

JADE BASED SOLUTIONS FOR KNOWLEDGE ASSESSMENT IN eLEARNING ENVIRONMENTS

Claudiu Anghel
aclaunic@yahoo.com
Cristian Godja
cgodja@yahoo.com
Mihaela Dinsoreanu
Computer Science Department
Technical University of Cluj-Napoca
Baritiu 26-29, RO-3400
Cluj-Napoca, Romania

Ioan Salomie
Department of Electronic and Computer Engineering
University of Limerick, Ireland
E-mail: Ioan.Salomie@ul.ie

KEYWORDS:

JADE, Mobile Agents, Student Assessment Service, Virtual Learning Environments.

ABSTRACT

E-learning is nowadays one of the most interesting of the "e-" domains available through the Internet. The main problem to create a Web-based, virtual environment is to model the traditional domain and to implement the model using the most suitable technologies. We analyzed the distance learning domain and investigated the possibility to implement some e-learning services using mobile agent technologies. From these e-learning services we focused on the student assessment service. We found JADE mobile agent framework to be the most suitable technology to achieve our goal.

INTRODUCTION

Almost every domain we know has nowadays its "e-" Internet-based counterpart. We talk about e-commerce, e-banking, e-learning etc. Each "e-" domain emulates the traditional one in a new, virtual, Web-based environment. The major problems of creating the virtual environment involve traditional domain modeling and implementing the model using the most suitable technologies.

As e-learning systems proved to be a handy and cheaper alternative to the classical face-to-face learning, they became in short time very used in learning processes. Moreover, there is a trend to backup the original face-to-face learning methods with the ones provided by an e-learning system. There is, already, a big interest in many countries worldwide to provide e-learning services for national education. Even if the replacement of the traditional learning with e-learning will not happen over night, the e-learning systems have to face a new set of requirements which come to meet the required services needed for a proper school education.

Our research is concerned with creating Web-based services for Virtual Learning Environments (VLE). This involves a complete analysis of the learning domain. The outcome of the analysis is the identification of the main concepts and relationships and building a conceptual model of the domain. On the other hand, the most appropriate technologies for implementing the model have to be analyzed and decided upon.

In this paper we focus on one aspect related to VLE, the Student Assessment. One of the most important educational components is the assessment of the student's acquired knowledge. There are several issues related to assessment that should be considered: communication issues, security issues, evaluation types, student answer analysis and grading.

This paper is structured as follows: an analysis of the Student Assessment domain is presented in Section 2, considering the most important concepts, constraints etc and building the conceptual model of the domain.

Section 3 presents our solution for the student assessment service based on the JADE Mobile Agent framework. Section 4 presents the architecture of our Student Assessment application that was built using the JADE technology described in the previous section.

We end with a discussion of some conclusions and possible developments in Section 5.

VLE

VLEs have to provide all the necessary resources for overcoming time and space limitations existent in traditional f2f environments. Students and instructors involved in a VLE can be located world-wide, they don't have to synchronize their communication, and their number is not limited.

Therefore, the services provided by VLEs should be designed considering issues like accessibility, scalability, security, communication etc. In our paper we will focus on one of the services of a VLE: the Assessment Service (AS). One big advantage of an automatized assessment process is that it eliminates the subjectivity that can be present in the traditional assessment process. Another advantage is that the automatized process is faster than the traditional one.

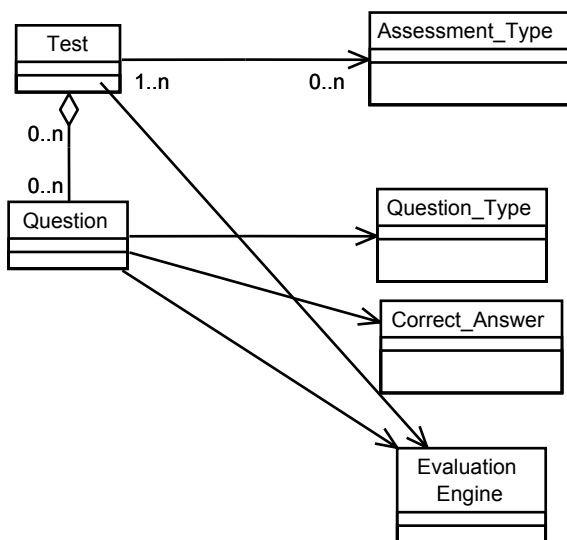
Student Assessment

AS provides the means of evaluating the students' acquired knowledge. It also provides the means for a student to get valuable feedback regarding his progress. AS is a highly dynamic component of the VLE, involving both synchronous and asynchronous communication between students and instructor. In order to build a model of AS, we analyzed the assessment process, different possible scenarios, and different assessment types. Based on the analysis we identified the main concepts involved and the relationships between them.

Main Concepts Identification

Analyzing the main concepts involved in student evaluation we identified the following:

- Learning entity (the Student)
- Teaching authority (the Instructor)
- Assessment or Test
- Assessment type (Compulsory Examination, Self-Assessment)
- Question
- Question Type
- Correct Answer
- Assessment procedure (as an Evaluation Engine)



The relationships between the concepts are depicted in a simplified manner in Figure 1.

Figure 1: Assessment Concepts and Relationships

The Teaching Authority provides the tests. A test may have different types and contains a set of questions. The two big categories we identified are Compulsory examinations and self-assessments. A question is associated to a question type, to one or more correct answers and also to an assessment procedure (implemented as an evaluation engine).

The evaluation engine provides the knowledge for evaluating the Student's answer against the correct answer(s) associated to a question. Student answers can be in a limited range, long natural language essays can not be analyzed.

The Student is able to access available self-assessments, to run a self-assessment and provide his answers. The student should also receive feedback regarding his performance (the grade, the correct answers etc).

For the compulsory assessments the assessment service should be able to get an attendance list containing the students that are ready to take the assessment, to give the assessment's questions to the each student, to get the answers from them, to set the conditions of the test i.e. the total available time for the test, the available time for each question and finally to evaluate the answers, to provide feed-back to the students and to store the results.

Main Functional Tasks

The next step in the analysis of the system is the representation of the functional requirements of the system. Considering the main external actors that interact with the system, the Student and the Instructor, we modeled the functional requirements in an UML-based manner as use-cases associated to the external actors.

The main functionalities of the system provided for the Student are: Visualization of Tests, Start a new Self-Assessment Test, View Test Results.

The functionalities provided for the Instructor are mainly: View Tests, Add new Test, Modify existing Test, Delete existing Test, View taken Tests, Schedule compulsory Tests

Other administration related functionalities are also considered

Non-functional Requirements

Besides the main concepts and functionalities described above, when modeling the AS service, some additional constraints have to be considered: independence of the VLE, independence of the implementation technology, scalability and accessibility. Another constraint is related to question/answer types. As mentioned above we did not consider Student answers as essays, therefore AS is more suitable for technical disciplines where the correct answers are in a limited range.

JADE MOBILE AGENTS – AN EFFICIENT SOLUTION

Since we are dealing with a highly distributed system and considering the constraints mentioned above, we investigated the possibility to provide a solution based on mobile agent technology.

The mobile agent technology can overcome some limitations of the well known server-client model:

1. scalability issues for many users – if the number of users increases over a threshold the server will not be able to process all the requests, causing long latencies and even requests loses.
2. bandwidth / latency – transferring all the evaluation engine information in one step (by using a mobile agent) reduces the overall network communications overheads, and also allows the user to access all required information instantly. More over, the evaluation of the test can be performed directly on the client's computer. In this way the server is freed from the task of grading students (which can be quite a consuming operation – if we think of the resources needed to evaluate natural languages answers for example)

Moreover an agent running on a client's (a student in our case) machine can bring many advantages; such an agent might:

- a. query some of the personal preferences of the student directly from the student's machine,
- b. integrate with some other office tools the student uses,
- c. monitor the activity of the student in the VLE - creating a user profile (areas of interest, learning speed, learning techniques)
- d. perform some background searches for materials that might be of interest for the student
- e. offer a message board to the student with the news of interest for the student.
- f. offer important messages for the student regarding his assessments / exams. This is achieved through the communication between this agent and an agent residing on the VLE's machine. In this way the student is always informed about his incoming exams, or about teachers that are online and available for direct chats, or about live discussion forums that might exist.

Maintaining a live connection to the server is one of the problems with the server-client models. The client must schedule a periodic request to the server to query for updates or news. To overcome this, the classic server-client model came with some solutions of its own, like the Push protocol. With Push protocol the server can initiate requests to the client, signaling that the data on the server has changed. This approach is used mainly for the PDAs that need some up-to-date information (like stock-exchange).

With mobile agents this drawback is overcome as the communication between the existing agents in the agency can happen in any direction.

Our goal was to design a multi-agent system that fulfills the functional requirements described respecting also the discussed constraints.

We started our design by mapping the functional requirements represented as use-case diagrams to a set of tasks the system has to perform.

In order to build a complete task model we considered a top-down approach decomposing more general tasks to specific subtasks. Next, we identified the necessary agent roles to perform the tasks. Agent roles define the position of the agent in the organization.

The organizational design actually consists of a set of models, each addressing one facet of the organization:

- **Environment Model**
The Environment model represents the available resources and also access protocols to resources.
- **Interaction Model**
The Interaction model represents the communication structure between agents.
- **Role Model**
The Role model is actually the authority structure in the organization. It links also tasks to roles.

Task Decomposition

Analyzing the use case diagrams that model the functional requirements of the system, we considered the following main tasks:

Communication Tasks

Communication is a key issue from both internal and external viewpoints. The organization obviously does not exist in isolation so it has to communicate to the exterior world. On the other hand we talk about an organization, so agents are supposed to communicate in order to achieve their goals. Therefore, we considered the two main communication types:

- Communication to external actors (Student, Instructor, VLE)
- Communication inside the system (modeled by Interaction Protocols)

To provide efficient communication to human external actors a **Personal Assistant Agent** was considered. The Personal Assistant (PA) is a stationary agent living on the client machine and providing the communication interface between the external actor (Student, Instructor) and the system.

Coordination Tasks

Besides communication, coordination of the organization is also a key issue. Coordination tasks involve: handling self-assessment requests, handling compulsory examinations, generating evaluation engines, performing evaluation etc.

The system was designed as a centralized coordinated system, the core of the coordination module being a **Server Agent**. The Server Agent (SA) is a stationary agent that lives on the AS machine and is responsible with handling self-assessment requests, examinations set by Instructors, generating corresponding evaluation engines etc.

For the evaluation itself we considered an **Evaluation Agent**. The Evaluation Agent (EA) is a mobile agent that migrates on the client (Student) machine and is able to perform the evaluation. EA is loaded with an Evaluation Engine containing the complete Test (questions, answer options, correct answer) and the assessment procedure.

The creation of the EA is also SA's responsibility.

We also considered other dependencies between tasks (Weiss 1999): pooled (results of one or more tasks jointly needed to perform another task), sequential (two or more subtasks should be performed in a specific sequence), reciprocal (two tasks depend jointly on each other).

Organizational Model

As previously stated, modeling an organization involves several concepts comprised in different sub-models of the organization. Our approach models a closed organization (no alien agents are allowed), containing benevolent, cooperative agents. We considered the following sub-models as components of our organizational model.

Environment Model

The Environment Model represents the resources available to the agents and the associated access protocols to them. We consider as resources both data and knowledge storage structures and other components (objects, servers etc) that provide specific services to agents. The design of the agents is independent of any specific resources. The Environment Model is represented by several UML-based package and class diagrams.

Role Model

This model contains the agent roles in terms of their tasks, interactions and accessible resources. As mentioned above we identified three agent roles like depicted in Figure 2.

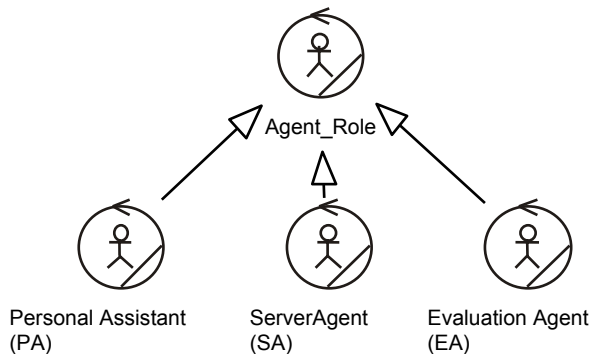


Figure 2: Agent Roles

Each role is associated to the set of tasks it's responsible for. We modeled the tasks as UML-type use-cases. In Figure 3 EA and its associated tasks is represented. EA is therefore responsible for traveling to the Student's site, for cooperating with the existing PA in order to perform the evaluation, for displaying the questions via a friendly graphical interface to the Student, for allowing the Student to enter his answers, for evaluating the answer and choosing accordingly the next question (adaptive behavior) and finally providing a result of the evaluation.

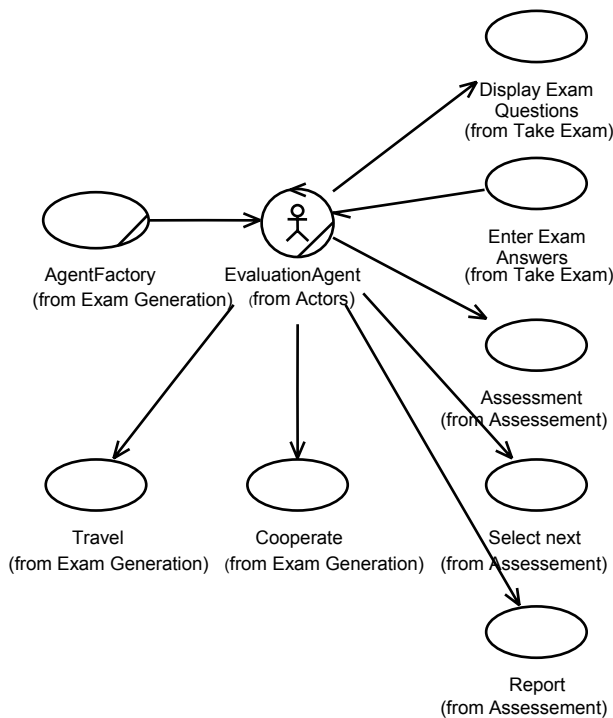


Figure 3: EA and associated tasks

Interaction Model

Interactions between agents are represented by communication protocols. These protocols are part of the social rules of the organization. The communication protocols details are represented as sequence diagrams (Bauer et al. 2001), (Bergenti and Poggi 2000), (Van Dyke Parunak and Odell 2002). We defined communication protocols between external actors and the system, and also between the agents inside the organization.

Now after we decided that our solution will be mobile agent based and after we identified the agents we need and the tasks these agents must perform as we explained above, a very important decision was to find the most suitable technology for implementation. Since we need a platform independent and on-line accessible system we choose Java, so we needed a Java based mobile agent system. More we wanted the mobile agent system to be FIPA compliant as FIPA is the organization responsible to standardize the agent based technology. We also wanted a system that is in growing and has possibilities to be extended in the future. We found JADE agent framework to be the best choice and the most suitable solution for our needs. We will show next how we used JADE features to implement our agents that are the core of our assessment service.

ServerAgent (SA)

This agent is the core of our assessment service. It is created on the main container of the JADE system and has three important parallel tasks to perform. Each of these tasks was implemented in separate JADE agent behaviour.

The first task is to load periodically from the data source, using the services provided by a data access layer, the information about the exams necessary to be started in the next period of time. This task was implemented by adding to the agent a **cyclic behaviour** in which the agent gets from time to time (e.g. every 12 hours) the necessary data from the data source (i.e. the database). The task was necessary to be added in order to limit the access to the data source which is time consuming.

The second task the server agent has to perform is to identify the exact moment when a scheduled compulsory assessment should be started and to trigger the assessment start. This task was implemented by adding to the agent another **cyclic behaviour**, in which the agent checks at very small time intervals if a new compulsory assessment should be started. If this is the case, then a new evaluation agent (EA) is created (see below) encapsulating all the data and knowledge required to perform the evaluation and a message, in form of an **ACL message** (ACL – Agent Common Language), is sent to all personal agents (PAs) residing on the student machines, informing these agents that a compulsory assessment must take place. The message is sent only to the PAs of the students that have to take the compulsory assessment. To this message, the PAs that are on-line will respond with a message requesting the migration of the evaluation agent.

The third task has to assure the communication with the personal agents (PAs see below). The communication is done by the exchange of ACL messages. The task is implemented by adding another behaviour, a **receiver behaviour** to the server agent. In this behaviour the server agent waits for messages from personal agents that may want to start self assessments for their students. When such a message arrives the server agent creates a new evaluation agent and responds to the personal agent when everything is ready. Finally the evaluation agent migrates on the student's machine for performing the whole assessment process.

Evaluation Agent (EA)

The evaluation agent is created by the server agent on the main container of the JADE system. It encapsulates all the necessary data and knowledge to perform an assessment (compulsory or self), migrates on the client's machine, communicates with the student giving the questions and getting the answers, performs the evaluation after getting the answers and finally goes back on the main container and persists the results within the data source of the assessment service.

Personal Agent (PA)

The personal agent has to assure a proper communication with both the human actors, i.e. the students, and the server agent. A particular task of PA is to start an assessment on a received SA request. This was implemented in a receiver behaviour. When receiving the request from SA the PA triggers the migration of the evaluation agent.

The most important problem we needed to solve was to assure the proper conditions on the client's machine for our agents EA and PA. In order to make the Student Assessment Service really usable we considered one important non-functional requirement: to allow the service to run on client's machine (this will be the student's machine) with a minimum (or no) installation required. The JADE system proved to be a very good solution since it offers the possibility for agents to live on the client's machine without them needing more than a Java enabled web browser to be installed on that machine. The adopted solution was to create an applet on the client's browser. Using the JVM, created by the browser to run the applet, a new agent container is created on the client's machine using JADE API calls.

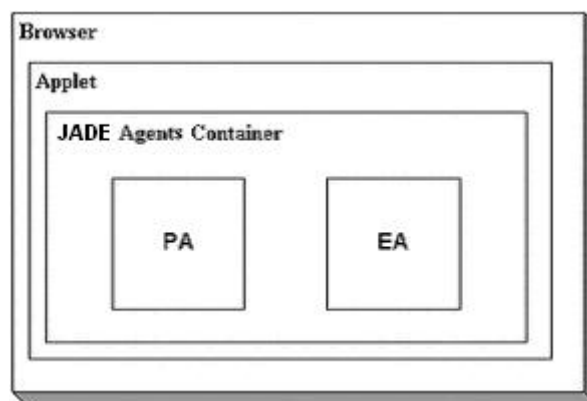


Figure 4. JADE Agent Container on client's machine

Once the agent container is ready, the Personal Assistant agent is created in this container with the behaviour described above. From now on the Personal Assistant can communicate with the Server Agent (SA) or with the EA through the ACL messages. There is one more problem to solve and that is the communication between our agents, the Personal Assistant (PA) and the Evaluation Agent (EA), and the student. The Personal Assistant must assist the student in his learning process so the PA needs to learn what the student does. For example if the student wants to start a self assessment the PA must determine that and send an ACL message to the SA with the student's request.

Starting a self assessment will be available for the student by clicking on the assessment name from a list in the available assessments sections. To be able to intercept this a javascript code will be executed from the HTML page that will send to the applet the students request, which in turn will sent the request to the PA.

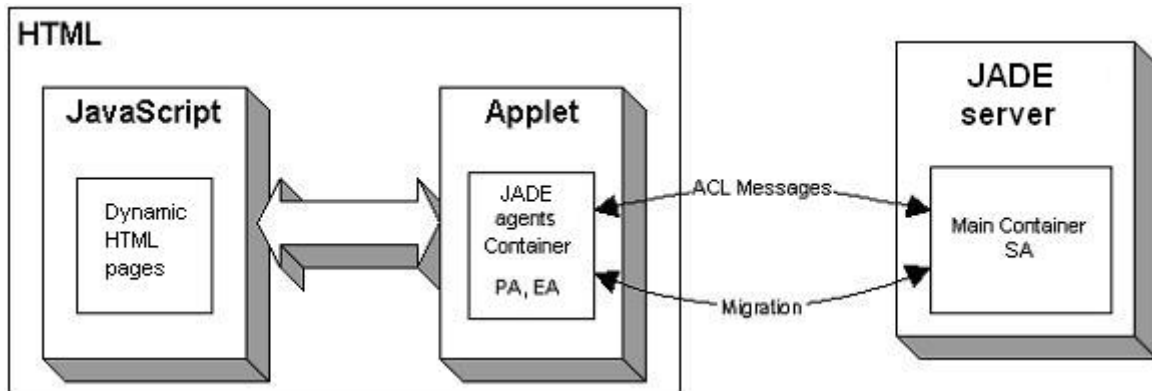


Figure 5. Complete interaction model

This interaction model is also used for the communication between the EA and the student, which takes place during the assessment.

Assessment Service Architecture

Considering the models developed in the analysis phase, we designed a general architecture of the application. The main considered issues were: distribution, reliability, scalability, platform independence, data storage independence, error proof. The architecture has to be therefore well structured and layered.

We considered a multi-layered structure containing well delimited, independent modules. Modules on lower levels provide services to modules on the upper levels.

The main modules of the system are:

- GUI – User Interface Module containing three submodules:
 - Instructor Interface Module
 - Student Interface Module
 - Admin Interface Module
- BL – Business Logic Module being together with MA (Mobile Agent Module) the core of the system
- MA – Mobile Agent Module – JADE based and described in the previous section
- DAO – Data Access Module – provides primitive data access operations (store, retrieve, update). Isolates the system from the data storage support assuring independence.
- Utility Server – provides services to other modules. Allows for different configuration settings.

The general architecture is presented in Figure 5.

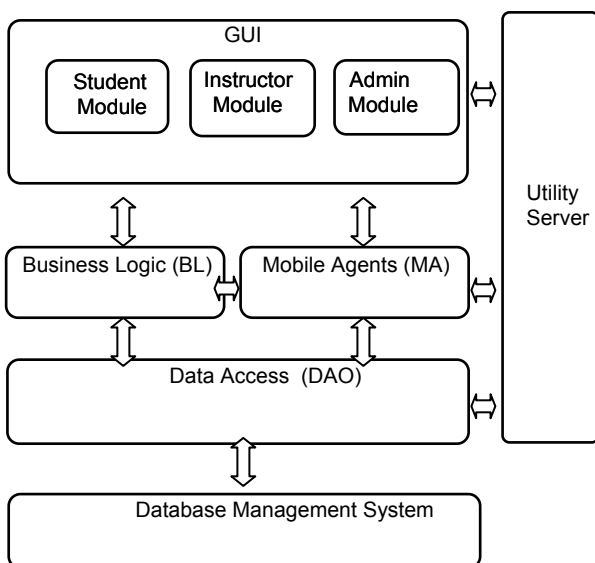


Figure 6: Assessment Service Architecture

The implemented application uses Java technologies (JDBC, RMI, Applets, JSP, Java Beans, etc) and JADE mobile agent platform (JADE 2002).

To provide a high level of flexibility an Evaluation Engine Factory is used inside the mobile agent module, that should create specific Evaluation Engines for specific assessment configurations. The Evaluation Engine is attached to an EA and provides its ability to analyze the student's answer and to match it against the expert answer, therefore being able to evaluate it.

Another important component is an Agent Factory that actually creates EA's. The EA is a mobile agent, loaded with assessment knowledge (the Evaluation Engine), with a set of questions and expert answers. The EA travels to the student's site and co-operates with the PAA in order to get the assessment done. The EA has an adaptive behavior depending on the student's answers.

The Evaluation Engine will be able to manage different test types like: multiple choice tests, short answers using natural language etc. using a natural language processor based on latent semantic analysis approach.

CONCLUSION

We are interested in our work to analyze and model specific areas of Virtual Learning Environments and to investigate the most suitable technologies to implement the developed models, particularly mobile agent-based technologies since we are dealing with a distributed and complex environment.

We believe that for the Student Assessment Service implementation, JADE technology offered us the support we needed.

We considered future developments using JADE for implementing other VLE services. Our next efforts will be in the direction of extending the e-Learning services to the wireless systems (like mobile phones) and we are confident in achieving this as JADE 3.0 is already "paving the way for wireless agent systems".

REFERENCES

- M.Dinsoreanu, C. Godja, C. Anghel, I.Salomie and T.Coffey 2003 - Mobile Agent based Solutions for Knowledge Assessment in eLearning Environments – Euromedia 2003
- Anghel C. 2003 <http://jade.csel.it/thirdpartysw.html#applets>
- JADE 2002. <http://sharon.csel.it/projects/jade/>
- Weiss G. 1999. "Multi-Agent Systems, A Modern Approach to DAI", *MIT Press*.
- Bauer B.; J.P.Mueller and J. Odell. 2001. "Agent UML: A Formalism for Specifying Multiagent Interaction", in P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*. Springer-Verlag Lecture Notes, Berlin, pp.91-103.
- Bergenti F., A. Poggi. "Exploiting UML in the Design of Multi-Agent Systems". 2000. In A. Omicini, R. Tolksdorf, F. Zambonelli, eds., *Engineering Societies in the Agents World - Lecture Notes on Artificial Intelligence*, volume 1972, pp.106-113, Berlin, Germany, Springer Verlag.
- DeLoach S.A.; M.F.Wood and C.H.Sparkman 2001, "Multiagent Systems Engineering", *IJSEKE*, Vol.11, No. 3 231-258.
- DeLoach S.A. 2000. "Specifying agent Behavior as Concurrent Tasks: Defining the Behavior of Social Agents", *AFIT/EN-TR-00-03*. Technical Report.
- Sparkman C.H.; S.A. DeLoach and A.L. Self. 2001 "Automated derivation of Complex Agent Architectures from Analysis Specifications", *AOSE – 2001*, Montreal, Canada.
- Van Dyke Parunak H. and J. Odell. 2002. "Representing Social Structures in UML", *Agent-Oriented Software Engineering Workshop II*, Michael Wooldridge, Paolo Ciancarini, and Gerhard Weiss, eds., Springer, Berlin, pp. 1-16.
- Wooldridge M., N.R. Jennings and D. Kinny. 2000. "The Gaia Methodology for Agent-Oriented Analysis and Design", *Autonomous Agents and Multi-Agent Systems*, 3(3): 285-312, September.
- Zambonelli F., N.R. Jennings, A. Omicini, M. Wooldridge. 2000. "Agent-Oriented Software Engineering for Internet Applications". In *Coordination of Internet Agents: Models, Technologies and Applications*. Springer-Verlag.
- Zambonelli F, N.R. Jennings, M. Wooldridge. 2001. "Organisational abstractions for the Analysis and Design of Multi-Agent Systems", in P. Ciancarini and M. Wooldridge, editors, *Agent-Oriented Software Engineering*. Springer-Verlag Lecture Notes in AI Volume 1957, January 2001.