A COMPARATIVE STUDY ON SEMANTIC WEB SERVICES FRAMEWORKS FROM THE DYNAMIC ORCHESTRATION PERSPECTIVE

Domenico Redavid, Floriana Esposito
Computer Science Dept., University of Bari, Via Orabona 4, 70126, Bari, Italy

Luigi Iannone
Computer Science Dept., University of Manchester, Kilburn Building, Oxford Road, M13 9PL, Manchester, U.K.

Keywords: Semantic web services, Dynamic orchestration.

Abstract: This paper presents a comparison between the two main existing frameworks for modelling Semantic Web Services, namely OWL-S and WSMO, based on the formal support they provide for reasoning. The adopted yardstick is the set of use cases dictated by one of the foremost tasks in the research field: the *dynamic* orchestration. As explained in the paper, the term *dynamic* denotes the capability of an agent to design and manage in an automatic way an orchestration schema using the semantic descriptions of some services. This capability, strictly related to the automatization of Discovery, Selection, Composition and Invocation use cases, is constrained by the knowledge representation strategies adopted by WSMO and OWL-S. As consequence, each approach has some limits discussed in this work.

1 INTRODUCTION

From the Web services perspective, an orchestration is a declarative specification that describes a workflow supporting the execution of a specific business process, operation, or service (Peltz, 2003). Currently, Web services technologies make it possible to describe orchestration, but only at design-time. Semantic Web Services (SWS) provide an ontological framework for describing services in a machine-readable format. In (McIlraith et al., 2001) a software agent applies logical reasoning on SWS descriptions in order to provide on the fly orchestration capabilities. With the adjective *dynamic* we denote the capability of an agent to design and manage in an automatic way an orchestration schema using the semantic descriptions of some services. The notion of *dynamic* orchestration becomes useful in scenarios where there is the need of run-time designing of process integration using semantic descriptions of the entities in play. This work aims at answering to the following questions:

- What does orchestrating mean for the Semantic Web?
- What are the requirements to enable the dynamic SWS orchestration?
- How well does each examined framework support such requirements?

In the Section 2 we revisit the notion of SWS orchestration taking into account the semantics of services. In Sections 3 we briefly describe and analyze the support offered to it by the two leading efforts for SWS representation (OWL-S and WSMO)[1]. In particular, we examine the formal support they provide for reasoning on the semantic descriptions of the services and the state of the art. In Section 4 a discussion on the limits of each approach is presented.

2 ORCHESTRATION AND SWS INFRASTRUCTURE

Orchestration for Web services has to rely on XML-based service description. Their merely syntactical

[1]Both the leading Semantic Web Services efforts - OWL-S (http://www.w3.org/Submission/OWL-S) and WSMO (http://www.w3.org/Submission/WSMO), are currently submissions in the Semantic Web Services Standardization process.

Redavid D., Esposito F. and Iannone L..
A COMPARATIVE STUDY ON SEMANTIC WEB SERVICES FRAMEWORKS FROM THE DYNAMIC ORCHESTRATION PERSPECTIVE.
DOI: 10.5220/0003071003550359
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2010), pages 355-359
ISBN: 978-989-8425-29-4

nature (Peltz, 2003) represents an unsurmountable obstacle towards the automatic implementation of the necessary operations to accomplish dynamicity in orchestration. Let us, indeed, consider the following scenario:
"Given a request (goal), an agent (dynamic orchestrator) *discovers* the possible SWSs able to accomplish the goal. At the same time, it *composes* the discovered services in order to have more possible ways to reach the goal. Having more choices available, it *selects* the best one exploring functional and non-functional properties of the different SWSs. The selection process can be used also during the composition (subgoal matching). Finally, the same agent manages the correct *invocation* of the selected services."

In this paper, dynamic orchestration of SWS means carrying out one or more of the operation mentioned above. This is very different from the design of the work-flow of the execution of simple Web services (Peltz, 2003). The first step to realize this scenario is the automatization of the emphasized operations (i.e.: discover, composition, selection and invocation). These operations, amongsts the others (namely, publishing, deployment and ontology management which are not directly involved in our scenario), are already been identified as use cases for the SWS infrastructures into the *Usage Activities* dimension described in Figure 4 in (Cabral et al., 2004). In the same work the architectural components and semantic descriptions needed to realize the *Usage Activities* are been individuated. The former had been grouped under the *Architecture* dimension and include a register, a reasoner, a matchmaker, a decomposer, and an invoker. The latter, make up the *Service Ontology* dimension and specifies the semantics of:

- Functional capabilities, such as inputs, output, pre-conditions and post-conditions;
- Non-Functional capabilities, such as category, cost and quality of service;
- Provider related information, such as company name and address;
- Task or goal-related information and domain knowledge defining, for instance, the type of the inputs of the service.

The dynamic SWS orchestration, as we defined above, needs to be matched with these dimensions (reported in Table 1) in order to individuate its requirements.

In the architecture dimension, the *reasoner* is the most important component to implement these use cases because it allows the semantic *matchmaking* of the services properties enabling the automatic *(de-)composition* of the services, and the automatic choice of the right parameters for the services *invocation*. Service ontology dimension is necessary because it makes possible to define the semantics of functional and non-functional properties owned by atomic and composite services that participate to the orchestration, as well as the semantics of their control constructs and data-flow. Ideally, the semantic descriptions of inputs and outputs should exist independently from the SWS specification. They should be chosen within existing ontologies describing a particular knowledge domain and, at most, refined. Hence, in order to define atomic SWS the semantic representation of conditions over inputs and outputs is needed, whereas to define composite SWS the semantic representation of control constructs and data-flow is needed too. Dynamic orchestration becomes, than, mainly a representation problem. This is why, in the following, when we will compare and contrast WSMO and OWL-S we will focus on the knowledge representation infrastructure.

Table 1: Orchestration in the SWS Infrastructure.

DIMENSION	DYNAMIC ORCHESTRATION
Usage Activities	Composition, Selection, Invocation, Discovery
Architecture	(De-)Composer, Reasoner, Invoker, Matchmaker
Service Ontology	Inputs, Outputs, Conditions, Atomic/Composite services

3 WSMO AND OWLS

The Web Service Modeling Ontology (WSMO) (Roman et al., 2005) provides a framework for semantic descriptions of Web services. It consists of four core elements that, properly linked, constitute the semantic description of the service: **Ontologies**, **Goals**, **Web Services**, and **Mediators**. The semantics of these entities can be specified using one of the formal languages defined by the Web Service Modeling Language[2] (WSML). WSML includes several language variants, based on three different logical formalisms, namely, Description Logics (Baader et al., 2003) (WSML-DL), First-Order Logic (Enderton, 1972) (WSML-Full) and Logic Programming (Lloyd, 1987) (WSML-Rule and WSML-Flight). Furthermore, it

[2]Web Service Modeling Language (WSML), W3C Member Submission, 3 June 2005. http://www.w3.org/Submission/WSML/

defines the variant **WSML-Core** that can be identified as the intersection between a particular Description Logic (a subset of $\mathcal{SHIQ}$) and Horn Logic (without function symbols and equality) (Grosof et al., 2003). WSML DL is, in general, incompatible with both WSML-Flight and WSML-Rule. Therefore, complete reasoning about services descriptions specified using WSML DL, Rule and Flight is unfeasible because WSML Full, the common super-language, is undecidable.

The Web Ontology Language for Services (OWL-S) is an OWL[3] ontology describing the three essential aspects of a service (i.e. its adervetisment, process model, and protocol details) using three different *modules*: **Service Profile**, **Service Model**, and **Service Grounding**. OWL-S itself is an OWL ontology. OWL, whose formal foundation is the Description Logics (Baader et al., 2003), provides three increasingly expressive sub-languages, in detail:

- **OWL-Lite** can be used to model classification hierarchies and simple constraints. OWL-Lite has the lowest computational complexity amongst OWL sub-languages.
- **OWL-DL** is used when the maximum decidable expressivity is required. It corresponds to $\mathcal{SHOIN}(\mathcal{D})$ DL (Horrocks et al., 2003).
- **OWL-Full**, although allowing for as much expressivity as RDF, is undecidable therefore it could be hardly be used for reasoning.

However, as pointed out in the following, SWRL (in its decidable fragment) represents a reasonable advancement in reconciling Logic Programming and Description Logics. For this reason, OWL-S is in a more better position than WSMO as SWRL can be seamlessly integrated with OWL.

The state of the art, both for WSMO and OWL-S, can be brefly analyzed w.r.t. the use cases reported in the usage activities dimension of Table 1, i.e. *Discovery*, *Selection*, *Composition*, and *Invocation*. Please, notice that, whilst for WSMO exist different implemented infrastructures that unify, amongst the others, all the aspects described above, to the best of our knowledge there is not any counterpart developed for OWL-S, yet. For WSMO, in general, the common starting point for the implementation of such functionalities is either WSMX[4] or IRS-3[5]. (Kifer et al., 2004) presents a logical framework that exploits WSMO formal descriptions to dynamically discover Web Services matching the requester goals. The SWS discovery in OWL-S, in most cases, aims to enhance the UDDI registry and to exploit semantic matchmaking techniques applied to the service profile model. Lynch et al. (Lynch et al., 2006) explore a potential alternative for UDDI registries[6], and investigate the feasibility of a general publish/subscribe model for Web Service Discovery. The research on services selection in WSMO is often related to the service discovery because both use cases require a semantic matchmaker component. The selection mechanism of OWL-S is related to the application of semantic matchmaking techniques both to the service profile and to the process model. The invocation use case is strictly related to the semantics of the protocol used to invoke the service. In (Walton, 2005) a particular protocol (Which can be used with OWL-S and WSMO) to represent the computational aspects of a Web service, such as invocation order, and inter-argument dependencies, is proposed. The research on composition of services encompasses more elements within service ontologies than other usage activities because it must consider the SWS conditions during the composition task. To the best of our knowledge, works on fully automatic composition of WSMO services have not been proposed yet. However, a semi-automatic composition approach is proposed in (Hakimpour et al., 2005) . Most OWL-S research activities focused on composition methods drawing inspiration mainly from AI planning, like reported in (McIlraith and Son, 2002), one of the first works about this topic. In (Redavid et al., 2008) a method for encoding OWL-S atomic processes by means of SWRL rules and composing them using a backward search planning algorithm is presented. The proposed solution has been realized using exclusively Semantic Web technologies.

4 DISCUSSION

The meaning of orchestration once we take semantics into account becomes different from the pre-arranged Web service composition, which is constrained only at syntactic level. To summarize, to orchestrate means to realize the automatic discovery, selection, composition, and invocation of services so as to achieve a goal. Solving this problem requires to respond to use cases organized along three dimensions in the SWS infrastructure (please see Sec. 2). Here we focus on the usage activities and service ontology to compare

[3]OWL-S is based on OWL 1.0 recently extended with the new recommendation OWL 2, as reported in http://www.w3.org/TR/owl-overview/

[4]Web Service Execution Environment (WSMX) - http://www.wsmx.org

[5]Internet Reasoning Service (IRS) 3 - http://kmi.open.ac.uk/technologies/irs/

[6]Universal Description Discovery and Integration (UDDI) - http://uddi.xml.org/uddi-org

and contrast the two main framework proposed so far: WSMO and OWL-S[7].

With respect to the *Usage activities* dimension, both WSMO and OWL-S take into account all the use cases that we have individuated for the dynamic orchestration. They differ only in the fact that WSMO requires the modeling of the service requester features. This implies the need for tackling the interoperability problems between different WSMO elements within the framework itself. In order to overcome this issue, different types of mediators are been defined and considered in every usage activity. This is reflected also in the *Service ontology* dimension where WSMO presents a richer service description model than OWL-S. Consequently, some extra features related to Web services execution (like goals, mediator and communication protocols) are represented. The shortcoming of this approach is that any WSMO services usage is limited to its declared goals at design time.

The formal semantics offered by WSMO and OWL-S is fundamental to achieve *dynamic* orchestration. In Figure 1 we compare the formalisms on which both frameworks are based.

WSMO appears unrelated to the Semantic Web stack of languages and formalisms, with the exception of its basic level (URI and XML) that guarantees syntactic interoperability on the Web. It offers mappings with the RDF syntax and the $\mathcal{SHIQ}$ Description Logic which, however, does not cover the whole OWL-DL.

WSMO global approach to service ontology definition can be summarized as follows:

- It started with the definition of a FOL-like language able to represent of all the possible aspects of a Web service;
- It defined several sub-languages restricting the original expressive power to well-known fragments (DL or LP);
- It (partially) mapped such fragments on to SW current standard languages.

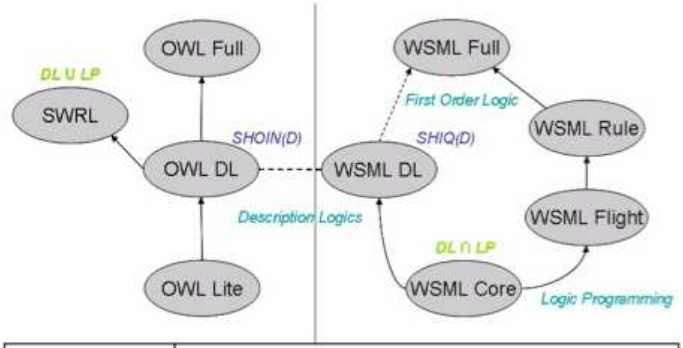
Given the known incompatibility between DL and LP (Borgida, 1996) currently the only implemented reasoning available for WSMO, when dealing with both rules and ontologies, is the one restricted to their common subfragment, i.e. DLP. Hence WSMO inference capabilities, as far as the implementation goal, to the best of our knowledge are significantly reduced with respect to a pure Semantic Web based counterpart. OWL-S, instead, is based natively in OWL whose natural extension towards rules is SWRL. SWRL decidable fragment (DL-safe rules (Motik et al., 2005)) is the largest possible union of primitives from both formalisms rather than their intersection. Current improvements of DL reasoners, now able to handle SWRL rules, allowed to develop early prototypes able to compose SWS described using OWL-S process model and mapped into SWRL (Redavid et al., 2008).

Therefore, from the perspective of two out of three main dimension of SWS infrastructures, OWL-S emerges has better equipped to encompass all the requirements dictated.

5 CONCLUSIONS

The SWS frameworks proposed in literature provide different support to dynamic orchestration. In this work we conducted a comparative study to elicit differences and analogies to do that, and remarked the capabilities necessary to enable dynamic orchestration: the needed requirements and the suitability of OWL-S and WSMO to support such requirements. As showed in the Section 2, the set of requirements are implied by means of the use cases, namely automatic discover, selection, composition and invocation, required to make dynamic the SWS orchestration. The formal language underlying the SWS frameworks is the key for an effective realization of these use cases. For this reason, in Sections 3 we described the formal support enabling reasoning on the semantic descriptions of the services offered by WSMO and OWL-S (based on OWL+SWRL and WSML, respectively). Finally, in Section 4, we have compared the formalisms underlying OWL+SWRL and WSML from the point of view of the expressive power effectively exploitable for reasoning on service descriptions. As result of this comparison, OWL-S emerges as more suitable candidates for the *dynamic* orchestration. As future work we will focus on two aspects. First, dealing with the Semantic Web natively issues, for instance, the absence of primitives for retracting knowledge due to the monotonic nature of a DL Knowledge base and the semantic interoperability. Second, the realization of the software agents able to manage the dynamic SWS orchestration by considering low-level details as, for instance, Quality of Service, Service Level Agreement and the coordination for the concrete Web services Invocation.

[7]Semantic Annotations for WSDL and XML Schema (SAWDL) - W3C Recommendation, http://www.w3.org/TR/sawsdl/- is not been considered because it does not allows the process model representation.



	Max usable expressive power combining the different types of SWS representation	
	WSMO	**OWL-S**
DL	WSML-DL (SHIQ(D))	OWL-DL (SHOIN(D))
LP	WSML-RULE (undecidable)	SWRL (undecidable)
DL & LP	WSML-Core (DL ∩ LP)	SWRL DL-Safe Rule (DL ∪ LP)

Figure 1: A comparison between the formal support provided by OWL+SWRL and WSML.

REFERENCES

Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P., editors (2003). *The Description Logic Handbook*. Cambridge University Press.

Borgida, A. (1996). On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1-2):353–367.

Cabral, L., Domingue, J., Motta, E., Payne, T. R., and Hakimpour, F. (2004). Approaches to Semantic Web Services: an Overview and Comparisons. In Bussler, C., Davies, J., Fensel, D., and Studer, R., editors, *ESWS*, volume 3053 of *Lecture Notes in Computer Science*, pages 225–239. Springer.

Enderton, H. B. (1972). *A Mathematical Introduction to Logic*. Academic Press, New York.

Grosof, B. N., Horrocks, I., Volz, R., and Decker, S. (2003). Description logic programs: combining logic programs with description logic. In *Proceedings of the 12th international conference on World Wide Web*, pages 48–57, New York, NY, USA. ACM Press.

Hakimpour, F., Sell, D., Cabral, L., Domingue, J., and Motta, E. (2005). Semantic web service composition in irs-iii: The structured approach. *cec*, 00:484–487.

Horrocks, I., Patel-Schneider, P. F., and van Harmelen, F. (2003). From SHIQ and RDF to OWL: the making of a Web Ontology Language. *J. Web Sem.*, 1(1):7–26.

Kifer, M., Lara, R., Polleres, A., Zhao, C., Keller, U., Lausen, H., and Fensel, D. (2004). A logical framework for web service discovery. In *In Proc. of the ISWC 2004 workshop on Semantic Web Services: Preparing to Meet the World of Business Applications*.

Lloyd, J. W. (1987). *Foundations of Logic Programming, 2nd Edition*. Springer-Verlag.

Lynch, D., Keeney, J., Lewis, D., and O'Sullivan, D. (2006). A Proactive approach to Semantically Oriented Service Discovery. In *Proceedings of the Second Workshop on Innovations in Web Infrastructure (IWI 2006), Edinburgh, Scotland*.

McIlraith, S. A. and Son, T. C. (2002). Adapting golog for composition of semantic web services. In Fensel, D., Giunchiglia, F., McGuinness, D. L., and Williams, M.-A., editors, *KR*, pages 482–496. Morgan Kaufmann.

McIlraith, S. A., Son, T. C., and Zeng, H. (2001). Semantic Web Services. *IEEE Intelligent Systems*, 16(2):46–53.

Motik, B., Sattler, U., and Studer, R. (2005). Query answering for owl-dl with rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60.

Peltz, C. (2003). Web Services Orchestration and Choreography. *Computer*, 36(10):46–52.

Redavid, D., Iannone, L., Payne, T. R., and Semeraro, G. (2008). OWL-S Atomic Services Composition with SWRL Rules. In An, A., Matwin, S., Ras, Z. W., and Slezak, D., editors, *ISMIS*, volume 4994 of *Lecture Notes in Computer Science*, pages 605–611. Springer.

Roman, D., Lausen, H., de Bruijn, J., Lara, R., Stollberg, M., and Polleres, A. (2005). Web Services Modeling Ontology. Journal of Applied Ontology, (1)1, 77-106, 2005.

Walton, C. (2005). Protocols for web service invocation. In *Proceedings of the AAAI Fall Symposium on Agents and the Semantic Web (ASW05)*.