

Towards Toolipse 2: Tool Support for the Next Generation Agent Framework

Michael Burkhardt
DAI-Labor, Technische
Universität Berlin
michael.burkhardt@dai-
labor.de

Marco Lützenberger
DAI-Labor, Technische
Universität Berlin
marco.luetzenberger@dai-
labor.de

Nils Masuch
DAI-Labor, Technische
Universität Berlin
nils.masuch@dai-
labor.de

ABSTRACT

Multi-agent system development is a complex task. In this paper we describe our idea of supporting the multi-agent system development within the JIAC framework by a unified tool solution. We illustrate an approach of providing a development platform, which enables comfortable, quick and comprehensive multi-agent system design and provides semantic searching for available services. At this we start with our latest three feature extensions to the JIAC framework, each one developed in the scope of a diploma thesis, and describe our planned adjustments and ideas to achieve the desired functionality.

1. INTRODUCTION

Over the last decade, Agent Oriented Software Engineering (AOSE) has gained attention as a suitable methodology for providing quality assurance within software development processes [4].

In order to counter nowadays requirements, the *DAI Labor* has developed *JIAC V*, the fifth version of its *JIAC* (Java Intelligent Agent Componentware) serviceware framework.

In this work, we describe our approach in developing such a tool solution by combining our latest three framework extensions — each one a result of a separate diploma thesis [5, 8, 11]. However, since we pursue the higher goal of providing a comprehensive development tool for *JIAC V*, we describe our ideas concerning this long-term task as well.

This paper is structured as follows: In the next section we will introduce the reader to the *JIAC V* development and provide a description of our three latest extensions to the agent framework. Thereafter we will describe our approach in combining these extensions to a development tool. Finally, we discuss related work and wrap up with the conclusion.

2. JIAC V TOOLS

The *JIAC V* serviceware framework [6] is a Java based agent framework which has been developed within industry- and government-funded projects. It complies with industrial requirements such as software standards, security and management, and allows the implementation of open, distributed

and scalable multi-agent systems in which agents are able to offer their abilities in a service oriented way. *JIAC V* combines agent technology with a service oriented approach. In order to extend the framework's capabilities and provide comfortable and effective agent engineering, we lately extended *JIAC V* with several elaborate features, which we describe below.

JADLedit is an Eclipse based editor for *JADL++* [5], a programming language which has been designed with particular focus on an easy usage to assure comfortable first steps in agent oriented programming. *JADL++* has been developed within the diploma thesis [5]. As one of its main features, *JADL++* uses the knowledge representation language *OWL* [3] as semantic foundation for its complex data types. **JADLedit** is divided into two parts, a *JADL++* source code editor which provides helpful features like syntax highlighting, code completion, error marking and many more. The second part is an ontology browser, which displays detailed information on the included ontologies, such as their classes and their properties.

The **Agent World Editor** [9], or **AWE**, is a tool which supports framework comprehensive design and deployment of multi-agent systems. **AWE** provides visual engineering and conveniently represents even complex multi-agent systems by means of an expressive notation in a single diagram. The tool uses a highly generic domain model as semantic foundation. A code generation routine translates the domain model instance of the current diagram into executable code. At this, **AWE** employs a sophisticated extension mechanism. **AWE** is implemented as plug-in to the development environment *Eclipse*¹ and thus employs its entire plug-in principle. To this end, support for each agent framework is encapsulated within a plug-in, which is loosely coupled to the **AWE** base application. **AWE**'s base application now defines a set of concepts and allows the developer to reference them for his *MAS* setup, while each Extension Plug-in defines a mapping from the abstract concepts defined by **AWE** to an according implementation of the respective agent framework. A standardised structure of these Extension Plug-ins encourages custom developments. Thus far, we implemented plug-ins for *JIAC V*, its predecessor *JIAC IV* and *JIAC*'s lightweight derivative *MicroJIAC*.

AWE has been developed within the diploma thesis [8].

The *JIAC SEMA*ntic *SE*rvice *MA*tcher (*SeMa*²) [11] provides a matching algorithm for the comparison between service enquiries and proposed service descriptions. Since agents shall find the appropriate services in an autonomous

Cite as: Towards Toolipse 2: Tool Support for the Next Generation Agent Framework, Michael Burkhardt, Marco Lützenberger and Nils Masuch, *Proceedings of 11th European Agent Systems Summer School (EASSS 2009)*, 31.August– 4.September, 2009, Torino, Italy.

¹<http://www.eclipse.org>

way, the latter are described by semantic information which allows for an automatic and detailed categorisation. The JIAC SeMa² algorithm is based upon the OWL service description ontology *OWL-S* [10], which allows to specify the purpose of a service by offering different parameters. Besides the name of the service these are in particular input/output parameters and preconditions and effects (*IOPE*). Preconditions and effects themselves are described in the Horn-like rule language *SWRL* [7], which extends the expressiveness of OWL.

JIAC SeMa² compares all search request parameters with each others separately. For the service names of request and advertisement it is checked whether they are identical or quite similar. Input and output parameters are expressed by using OWL classes, therefore these parameters are not only checked for identity but also for the taxonomical dependencies between them. The respective preconditions and effects are compared to their structural similarity (taxonomy matching) and whether they can be fulfilled (e.g. do the parameter instances of the requester fulfil the precondition of the advertised service). Each of these matching tasks is processed modularly and leads to a numerical result, which is deduced from the degree of similarity between request and advertisement. The sum of each of these results represents the overall similarity between a service request and service advertisement.

JIAC SeMa² has been developed within the diploma thesis of the third author [11].

3. COMBINING FEATURES

The main concern of our latest feature extensions to JIAC V was to increase the framework's overall performance. We already evaluated SeMa² within the last year's edition of the Semantic Service Selection Contest [1] and received remarkable results. An evaluation of AWE and JADLedit is currently done in the context of this year's Multi Agent Contest [2], in which we support the JIAC team developers with our tools. At the moment, we are working on a combined tool solution in which the developer can profit from each of the three presented features from a central point.

While AWE allows for the appending of services to an agent an existence of those is still assumed. The overall MAS development process is consequentially determined by an alternating usage of JADLedit, which is used to develop the required services, and the Agent World Editor, which is used to attach the latter to agents and design the overall multi-agent system structure.

Improving this situation, we are working on a combination of JADLedit and the Agent World Editor in order to benefit from a unified development and design support.

Our basic idea at this is to use JADLedit as editor for services selected in AWE. This provides not only detailed knowledge of existing services, but also allows for additional adjustments and developments from scratch, which is moreover supported by a comprehensive overall MAS representation. Since both, AWE and JADLedit, have been developed as plug-ins to the Eclipse IDE, our main task remains in defining the co-operation between both tools. The loose coupling of this plug-in architecture furthermore allows us to perform separate improvements of each part with only a minimum of dependencies and thus increases the application's service and its maintainability.

Although the combination of AWE und JADLedit makes

the service oriented agent development more comfortable, the capabilities of the service paradigm — with reusability aspects in particular — are as yet not fully utilised. The development support is still limited to the implementation and the appending of existing services to agents, however, an effective search mechanism for these specific services in the framework's libraries is currently not provided. At this point we are pursuing an application of SeMa².

Our approach here is similar to that of the previous combination of AWE and the JADLedit. Again, we are utilising Eclipse's plug-in mechanism and encapsulate the entire service lookup feature within a separate plug-in. The plug-in will contain a visual frontend (including a search mask, a search result table and features to add the retrieved services to an agent) and the service matcher itself. In the search mask, we will provide service retrieval in different granularity. The developer will be able to search for available services (i.e. library or custom developed) by name or by an OWL-S service description, which allows for the specification of detailed parameters, such as preconditions or the service's effects. Matching results will be displayed within a table and comprise a detailed description, while buttons allow the developer to append the retrieved results to the current MAS setup.

The combination of AWE, JADLedit and SeMa² will support us as multi functional tool in the design of multi-agent systems, in the accompanying service selection and in their development.

4. CONCLUSION

In this paper we described our idea in combining the results of three separate diploma theses to a MAS development tool for the JIAC V framework. In doing so, we started with an introduction of the JIAC V framework with particular focus on its service feature and motivated the necessity for a supporting tool solution. Subsequently, we introduced the results of the mentioned diploma theses, namely SeMa² as semantic service matcher, AWE as MAS design tool and JADLedit as service development tool and described the usage of the latter two within the overall JIAC V development process. We criticised the alternating usage of both and proposed our idea and our approach in combining AWE and JADLedit to one single tool solution. In order to provide an effective search mechanism for available services, we described our intention to include SeMa² in this tool combination as well. After introducing these short-term tasks, we presented our long term intention of extending this tool combination to a comprehensive development platform for the JIAC V framework. With Toolipse 2, we seek for methodology guided support in every aspect of the JIAC V development process. We want to achieve this task by integrating a set of other tools, each one streamlined to a specific development purpose. While some of these tools already exist (such as the VSDT) we outlined scheduled developments and thus depicted our approach in developing a tool for the next generation agent framework.

5. REFERENCES

- [1] Annual International Contest S3 on Semantic Service Selection. <http://www-ags.dfki.uni-sb.de/~klusck/s3/s3c-2008.pdf>.
- [2] Multi Agent Contest. <http://www.multiagentcontest.org/>.

- [3] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. OWL Web Ontology Language Reference. W3C Recommendation, February 2004. <http://www.w3.org/TR/owl-ref>.
- [4] Federico Bergenti and Michael N. Huhns. *Methodologies and Software Engineering for Agent Systems*, volume 11 of *Multiagent Systems, Artificial Societies, and Simulated Organizations*, chapter On the use of Agents as Components of Software Systems, pages 19–32. Kluwer Academic Publishers, 2004.
- [5] Michael Burkhardt. Integration von OWL in einem Interpreter als Konzept für komplexe Datentypen. Diploma thesis, Technische Universität Berlin, Berlin, Germany, March 2009.
- [6] Benjamin Hirsch, Thomas Konnerth, and Axel Heßler. Merging Agents and Services — the JIAC Agent Platform. In Rafael Bordini, Mehdi Dastani, Jürgen Dix, and Amal El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Tools and Applications*, pages 159–185. Springer Berlin/Heidelberg, 2009. To appear.
- [7] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosz, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. W3c member submission, World Wide Web Consortium, 2004. <http://www.w3.org/Submission/SWRL/>.
- [8] Marco Lützenberger. Development of a Visual Notation and Editor for Unifying the Application Engineering within the JIAC Framework Family. Diploma thesis, Technische Universität Berlin, Berlin, Germany, March 2009.
- [9] Marco Lützenberger, Tobias Küster, Axel Heßler, and Benjamin Hirsch. Unifying JIAC Agent Development with AWE. In *Proceedings of the Seventh German Conference on Multiagent System Technologies, Hamburg, Germany, 2009*. To appear.
- [10] David Martin, Mark Burstein, Erry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srinivasan Narayanan, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. OWL-S: Semantic Markup for Web Services. Technical report, November 2004. <http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/>.
- [11] Nils Masuch. Development of a Standard-Based Service Matcher Component within an Agent-Oriented Framework. Diploma thesis, Technische Universität Berlin, Berlin, Germany, April 2009.