# Semantic Web Services for Multi-Agent Systems Interoperability

Alda Canito[1], Gabriel Santos[1], Juan M. Corchado[2], Goreti Marreiros[1] and Zita Vale[3]

[1] Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development, Institute of Engineering - Polytechnic of Porto (ISEP/IPP), Porto, Portugal
[2] BISITE - Research Centre, University of Salamanca, Salamanca, Spain
[3] Institute of Engineering - Polytechnic of Porto (ISEP/IPP), Porto, Portugal
[1] {alrfc, gajls, mgt}@isep.ipp.pt
[2] corchado@usal.es
[3] zav@isep.ipp.pt

**Abstract.** Agent-based technologies are often used including existing web services. The outputs of some services are also frequently used as inputs for other services, including other MAS. However, while agent-based technologies can be used to provide services, these are not described using the same semantic web technologies web services use, which makes it difficult to discover, invoke and compose them with web services seamlessly. In this paper, we analyse different agent-based technologies and how these can be described using extensions to OWL-S. Additionally, we propose an architecture that facilitates these services' usage, where services of any kind can be registered and executed (semi-)automatically.

**Keywords:** Semantic Web Services, Multi-Agent Systems, OWL-S.

## 1 Introduction

The execution of complex tasks often requires the composition of several, atomic services. While a wide variety of these are available on the web as web services, the design and development of a workflow is still very time-consuming, considering that: they need to be found, they may have different interaction protocols, and proper description of their workings, inputs and outputs if often lacking [1, 2]. Semantic Web technologies have been proposed for the description of web services in order to make those descriptions richer: by providing them in a machine-readable way, the processes of discovery and composition of services by intelligent software agents become easier [2–4].

Individual agents, capable of solving specific tasks in their systems, can be seen as service providers, as well as the Multi-Agent Systems (MAS) that are able to solve more complex tasks. Both agents and MAS are capable of providing services, as is the case of decision support agent-based systems [5–7]. Additionally, existing MAS-based Decision Support Systems often execute tasks that depend on the outputs of services [5, 8], but also of other known MAS systems, as is the case of [6, 9–11]. These services are often not available outside of their environment for several reasons, e.g. the

complexity of the network configurations required to ensure secure communications. Service providers can be shifted to different servers, and systems requiring those services need to be reconfigured. To overcome these issue, one possible solution would be to have a services' catalogue where both agents and web services could register, expose the service(s) they provide, making them publicly available for other systems that might be potentially interested in using them. On the other hand, systems interested in using services would be able to search for a type of service, and as response would receive a list of available services – considering the type of service, the service provider, the expected input(s), and the result output.

In order to overcome the necessary services' heterogeneity, we propose an architecture featuring semantic description of services that that facilitates service publishing, discovery, composition and interoperability. Additionally, we present an extension to OWL-S for Agent-based services, using JADE agents as an example. Agent-based services and web services are described rather differently, with the former requiring information to be communicated with, including, but not limited to, host, port, agent identifier, performative(s), language(s) and ontology(ies). For the invocation of the later, in turn, information regarding its URI, protocol (e.g.: HTTP, HTTPS), port, technology (e.g. SOAP, REST) and method (e.g. GET, POST), among others, must be known.

This document is structured as follows: (1) Introduction, where the problem and motivations are exposed; (2) Multi-Agent Systems and Semantic Web Services, where existing approaches to semantic web service description are presented and we discuss the relevance of agent-based approaches to service composition and how intelligent agents could be described as service providers, (3) Semantic Description of Agent-Based Services, where we propose an extension to OWL-S to allow the description of agent-based services, (4) Architecture, in which we propose an architecture that would use these semantic notations to invoke MAS-based services alongside with other types of services and (4) Conclusions.

## 2 Multi-Agent Systems and Semantic Web Services

Intelligent Agents are often used to solve and simulate problems where the involved parties have different goals and objectives which require different levels of proactivity. While using Multi-Agent Systems in these scenarios is adequate, more complex scenarios can arise, as is the case of [5], where several multi-agent systems are invoked and the outcomes of them must be processed and combined in order to generate a bigger picture. Running several agent-based simulations concurrently in order to compare results is a fairly common task to perform, especially in scenarios where different configurations have immediate impact on the results or where different systems must communicate. Similarly, it is often necessary to perform simulations sequentially, where the results of the first serve as input or influence those that follow. While in some scenarios these systems directly communicate with each other [9, 11], such is not always the case [5, 8]. This problem becomes even more complex if the data must be transformed through other processes, such as those provided by services or tools [12], whose

availability must be assessed. As such, establishing which systems and services are available and how they can be combined becomes an important issue.

As proposed by McIlraith in [3], describing web services in a semantically rich way – not only in terms of their inputs and outputs, but also by describing the inner processes of these services themselves and the tasks they perform – allows for a bigger automation in the processes of service discovery, composition and compensation [2, 13–15]. The principles of interoperability and coordination between agents follow the same vision as the semantic web [4]; however, while web services represent atomic, mostly stateless tasks, intelligent agents are proactive entities with specific goals [16]. This proactivity manifests, among other things, in locating services and partners which will help the agent to fulfil its tasks. In [17], the importance of the semantic description of services for this task is discussed, establishing that web services and agents are similar when it comes to their discovery and that any matching to be made between different service groundings must ultimately rely on semantic abstractions.

While there are many concurrent semantic approaches to the description of we services [18–22], OWL-S includes a number of concepts and properties to allow the proper description of SOAP services, with extensions also available for the RESTful kind [23]. Additionally, with OWL-S it is possible to describe not only atomic services, but also workflows resulting of the composition of several atomic ones. In order to use OWL-S to also describe agents and multi-agent systems, new concepts must be added to the ontology. As such, we will study the properties of agent-based technologies in order to establish how these can be described, and then proposing the necessary additions to OWL-S.

## 3 Semantic Description of agent-based services

OWL-S includes a number of concepts and properties to allow the proper description of SOAP services, with extensions are also available for RESTful services [21]. In our proposed architecture, agents will also be described with OWL-S; we will discuss if its existing concepts and properties are sufficient for describing intelligent agents or if extensions are necessary.

OWL-S is divided in three main components: (i) Service Profile, (ii) Process Model and (iii) Grounding. The first is meant to be read by humans and features the name of the service, its description, provider, limitations and other relevant information. The Process Model describes how the service works, describing its inputs and outputs, preconditions and effects. Finally, Grounding specifies interaction details such as the interaction protocol and message formats [24].

Agents do not expose their services through any standardized description formats like WSDL [25] or WADL [26]. In order to define the properties that an entity must know in order to interact with agents – in our case, specifically JADE agents [27] – an abstract agent ontology was defined, from which *JADE* agent has been extended. Figure ***Fig. 1*** introduces the abstract *Software Agent* ontology including possible Subclasses, such as *OAA* [28], *ZEUS* [29], *Jadex* [30], *EMERALD* [31], *SeSAm* [32] and *JADE* agents.

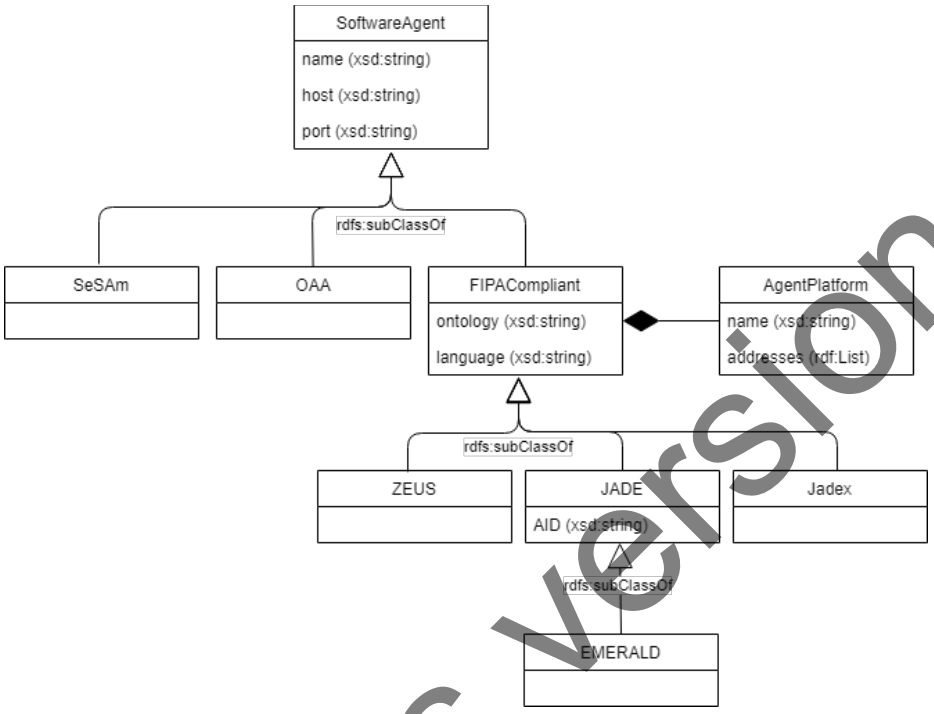


**Fig. 1.** The *Software Agent* ontology.

The *Software Agent* ontology is described by the *SoftwareAgent* abstract Class which is composed by the Data Properties *name*, *host*, and *port* that identify the agent. The *FIPACompliant* agent SubClass includes the *ontology* and *language* Data Properties for the effective communication with the agent, and also the *AgentPlatform* Class which identifies the agent platform *name* and the *addresses* list. From the *FIPACompliant* agent Class *ZEUS* agent, *JADE* agent, and *Jadex* agent SubClasses are derived. The *JADE* agent Class is expressed by the Data Property *AID* which is the agent unique identifier in the agents' community. The *EMERALD* agent is Subclass of JADE.

By definition, Software Agents run on a given host through a specific port and have a unique name for identification purposes. While additional information may be required for interaction with an agent, these basic properties are used in the *SoftwareAgent* abstract Class. Other properties, which may be required by different agent implementations, are exposed in different Subclasses.

FIPA provides a number of standards for communication between heterogeneous agents and the services they provide [33], specifying that all agents must state which ontology they use to describe their message and the language they use (e.g.: XML, JSON, RDF/XML, TURTLE, etc.). Therefore, the *FIPACompliant* agent Subclass includes the *ontology* and *language* Data Properties for the effective communication with the agent, and the *AgentPlatform* Class which identifies the agent platform *name* and the *addresses* list. ZEUS, JADE, EMERALD, and Jadex-based agents, being FIPA

compliant, are expressed though four new Subclasses, named accordingly. JADE agents, in particular, require an identification for individual agents in a given community, which is provided by the Data Property AID (agent unique identifier).

The *JADE* agent Class supplies atomic services via the *JadeAtomicService* Class. The interaction protocol for this scenario, i.e. its Grounding, is provided by the *JadeGrounding* Class. The relationships between these entities and those supplied by the OWL-S and RESTFul Grounding are represented in Figure ***Fig. 2***, below:

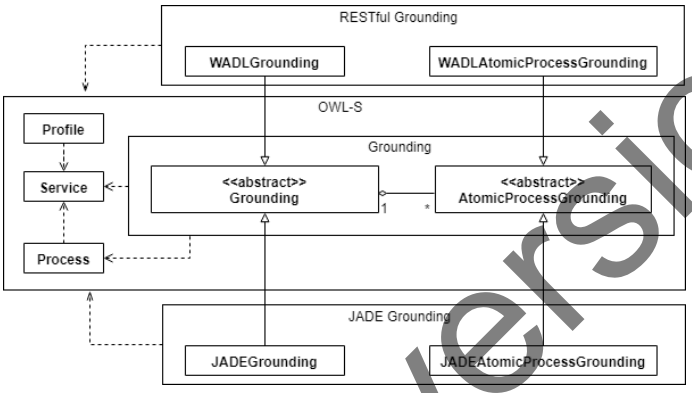


**Fig. 2.** JADE and WALD Grounding Classes.

For our example, let's consider a Forecast Service, which is provided either by a RESTFul web service and by an agent. This is a generic forecasting service based on an artificial neural network algorithm. As input, the algorithm expects to receive a training set composed of two arrays, and a testing set array to determine the output. The training set arrays are the *TrainInput* and *TrainOutput*. The *TrainInput*, is an array of arrays where each array outputs the value of the corresponding position of the *TrainOutput* array. Given the training set, the algorithm determines the output of the *TestInput* array. Fig. 3 presents the *Forecast* ontology.

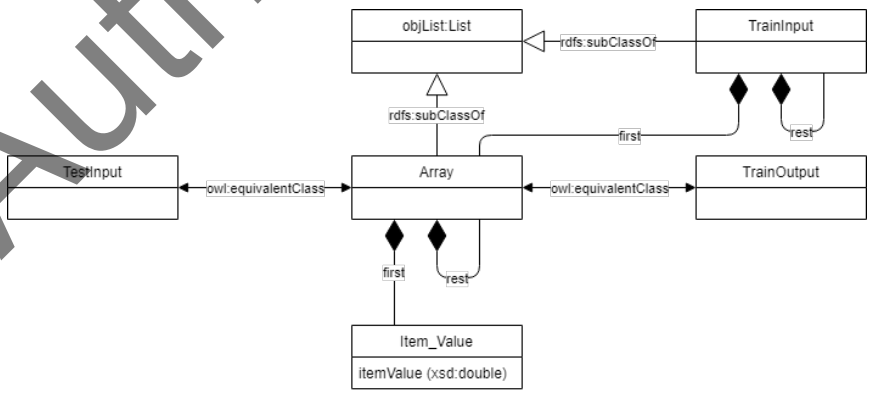


**Fig. 3.** The *Forecast* ontology.

The *Forecast* ontology is composed by the classes *Array*, *TestInput*, *TrainInput*, *TrainOutput* and *Item_Value*. The *Array* Class is an abstract class equivalent to the *objList:List*[1] Class in conjunction with the Object Property *first* with range *Item_Value*, and the Object Property *rest* with recursive range *Array*. The *objList:List* Class is part of the OWL-S ontology and it is used due to the need of representing lists of objects. The *Array* abstract Class has been defined to be reused in both *TestInput* and *TrainOutput* Classes, being these last equivalents to the *Array* Class. The *Item_Value* Class is defined by the Data Property *itemValue* which holds the double value of each item. Finally, the *TrainInput* Class is equivalent to the *objList:List* Class in conjunction with the Object Property *first* with range *Array*, and the Object Property *rest* with recursive range *TrainInput*.

Now that the entities regarding inputs and outputs of the service are established, a definition of the service itself is in order. Figure ***Fig. 4*** introduces the Forecast Service definition.

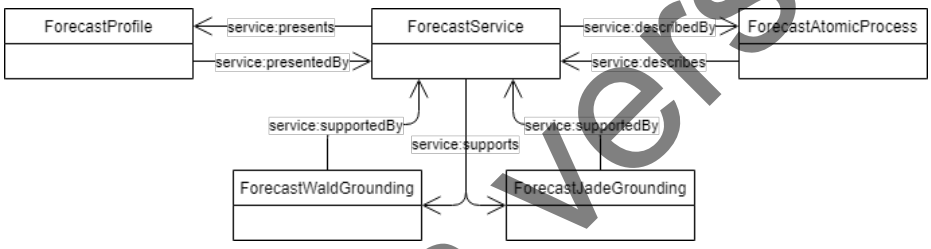


**Fig. 4.** Forecast Service definition.

The *ForecastService* presents a *ForecastProfile*, is described by the *ForecastAtomicProcess*, and supports both *ForecastWaldGrounding* and *ForecastJadeGrounding*. Two different Groundings are provided for this Forecast Service: *ForecastWaldGrounding* and *ForecastJadeGrounding,* instances of *WadlGrounding* and *JadeGrounding* respectively, representing the two possible ways to invoke this service.

The *WadlGrounding* adds a semantic definition to the parameters of the service, but when it comes to invocation, it ultimately refers to its WADL file for details. *JadeGrounding* provides the description for the forecast service provided by the software agent. It is very similar to the *WadlGrounding* definition, being the main difference the use of the *Software Agent* semantic model instead of the WADL file definition for the service's invocation.

The complete service definition, along with instantiation files, as well as the semantic data models are publicly available online[2] for appreciation.



[1] http://www.daml.org/services/owl-s/1.2/generic/ObjectList.owl#List

[2] http://www.gecad.isep.ipp.pt/epia/19/services/forecast/

## 4 Architecture

The proposed architecture must: (i) enable semantic services to register/deregister in/from the platform; (ii) enable client applications to search for services by different filter parameters and (iii) provide clients with machine-readable information about the available services (including services' parameterization, inputs and outputs). This way, systems will be able to perform these processes automatically when a service is required. **Fig. *3*** introduces the application level architecture, where three different entities are easily identified: (i) the Service's Catalogue, (ii) Service Providers (e.g. Agent-based or web service) and (iii) the Services' Clients.

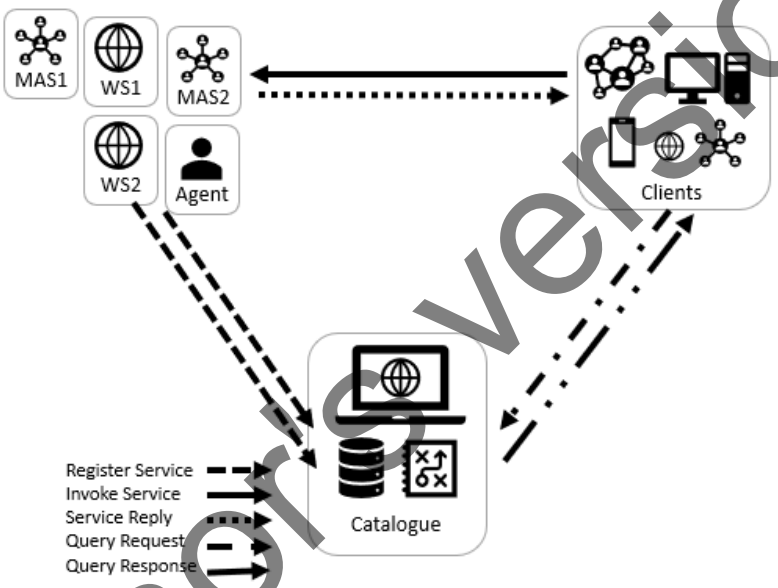


**Fig. 5.** Application level architecture.

Service providers can be any Agent, Multi-Agent System or Web Service that is able to execute a specific atomic task. This task must clearly specify its input and output parameters and supply a description as to what processes it entails.

Concurrently, the Services' Client represents the client applications searching for available services. These applications can use the information provided by the Services' Broker response to request for the services' execution directly to the service provider(s). It should be noticed that any registered service is also a potential service client: for instance, an agent-based service may be composed by several web-services, also available independently in the broker's application platform.

The Catalogue is the main application. It is responsible for proving semantic service's registration, deregistration and search services. The Catalogue's modules are distributed through three layers, namely: (i) Client Interface, which fulfils tasks such as Service Registration and De-Registration, supplies descriptions of services and allows Clients to search for the services they need; (ii) Composition Suggestion, which is used

for supplying descriptions and services when a specific task requires more than one known service to be fulfilled, and (iii) Service Database, where the descriptions of the known services are stored and which can be queried via a SPARQL endpoint. The layers and modules, along with the relationships between them are shown in ***Fig. 6***, below:

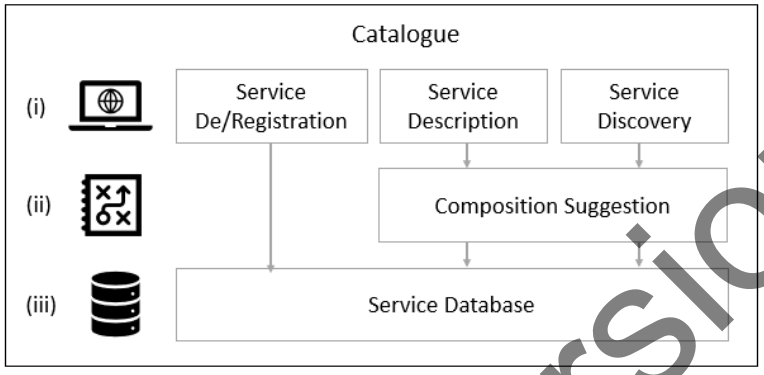


**Fig. 6.** The Catalogue's inner modules

Upon registration, service providers must announce what type of service they supply, a description of its purpose, where and how it can be invoked and its input and output parameters. When it comes to service discovery, different types of searches can be performed [34], allowing not only for syntactic similarity but for semantic similarity as well: e.g., by being able to compute how similar two ontological entities are or if mappings between them are available. Additionally, as single service may not be able to fulfil a certain task, but the combination of two or more known services may generate the desired outcome, the Composition Suggestion module can be invoked. As the name suggests, it will try to break down a discovery request into multiple queries and confirm is a composition can be made with those results. Different techniques of service discovery and composition will be employed in this task. If the Client agrees with the suggestion, it will be stored in the database as a composite service, with its own OWL-S description.

## 5 Conclusions

In order to discover, invoke and compose agent-based alongside with web services, a common framework for their description is required. Existing semantic web technologies for the description of web services exist, with OWL-S being one of the most commonly applied. In this paper, we explored the possibility of viewing agent-based solutions as web services - as those provided in SOAP or RESTful applications - but with a different interaction protocol. We therefore analysed the specific needs of agent-based solutions and how OWL-S could be extended to properly represent them. Finally, we proposed an architecture that would allow entities, agents or otherwise, to register, deregister an invoke each other seamlessly, while also providing services for discovery and composition suggestion.

The most common implementations of OWL-S ontology, namely the instantiation of Grounding ontology module, end up by pointing to an XML file. In the future, it would be interesting to study the implications of using a purely semantic description for the invocation of all services, as we proposed in this paper to be done for agents, such that no WADL or WSLD files would be necessary, especially when it comes to service definition change over time.

## Acknowledgements

The present work has received funding from FEDER Funds through NORTE2020 program and from National Funds through Fundação para a Ciência e a Tecnologia (FCT) under the project UID/EEA/00760/2019. Gabriel Santos is supported by national funds through FCT PhD studentship with reference SFRH/BD/118487/2016.

## References

1. Lemos, A.L., Daniel, F., Benatallah, B.: Web Service Composition. ACM Comput. Surv. 48, 1–41 (2015).
2. Klusch, M., Kapahnke, P., Schulte, S., Lecue, F., Bernstein, A.: Semantic Web Service Search: A Brief Survey. KI - Künstliche Intelligenz. 30, 139–147 (2016).
3. McIlraith, S.A., Son, T.C., Honglei Zeng: Semantic Web services. IEEE Intell. Syst. 16, 46–53 (2001).
4. Huhns, M.N.: Agents as Web services. IEEE Internet Comput. 6, 93–95 (2002).
5. Teixeira, B., Pinto, T., Silva, F., Santos, G., Praça, I., Vale, Z.: Multi-Agent Decision Support Tool to Enable Interoperability among Heterogeneous Energy Systems. Appl. Sci. 8, 328 (2018).
6. Pinto, T., Morais, H., Sousa, T.M., Sousa, T., Vale, Z., Praca, I., Faia, R., Pires, E.J.S.: Adaptive Portfolio Optimization for Multiple Electricity Markets Participation. IEEE Trans. Neural Networks Learn. Syst. 27, 1720–1733 (2016).
7. Pinto, T., Vale, Z., Praça, I., Pires, E.J.S., Lopes, F.: Decision Support for Energy Contracts Negotiation with Game Theory and Adaptive Learning. Energies. 8, 9817–9842 (2015).
8. Teixeira, B., Silva, F., Pinto, T., Santos, G., Praca, I., Vale, Z.: TOOCC: Enabling heterogeneous systems interoperability in the study of energy systems. In: 2017 IEEE Power & Energy Society General Meeting. pp. 1–5. IEEE (2017).
9. Santos, G., Femandes, F., Pinto, T., Silva, M., Abrishambaf, O., Morais, H., Vale, Z.: House management system with real and virtual resources: Energy efficiency in residential microgrid. In: 2016 Global Information Infrastructure and Networking Symposium (GIIS). pp. 1–6. IEEE (2016).
10. Santos, G., Pinto, T., Praça, I., Vale, Z.: MASCEM: Optimizing the performance of a multi-agent system. Energy. 111, 513–524 (2016).
11. Kravari, K., Bassiliades, N., Boley, H.: Cross-community interoperation between knowledge-based multi-agent systems: A study on EMERALD and Rule Responder. Expert Syst. Appl. 39, 9571–9587 (2012).

12. Carneiro, J., Alves, P., Marreiros, G., Novais, P.: A Multi-agent System Framework for Dialogue Games in the Group Decision-Making Context. Presented at the (2019).
13. Klusch, M., Fries, B., Sycara, K.: Automated semantic web service discovery with OWLS-MX. In: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems - AAMAS '06. p. 915. ACM Press, New York, New York, USA (2006).
14. Lord, P., Alper, P., Wroe, C., Goble, C.: Feta: A Light-Weight Architecture for User Oriented Semantic Service Discovery. Presented at the (2005).
15. Rodriguez-Mier, P., Pedrinaci, C., Lama, M., Mucientes, M.: An Integrated Semantic Web Service Discovery and Composition Framework. IEEE Trans. Serv. Comput. 9, 537–550 (2016).
16. Greenwood, D., Calisti, M.: Engineering web service - agent integration. In: 2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583). pp. 1918–1925. IEEE.
17. Martin, D., Burstein, M., McIlraith, S., Paolucci, M., Sycara, K.: OWL-S and Agent-Based Systems. In: Extending Web Services Technologies. pp. 53–77. Springer-Verlag, New York.
18. Web Service Modeling Language (WSML), https://www.w3.org/Submission/WSML/.
19. Semantic Annotations for WSDL and XML Schema, https://www.w3.org/TR/sawsdl/.
20. Pedrinaci, C., Cardoso, J., Leidig, T.: Linked USDL: A Vocabulary for Web-Scale Service Trading. Presented at the (2014).
21. Kopecký, J., Gomadam, K., Vitvar, T.: hRESTS: An HTML Microformat for Describing RESTful Web Services. In: 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. pp. 619–625. IEEE (2008).
22. SA-REST: Semantic Annotation of Web Resources, https://www.w3.org/Submission/SA-REST/.
23. Filho, O.F.F., Ferreira, M.A.G.V.: SEMANTIC WEB SERVICES: A RESTFUL APPROACH. (2009).
24. OWL-S: Semantic Markup for Web Services, http://www.ai.sri.com/~daml/services/owl-s/1.2/overview/.
25. Christensen, E., Curbera, F., Meredith, G., Weerawarana, S.: Web Service Definition Language (WSDL), https://www.w3.org/TR/2001/NOTE-wsdl-20010315.
26. Hadley, M.: Web Application Description Language, https://www.w3.org/Submission/wadl/.
27. Bellifemine, F.L., Caire, G., Greenwood, D.: Developing multi-agent systems with JADE. John Wiley (2007).
28. The Open Agent Architecture, http://www.ai.sri.com/~oaa/.
29. The Zeus Technical Manual, http://zeusagent.sourceforge.net/docs/techmanual/TOC.html.
30. Braubach, L., Pokahr, A., Lamersdorf, W.: Jadex: A BDI-Agent System Combining Middleware and Reasoning. In: Software Agent-Based Applications, Platforms and Development Kits. pp. 143–168. Birkhäuser-Verlag, Basel (2005).
31. Kravari, K., Kontopoulos, E., Bassiliades, N.: EMERALD: A Multi-Agent System for Knowledge-Based Reasoning Interoperability in the Semantic Web. Presented at the (2010).

32. Klügl, F., Herrler, R., Klügl, F., Herrler, R., Fehler, M.: SeSAm: implementation of agent-based simulation using visual programming Policy Agents I, II, III View project SeSAm: Implementation of Agent-Based Simulation Using Visual Programming. (2006).
33. Foundation for Intelligent Physical Agents: FIPA A.C.L., http://www.fipa.org/specs/fipa00061/SC00061G.html.
34. Negi, A., Kaur, P.: Examination of Sense Significance in Semantic Web Services Discovery. Presented at the (2019).