Engineering JIAC Multi-Agent Systems (Demonstration)

Marco Lützenberger, Thomas Konnerth, Tobias Küster,
Jakob Tonn, Nils Masuch, and Sahin Albayrak
Technische Universität Berlin, DAI-Labor
Ernst-Reuter-Platz 7
10587 Berlin, Germany
marco.luetzenberger@dai-labor.de

Categories and Subject Descriptors

I.2 [**Computing Methodologies**]: Distributed Artificial Intelligence—*Multiagent systems*

Keywords

agent platform; development environments

1. INTRODUCTION

The development of high quality software is considered to be extremely difficult. In fact, it has been argued that such endeavours are one of the most difficult construction tasks that humans undertake [2]. Contemporary IT-infrastructures facilitate the need for distributed applications—a system design which can be implemented with *Agent Oriented Software Engineering* [2]. Yet, despite the requirement for distributed systems and despite the availability of frameworks, methodologies and tool suites that support the development of distributed applications, there are only few attempts to use agent technology for commercial or industrial applications.

The reasons for this are not entirely clear, though, analyses [6, 7, 9] show that there are several factors that may foster industrial adoption of agent technology. These factors include: awareness for the need for agent technology, compliance with standards, reasonable costs, scalability and performance, stability and robustness, extensibility, mature methodologies and tool support, and a comprehensive user documentation.

To facilitate the industrial adoption of agent technology, we developed the *Java Intelligent Agent Componentware*, or *JIAC* [4]. JIAC supports a comprehensive set of standards, ensures reliable applications, supports custom extensions, and provides a whole set of user friendly and well documented tools for the design, implementation, deployment, and monitoring of multi-agent systems. Agent frameworks with a focus on industrial requirements, however, find it difficult to gain foothold in the realm of research where other factors determine their level of acceptance. The aim of JIAC is to show that industrial requirements and research needs can be combined in one solution.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

2. THE JIAC AGENT FRAMEWORK

JIAC [4] is a Java-based multi-agent framework and runtime environment, which integrates the agent paradigm with Service Oriented Architectures. JIAC's discovery and messaging infrastructure is based on *ActiveMQ*[1], which facilitates a transparent distribution of JIAC agents—even beyond network boundaries.

An agent-platform comprises one or more 'agent nodes', which are physically distributed and provide the runtime environment for JIAC agents. New agents, services, as well as further agent nodes can be deployed at runtime. Agents can interact with each other by means of service invocation, by sending messages to individual agents or multicast-channels, and by complex interaction protocols. Each individual agent's knowledge is stored in a tuple-space based memory. Finally, JIAC agents can be remotely monitored and controlled at runtime by means of the *Java Management Extension Standard* (*JMX*).

Each agent contains a number of default components, such as an execution-cycle, a local memory and communication adaptors. The agents' behaviours and capabilities are implemented in so-called *AgentBeans*, which are controlled by the agent's life cycle.

AgentBeans are capable of reactive-, proactive-, iterative-, or controlled behaviour and thus comply with the key characteristics of software agents [10]. The assembly of the entire multi-agent system is described by one ore more Spring[2] configuration files.

3. ENGINEERING JIAC AGENTS

The JIAC framework comes along with a number of editors and tools, most of which are *Eclipse* plugins.

First, a specific Eclipse plugin, the *JIAC Project Plugin*, provides a special kind of Eclipse Project Nature for JIAC projects. The project plugin comes along with a project wizard, which creates a *Project Object Model*, allowing *Apache Maven*[3] to download and install the JIAC libraries and all its dependencies and create an according Eclipse project. Besides the JIAC Project plugin there are additional Eclipse views, each providing a particular functionality to ease the development of JIAC agents and -applications. These views include lists of components of the current JIAC project, lists of all the JIAC nodes running in the same network, or func-

[1]ActiveMQ: `http://activemq.apache.org`
[2]Spring: `http://www.springsource.org/`
[3]Maven: `http://maven.apache.org`

tionality to access the *AgentStore* [1], in order to include deployed components into the current project, and for packaging the current project and deploying it to the store itself.

As mentioned above, JIAC uses Spring configuration files to describe the assembly of multi-agent systems. Being based on *XML*, Spring configuration files are considerably difficult to edit. The *Agent World Editor* (*AWE*) [3] was developed to alleviate this problem by providing a graphical overview and an editor for these files. After modelling the multi-agent system, using intuitive graphical symbols for agents, nodes and beans, the AWE can be used to generate both, the configuration files and stubs for the individual agent beans. AWE can also be used for managing agent configurations that are laid out across several files and for importing and editing existing files.

The *Visual Service Design Tool* (*VSDT*) [3] is an Eclipse-based editor for the *Business Process Modeling Notation*. It allows to design and implement services and entire multi-agent systems in terms of process diagrams. VSDT features: modelling assistance, structural validation, and a simple interpreter/simulator for stepping through the processes and interpreting the diagrams, which has proven very useful for debugging. Processes modelled in the VSDT can be transformed and exported to a number of JIAC Agent Beans. These processes can be exposed as a JIAC service or triggered at a given time, or on receiving a particular JIAC message.

ASGARD [8] is the next tool in the development chain of JIAC. ASGARD is a visual monitoring application with management capabilities for the JIAC platform. ASGARD provides a three-dimensional graphical overview of currently running JIAC entities within a local network and uses the visualisation to indicate properties and states of-, as well as interaction between JIAC entities. ASGARD's main use is to support developers during implementation and testing of distributed JIAC applications as well as to check the current state of deployed applications at runtime for maintenance purposes. ASGARD uses JIAC's integrated monitoring interface to connect to local and remote entities via JMX. Automatic discovery of running JIAC entities is provided by the use of multicast technology, *RMI* registries, and peer-to-peer forwarding of known entity addresses.

In order to provide a high degree of flexibility in changing environments, agents must be able to dynamically interpret and invoke the functionality of other agents. The enhancement of services by semantic information allows for such an approach. The *Semantic Service Manager* [5] directly addresses this problem by offering an editor for the description of JIAC actions as *OWL-S* services at design time. It consists of an *OWL* ontology manager, which provides automated integration of new OWL ontologies and a transformation procedure from *EMF Ecore* models to OWL. In general, the manager supports the developer by automatically annotating JIAC actions with OWL-S descriptions, which can be refined manually afterwards. After the agent systems have been developed, they can be deployed to the AgentStore [1]. Inspired by the popular 'app store' metaphor, as known e.g., from *Apple* or *Google*, this AgentStore can be used for deploying, sharing, and reusing multi-agent systems.

4. REFERENCES

[1] A. Heßler, B. Hirsch, T. Küster, and S. Albayrak. Agentstore — A pragmatic approach to agent reuse. In F. Dechesneand, H. Hattori, A. ter Mors, J. M. Such, D. Weyns, and F. Dignum, editors, *Advanced Agent Technology. AAMAS 2011 Workshops, AMPLE, AOSE, ARMS, DOCM3AS, ITMAS, Taipei, Taiwan, May 2–6, 2011. Revised Selected Papers*, volume 7068 of *Lecture Notes in Artificial Intelligence*, pages 128–138. Springer, 2012.

[2] N. R. Jennings and M. Wooldridge. Agent-oriented software engineering. *Artificial Intelligence*, 117:277–296, 2000.

[3] T. Küster, M. Lützenberger, A. Heßler, and B. Hirsch. Integrating process modelling into multi-agent system engineering. *Multiagent and Grid Systems*, 8(1):105–124, January 2012.

[4] M. Lützenberger, T. Küster, T. Konnerth, A. Thiele, N. Masuch, A. Heßler, J. Keiser, M. Burkhardt, S. Kaiser, and S. Albayrak. JIAC V — A MAS framework for industrial applications (extended abstract). In T. Ito, C. Jonker, M. Gini, and O. Shehory, editors, *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, Saint Paul, Minnesota, USA*, pages 1189–1190, 2013.

[5] N. Masuch, P. Brock, and S. Albayrak. Semi-automated generation of semantic service descriptions. In J. B. Pérez, J. M. C. Rodríguez, J. Fähndrich, P. Mathieu, A. Campbell, M. C. Suarez-Figueroa, A. Ortega, E. Adam, E. Navarro, R. Hermoso, and M. N. Moreno, editors, *Trends in Practical Applications of Agents and Multiagent Systems*, volume 221 of *Advances in Intelligent Systems and Computing*, pages 155–162. Springer International Publishing, 2013.

[6] J. McKean, H. Shorter, M. Luck, P. McBurney, and S. Willmott. Technology diffusion: Analysing the diffusion of agent technologies. *Autonomous Agents and Multi-Agent Systems*, 17:372–396, 2008.

[7] M. Pěchoučyek and V. Mařík. Industrial deployment of multi-agent technologies: review and selected case studies. *Autonomous Agents and Multi-Agent Systems*, 17(3):397–431, 2008.

[8] J. Tonn and S. Kaiser. ASGARD — A graphical monitoring tool for distributed agent infrastructures. In Y. Demazeau, F. Dignum, J. M. Corchado, and J. B. Pérez, editors, *Advances in Practical Applications of Agents and Multiagent Systems*, volume 70 of *Advances in Intelligent and Soft Computing*, pages 163–175. Springer, 2010.

[9] D. Weyns, H. Van, D. Parunak, and O. Shehory. The future of software engineering and multi-agent systems. *Special Issue on Future of Software Engineering and Multi-Agent Systems, International Journal of Agent-Oriented Software Engineering (IJAOSE)*, 2008.

[10] M. Wooldridge. Agent-based software engineering. *IEEE Proceedings — Software*, 144(1):26–37, 1997.