Towards Toolipse 2
Tool Support for the Next Generation Agent Framework

Michael Burkhardt                           Marco Lützenberger                           Nils Masuch
michael.burkhardt@dai-labor.de
marco.luetzenberger@dai-labor.de
nils.masuch@dai-labor.de

DAI Labor Berlin
Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany

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.

Keywords
JIAC V, Tool, Services, Development, Multi-Agent Systems.

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 [8].

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. Regarding JIAC V's comprehensive capabilities of transparent distribution, service based interaction, semantic service descriptions, generic security and management mechanisms and support for flexible and dynamic reconfiguration in distributed environments [14], the framework's need for tool solutions becomes apparent.

In this work, we describe our approach in developing a tool solution which supports the development of multi-agent systems, by combining our latest three framework extensions - each one a result of a separate diploma thesis [9, 19, 22]. Due to the nature of these extensions, we are focusing this work on the design stage of the surrounding AOSE process, while we put particular focus on the JIAC's service aspects. However, since we pursue the higher goal of providing a comprehensive and methodology guided development tool for JIAC V, we describe our ideas concerning this long-term task as well.

This article is structured as follows: In the next section we will introduce the reader to the JIAC V development and provide a detailed description of our three latest extensions to the agent framework. Subsequently, we will describe our approach in combining these extensions to a development tool, which supports the design and service engineering of JIAC V multi-agent systems. After this, we introduce our long-term intentions with the tool combination and
describe further steps in developing a comprehensive tool solution. Finally, we discuss related work and wrap up with the conclusion.

2. The JIAC V Agent Framework

The JIAC V serviceware framework [14] 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 and has been applied in a number of different projects, ranging from personal information agents [5] and service delivery platforms [15] to simulation environments [10]. Development is supported by a rich library, which provides services and agents with frequently required abilities. The developer can reference these predefined agents for his application and extend their behaviour; for example by custom defined services. JIAC V provides a script language for this purpose. In its second incarnation, the JIAC Agent Description Language (JADL++) [9] has been geared towards the requirements of agent oriented service specification. Since JIAC V is the latest representative of the ever adapted and improved JIAC framework family [14], we also refer to it as The Next Generation JIAC, or simply, JIAC TNG.

In order to extend the framework's capabilities and provide comfortable and effective agent engineering, we lately extended JIAC TNG with several elaborate features, which we describe below.

2.1 Service Development with JADLedit

JADLedit is an Eclipse based editor for JADL++ [9], a programming language for services, which has been designed with particular focus on an easy usage to assure comfortable first steps in agent oriented programming. Agents which are equipped with a JADL++ interpreter are able to offer services which are based on the language's syntax. As one of its main features, JADL++ uses the knowledge representation language OWL [6] as semantic foundation for its complex data types. This feature allows for type safe editing of knowledge artefacts (namely facts) within the JADL++ programming language. We divided JADLedit into a development- and a browse-section. While the development-section consist of a state-of-the-art source code editor which provides features such as syntax highlighting, code completion, error marking and many more, the browse-section describes an OWL ontology browser, which displays detailed information on the ontologies, included by the current service, such as their classes or their properties. The combination of both sections allows the developer to access information on the artefact's data type he is using in his code. The additional information no longer necessitates the consultation of an external documentation and thus improves the overall development process.

JADL++ has been developed within the diploma thesis of the first author [9].

2.2 System Design with AWE

The Agent World Editor [20], 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. These diagrams are grounded by an underlying domain model, which acts as formal specification. Due to the generic nature of this domain model, AWE is capable of representing MAS designs without any concrete framework affinity. This allows for conceptual work which is furthermore supported by AWE’s picture export feature, which in turn allows for easy exchange or for the presentation of MAS designs without making use of the tool itself.

By invoking a translation routine, AWE produces executable code from the framework independent domain model instance of the diagram. At this, AWE currently provides support for the entire JIAC framework family, which comprises version IV, version V and MicroJIAC, JIAC’s derivative for devices with limited resources. However, the scope of support is not limited. We developed AWE under the premise of providing easy extensibility, which we realised by using a plug-in architecture for the implementation. In this architecture, each plug-in realises a separate and distinct part of functionality, which allows for the easy exchange of system crucial components (such as the domain model) as well as for the appending of additional functionality by providing
custom plug-ins, the so called Extension Plug-ins. Support for a specific agent framework is now encapsulated within an Extension Plug-in and mainly comprises a translation routine and a meta model of the framework’s code syntax.

AWE also provides library support during design. In order to avoid any framework affinity, we integrated a set of abstract concepts in the AWE base application, which define common and frequently used standard agents. The MAS designer can reference these so called ConceptElements as basis for his own agents and extend their behavior at will.

In order to provide framework support, each Extension Plug-in provides a mapping from the abstract concepts to a concrete implementation of the respective framework. The code translation routine accesses this mapping and replaces the abstract placeholders with references to concrete implementations.

Currently we support the entire JIAC framework family with AWE, however, the generic domain model, the modular architecture and the standardised structure of the Extension Plug-ins facilitates developments beyond this scope. In this course, we are planning an implementation for the JACK framework Extension Plug-in in the near future.

AWE has been developed within the diploma thesis of the second author [19].

2.3 Semantic Service Matching with SeMa²

The JIAC SeMantic Service Matcher (SeMa²) [22] provides a matching algorithm for the comparison between service enquiries and proposed service descriptions. Since agents shall find the appropriate services in an autonomous way, the latter are described by semantic information which allows for an automatic and detailed categorisation. The JIAC SeMa² is based upon the OWL service description ontology OWL-S [21], 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 [16], 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 fulfill 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 [22].

3. Short-Term Intentions

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 has been done in the context of this year's Multi Agent Contest [2], in which we supported the JIAC team developers with our tools.

At the moment, we are working on a combined tool solution which integrates the three presented developments and improves current lacks of the overall MAS engineering process.

At this, we start with a combination of AWE and JADLedit. 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 and 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 service implementations 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 JADLeditor. 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 front-end (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 in the framework’s library or the local file system for available service implementations 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. Figure 1 illustrates this mechanism.

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.

We provide an illustration of the intended appearance of this tool combination in Figure 2.

4. Long-Term Intentions

Although we are still working on the previously presented integration, we are already discussing several additional improvements for the near future. We describe these long-term intentions below.

4.1 Service Composition

The service matcher's analysis is currently limited to the analysis of single services. For the near future we plan to extend this feature to provide a planning capability. A developer's service request will no longer be matched against single services, but also against combinations of them. Fitting service combinations are then proposed to the developer with the ability to append them to the agents of his MAS design.

Due to the strict compliance of modularity in our work, we will be able to implement this service composition feature without any side effects or necessary adaptations to the AWE or JADLedit.
4.2 Semantic Service Matching with SeMa²

In the near future we plan to develop an interface between the JIAC V environment and web services, such as WSDL files. This shall enable the integration and assignment of any web service to agents. At the same time JIAC services shall be provided as web services. In its pure form WSDL files do not contain any semantic information. In order to provide a common representation of services within the architecture we envision a feature that enables the developer to enrich any service, independent of its underlying service type, with semantic information according to the OWL-S structure. Therefore we plan to integrate an editor that allows for the selection of WSDL services and provides a mechanism to wrap a semantic description in the form of an OWL-S file around it. Doing so, these services can be fully matched and categorised by using the advantages of semantic algorithms. This feature allows for further automated processing of the available services.

4.3 Semantic Service Specification

The integration of SeMa² into our tool combination already proposed our idea of a semantic search for services during the MAS design stage. The user is able to search for available service implementations and append the matching results to the agents of his MAS. At this, we are currently using JIAC’s native syntax, which defines a reference to the service's implementation file in the application’s class path. However, instead of providing this concrete reference, we plan to append services solely as OWL-S based description. In combination with the previously described service composition, the approach allows us to abstract from a manual service selection at design time and delegates this task to the compile time were the service matcher is used to determine a successful service constellation for the given description and to append it to the respective agents. In order to create the mentioned OWL-S descriptions, we will make use of JADLedit which is already capable of supporting ontology based development by its included ontology...
browser. Altogether, this additional feature will simplify the service engineering within JIACV.

### 4.4 The Toolipse 2

In this paper we merely touched the surface of JIAC V’s capabilities since we only focused on the framework’s service aspects during the design stage. In fact, JIAC V comes along with its own development methodology [14], which supports MAS development from analysis to deployment. In a long-term attempt, we plan to include more and more tools in the same way we described in this work and thus aim at a comprehensive and methodological guided development support for the JIAC V agent framework. Several development tools already exist, such as the Visual Service Design Tool (VSDT) [18], which allows service and process modelling on a visual level. Altogether we will align our work at the Toolipse [23], a tool suite, similar to the one we are aiming for and which is used as development platform for JIAC V’s preceding version, JIAC IV. We will combine a versatile set of tools, including but not limited to service, ontology and protocol editors and thus provide the Toolipse for a Next Generation JIAC, or the Toolipse 2.

### 5. Related Work

We have evaluated a number of similar tools, some of them already seeing the third Generation [3, 23].

#### 5.1 The agentTool System

The agentTool system [3; 24, p. 245-259] is a visual design environment for top-down design of multi-agent systems, supporting the steps of the MaSE methodology [25]. In its third edition it supports O-MaSE [11] with its nine different model types and is supplemented by a consistency checker, code generation, a metrics calculator and process engineering support. All aspects of an agent-oriented design can be modelled nicely, whereas the code generation feature is more or less an open issue. The paradigm of modelling services is provided by O-MaSE, but the underlying service realisation is not specified.

#### 5.2 The INGENIAS Development Kit

The INGENIAS Development Kit (IDK) [12] is a visual development tool supporting the INGENIAS Agent Framework. The IDK provides extensibility features and has been developed as open source project on the basis of an extensible plug-in architecture [12]. The focus here lies on code generation plug-ins such as the most popular INGENIAS Agent Framework [13]. The plug-in is included within the standard IDK setup and provides a translation of the design into executable JADE code. JADE [7] itself provides the creation of services and their association to agents. Furthermore it allows the integration of self-defined ontologies for the description of attributes. Nevertheless services and ontologies are not related to standards such as OWL and OWL-S, but follow a proprietary approach.

#### 5.3 The JACK Development Environment

The JACK Development Environment [24, p. 261-277] supports the development of JACK based applications. It provides for the creation and manipulation of each JACK component by means of visual system engineering and includes several other specialised tools. Project management is accomplished by the JDE Browser and MAS design by the JDE Design Tool [4]. The latter allows visual system engineering on the basis of drag and drop. Code generation and execution is provided by the JDE as well. The Compiler Utility Tool translates the developed diagrams into Java classes and supports both, execution and debugging. A service-oriented approach is not provided by JACK, since it concentrates on the BDI software model. Therefore agents possess plans which consist of several tasks.

#### 5.4 The CAFnE Toolkit

The CAFnE Toolkit [17] (Component Agent Framework for domain Experts) has been developed as a successor to the Prometheus Design Tool (PDT). The tool provides domain experts a suitable way to easily build and modify multi-agent systems and accomplish modifications in complex agent-based domains, while its intuitive operability particularly addresses persons with limited programming skills. The toolkit allows visual modelling, code generation, compilation and execution of agent based applications.
MAS development is done by the specification of diagrams. CAFnE operates on a framework unspecific and simplified BDI domain model, which provides platform independent as well as BDI compliant MAS configurations. In addition to PDT, CAFnE provides an effective error avoidance mechanism and the generation of framework dependent executable code inclusive transformation to JACK. Since CAFnE is independent of the underlying framework and its architecture it does not specify how the service's approach within multi-agent systems is realised.

5.5 The Toolipse

The JIAC IV Toolipse [23] is an IDE based on the Eclipse platform, supporting the JIAC Methodology. It facilitates visual design of multi-agent systems within the JIAC IV framework. Each JIAC IV feature is realised by one or more Eclipse plug-in(s), e.g., wizards, editors and views. The editors are providing direct source code manipulation or system engineering on a visual level. While the source code editors provide state-of-the-art programming, the graphical editors allow specifying diagrams visually. The foundation of these diagrams is a formal specification by means of an underlying meta-model. Particularly interesting for this work is the Agent Role Editor (ARE), which is used during the application deployment, in order to define the MAS design. Since ARE has been applied for several years now, various improvement ideas have initiated the development of its predecessor, the Agent World Editor. With regard to the service paradigm, JIAC IV supports JADL, the predecessor of JADL++, which also allows the definition of ontologies. But in contrast to JIAC V its approach is not based on standards.

6. 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 [22], AWE as MAS design tool [19] and JADLedit [9] 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 planned developments and thus depicted our approach in developing a tool for the next generation agent framework.

References


