Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 32 (2014) 920 – 927

Procedia Computer Science

The 1st International Workshop on Developing and Applying Agent Frameworks (DAAF)

Model-Driven Development and Validation of Multi-Agent Systems in JIAC V with the Agent World Editor

Christian Kuster[a,*], Tobias Küster[b], Marco Lützenberger[b], Sahin Albayrak[a,b]

[a] German-Turkish Advanced ICT Research Centre, Ernst-Reuter-Platz 7, 10587 Berlin, Germany
[b] DAI-Labor, Technische Universität Berlin, Ernst-Reuter-Platz 7, 10587 Berlin, Germany

Abstract

In this work we describe the Agent World Editor, a graphical editor that supports the development and validation of multi-agent systems in JIAC V. For the development of this editor, which complies with the OSGi technology, a model-driven approach was used. Concepts that support the developer to design multi-agent systems in JIAC V have been developed as well. Developers do not need to configure their systems by hand, but can model them in a graphical manner. Errors are detected at design time, and the adaptive user interface offers additional guidance in the development process.
© 2014 Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license
(http://creativecommons.org/licenses/by-nc-nd/3.0/).
Selection and Peer-review under responsibility of the Program Chairs.
Keywords: AOSE, MAS, JIAC V, MDSD, OSGi

1. Introduction

Due to the increasing computerization and networking of technical systems the agent-oriented software engineering (AOSE)[1] is becoming more and more important. AOSE "comprises approaches, methods, techniques and tools for the development and execution of agent-oriented software"[2]. An Agent is a software program that acts on behalf of a user or another program to fulfil the assigned goals in consideration of its own interests and capabilities. It interacts with its environment and other agents autonomously. Providing a concept that facilitates specifications in terms of behaviour, AOSE significantly increases the flexibility of object-oriented software engineering (OOSE). Where OOSE deals with passive objects, AOSE defines active agents.

In order to establish AOSE in the software industry, the used agent frameworks have to fulfil the requirements of project and quality management[3]. This includes tool support during the development process, though, there is still a lack of appropriate and mature tools[3,4]. Plenty of theories, methodologies, tools and frameworks emerged through the last decade[5], but most of these approaches are not aware of industrial needs and gear towards research topics.

The Java-based Intelligent Agent Componentware V (JIAC V)[6], a framework with focus on industrial requirements[7], such as support of the service-oriented architecture paradigm, standardization of all components, deployment and undeployment at runtime, robustness, scalability, modularity and extensibility[5]. It also includes many tools for

* Corresponding author. Tel.: +49-(0)30-31474218 ; fax: +49-(0)30-31474003.
E-mail address: christian.kuster@gt-arc.com (Christian Kuster).

1877-0509 © 2014 Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license
(http://creativecommons.org/licenses/by-nc-nd/3.0/).
Selection and Peer-review under responsibility of the Program Chairs.
doi:10.1016/j.procs.2014.05.512

specific aspects[8,9,10]. However, JIAC V still requires better support in configuring MAS architectures. Such a tool may facilitate the development process, and further the adoption process of agent technology in the software industry.

Since the modelling of the structure of the architecture is an important aspect in the development of *multi-agent systems (MAS)*, the *Agent World Editor (AWE)*, which is described in this paper, attempts to fill this gap. This paper is based on the diploma thesis of the first author.

The rest of the paper is structured as follows. In Chapter 2 the agent framework JIAC V is introduced, Chapter 3 gives an overview about the concept of model-driven software engineering. In Chapter 4 current tool support for AOSE is elaborated. Chapter 5 presents AWE, and its integrated key concepts. Finally, Chapter 6 draws a conclusion and names future steps.

2. The JIAC V Framework

The Java-based Intelligent Agent Componentware V (JIAC V)[6] is a multi-agent development framework and runtime environment that has been applied in a number of research- and industrial projects[7]. JIAC integrates the agent- with the service-oriented architecture paradim.

In JIAC V, an agent platform consists of one or more agent nodes, which represent the runtime environment for the agents. Agent nodes run in their own Java virtual machine and, typically, on their own physical machine. Agents can communicate and use services provided by other agents. These services are called actions in JIAC V and are encapsulated in the agent beans. Agent beans are allocated to one or more agents, thus, they define the possible services of the agents. JIAC V MAS architectures are specified by Spring[1] configuration files. JIAC V contains several pre-defined configurations with standard elements such as a basic agent or a bean for communication that can be reused.

3. Model-Driven Software Engineering

Model-driven software development (MDSD)[11] denotes a development process where models are not only used for documentation but are seen as equal to source code. The translation process is automated. Stahl et al.[11] define MDSD as follows: “Model-driven software engineering is the generic term for techniques that generate runnable software out of formal models” (p. 11). Here, a formal model means a model written in a domain specific language (DSL) that describes an aspect of the software entirely. The structure of the DSL is specified in the meta model by the *abstract syntax* and the *static semantic*[12]. The DSL further consists of the *dynamic semantic* and the *concrete syntax*. The software is generated by either interpreters or generators, both having advantages and disadvantages. Interpreters parse the model, analyse it at runtime and invoke appropriate actions. Generators on the other hand generate executable source code that is static at runtime.

A key benefit of MDSD compared to other types of software engineering is the higher degree of abstraction, leading to a clearer view on the problem and avoiding redundancy. Furthermore, the model can be separated from the technical implementation. Due to the automatic source code generation the software architecture is homogeneous and therefore easier to test. However, MDSD means higher initial costs and makes high demands on the meta model.

4. Related Work

Over the last years many agent frameworks have emerged in the area of multi-agent systems[13]. Some of them have prevailed as promising. Bordini et al.[14] and Baldoni et al.[15] give an overview over the most important platforms for the development of MAS. Garneau and Delisle[16] present different frameworks for multi-agent systems as well, but focus on provided tools, formulate criteria and evaluate the presented tools.

One of the most popular frameworks including a graphical editor is the commercial platform JACK [17,18], which is based on the Belief-Desire-Intention (BDI) agent architecture[19]. The related development environment is the JACK Development Environment (JDE)[20] providing a wide range of tool support and documentation.

[1] http://projects.spring.io/spring-framework

The development of a MAS can be done via *Drag & Drop*. Further, an automatic generation of the source code and a debugger for the application are included. On the other hand, the JDE does not provide any methodology[21].

The INGENIAS Development Kit (IDK)[22] is a visual development environment, targeting the INGENIAS Agent Framework[23]. INGENIAS has been developed with the goal to extend MAS by the concept of organizational structures. The idea is that the organization that will use the MAS has a big impact on the MAS itself. The graphical editor is based on an extensible plug-in architecture and currently supports the export to JADE source code.

The development environment agentTool III (aT^3)[24] supports the Organization-based Multiagent Systems Engineering (O-MASE)[25] methodology. In this methodology the MAS is built from method fragments that are based on a common meta model. This meta model defines analysis-, design and implementation concepts that help modelling the MAS. O-MASE concentrates on the design phase; the implementation only plays a subordinate role[26], although the tool supports the export to JADE source code[25].

The Component Agent Framework for domain-Experts (CAFnE)[27,28] extends the Prometheus Design Tool (PDT), which implements the Prometheus methodology. As opposed to other development environments, the focus lies on support of domain experts that only have little programming experience in the modification of already existing agent applications. Furthermore, CAFnE uses a framework-unspecific domain model allowing platform-independent design[26].

An earlier version of the Agent World Editor (AWE) was developed as a part of a diploma thesis[29] at TU Berlin and was developed as a graphical editor for the modelling of MAS in JIAC. The focus was to complete the set of already existing tools of the IDE Toolipse[8,9]. Of particular interest was, similar to CAFnE, to keep the used domain model free from any association to a specific framework; the editor builds on GMF and was developed as an Eclipse plug-in. The export is done via additional plug-ins, with JIAC V being supported from the beginning.

Another framework that focuses on the development of goal-based MAS in the semantic web is SEAGENT[30]. Similar to AWE, SEAGENT provides a graphical editor being realized as an Eclipse plugin. However, there is no further support of the modeling process, but a monitoring component for runtime.

Summarising the presented work, the one thing that all of the above-presented works have in common is to establish the use of agent-based systems in the industry[21]. As they concentrate on the implementation of these methodologies and disregard the usability as well as the development process as a whole they can be seen more as a proof-of-concept. AWE is proposed as a tool to support the developer in implementing MAS from scratch as well as further validate and optimize existing systems.

5. The Agent World Editor

AWE is a graphical editor that allows the user to configure the assembly of JIAC V MAS in a single diagram (see Figure 1). The focus in AWE lies on the structure of the system, and not on the behaviour. The editor follows a model-driven design approach and is itself developed after this methodology using GMF, a framework of Eclipse. It makes use of the plug-in concept of Equinox, a framework that implements the OSGi specification and constitutes the base of the Eclipse architecture. All concepts follow usability and support the user through the whole modelling process. An export functionality generates the files that are needed to run the modelled JIAC V MAS, including stubs for the agent beans.

AWE covers all the three important steps *initialization*, *modelling* and *export*. For each of these steps concepts were found to support the developer in the design process. The most important concepts are presented in the following.

5.1. Concept

As AWE has been developed by means of model-driven engineering, the core is built on a domain model representing the JIAC V architecture. The focus of the domain model is on the structure of the MAS; no attention was paid to more complex relationships between the agents that are needed to implement behaviour. It was the aim to develop a domain model that is able to cover as much as needed while keeping it as simple as possible. In order to simplify the modelling process, the abstract components were combined with the technical representation of the MAS in form configuration files. In doing so, it is always apparent where in the system the agents are defined. In the domain model configuration files can be of two types: The first are library files that consist of predefined elements and

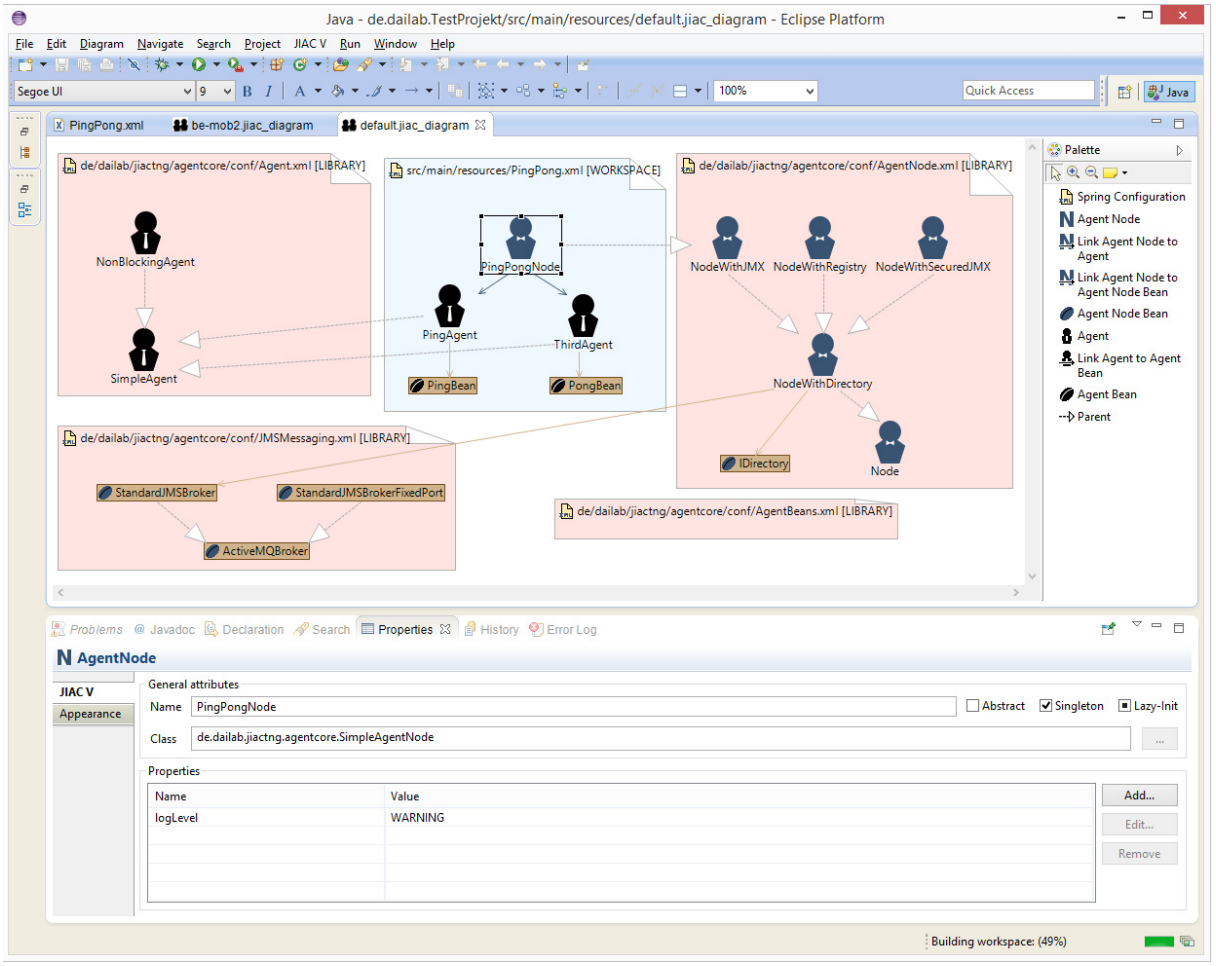

Fig. 1. Agent World Editor: model (top left), toolbar (right) and property sheet (bottom).

therefore are loaded into the diagram (write-protected). They allow for a better view on the system and ease the work with these elements. Developers quickly see which predefined libraries and elements are available and can reference them. In Figure 1 they are coloured in light red. The second are editable configuration files of the current system. In Figure 1 they are coloured in light blue. Constraints were used to prohibit unwanted ambiguities for the description of model states, e.g. spring configurations must have a valid name and path, and must point to folder that is intended for configuration files.

5.1.1. Integration into JIAC V framework

One of the most difficult task was to integrate AWE into the JIAC V framework properly to provide extended usability. This applies to the whole modelling process: When initializing the diagram the developer is guided through a process in which the settings for a valid data model can be made. The goal here is to prevent the developer from initializing invalid diagrams, e.g. containing references to elements from not loaded configuration files. For this reason the standard configuration files that are needed by all JIAC V MAS are loaded automatically into the diagram. More configurations that are presented in a dialogue can be chosen. In addition, dependencies will automatically be resolved if possible, otherwise an error is displayed. Due to the fact that MAS consist of similar structures a template system was developed that contains typical use cases, and is further extendible by plug-ins. Thereupon, in the initialization process the declared elements are created.

```
<bean name="Node" parent=".." singleton="false">

  <property name="agents">
    <list>
      <ref bean="Agent1" />

      <ref bean="Agent2" />



    </list>
  </property>
</bean>

<bean name="Agent1" parent=".." />


<bean name="Agent2" parent="..">
  <property name="agentBeans">
    <list>
      <ref bean="Bean2" />
    </list>
  </property>
</bean>

.
```

```
<bean name="Node" parent="..">
  <!-- AWE3: value for "singleton" = "false" ↩
       ↪ deleted -->
  <property name="agents">
    <list>
      <ref bean="Agent1" />
      <!--commented out
      <ref bean="Agent2" />
      commented out-->
      <!--added-->
      <ref bean="Agent3" />
    </list>
  </property>
</bean>

<bean name="Agent1" parent=".." />

<!--commented out
<bean name="Agent2" parent="..">
  <property name="agentBeans">
    <list>
      <ref bean="Bean2" />
    </list>
  </property>
</bean>
commented out-->

<!--added-->
<bean name="Agent3" parent="..">
  <property name="agentBeans">
    <list>
      <ref bean="Bean3" />
    </list>
  </property>
</bean>
```

Fig. 2. example for a changed configuration file: on the left is the former, on the right is the latter

To keep the data models and therefore the diagram compact, only necessary information is shown. Apart from the graphical representation, this only includes the name of the element. Other information is shown in a textual property view. In order to help the developer keep the model valid, all options in dialogues are limited to type-specific valid entries. In doing so, most of the constraints are correct by construction.

5.1.2. Bi-directional updates

In the actual modelling phase it is possible to make changes to the visual elements and, based on these changes, modify the corresponding configuration file. Since this mechanism is not trivial it is further described in Section 5.1.3. The other possible use case is that the configuration file itself is changed; this case is frequently not supported by graphical editors having only export functionality. However, AWE is capable of updating the diagram according to made changes. For this, the diagram elements of the configuration files are linked to the configuration files. The user is able to manually trigger updates of the graphical elements. In this update process all sub elements are deleted from the diagram and re-included. This procedure easily solves the problem of finding all differences in both versions. In order to avoid that connections to external elements are unnecessarily cut, connections from old elements are redirected to their corresponding new elements. Connections to no longer available elements are removed automatically.

5.1.3. Structure-preserving export

One key feature of MDSD is the automatic transformation from the domain model to source code. Contrary to the initialization of the diagram, for the export there is no certain need for a guided process, but it can be executed

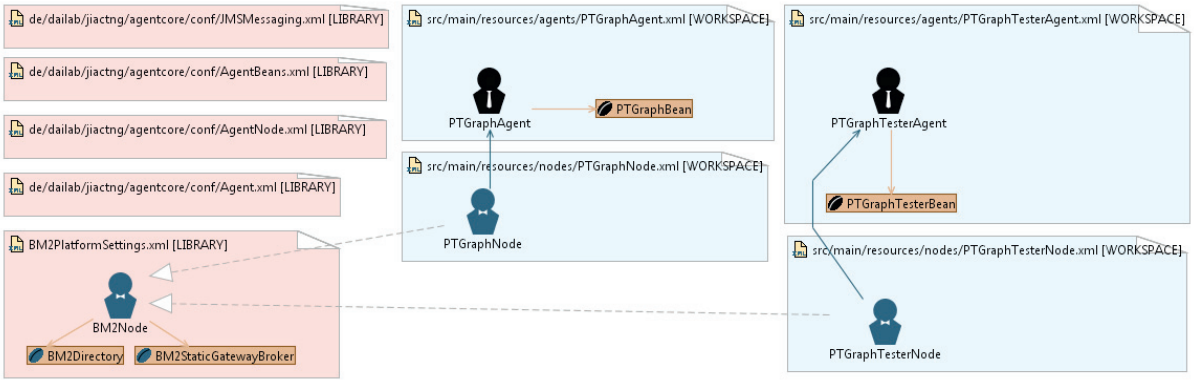

Fig. 3. Example for a JIAC V configuration file

automatically. For the file generation it is important to not override existing files and to preserve their inner structure. Instead, the changes are added. To give the developer the control over the export mechanism it has to be manually invoked. This overcompensates the fact that the diagram and the underlying files are out of sync. The export must generate two types of files, namely XML configuration files and Java files containing stubs for the agent- and agent node beans. The stubs will only be generated if the referenced class points to a non-existing Java file or the existing Java class is annotated with "@generated" in the corresponding JavaDoc.

The XML configuration file export is the more important one. If the file already exists, it is compared with the new content on a textual level. Elements that are deleted are marked as comment in the file, existing elements will be edited in the appropriate places. For this the old values are commented out and the new values are inserted. As a result, changes can be retraced. Figure 2 shows an example of the structure-preserving insertion of changes. On the left the former configuration can be seen before it is modified by the editor. On the right the configuration is overridden successfully. In this example an agent element (Agent2) is deleted, and a new agent element (Agent3) is added. The deleted element is merely marked as comment at defining as well as referencing places. This way the former structure of the file stays visible to the developers. The agent node element (Node) shows how the changed reference to Agent2 and the attribute *singleton* are being labelled.

The Java files implement the behaviour of the agents. Since the focus in AWE lies on the structure of the MAS, these files are stubs, extending the Java super class. Agent- and agent node beans, thus, can contain actions that are provided by the corresponding agents. Method stubs for these actions are generated as well.

5.2. Implementation

The editor has been implemented as an Eclipse plug-in using the *Graphical Modeling Framework (GMF)* as it facilitates the model-driven approach by providing frameworks and functionalities to cover the whole development process of graphical editors for data models. The SOA architecture of Eclipse makes it possible to add new features to the editor by installing new plug-ins. Furthermore, GMF already implements a set of commonly used features for graphical editors and gives access to features of the Eclipse IDE.

6. Evaluation

One parameter that provides information about the quality of AWE is the completeness of the meta model. In MDSD, a possibility to check the completeness is to test different scenarios and find constructs that cannot be described with this meta model [11]. Since JIAC V has already been used in various projects, several configurations exist; representative samples defining real world scenarios were chosen and remodelled with AWE. There were no configuration files that could not be described with the meta model. An interesting side effect had shown: Some elements had been defined with names from JIAC V libraries and therefore shadowed them, what could lead to problems in further

development. This were not been detected before using AWE and emphasises its importance. In AWE, shadowing results in a warning message.

To illustrate the time savings and improvement of usability, we present a small example (see Figure 3); a further evaluation with multiple participants still has to be made. The details of the system are not important, but the two development processes, namely manually and by using AWE. To manually write the configuration files it was necessary to first locate the file "BM2PlatformSettings.xml" in the libraries, so that the two agent nodes could inherit the element "BM2Node" containing some specific settings of a project "BM2". Subsequently, configuration files for "PTGraphNode", "PTGraphAgent", "PTGraphTesterNode" and "PTGraphTesterAgent" were created by the *Copy & Paste Programming* anti-pattern. In all of these steps errors due to wrong written name references occurred. Errors could only be detected during start-up of the system, leading to a development time of ca. 30min. With AWE, everything could be done in a graphical manner. Configuration files from libraries were automatically detected and could be included in the diagram using a dialogue. Invalid references, such as *Copy & Paste* errors, were avoided by design. The development time with AWE was ca. 10min. This means a decrease by factor 3. For larger systems an even better performance is expected as errors, e.g. invalid references, are harder to find.

7. Conclusion and future work

AWE is a graphical editor that allows an user-friendly visual design of multi-agent systems. It minimizes the need for manual programming and therefore the error potential. The development was in compliance with the MDSD principle to increase the quality due to a higher abstraction level and the automated generation of the system's source code. The editor combines the conceptional with the technical view in the form of configuration files to give an intuitive view on the MAS. Often tools are not used because they provide a very limited work flow that cannot be adapted to the project's and the user's needs. In contrast, from the outset the main goal of AWE was to be as complete as needed while providing the most usability as possible. This includes the functionality like changes are incorporated in the graphical editor as well as in the linked configuration files so that the user is not unnecessarily restricted. Existing systems can be imported for modification and resulting changes of configuration files can be followed.

Several extensions to the editor are possible. This work has concentrated on the structure of MAS. The next step would be to implement the behaviour of the agents and relationships between them. Another possible development is support of different perspectives of MAS. The current editor shows a view that is close to the technical system. An additional view could group the agents logically or visualize interactions. Furthermore, to increase the usability, the editor could provide deployment and execution of agent systems. Furthermore, changes in the configuration files could be traced to provide a version control system.

Acknowledgements

This work is partially funded by the German Federal Ministry of Education and Research under the funding reference number 01IS12049A.

References

1. Jennings, N.R.. On agent-based software engineering. *Artificial intelligence* 2000;**117**(2):277–296. URL: `http://www.sciencedirect.com/science/article/pii/S0004370299001071`.
2. Weiß, G.. Agentenorientiertes software engineering. *Informatik-Spektrum* 2001;**24**(2):98–101. doi:10.1007/s002870100147.
3. Pěchouček, M., Mařík, V.. Industrial deployment of multi-agent technologies: review and selected case studies. *Autonomous Agents and Multi-Agent Systems* 2008;**17**(3):397–431.
4. Weyns, D., Parunak, H.V.D., Shehory, O.. The future of software engineering and multi-agent systems. *International Journal of Agent-Oriented Software Engineering* 2009;**3**(4).
5. Lützenberger, M., Küster, T., Konnerth, T., Thiele, A., Masuch, N., Heßler, A., et al. A multi-agent approach to professional software engineering. In: *Engineering Multi-Agent Systems*. Springer; 2013, p. 156–175.
6. Lützenberger, M., Küster, T., Konnerth, T., Thiele, A., Masuch, N., Heßler, A., et al. Jiac v: A mas framework for industrial applications. In: *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-agent Systems*; AAMAS '13. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-1993-5; 2013, p. 1189–1190.

7. Lützenberger, M., Küster, T., Konnerth, T., Thiele, A., Masuch, N., Heßler, A., et al. Engineering industrial multi-agent systems – the JIAC V approach. In: Cossentino, M., Seghrouchni, A.E.F., Winikoff, M., editors. *Proceedings of the 1st International Workshop on Engineering Multi-Agent Systems (EMAS 2013)*. 2013, p. 160–175.
8. Küster, T., Lützenberger, M., Heßler, A., Hirsch, B.. Integrating process modelling into multi-agent system engineering. *Multiagent and Grid Systems* 2012;**8**(1):105–124. URL: `http://iospress.metapress.com/index/88124719M6587N37.pdf`.
9. Tuguldur, E.O., Hessler, A., Hirsch, B., Albayrak, S.. Programming multi-agent systems. chap. Toolipse: An IDE for Development of JIAC Applications. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-642-03277-6; 2009, p. 184–196. doi:10.1007/978-3-642-03278-3_12.
10. Tonn, J., Kaiser, S.. Asgard–a graphical monitoring tool for distributed agent infrastructures. *Advances in Practical Applications of Agents and Multiagent Systems* 2010;:163–173.
11. Stahl, T., Völter, M., Efftinge, S., Haase, A.. *Modellgetriebene Softwareentwicklung: Techniken, Engineering, Management*. Heidelberg: dpunkt; 2 ed.; 2007. ISBN 978-3-89864-448-8.
12. Gronback, R.C.. *Eclipse Modeling Project: A Domain-Specific Language (DSL) Toolkit*. The Eclipse Series. Pearson Education; 2009.
13. Akbari, O.. A survey of agent-oriented software engineering paradigm: Towards its industrial acceptance. *Journal of Computer Engineering Research* 2010;**1**(2):14–28.
14. Bordini, R.H., Braubach, L., Dastani, M., El, A., Seghrouchni, F., Gomez-sanz, J.J., et al. A survey of programming languages and platforms for multi-agent systems. 2006.
15. Baldoni, M., Baroglio, C., Mascardi, V., Omicini, A., Torroni, P.. Agents, multi-agent systems and declarative programming: what, when, where, why, who, how? *A 25-year perspective on logic programming* 2010;:204–230.
16. Garneau, T., Delisle, S.. A new general, flexible and java-based software development tool for multiagent systems. In: *Proceedings of the International Conference on Information Systems and Engineering (ISE 2003)*. 2003, p. 22–29.
17. Howden, N., Rönnquist, R., Hodgson, A., Lucas, A.. Jack intelligent agents-summary of an agent infrastructure. In: *5th International conference on autonomous agents*. 2001, .
18. Winikoff, M.. Jack intelligent agents: An industrial strength platform. *Multi-Agent Programming* 2005;:175–193.
19. Rao, A.S., Georgeff, M.P., et al. Bdi agents: From theory to practice. In: *ICMAS*; vol. 95. 1995, p. 312–319. URL: `http://www.aaai.org/Papers/ICMAS/1995/ICMAS95-042`.
20. Bordini, R., Dastani, M., Seghrouchni, A.. *Multi-Agent Programming:: Languages, Tools and Applications*; vol. 2. Springer; 2009.
21. Mascardi, V., Martelli, M., Gungui, I.. Dcaselp: a prototyping environment for multilanguage agent systems. 2008.
22. Gomez-Sanz, J.J., Fuentes, R., Pavón, J., García-Magariño, I.. Ingenias development kit: a visual multi-agent system development environment. In: *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: demo papers*; AAMAS '08. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems; 2008, p. 1675–1676.
23. Pavón, J., Gómez-Sanz, J.J., Fuentes, R.. The ingenias methodology and tools. *Agent-oriented methodologies* 2005;**9**:236–276. URL: `http://www.igi-global.com/chapter/agent-oriented-methodologies/5061`.
24. Garcia-Ojeda, J.C., DeLoach, S.A., Robby, . agenttool iii: from process definition to code generation. In: *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*; AAMAS '09. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-0-9817381-7-8; 2009, p. 1393–1394.
25. DeLoach, S.A., Garcia-Ojeda, J.C.. O-mase: a customisable approach to designing and building complex, adaptive multi-agent systems. *Int J Agent-Oriented Softw Eng* 2010;**4**(3):244–280. URL: `http://dx.doi.org/10.1504/IJAOSE.2010.036984`. doi:10.1504/IJAOSE.2010.036984.
26. Lützenberger, M., Küster, T., Hessler, A., Hirsch, B.. Unifying jiac agent development with awe. In: *Proceedings of the 7th German conference on Multiagent system technologies*; MATES'09. Berlin, Heidelberg: Springer-Verlag. ISBN 3-642-04142-6, 978-3-642-04142-6; 2009, p. 220–225.
27. Jayatilleke, G., Thangarajah, J., Padgham, L., Winikoff, M.. Component agent framework for domain-experts (cafne) toolkit. In: *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*; AAMAS '06. New York, NY, USA: ACM. ISBN 1-59593-303-4; 2006, p. 1465–1466. doi:10.1145/1160633.1160917.
28. Jayatilleke, G.B., Padgham, L., Winikoff, M.. Evaluating a model driven development toolkit for domain experts to modify agent based systems. In: *Proceedings of the 7th international conference on Agent-oriented software engineering VII*; AOSE'06. Berlin, Heidelberg: Springer-Verlag. ISBN 978-3-540-70944-2; 2007, p. 190–207.
29. Lützenberger, M.. *Development of a Visual Notation and Editor for Unifying the Application Engineering within the JIAC Framework Family*. Master's thesis; Technische Universität Berlin; Berlin, Germany; 2009.
30. Dikenelli, O., Erdur, R.C., Gumus, O.. Seagent: a platform for developing semantic web based multi agent systems. In: *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*; AAMAS '05. New York, NY, USA: ACM. ISBN 1-59593-093-0; 2005, p. 1271–1272. doi:10.1145/1082473.1082728.