Semi-automated Generation of Semantic Service Descriptions

Nils Masuch, Philipp Brock, and Sahin Albayrak

Abstract. The increasing complexity and dynamics of distributed systems make the management and integration of new services more and more difficult. Automation processes for the definition, selection and composition of services for goal achievement can produce reliefs. However, a high degree of self-explanation of the services is obligatory for this. Today's multi-agent frameworks only provide insufficient solutions to this. Within this paper we will outline an approach, which enables the integration of semantic service descriptions into multi-agent systems with reasonable effort.

Keywords: OWL-S, Multi-Agent Framwork, Semantic Service Description, JIAC.

1 Introduction

In the past, the demand for modular, distributed and dynamic computer systems has increased rapidly. The reasons for this are manifold, ranging from maintainability and reliability to adaptability aspects, just to name a few [14]. In the field of multiagent systems (MAS) many of the current approaches try to account for these requirements. However, these systems usually are characterised by a high degree of complexity, which makes it difficult to provide a transparent way to define, lookup and invoke functionalities of software entities, such as agents. A promising approach to this is the service paradigm which leads to the development of service-oriented architectures (SOA) and is an ideal complement to multiagent systems [13]. One of its inherent strengths is typically the definition of a clear autonomy of each service, which means that it is represented as a separate module. Further, services are designed for enhancing the interoperability which is one of the key issues for distributed systems. Especially when talking about huge computer systems with different providers and parties involved these parameter are essential.

Nils Masuch · Philipp Brock · Sahin Albayrak
DAI-Labor, TU Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
e-mail: nils.masuch@dai-labor.de

J.B. Pérez et al. (Eds.): *Trends in Prac. Appl. of Agents & Multiagent Syst.*, AISC 221, pp. 155–162.
DOI: 10.1007/978-3-319-00563-8_19

Beyond that, these systems also require mechanisms to adapt their behaviour to the environment or to current offers, since software services might be added or removed dynamically. In many approaches of current software systems the dynamic aspect is not regarded sufficiently. In order to encounter these challenges automated techniques that minimize the necessary intervention by developers are a promising approach [10]. One important issue in the automation process is the automated interpretation of services, allowing for a discovery of suitable services. Therefore the system has to be able to understand specific service parameters such as preconditions or effects. In practice a lot of so-called semantic service matcher components have been developed (e.g. [7], [9]), but, to the best of our knowledge, none of them has been fully integrated into a multi-agent framework. Doing so, the basic functionality is provided for the next step, which is the autonomic composition of services to reach a certain goal. The basic condition to develop these automation processes is a knowledge representation that enriches descriptions semantically, often described as ontologies. In practice the enrichment of real applications, which are more and more developed based on the agent-oriented programming paradigm (AOP), by ontologies is currently very troublesome if available at all. This means there is a clear lack of concepts combining the AOP world, which is widely used for programming, and the world of standardized formal semantics not only at runtime but also at design time.

In this paper we present a conceptual approach for integrating semantic service information into a MAS, taking the first step towards standard-based, dynamic and automatic service matching and composition systems.

In the remainder of this paper we will give some background information about the semantic service description language OWL-S. Thereupon we will introduce JIAC V (Java Intelligent Componentware, Version 5) [5], our own Java-based multi-agent development framework. Based on this framework, we will then present our approach of integrating semantic service descriptions into the development methodology of dynamic, distributed systems based on JIAC. Consequently we will present a use case, in which the dynamic invocation of services can be gainful. Finally, we will conclude with a short outlook of our next steps.

2 Background

In the context of semantic enrichment of services the semantic web community developed multiple approaches, such as SAWSDL[1], WSMO [2], hRests [3] and OWL-S [1] which are focusing on many identical aspects but are at the same time distinct in some relevant attributes. Due to the lack of space we are not able to discuss them thoroughly and refer to [8]. In the following we will focus on one of the most comprehensive ones, namely OWL-S, and describe it in more detail.

[1] `http://www.w3.org/2002/ws/sawsdl/`

2.1 OWL-S

OWL-S is based upon on the Web Ontology Language (OWL)[2], more precisely it is a specific OWL ontology, which is structured for describing service attributes. The ontology is split up into three parts, namely Service Profile, Service Grounding and Service Model.

The Service Profile describes all relevant parameters of the service for searching components. The most important attributes are the so-called IOPEs, which stands for Input, Output, Precondition and Effects. The Service Model enables the description of the service's processment. It defines how the service works, and if necessary which interactions will take place. For example a Service Model can combine multiple atomic processes to more complex components using constructs like loops or conditional expressions. The Service Grounding defines the invocation details of the service. The OWL-S standard leaves the kind of invocation technique open, allowing the integration of different transport protocols. For the definition of preconditions and effects OWL-S does not adhere to a specific rule language. One of the propositions to use is the Semantic Web Rule Language (SWRL) [6].

SWRL has been designed to formulate implication rules, which can be used on entities from OWL ontologies. A SWRL rule consists of two parts, the antecedent (body) and the consequent (head). For the evaluation of the rules the information from the head will be added to the knowledge base, if the conditions in the body are fulfilled. SWRL rules are comprised of any number of conjugated atoms. In order to evaluate the rules in acceptable time, the language must remain decidable. Therefore all variables which are required in the head part, have to be already present in the body part and further no complex class constructs may be used. Using SWRL according this way, it can be used together with OWL-DL and remains decidable.

2.2 JIAC V

Java **I**ntelligent **A**gent **C**omponentware V (JIAC V) is, as the name indicates, an open Java-based multi-agent framework currently being developed at Technical University of Berlin. The framework combines the agent-oriented with a service-oriented view. Services can either be invoked via explicit invocations of specific services or via an incomplete service description that leads to a service matching process returning the most appropriate one. Agents in JIAC V are located on Nodes. These nodes can be executed on different machines and are also responsible for middleware management and communication issues. For example each node manages the dynamics via a service- and an agent-directory according to FIPA Agent Management Specification[3].

The agents in JIAC V are component-based. Their structure can be very roughly divided into an execution cycle, a local memory and Agent Beans. The memory is a tuple space implementation, and provides parallel access of the data to all agent

[2] `http://www.w3.org/2004/OWL/`

[3] `http://www.fipa.org/specs/fipa00023/XC00023H.html`

components. The components also have the opportunity to sign up for listeners on memory data, in order to be informed about every change. The execution cycle is the heart of JIAC agents. This main thread processes existing events at regular intervals.

The different functionalities and behavioural rules of an agent are encapsulated in Agent Beans. These functions are defined as actions and can be made available in different scopes. Thus, an action can be defined as being visible to the agent itself, to all agents of one node or globally to all nodes.

One central component of JIAC V is the Semantic Service Matcher SeMa^2 [9], which enables the system to match OWL-S based service requests and advertisements. The matching process follows a hybrid approach, which considers both syntactic and semantic elements. For the evaluation of rules SeMa^2 utilises a OWL-DL Reasoner named Pellet [11], which is able to reason over SWRL constructs. The overall evaluation result is composed of multiple matching algorithms, namely Service Name Matching, Text Similarity Matching, Taxonomy Matching, Rule Structure Matching and Rule Reasoning. As a result the service matcher provides the requester with an assessment of all suitable service options in a ranking order.

3 Approach

As described within the previous chapter, OWL-S is a very comprehensive service desciption ontology with a huge flexibility in describing the IOPEs. However, this complexity leads to a significant drawback when it comes to the creation and integration into an application environment, since there are no sufficient tools available. In the following we propose our conceptual approach of integrating OWL-S into JIAC V by describing the different methodological steps towards a dynamic service platform.

The first essential step for an efficient interoperability is the definition of a domain ontology (see Figure 1). When developing project applications based on JIAC V often Eclipse Modeling Framework (EMF[4]) is being used. EMF comes with a direct transformation into Java. Further, there is an intensive research in transforming EMF models into OWL ontologies [12, 4]. Consequently, EMF is an ideal candidate to describe ontologies and transform them to the implementation language (Java) and the service description language (OWL / OWL-S).

As described in the JIAC V section, Java methods developed within Agent Beans can easily be declared as JIAC Actions by annotating the method heads. Figure 2 shows such a method head exemplarily. During runtime all relevant service information is being extracted and added to the agents service directories. Therefore the developer can fully concentrate on the implementation of the service's functionality.

However, what is missing here is the semantic enrichment of services for a detailed analysis of their functionality. Figure 3 shows a process diagramm illustrating the workflow of how semantical information should be enhanced to the core service data. At first the annotation head will be extended by a further tag named *semantify*, which specifies that the service should be descibed via an OWL-S ontology. Then,

[4] `http://www.eclipse.org/modeling/emf/`

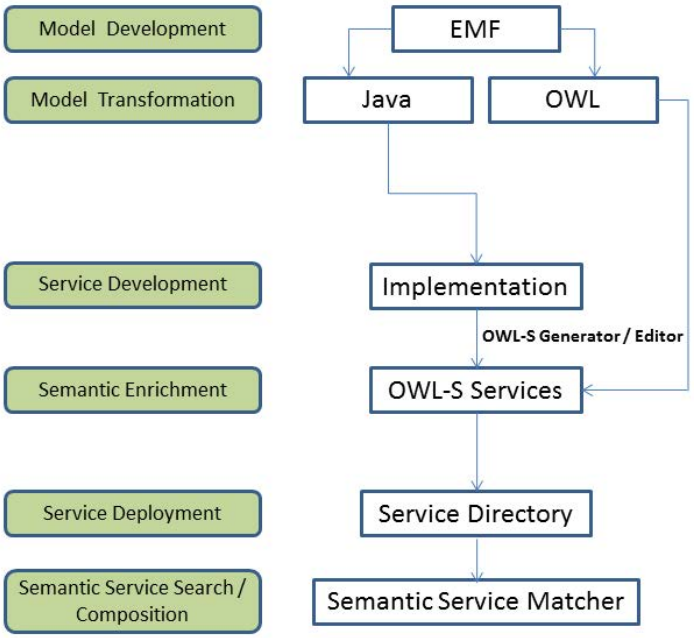

Fig. 1 Service integration methodology

```
@Expose(name = MobilityServiceBeanActions.ACTION_SEARCH_ROUTING_OPTIONS, scope = ActionScope.NODE)
public ArrayList<JourneyEvent> getJourneyOptions (Position location1, Position location2, Date time) {
```

Fig. 2 The JIAC V annotation mechanism

at compile time, the system checks whether a service description for this action has already been created. If not, a component named *OWL-S Generator* extracts all available information of the method head and the comment part (for now service name, service description, Java input params, Java output param) and maps the input and output parameters to the concepts of the corresponding ontology, which has to be defined as the reference ontology within the JIAC application settings. Consequently, a first OWL-S description will be created and loaded into an Eclipse based Editor. The developer is now able to modify and extend the service description (e.g. preconditions and effects). After the adaption phase the OWL-S description is stored persistently and a reference from the service to the description is being stored in the service header. At runtime the framework deploys and registers the service within the agent's service directory.

In order to find and integrate an appropriate service into a new application, the developer has to be able to specify the desired functionalities. Therefore there has to be some sort of Semantic Service Search Editor at design time, which is supporting the definitions of IOPEs and further restrictions, such as QoS parameters. The search

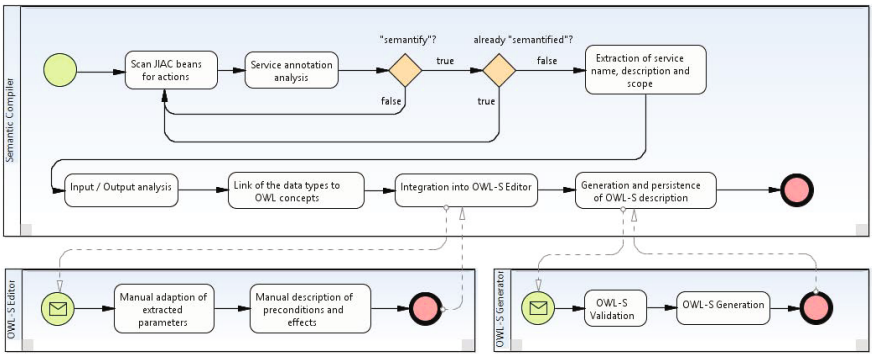

Fig. 3 Workflow of the OWL-S generation process

algorithm is then based upon the semantic service matcher SeMa^2 and returns all suitable and available services. The developer can select the most appropriate one, which is then hard coded into the invocation details. In a next step a hybrid approach is intended, in which the developer defines a specific service, but at times the service is not available, SeMa^2 is searching for an alternative and, if available, invokes it instead. Finally, the service matcher should be able to compose multiple services to reach a specific goal.

Our main intention with this approach is to enable the developer to easily utilise semantic service descriptions in order to have an increased flexibility and automation in highly dynamic application environments. In the next chapter we will identify a use case in which the additional overhead is justified.

4 Example

A domain that is well-suited for the integration of service automation techniques into huge computer systems is the field of mobility and transportation. Especially when thinking about urban traffic, people are asking for more flexible solutions regarding their personal needs. Therefore intermodal solutions or new approaches such as spontaneous car- or ride-sharing become more and more popular. Integrating different mobility providers into a distributed platform is an ideal use-case for an assistance system where dynamic aspects such as the availability of resources and context information (e.g. traffic jams) have to be considered for creating a user-specific solution. The combination of these offers can lead to composite services that support the user with intermodal and adaptable travel assistance. In this scenario services will be dynamically added or removed. Having a platform with comprehensive, but distributed information about mobility and routing options together with value added services offers the opportunity to develop a highly automatic service

assistance planner. However, it requires an architecture which enables the developers to easily extend their services with semantical information, which is interpretable for standard-based matching and composition components. The concept described in the paper shall be a basic step towards this goal.

5 Conclusion

In this paper we presented a methodological and procedural approach of integrating semantic service descriptions into a multi-agent framework, in order to address the challenges of highly dynamic and complex application platforms. Our approach shows how OWL-S descriptions can be semi-automatically generated via standard Java method information and multi-agent framework specific annotations and integrated into the running system. Doing so, the software developer will be unburdened from declaring all service description information manually.

In a next step, we will implement the described concept within the JIAC V agent framework and upon that we aim to develop an open, intermodal mobility service platform as a first evaluation use case within a national government funded project.

References

1. Ankolekar, A., Burstein, M., Hobbs, J.R., Lassila, O., Martin, D.L., Mcilraith, S.A., Narayanan, S., Paolucci, M., Payne, T., Sycara, K., Zeng, H.: DAML-S: Semantic Markup for Web Services
2. Feier, C., Roman, D., Polleres, A., Domingue, J., Fensel, D.: Towards intelligent web services: The web service modeling ontology. In: International Conference on Intelligent Computing, ICIC (2005)
3. Gomadam, K., Vitvar, T.: hRESTS: an HTML Microformat for Describing RESTful Web Services. In: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2008 (2008)
4. Hillairet, G., Bertrand, F., Lafaye, J., et al.: Bridging EMF applications and RDF data sources. In: Proceedings of the 4th International Workshop on Semantic Web Enabled Software Engineering, SWESE (2008)
5. Hirsch, B., Konnerth, T., Heßler, A.: Merging Agents and Services — the JIAC Agent Platform. In: Bordini, R.H., Dastani, M., Dix, J., El Fallah Seghrouchni, A. (eds.) Multi-Agent Programming: Languages, Tools and Applications, pp. 159–185. Springer (2009)
6. Horrocks, I., Patel-Schneider, P., Boley, H., Tabet, S., Grosof, B., Dean, M., et al.: SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission 21, 79 (2004)
7. Kapahnke, P., Klusch, M.: Adaptive Hybrid Selection of Semantic Services: The iSeM Matchmaker. In: Blake, M.B., Cabral, L., König-Ries, B., Küster, U., Martin, D. (eds.) Semantic Web Services, pp. 63–82. Springer, Heidelberg (2012)
8. Klusch, M.: Semantic web service description. In: Schumacher, M., Schuldt, H., Helin, H. (eds.) CASCOM: Intelligent Service Coordination in the Semantic Web. Whitestein Series in Software Agent Technologies and Autonomic Computing, pp. 31–57. Birkhäuser, Basel (2008)

9. Masuch, N., Hirsch, B., Burkhardt, M., Heßler, A., Albayrak, S.: SeMa^2: A Hybrid Semantic Service Matching Approach. In: Blake, M.B., Cabral, L., König-Ries, B., Küster, U., Martin, D. (eds.) Semantic Web Services, pp. 35–47. Springer, Heidelberg (2012)
10. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-Oriented Computing: a Research Roadmap. Int. J. Cooperative Inf. Syst. 17(2), 223–255 (2008)
11. Parsia, B., Sirin, E.: Pellet: An OWL DL Reasoner. In: Proceedings of the International Workshop on Description Logics, p. 2003 (2004)
12. Rahmani, T., Oberle, D., Dahms, M.: An adjustable transformation from OWL to Ecore. In: Petriu, D.C., Rouquette, N., Haugen, Ø. (eds.) MODELS 2010, Part II. LNCS, vol. 6395, pp. 243–257. Springer, Heidelberg (2010)
13. Ribeiro, L., Barata, J., Colombo, A.: MAS and SOA: A Case Study Exploring Principles and Technologies to Support Self-Properties in Assembly Systems. In: Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2008, pp. 192–197 (2008)
14. Wooldridge, M.J.: An Introduction to MultiAgent Systems, 2nd edn. Wiley (2009)