single instance can handle all the knowledge, e.g. ontologies and service descriptions, and all the requests) and "closeness" (in which all the relevant knowledge is available in a single instance, controlled by a single authority). There are however scenarios in which the above assumptions do not hold, and which raise the need to support a distributed deployment. In particular, we can distinguish between "bottom-up" and "top-down" distributed scenarios:

- In a *bottom-up* scenario, different SEE instances are established independently, e.g. in different platforms owned by different providers or groups. The respective owners decide to federate them, so that users of each one can benefit also from services provided by other ones. The individual instances can have some peculiarities, likely related to their respective owners (e.g., in terms of domain or geographical location of registered services) but this is not by design. The federation is likely not transparent to users, which may want to distinguish between services provided by their "own" platform and services provided by other platforms.

[1] http://www.wsmx.org/

– In a *top-down* scenario, a single SEE instance is not enough (e.g. for scalability reasons), thus the owner needs to deploy a distributed architecture which includes multiple instances. In this case, some specialisation of the individual instances (e.g., in distributing service descriptions among them) can be an explicit design choice. The distributed architecture is transparent to users, who see it, logically, as a single entity.

It is worth noting that the two scenarios are not mutually exclusive: a top-down "distributed instance" can act as a single member of the federation in a bottom-up scenario.

2 Towards Federation Support in WSMX

While there exists some preliminary work on a peer-to-peer discovery mechanism in WSMX [5], the overall issue of SEE federation through WSMX has not been addressed yet, and the current version of WSMX can only support single-instance deployments.

WSMX relies on a local repository for storing semantic Web service descriptions and other relevant artefacts. Rather than building a network of discovery engines, a solution can be envisioned in which this repository is distributed and transformed into a larger-scale shared infrastructure that can be accessed by multiple discovery engines, as proposed in [2]. Such solution does not however suit very well scenarios (typically bottom-up) in which owners of SEE instances are likely not willing to directly publish their knowledge in a more or less open / shared environment, but only to allow a more limited and controlled access to it through the services provided by their own instance of SEE middleware. Therefore, we envision a network of WSMX nodes (able to operate also independently), each one with its own repository and its own local knowledge. In a network made of multiple WSMX nodes, each node can thus support the life-cycle of locally registered services as in a single-instance deployment. In a distributed architecture, however, a WSMX node needs to be able to communicate with other nodes, in order to perform discovery, invocation, etc. also for services that are not locally known.

Discovery is the first and probably most relevant functionality to be supported in a federation of WSMX nodes. In order to discover services which are not registered locally, WSMX needs to forward the request (i.e., the goal) to other nodes. A first issue to be addressed in this respect is which other WSMX nodes should be contacted. Since discovery is a service, a research question here is how to describe and discover discovery services, and if it is feasible to use WSMO and existing discovery mechanisms in WSMX for this purpose. Simpler approaches, e.g. based on some categorisation of business domains covered by each node as in [4], are also possible. It needs to be noted that in a bottom-up scenario the distribution of knowledge among nodes may not reflect any particular strategy, thus making sophisticated techniques not very effective or even applicable. In such cases, if the number of nodes is small, contacting all the nodes can actually be a feasible solution. In larger scenarios, a clustering approach as proposed in [3] can be feasible. WIth such approach, relevant clusters (rather than individual nodes) are identified, and all nodes in these cluster(s) can be contacted.

Discovery is further complicated if we consider service composition. In a single-instance deployment, WSMX can identify a set of locally registered services that, if composed together, can fulfil the user's goal. In a distributed setting, however, it may be the case that no single WSMX node has knowledge of a complete set of services that can fulfil the goal, but that this can be achieved by composing services registered at different WSMX nodes. Moreover, the knowledge about how to decompose a goal into multiples subgoals can be present in different nodes, and multiple (and successive) decompositions can be possible. Therefore, in order to fully support a distributed and cooperative scenario, a mechanism to control distributed decomposition of goals and forwarding of subgoals between nodes is needed.

Finally, a distributed architecture has an impact also on other functionalities provided by WSMX. Regarding ranking, ranked lists of matching services coming from different WSMX nodes need to be merged. Regarding service invocation, both the WSMX node which received the request from the user and the one where the service is registered likely need to be involved.

An enhanced WSMX, capable to cooperate with other federated WSMX instances, can support both the distributed scenarios previously introduced through different deployment strategies:

- The bottom-up scenario suggests a peer-to-peer architecture, in which each WSMX node is aware of the federation and is able to initiate a federated discovery when requested by a user. Each user communicates with its "own" node, which may in turn communicate with other nodes. Therefore, any WSMX node can potentially communicate with any other WSMX node in the architecture. We call this the *P2P deployment*.
- In a top-down scenario, there can be one WSMX node (the "front-end node") which has the functionality for federated discovery and forwards requests to other nodes[2]. The front-end node basically plays the role of a single point of access, and all users communicate only with it. Other WSMX nodes only respond to requests coming from the front-end node, performing only local operations. We call this the *front-end deployment*.

3 Conclusions and Future Work

In the upcoming versions of WSMX we plan to address the issues outlined here, enabling the realisation of a federation of Semantic Execution Environments. We plan to address discovery first, starting with the support for forwarding goals and subgoals resulting from goal decomposition between WSMX nodes. This functionality, coupled with support for invoking remotely registered services, can already enable simple federations composed by a few nodes, where it is feasible

[2] This front-end node can be easily replicated, if needed for scalability reasons.

to forward the request to all the members of the federation. The next step will then be to define one or more mechanisms to identify only a subset of nodes as the target, depending on the goal.

References

1. Roman, D., Keller, U., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., Polleres, A., Feier, C., Bussler, C., Fensel, D.: Web Service Modeling Ontology. Applied Ontology 1(1), 77–106 (2005)
2. Sapkota, B., Roman, D., Kruk, S.R., Fensel, D.: Distributed Web Service Discovery Architecture. In: AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services. p. 136. IEEE Computer Society, Washington, DC, USA (2006)
3. Sapkota, B., Vasiliu, L., Toma, I., Roman, D., Bussler, C.: Peer-to-Peer Technology Usage in Web Service Discovery and Matchmaking. In: Ngu, A., Kitsuregawa, M., Neuhold, E., Chung, J.Y., Sheng, Q. (eds.) Web Information Systems Engineering – WISE 2005, Lecture Notes in Computer Science, vol. 3806, pp. 418–425. Springer Berlin / Heidelberg (2005)
4. Sivashanmugam, K., Verma, K., Sheth, A.: Discovery of Web Services in a Federated Registry Environment. In: ICWS '04: Proceedings of the IEEE International Conference on Web Services. p. 270. IEEE Computer Society, Washington, DC, USA (2004)
5. Toma, I., Sapkota, B., Scicluna, J., Gomez, J.M., Roman, D., Fensel, D.: A P2P Discovery Mechanism for Web Service Execution Environment. In: Second WSMO Implementation Workshop (2005)
6. Vitvar, T., Zaremba, M., Moran, M., Zaremba, M., Fensel, D.: SESA: Emerging Technology for Service-Centric Environments. IEEE Software 24(6), 56–67 (November 2007)