CIT. Journal of Computing and Information Technology, Vol. 27, No. 1, March 2019, 25–42
doi: 10.20532/cit.2019.1004464

Role of Agent Middleware in Teaching Distributed Systems and Agent Technologies

Costin Badica[1], Milan Vidaković[2], Sorin Ilie[1], Mirjana Ivanović[3] and Jovana Vidaković[3]

[1]University of Craiova, Department of Computers and Information Technology; Romania
[2]University of Novi Sad, Faculty of Technical Sciences, Department of Computing and Control; Serbia
[3]University of Novi Sad, Faculty of Sciences, Department of Mathematics and Informatics; Serbia

Computer science and information communication technologies are among the fastest changing areas and it is essential to follow this world-wide trend also in education, constantly innovating and adapting curricula. In this paper, we introduce the structure, methodological aspects and educational experiences of teaching two courses on distributed systems and agent technologies at two different universities and countries. The presentation is focused on the role of agent middleware and multi-agent systems in teaching various theoretical and practical aspects of these courses. At the University of Craiova, the conclusion is that the use of agent middleware in general and of JADE platform in particular for teaching the course Distributed Systems certainly brings many advantages, but also has some limitations. At the University of Novi Sad, within the Agent Technologies course, agent middleware, initially developed as part of the research project, has been successfully used for educational purposes, too. For both courses, we present the structure, the tools, teachers' and students' experiences and joint useful conclusions and lessons learned with regard to courses delivery.

ACM CCS (2012) Classification: Computing methodologies → Artificial intelligence → Distributed artificial intelligence → Multi-agent systems

Applied computing → Education → Interactive learning environments

Computing methodologies → Distributed computing methodologies → Distributed algorithms

Computer systems organization → Architectures → Distributed architectures → Client-server architectures

Keywords: distributed software technology, multiagent middleware, educational experience

1. Introduction

Computer science (CS) and Information communication technologies (ICT) curricula must be under constant reevaluation and development as nowadays these two areas have been rapidly changing. Therefore, it is significant to impose and suggest adequate approaches for revitalizing CS and ICT education and curricula. Significant changes in developing modern software have happened in the last two decades. We are now in a world of computing, where basically everything is distributed in the broader sense, *i.e.* computing devices are interconnected, they use heterogeneous software and hardware platforms, and they exchange information via heterogeneous network communication channels. Facing this reality, technologies of distributed computing are developed and diversified with the spread of new platforms, architectures and languages for applications that could not have been imagined before, like ubiquitous and pervasive computing, mobile computing, sensor networks, high-performance computing, cloud computing, or the Internet of things. Therefore, rigorous design, integration, and harmonization of various topics of distributed systems into CS and ICT curricula, based on the most recent technological developments, presents a quasi-permanent challenge taking into account the various constraints of time, resources, effort, and expertise of educators and students.

Motivation and discussion of the structure of a core Distributed Systems and Agent Technologies courses, as well as their integration into

CS and ICT curricula, are not an easy task and it would probably require more space than is available in this paper.

At the University of Craiova, based on our 6 years experience in teaching a one-semester mandatory course in Distributed network application development (DNAD hereafter) to undergraduate CS and ICT curricula, in teaching the various concepts of distributed systems, we focus on the role played by multi-agent distributed middleware.

Our investigation is triggered by the following core question:

> Q: *Why and how can agent middleware play a relevant role in teaching topics of distributed systems in CS and ICT curricula?*

Based on our research and educational experiences, we do believe that agent middleware is relevant for teaching several theoretical and practical aspects of distributed systems and agent technologies. We will provide arguments for this statement in the paper.

The same core question Q inspired us at the University of Novi Sad to select particular agent middleware for elective Agent Technologies (AT hereafter) course and draw some useful educational and methodological conclusions. At the University of Novi Sad, until recently, undergraduate CS and ICT curricula only had one course in artificial intelligence (AI) where general AI topics were covered. Rarely such courses presented distributed systems and agent technology in more details. We realized that distributed systems and agent technologies are sufficiently mature, and it was a challenge to introduce such topics to undergraduate studies. Unfortunately, we faced some obstacles. First, there is a great discrepancy between our students' previous knowledge and ambitions. The majority of students lack motivation for studying and gaining a higher level of knowledge and skills in any course and they try to avoid demanding courses. On the other hand, a minority of students have shown to be highly motivated and prefer to be challenged with new, interesting and innovative courses. We have also faced some general obstacles such as the lack of a choice of appropriate textbooks and teaching materials. Nevertheless, that this course is elective, students' motivations to enroll in it are very diverse. With the need of an appropriate software tool for the practical part of the course, we also concluded that use of the in-house developed system would obtain easier maintenance of students practical and laboratory activities.

In this context, the paper brings to the readers some experiences gained in teaching two similar courses at two universities from two countries, delivered by the teachers who have been collaborating for a decade.

On one hand, we present our approach and conclusions on using agent middleware to support the lectures, lab and project activities during the DNAD course that was taught for 6 years to CS undergraduates at the University of Craiova, Romania.

On the other hand, we also elaborate on some challenges and initial experiences in the delivery of elective courses on agents and multi-agent systems, to undergraduate CS curricula at the University of Novi Sad during a time period of 3 years.

Some common experiences in delivering these courses are identified and we will present our joint conclusions that could be useful for those teachers who are considering introducing topics/courses on distributed systems and agent technologies within CS and ICT curricula at their universities.

The paper is organized as follows. We start in Section 2 with a background and an overview of various sources for developing a computer science course focused on topics in distributed systems and multi-agent technology. In Section 3, we follow with an overview of the course on distributed network applications development. Section 4 brings an overview of Agent Technology course. In Section 5, educational experiences from delivering both courses are presented and discussed. They could be useful for teachers and educational institutions that plan to introduce similar courses in their curricula. Last section brings some concluding remarks.

2. Background

2.1. Related Work

Rapid development of ICT has influenced important trends oriented towards distributed, pervasive networks and Internet of things, where agents and agent technologies play essential role. For their future jobs in ICT companies, students would benefit from learning and understanding new and challenging environments and implementations of softwares such as ubiquitous and pervasive computing, sensor networks, Internet of things, high-performance computing, mobile computing, cloud computing and even collective intelligence [1]. Therefore, rigorous design, integration, and harmonization of various topics of distributed systems and agent technologies into CS and ICT curricula presents a permanent task requiring expertise adaptation of educators and students [2].

Despite the fact that there are different courses on distributed systems and agent technologies delivered at universities world-wide, there are not too many papers that report on educational effects and students' motivation, results and achievements. It is also worth noting that multi-agent approaches are rather diverse, providing a wider perspective on computer science methods, spanning various topics reaching applications, intelligent methods, new programming paradigms and software technologies, possibly, but not necessarily, in connection with distributed systems and technologies.

There are some papers that present experiences in developing and using a variety of agent environments. For example, [3] and [4] provide an extensive overview of such environments, but there are only a few papers describing effects and students'/teachers' experiences in delivering distributed systems and agent-oriented courses.

An interesting approach for revitalizing introductory undergraduate CS curricula through the integration of agent-based modeling and multi-agent systems is presented in [5]. Authors decided to use their own system, *i.e.* MAgICS (Multi-Agent Introduction to Computer Science) framework. They introduced a range of rather standard topics (searching and sorting, machine learning, networks and security), but put a special focus on parallel, distributed, and stochastic methods. Teachers' primary teaching/learning goal was to enable students to think in decentralized manner but also to understand the trade-offs between centralized and decentralized approaches. Additionally, rather advanced achievement was that students were able to consider issues of distribution and parallelism from the programming, as well as from the conceptual design of systems point of view.

NetLogo was proposed as an excellent platform for teaching intelligent agents within Multi-agent systems course, in paper [6]. The authors discussed a number of interesting features of the platform and their educational value: expressive and rather simple programming language with a small learning curve, rapid GUI creation and custom visualizations, significant features that facilitate modeling of complex environments and agents, *etc*. In the first several years of their course delivery, the students enjoyed the course in spite of the fact that it was rather theoretically oriented. Obviously, courses on distributed systems and agent technologies require practical aspects and program development. It is one of the significant drawbacks of these authors' approach. But they persistently continued to teach similar topics in their CS studies and reported in [7] about new, rather specific experiences. In fact, they presented a series of modules within study program, that progressively address other related topics necessary for their course. Finally, in the last year of study they deliver a course on multi-agent systems and principles of robotics. Within the course, they organized a Robotics Challenge which gave students the opportunity to integrate gained knowledge and skills in order to solve a real problem. According to the authors' report, this approach, which seems a little bit demanding compared to our approaches, was very well received by students.

Another educational approach based on NetLogo platform is presented in [8]. It was used in elective AI course, fifth semester of the Bachelor of Computer Science. Significant part of this course was devoted to autonomous agents and multi-agent systems. In fact, the main idea presented in the paper was to extend a BDI (Belief-Desire-Intention) library in NetLogo based on a specific case study and students' solutions. Students were given the "possibility to choose either to complete a final exam to evaluate the module Autonomous Agents and Multi-Agent Systems, or to work on a course project using NetLogo and BDI" [8]. It is interesting to note that almost all students decided to follow the practical work on projects. Authors specified the undergraduate students' research projects from computational economics and the main task was to enhance the existing library in order to make students familiar with a more sophisticated form of the BDI model in NetLogo.

Extensive experiences in teaching several undergraduate courses on distributed systems and agents and multi-agent frameworks at our two universities are presented in [2], [9]. We paid attention to three key issues in our courses.

The first aim was to support students' intuitive understanding of vital concepts of distributed systems and agent technologies. The second aim was the selection of appropriate teaching material and we pointed out some criteria useful for selecting and/or preparing adequate teaching material. The third aspect was oriented towards specification of good motivational and inspiring examples and problems that students have to solve practically during lab classes, home-work and projects.

An important part of courses delivery is devoted to lessons learned and feedback from students in order to try to improve the courses from year to year.

Practical aspects of courses on distributed systems and agent technologies can be of great importance for students and their future jobs in companies. So the decision which environment, platform, middleware to use is important for teachers and it can significantly influence students' motivation, learning curve, acquiring practical skills and general satisfaction with courses' effects.

Following [10], agent systems are deployed over specialized software infrastructures that provide a basic set of functionalities for the existence of a realistic multi-agent application. Seen from the perspective of distributed systems and agent technologies, such infrastructures are placed at the middleware level. They define a software layer that

(i) assures platform (here understood as hardware + operating system) independence and

(ii) provides a collection of software functionalities and services, including: agent lifetime management, agent communication and message transport, agent naming and discovery, mobility, security, *etc.*

An *agent framework* is a software infrastructure that is available as software library, programming language environment, or both, and provides the core software artifacts needed for creating the skeleton of a multi-agent system. A software package that provides the core runtime functionalities for deploying and running distributed multi-agent applications is traditionally known as *agent platform*. Typical *agent middleware* provides both, an agent platform and an agent framework.

More than a hundred of agent platforms and toolkits that differ in maturity, quality, standards compliance, and complexity are reported in the literature [10]. One of the most popular, well-documented, FIPA[1] compliant, and easy-to-program agent packages is JADE [11]. So at the University of Craiova within our DNAD course, we decided to use JADE.

Our intention within AT course at the University of Novi Sad was to prepare a smooth introduction in agent technology and multi-agent systems (MASs) using, as much as possible, our previously developed teaching resources and self-developed software tools for creation of MASs [2], [9]. The main advantage of our approach is that we use in-house agent middleware developed within our research activities, for teaching as well. So we are able to offer thorough knowledge of our system to the students. Also, it is possible to easily make corrections and extend the system, because we do not depend on external programmers. Students performed different tasks with the system and used it to implement simple MAS, while during programming and testing their solutions, they could also test our framework.

2.2. Computer Science Curricula Recommendations

ACM and IEEE are continuously developing, revising, refining, and adapting recommendations to help academic educators with the design and further adaptation of CS and ICT curricula by incorporating the most recent results of CS and ICT research and by addressing the new market requirements for such professionals. The recommendations are available as Computer Science Body Knowledge – CSBK, the last version being issued in 2013 [12].

Moreover, CS scholars are providing textbooks in distributed systems, like for example [13].

[1]http://www.fipa.org/

There are also some well-known introductory textbooks on Agent Technologies [14]. Those textbooks, as well as CSBK, are valuable sources of knowledge and methodology for teaching Distributed Systems and agent topics to CS and ICT undergraduates. Unfortunately, students usually avoid reading the books, while looking for some compressed/digested versions of teaching materials, so we also had to take care about that in delivering our courses.

CSBK provides comprehensive structuring of CS knowledge into 18 *knowledge areas* – KA that we had to take into account when preparing our courses. Each KA is decomposed into a number of *knowledge units* – KU, while each KU is further divided into a number of *topics* with associated *learning outcomes* – LO. Each LO must have associated a certain *level of mastery* from the available set: familiarity, usage, and assessment.

Three CSBK KAs that are relevant for the development of courses, including Distributed Systems and agent topics were recently updated to reflect the current developments in computing and information technology industry:

- Networking and Communication (NC) KA was split, because of its growth and divergence. A part of it was included in PBD (see below).
- Platform-Based Development (PBD) is a new KA, with its content mainly grown from the NC KA.
- Parallel and Distributed Computing (PD) is a new KA that has acquired topics that were previously spread in other KAs.

3. Distributed Network Application Development (DNAD) Course

3.1. Overview of DNAD Course

In developing Distributed Network Applications Development (DNAD) course at the University of Craiova, we took into consideration the definition of course learning objectives and the availability of course prerequisites. The DNAD course is scheduled in the 3rd year, 6th semester of Computer Science curricula.

Firstly, the course learning objectives were formulated in accordance with recommendations provided by CSBK on the topics that were considered relevant for distributed systems, as well as based on standard textbooks in distributed systems:

- **LO1**: To introduce the principles and concepts of distributed software.
- **LO2**: To introduce the basic technologies of distributed software with a focus on core middleware technologies based on Internet.
- **LO3**: To provide an opportunity to obtain practical experience in applying these techniques for programming small-scale distributed software applications.

Secondly, the course prerequisites were established, taking into account the current structure of our computer science curricula. The students must be familiar with theory and practice of computer programming (including object-oriented programming using the Java programming language), operating systems and computer networks. The courses that directly benefit from DNAD are Electronic Commerce and Web Application Design.

Thirdly, a course structure was defined, including lectures, labs, and project activities. Finally, we added topics on agent middleware that were aimed to support teaching of several conceptual and practical aspects of distributed systems, while trying to correctly fit agent middleware into the concepts and approaches of distributed systems.

The DNAD course is structured in two modules with separate grading:

(i) Course module comprising lectures and labs with 4 ECTS points;
(ii) Project module with 1 ECTS point.

Both modules last 14 weeks with lectures 2h/week, lab 2h/week and project 1h/week.

DNAD course uses a technology-centered pragmatic approach to teach distributed systems. The course is focused on applications and programming in the spirit of [15] and [16], rather than on theory and algorithms, as for example in [17]. Nevertheless, important concepts are firstly introduced and then exemplified with the help of technologies and frameworks.

Similar pragmatic approaches in teaching distributed systems applications to undergraduates were already used, like for example [18]. However, the special feature of our approach is the significant role that we assigned to distributed agent middleware in teaching the concepts and practical applications of distributed systems.

There is no single textbook to cover the course content. Nevertheless, for our course, a useful starting reference is [13]. The course lectures comprise the following list of topics:

- Introduction to Distributed Systems: definition, classification and characteristics,
- Models of Distributed Systems: physical, architectural, and fundamental models,
- Inter-process communication in Distributed Systems: TCP, UDP, group communication,
- Core technologies for Web-based Distributed Systems: HTML/CSS, XML, HTTP; Web clients and servers; Servlets and Apache/Tomcat,
- Object-based Distributed Systems and Remote Method Invocation: Design of RMI, Programming Java RMI,
- P2P systems: Structured and unstructured overlay networks,
- Agent-based Distributed Systems: FIPA and JADE,
- Web Services: Concepts and standards; Axis2.

DNAD course addresses several KUs that are part of three KAs of CSBK, as follows (note that two KUs are on our wish-list, as future work):

- KA: Networking and Communication (NC): KU – NC/Networked Applications,
- KA: Platform-Based Development (PBD): KU – PBD/Introduction; KU – PBD/Web Platforms; KU – PBD/Mobile Platforms (on the wish-list as future work),
- KA: Parallel and Distributed Computing (PD): KU – PD/Distributed Systems; KU – PD/Cloud Computing (on the wish-list as future work).

Grading of the DNAD course module is based on final exam and lab assignments. The final exam counts 60% of the final mark and comprises 30% based on a questionnaire of knowledge questions and 30% based on "apply skills" exercises involving the design of a small-scale distributed software application. The lab grading counts 40% of the final mark and it is determined from the outcome of a set of lab assignments. Their number depends on difficulty, and it is usually chosen between 3 and 5. During the semester, as part of the lab activity, the students are also exposed to a number of tutorials that introduce the software packages and tools required for carrying out their lab and project assignments.

Grading of the DNAD project module is based on a project assignment and associated deliverables: a project report and a software package. The grade is split as 20% for the intermediary report and 80% for the final report and software deliverables.

3.2. Role of Agent Middleware in DNAD Course

Agent middleware can play (and actually played) an important role in teaching our DNAD course. In order to support this statement, we performed a thorough analysis of the JADE agent middleware – JADE [11] with respect to the requirements set by the learning objectives, structure and topics of the DNAD course.

Firstly, JADE agent middleware supports all the course learning objectives. It supports **LO1**, as using JADE examples, we can explain many principles and concepts of Distributed Systems, including: platform heterogeneity management, transport protocols, white and yellow pages (naming and directories), code mobility, fault tolerance (JADE supports a limited form of fault tolerance), and interaction protocols based on message exchange. Conceptually, following the classification of architectural models proposed in [13], JADE uses a *software component-based model*. This means that JADE agents are actually software components (*i.e.* dynamically loadable objects enabling runtime configuration) that support asynchronous message passing for agent communication in the P2P style (although JADE itself cannot be characterized as a true P2P system). Moreover, JADE is standards' (*i.e.* FIPA) compliant. Partly, it also supports **LO2**, as JADE itself can be described as a middleware platform. It also well supports **LO3** with the low cost of a smooth learning curve by providing a meaningful and well-documented API that helps students to acquire skills for developing JADE-based small-scale distributed software in due time and with reasonable effort.

Secondly, using JADE as an example, we can cover part of the topics included in DNAD course – both lectures and laboratories. In particular, JADE is a good example of component-based distributed middleware platform, well supporting students' practical work.

The interaction model (one of the fundamental models [13]) of a distributed system actually corresponds to a distributed algorithm and it can be described in a disciplined way as a set of communicating state-machines [17]. This model can be naturally implemented using JADE agents with the help of finite-state machine behaviors, *i.e. FSMBehaviour* class [11].

JADE can be also used to introduce the service-oriented architecture. JADE agents can expose services registered in a yellow pages directory – the *Directory Facilitator* agent. Services can be named, searched in this directory, and then invoked using interaction protocols, thus supporting one of the basic architectural patterns of service-oriented computing. Services provided by JADE agents can be encapsulated into and exposed as Web Services [19], allowing the integration of distributed multi-agent applications into the Web environment, and enabling use of JADE as integration middleware of distributed heterogeneous applications.

Using JADE, we can also exemplify an elegant model of object serialization based on Java Beans and semantic web technologies. Firstly, JADE provides an API for manipulating ontologies that supports packing and unpacking of complex objects when they are exchanged between agents via FIPA ACL messages. Moreover, the specialized *Ontology Bean Generator* tool is available to facilitate the engineering process of JADE ontologies [20].

One of the weaknesses of JADE, however, is the integration with standard Web technologies. Although possible, the development of a Web-based application that integrates JADE on the server-side is not natural and it requires the use of additional software glue known as *JADE Gateway* [21]. This facility was nevertheless useful for project activities, where students chose to develop a Web-based interface to a JADE-based distributed multi-agent application.

Finally, it is worth mentioning that JADE can also support the development of mobile computing applications on Android-based smartphones [22]. This can be very useful with regard to the extension of DNAD course with topics covering the *Mobile Platforms* KU. Moreover, JADE can be also useful in the near future to support lab and project activities involving the development of distributed mobile applications with JADE agents running on Android-based smartphones.

Before concluding this section, we would like to mention that although JADE (and multi-agent middleware in general) can support the teaching of distributed systems in multiple ways, it is by far not the silver bullet. There are many other aspects related to concepts and technologies of distributed systems that require additional examples and tools to support a good coverage of the subject. Some example topics that require different tools are web technologies, RMI, P2P systems, cloud computing, ubiquitous and pervasive computing. Taking also into account that it is probably impossible to find a single tool to cover all the topics, we can conclude that the correct approach, we also followed, is to carefully analyze the course curricula (lectures, labs, and project) and decide where precisely agent middleware can serve as a relevant example, as well as a practical implementation tool.

4. Agent Technologies (AT) Course

4.1. Overview of the AT Course

At the University of Novi Sad, Faculty of Technical Sciences, the main idea within the elective course on Agent Technologies (AT) was to motivate better students to learn some new, modern and challenging technologies, but also to give them opportunity to bridge the gap in the understanding of a particular area of distributed systems *i.e.* multi-agent systems as one among essential components of smart environments and computational collective intelligence areas. The course on AT is scheduled in 4th year 8th semester and is based on multi-agent middleware that has been developing at our University during the last decade [9].

Firstly, the course learning objectives also arose as a consequence of recommendations provided by CSBK and were formulated in accordance with some of those relevant for distributed systems and agent technologies. AT course addresses several KUs that are part of three KAs of CSBK, as follows:

- KA: Networking and Communication (NC): KU – NC/Networked Applications,
- KA: Platform-Based Development (PBD): KU – PBD/Introduction; KU – PBD/Web Platforms,
- KA: Parallel and Distributed Computing (PD): KU – Distributed Systems.

As we could not find appropriate textbook for our approach to AT course, we prepared the teaching materials in the following way. We started with selecting some topics of textbooks in Agent technologies, but predominantly used our own experience in the field, as well as contemporary freely available papers published in the past 15 years. Accordingly, learning objectives were formulated:

- **LO1**: Present essential concept of intelligent agents, different types of agents, communication protocols, and supporting architecture for their operation.
- **LO2**: Explain essential differences between single agent and multi-agent systems and crucial facts about agent lifecycle management, communication, and interaction.
- **LO3**: Discuss possible applications of MASs and emphasize advantages of the agent-based approach for solving complex engineering problems. Also, provide the opportunity to obtain practical experience in applying gained knowledge to solve slightly complex problems, preferably in the AI realm.

Secondly, the course prerequisites by some other courses at our study program, more or less related to AI, like "Basic Artificial Intelligence Techniques" and "Business Intelligence". Through mentioned courses, students gained essential knowledge of characteristic AI topics, including planning, knowledge-based systems, fuzzy rule-based systems, genetic algorithms, and machine learning. As they are 4th year students, we expect that they:

(i) are familiar with and skilled in programming and using the Java and JavaScript programming languages,

(ii) have adequate knowledge in the areas of operating systems and computer networks.

With such previously adopted knowledge, the course on AT is not too demanding and is valued by 4 ECTS points.

The structure of the course is prepared in a stepwise manner. The course encompasses regular lectures about theoretical aspects, and lab exercises, where students use middleware for solving smaller lab assignments distributed over the semester and supervised by teaching assistant. In the last part of the course, students have to implement a slightly complex problem (*i.e.* the design of a small-scale distributed agent middleware) by themselves. The course lasts 14 weeks, with 3 hours/week for theoretical lectures and 3 hours/week for laboratory work. During theoretical classes, students learn fundamentals of the agent technology, organization of agent code, agent lifecycle, FIPA protocols and FIPA ACL. They expand their technical knowledge with all necessary skills for the agent middleware development, like: Java EE, aspect-oriented programming, and advanced JavaScript. Laboratory tasks are closely related to the topics presented during theoretical classes. For lab exercises, students are divided into small teams (3 or 4 members) with the primary task to create agents capable of communicating with several different types of agents on the same computer, to employ FIPA communication protocols, and finally to perform a communication between multiple agents on several computers. Students are given several examples of inter-agent communication based on the FIPA Protocols.

The final grade is composed of three components:

1. 30% is based on a questionnaire for assessing gained theoretical knowledge;
2. 30% is based on lab achievements and

3. 40% is based on the evaluation of quality of the project completed as part of homework activities.

During the course, students are also introduced to several tutorials of necessary frameworks, libraries, and application servers, to be able to cope with project activities that require implementation of small-scale distributed agent middleware.

Since 2014/2015 academic year, between 12 and 29 students have been selecting this elective course every year.

4.2. Role of the In-house Developed Agent Middleware in AT Course

Similarly, as within DNAD course at the University of Craiova, we concluded that successful delivering of a highly practically oriented AT course, requires appropriate agent middleware. We have been considering several possibilities, including a thorough analysis of the JADE agent middleware functionalities. Our motivation and intention was to use agent middleware that will satisfy, as much as possible, AT course learning objectives with the low cost of a smooth learning curve. We also thought that it is important to provide a meaningful and well-documented software that helps students to acquire skills for developing distributed solutions in due time and with reasonable effort.

Therefore, we finally decided to use, for all course activities, an in-house implemented very high-quality and robust agent development framework *i.e.* the Siebog agent middleware [9]. Siebog integrated two (see Figure 1), also in-house, developed, components: XJAF and Radigost, to support server-side and client-side

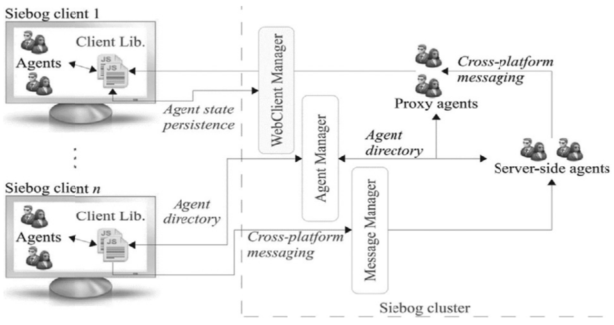

Figure 1. Siebog Architecture.

agents. Server-side agents are supported by the XJAF component of the Siebog framework.

Agents developed in Siebog are load-balanced and can be executed in the distributed, clustered environment, safely transferred to another node if their node in the cluster fails [9], [23].

Client-side agents (written in JavaScript) are supported by Radigost component and can migrate to the server-side if necessary. Being implemented as JavaScript objects and loaded within the web page, they can execute in a wide range of devices, such as desktops, laptops, tablets, smartphones, smart TVs, *etc*. Since the state of a client-side agent can be destroyed when the page is left (or the browser is closed), it was necessary to develop a way of persisting the state of an agent to the server, so it could continue with the execution when the page is reloaded. Client-side agents are also able to communicate with both client and server-side agents.

5. Educational Experiences

In this section, we will present experiences in delivering these two courses and gained with teaching them for several years at our universities. Numerous positive experiences are gained by teachers and students as well, which proves that such courses should be a regular part of study programs in informatics and computer science.

5.1. Educational Experiences with DNAD Course

Experiences with teaching the DNAD course have been gained during the academic years between 2009–2015. During the course lectures, we followed the course topics outlined in Section 3. They included a chapter on agent middleware covering FIPA and JADE. Moreover, agent middleware examples were often used to discuss concepts of distributed systems. On the other hand, we experimented with different approaches and assignments in each year for the lab and project work in order to better adapt to the special needs of our students, so the following presentation is more focused on those practical aspects of the course.

Agent middleware was firstly added to the DNAD course in 2009/2010 academic year. The lectures included a chapter on FIPA and JADE. Also, the students were exposed to the design and implementation of simple multi-agent systems during the laboratory classes. The process started with the presentation of a scenario, followed by the identification of agent types, design of interaction protocols and agent behaviors. However, while we noticed during the exams that students received well the discipline of MAS design, they had difficulties with the implementation of the agent system. Actually, very few of them were able to produce a working JADE-based MAS at the end of the lab activity. We learnt that one weakness of our approach was the schedule of JADE introduction too late, towards the end of the course. The students needed more time to gain better familiarity with the technology in order to successfully finalize some concrete programming tasks.

Therefore, during the next 2010/2011 year, we decided to direct more lab assignments towards MAS design and implementation, as well as to understanding of the underlying technologies in connection with agent middleware. So, apart from the course lectures dedicated to FIPA and JADE, students received a lab task to implement distributed entities called "agents" that can interact using a simple "ping" protocol. For the implementation, students had to use several Java middleware technologies, including: sockets, RMI, servlets, web services and JADE. They were introduced to the MAS design methodology during the first lab task, while the rest of the laboratory work was concentrated on various implementations.

Learning from past experiences, we helped students by creating a JADE bootstrap class that instantiates the JADE platform as well as the *Remote Management Agent* – RMA that provides a default Graphical User Interface to the MAS. Then we taught students how to easily add agents to their system using sample Java code. This proved to be a better approach, since all students were able to create agent running examples more easily.

During academic year 2010/2011, we also directed the project activity towards using agent middleware. Therefore students received an "agent stress" experiment as project assignment. Their first task was to define an agent organization with a fixed communication topology of their choice (*i.e.* ring, mesh, linear *etc.*) and to run an experiment to determine how many "ping" protocols they can instantiate on a single machine before their system started to exhibit any kind of failures, like lost messages or agent crashes. Their second task was to rerun their experiment on a computer network using more machines and then compare the results. The implementation technology of the "agents" was left to their choice. Out of 28 students that presented the project in the first exam session, 25 of them decided to use JADE. However, this result is only partly positive, as 29 out of 57 students did not present the project at all.

We generally concluded that students liked working with JADE. However, as a "side-effect", we noticed that they prefer to use a simple development environment that automatically takes care of general repetitive tasks that are part of the system setup. This conclusion is independent of the technology taught, as we noticed the same problems with teaching Java servlets. Although it can be argued that doing such actions manually can be more educative as students are faced with "more realistic" problems, those activities are also tedious and discouraging, thus hindering the focusing of students' attention on more interesting problems. One negative effect is that most students just stopped doing the laboratory duties altogether, accepting lower grades. So we decided to focus them on programming and experimentation activities, rather than forcing them to do the configuration and setup activities themselves.

Doing experiments on multiple computers also captured well the students' attention, in spite of the obvious complexities of setting up a distributed application on a computer network. The JADE RMA agent was very well received by the students for monitoring the setup of their distributed system before actually starting the agents' interactions.

Most of the students needed personal assistance from the lab teacher until they were able to produce a working lab assignment. This resulted in students' asking so many questions that the professor ended up by handing out pieces of code to them. Some students finalized their lab assignments before the schedule, thus finishing their work faster. Consequently, the teacher took advantage to assign them with helping slower students that ran out of time. However, this approach turned out to be not so "competitive", as slower students actually received more attention. The negative effect was that the number of these students was increasing by the end of the semester. Nevertheless, this was compensated during the grading process.

During academic year 2011/2012, we decided to continue with the same lab assignments, while applying an "elitist" type of assistance. If a student did not present a reasonable personal attempt, then he/she would not get any help. But in order to make sure that all students could make that first attempt, we provided them with a review of Java programming in the first lab session. For diversity, we slightly modified the project assignment by adding the requirement to measure the MAS setup time for instantiating an increasing number of ping protocols on a single machine, as well as on two machines and then compare the results. As a consequence, only 15 out of 45 students were able to present their project during the first exam session. Also, only one student chose to use sockets instead of JADE agents.

For the following academic year 2012/2013, we chose to make a radical change in lab and project activities. We devised two large assignments:

- a distributed master-slave password cracker implemented using a safe socket-based communication protocol over UDP, and
- a Twitter-like system implemented using JADE agents and equipped with a Web based GUI.

We also continued with the presentation of a Java programming tutorial during the first lab. For the project task, the students were asked to perform a "stress" experiment on one of the lab assignments of their choice.

During their assignments, we identified two most frequent difficulties encountered by the students:

- dealing with the potential failures of the UDP protocol, and
- interfacing JADE with the Tomcat Web server.

They determined many students to actually refuse to even try to achieve these particular requirements of the assignments. Out of 90 students, only 21 of them were able to implement the safe communication protocol over UDP and only 17 created some kind of Web interface to their agent-based Twitter-like system. Just 25 out of 90 students presented the project in the first exam session, which was in fact an overall negative result. We concluded that the main cause was the significantly higher complexity of the lab assignments, as compared with the previous years.

During the academic year 2013/2014, we presented the students with the challenge of building a mobile app that allows users to chat peer to peer. The students were briefly introduced to agents as distributed software components. They were also given an example of how to programmatically setup a JADE framework using Java code, how to connect to it from a mobile device and how to dynamically create agents. Students were not granted credit for attending the lab, unless they successfully achieved the framework setup. We found that this was a great source of motivation for below average graded students. While most of the students attempted to do everything by themselves, they usually failed because of the lack of experience. So they were tempted to ask for help from better students, which was offered happily. This practice was beneficial to all students involved, so it was not discouraged. Therefore, all 88 attending students (out of 102 enrolled in the course) have successfully instantiated the JADE framework.

The decision to use mobile devices was based on two elements: demand from students and demand from software companies, including local ones. We made two local pilot workshops for mobile programming to get students' feedback on this decision. Based on positive students' response and the high interest of local software companies, we decided to include this technology into our lab.

Note that this positive feedback was partly due to the low number of possibly unrepresentative, but highly motivated attendees. When all students were presented with our distributed mobile challenge at the lab, a lot of them seemed overwhelmed. However, again by conditioning their attendance with providing a working demo application, most students (91/102) were able to produce a simple Android app in the first lab.

At the end, we had a total of 88 students that gathered both the know-how of creating a mobile application, as well as setting up a JADE framework. The last part of the challenge required creation of an agent from the app and then sending it a message. Out of 88 students, only 36 (40%) were able to actually finalize it. Based on students' feedback about the lab assignments and the implementation effort required, it became clear that actually only half of these 36 students would have been able to do the assignment by themselves, while the other half required assistance from the rest.

A general conclusion of 2013/2014 lab activities was that merging different technologies might intimidate students, possibly with a negative effect on the educational process since most students (the remaining 102 – 36 = 66) did not achieve the target set by the lab of developing an actual distributed application. On the positive side, we have initiated a few students in mobile computing, which we consider beneficial for the local software development industry. Nevertheless, the mobile software technology and tools evolved too rapidly, so a large part of our tutorials and lab presentations were rendered obsolete for the next years. So, we concluded that, at that time, this was not a sustainable way of conducting the lab activities.

In academic year 2014/2015, the students were challenged with developing a mobile code application. For the sole purpose of *captatio benevolentiae*, this was presented as "building a computer virus". The scenario involved two computers running "trojan code" such that one was sending the virus code, and the receiver would then run the virus on the host computer. What the virus did was unimportant for the purpose of the lab, so a simple "hello world code" was enough for demoing the application.

At this point, students were allowed to use any technology in order to achieve the desired functionality. JADE agents were presented as a possible alternative supported by mobile agents. As usual, the students were given code to setup the framework and two demo agents. At the time of this challenge, the students were also introduced to Java Sockets, remote method invocation and web services. Given the freedom to choose, 39 (42%) students of a total of 92 used JADE agents. This is an objectively high percentage given that only 72 students attended that laboratory. Counting only the physically present students who actually completed the assignment, 55% of them used JADE. We believe that this was partly due to the concise and comprehensive tutorial on agent mobility provided by JADE community. Even so, we felt that many "lazier" students preferred to apply the knowledge gathered at the beginning of the course by implementing code mobility using Java Sockets (no other way of implementation was attempted). In other words, 33 (35%) students preferred to practice acquired knowledge, rather than to learn and experiment with the new agent technology. We believe that such students were mostly interested in being hired by local companies as soon as possible, rather than spending more time and effort in mastering new skills, not immediately useful to a local employer. The group of students that used JADE mobile agents seemed to be more research oriented. We found these students to be better programmers, by rapidly adapting to new programming styles and technologies, thus leaving also enough time for experimentation, rather than only learning to code.

Based on our experiences we can speculate that agent middleware in general and JADE in particular could offer certain advantages as compared with other frameworks for teaching several aspects of distributed systems technologies and applications. JADE has a smooth learning curve and requires considerably less effort than other enterprise technologies – like Enterprise Java Beans, for example. Students enjoyed programming simple JADE-based distributed applications. Moreover, agent-based approach offered a disciplined approach for design and development of distributed applications that can be successfully transferred to other distributed software technologies.

However, there are aspects where the use of JADE presents difficulties. Students needed help with the setup and creation of simple applications. This was achieved by the creation of a JADE bootstrap class, as well as a special application configuration of Eclipse platform to facilitate the development and running of agent applications. Moreover, the implementation of a Web-based GUI for JADE-based MAS requires a tedious and discouraging work of interfacing two different distributed technologies: FIPA agents on one hand and web servers or mobile technology on the other. This fact requires further investigation before deciding to consider this as a limit of our lab or just a trigger for creating better ways of interfacing JADE other distributed software technologies.

Finally, we also noted that students did not perceive JADE as an actual enterprise technology. We have reasons to believe that this is one of the main causes of their low turn up with the project presentations in the first exam session. It is quite hard to capture the full attention of all the students when the only foreseeable result of using JADE is a basis for a prototype or concept implementation.

Before concluding the discussion on educational experiences with DNAD course at the University of Craiova, we mention that, starting with 2016, the CS curricula was updated according to new regulations and, although quite successful in our opinion, the topics of the DNAD course had to be dispersed into other courses, as follows. Firstly, the course itself was shifted one semester earlier and it was oriented more on concurrent programming including modeling and verification of concurrency, as well as on programming with threads on multicore architectures, with examples in Java. Only a small part related to the introduction of distributed systems and programming by asynchronous message passing using sockets and remote method invocation, was kept in the new course. More advanced distributed computing aspects, including distributed and parallel algorithms are now being taught as a separate course. Multi-agent systems topics were also included into an advanced course that is taught within the Master program. However, the aim of this course is more in the spirit of multi-agent, than distributed systems, including, for example, topics of game theory, artificial intelligence (Belief-Desire-Intention programming) and Machine Learning. Note also that an advanced course on Distributed Programming focused on distributed software technologies is also included in the Master program. So, we can conclude that our positive experiences with teaching DNAD topics were not lost, but rather reused by enriching other courses that are now part of the CS curricula at the University of Craiova.

5.2. Educational Experiences with AT Course

Teaching experiences with the elective AT course at the Technical Faculty, University of Novi Sad have been gained during the academic years 2014/2015, 2015/2016 and 2017/2018. During the course lectures, we followed the course content and methodology mentioned in Section 4. Moreover, like our colleagues from the University of Craiova, we experimented with different approaches and assignments, especially in the last year compared to the previous ones. Particularly for the project work, we introduced new elements in the last academic year in order to offer students better understanding of agent concepts and technology, thus allowing them to be more focused on practical aspects of the course.

AT course was introduced as elective course for the first time during the 2014/2015 academic year. Out of 12 students attending this course, 4 proceeded with the work in the field of agent technology by taking upon (and finishing on time) a bachelor thesis related to agent frameworks. The same trend continued in the 2015/2016 year: 5 out of 20 students advanced to doing a bachelor thesis related to agent frameworks. Finally, during the school year 2016/2017, 6 out of 29 students, took bachelor thesis related to agent frameworks. All students passed the final exam for this course within six months from the end of the course.

From our point of view, we applied important methodological approach *i.e.* we have adopted in our AT course the organization of practical activities as team work. Similar to the real-life tasks and assignments, usually in IT companies, students were organized in groups/teams and each team had to implement the same API given by the teachers.

That way, all the teams had to produce interoperable code, which would be compatible with the solution from any other team. Another advantage of having team work is in giving more complex assignments to the students. Both students and teachers were satisfied with the results achieved from performing the tasks within the teams, since all the solutions were produced on time and the code was interoperable. However, there is a potential drawback of having teams instead of individual assignments. In this approach, it is quite usual that not all students contribute equally to solving joint tasks and problems. That is the general problem in having teams and one solution can be to set up a fair project management able to assess roles in the team, which is in our course equally done by the teachers and students. Student polls indicate that they are reasonably satisfied with the load balancing between team members.

Finally, as AT is an elective course, a very important issue to be considered is how to try to raise the number of students and motivate them to select and attend this course. During academic year 2015/16, we had more students than in the previous year (20 of them). Furthermore, during the 2016/17 year, 29 students attended our course. This is a valuable information, as it seems that more and more students are becoming aware of the importance and necessity of application of distributed programming and agent technology in real-life domains. They have realized that such a course would give them the opportunity to improve distributed programming skills necessary for their future careers.

For example, one of the characteristic examples that students have to solve during their lab classes is the FIPA Subscribe Interaction Protocol [9] that has been realised on the Siebog platform. In this example, students need to implement several types of agents to provide message exchange and implement basic agent intelligence. Furthemore, students are required to implement this example in the distributed environment, having multiple agents deployed on multiple servers. This way, students are also trained to solve problems in the field of distributed computing.

Students develop the agent code in the Eclipse IDE, and deploy the solution into the JBoss application server. Siebog middleware has the built-in web console for the agent management. This console also dumps all the logged messages and significantly improves debugging and analysis. It also visualizes the interactive way of monitoring the Siebog operations and helps students in better understanding of the agents' execution.

After this initial phase, the clustering of application servers and realization of agents in clustered environments is presented to students. Needed technologies (JavaEE) for the realization of agent middleware are also discussed: Enterprise Java Beans – EJB, Java Naming and Directory Interface – JNDI, Java Messaging System – JMS, Message Driven Beans – MDB, JAX-RS specification for the REST services, and WebSockets. Students also widen their knowledge about reflection, class loading, security managers, and advanced JavaScript.

Last part of the course is devoted to students' building their own agent middleware using the Siebog as a model. To simplify students' activities, their task was to implement only a subset of all Siebog features and demonstrate agent functionality by implementing chosen FIPA agent communication protocol on a real-life problem.

Each team of students has to implement particular problem emphasizing the interoperability (following the same API). So, all instances must be able to communicate with each other, providing interoperability (by REST services). This means that agents developed by different teams and deployed on different platforms must communicate with each other without any additional requirements or implementation efforts.

Our experience with previous years and tasks that students have successfully completed is very positive. On the other hand, as lab tasks in previous 2 years were very similar, we would like to prevent possible students' cheating, but also to make further innovations within the course. As a consequence, during the last academic year, we introduced additional assignments for students – besides creating the agent framework, they had to perform one more task – to solve more a complex problem, preferably in the AI realm. The problems that students were solving during the last academic year are very briefly presented below.

This way, they were able to apply their knowledge of AI in the Intelligent Agents realm, thus broadening their skills.

Example 1. Fruit purchasing agents – Agents are trained to perform the image recognition of the fruits to be purchased. It is necessary to distinguish two types of agents: merchant and estimator. Merchant is sending images of the fruits to be purchased to the estimator agent. Estimator agent returns the estimated quality of the fruits, so the merchant can make a decision whether to buy this fruit or not. Estimator agent does its job by employing the neural network approach.

Example 2. Sport match result prediction – In this example, the idea is to implement agents and train them to predict the outcome of the sport matches. Generally speaking, there are multiple agents doing the same job, doing the same prediction based on particular input data. The key agent here is the master agent, which gathers the predictions from other agents and sends the aggregated results to the client. As in the previous example, concept of neural network is utilized for the sport matches results prediction.

Example 3. Weather prediction – In this example, the idea is to implement agents and train them to perform the weather prediction. Again, the master agent acts on the client request, activates a number of prediction agents and gathers their predictions. The master agent sends the aggregated data back to the client. Neural network model is used for the weather prediction.

Example 4. Chess playing – For each chess figure type, it is necessary to develop a corresponding type of agent. Also, for each concrete figure, it is necessary to activate an instance of a corresponding agent. The master agent coordinates figures (agents) and is trained to move them on the chess board according to chess rules. Master agent uses open source chess engine to make adequate moves. Human opponent is playing against agents.

All the above-mentioned projects have been proposed by the students themselves. The proposed projects were carefully considered by the professor and teaching assistant, adjusted to the content of the course and available resources and improved to be finally approved. All the proposed projects were implemented properly and defended successfully. As these projects were obligatory part of the course realization, obtained grades together with the final exam created the final grade for each student.

By the end of the course, we expect each student to have gained appropriate knowledge, acquired adequate agent programming skills, and achieved main course goals:

- Understand the concept of agents and agent middleware, the agent lifecycle, message exchange, mobility and other related concepts.
- Understand the diversity between specificities of agents and other software entities.
- Be able to design and implement distributed, agent-based solutions.
- Be able to solve problems being part of the team.
- Critically analyze the expected benefits of using agent technology.
- Apply the newly obtained knowledge of agent technology to the AI problems, making agents really intelligent.

6. Conclusion

In this paper, we presented both a comprehensive overview and experiences of delivering two courses on distributed systems and agent technology at two universities in two countries. For both courses, we presented the structure, the tools, teachers' and students' experiences and lessons learned.

Concerning the course on Distributed Network Application Development, which was delivered in the period of six years, we highlighted the role of agent middleware and multi-agent systems on teaching various theoretical and practical aspects of the distributed systems. Our conclusion is that the use of agent middleware and of JADE platform for teaching topics of distributed systems certainly brings many advantages, but also has some limits and poses a few difficulties. As future work, we plan to adapt our course curricula and methodology to address some of these issues. We also plan to expand our course curricula by adding new topics in mobile computing and cloud computing, while maintaining the significant role of agent middleware.

Another course we deeply considered in this paper is the Agent Technologies course, which has been offered for the past several years at the Faculty of Technical Sciences, UNS, Serbia, inspired by [9] and subsequently improved [2]. Students have had classes and exercises performed using Siebog in-house agent middleware, which is being actively developed at the UNS for more than 10 years.

Experience with Siebog showed that the in-house solution used for teaching such advanced courses is very welcomed by students, since they can obtain instant help regarding problems or understanding complex concepts. Practical projects for students were organized in the form of team work. Team work provided students with the possibility to develop distributed applications in the way it is usually done in IT companies, which prepares them for the real-life working situations in their future jobs.

Based on previous experiences in delivering this course, we are aware of the fact that there are still a wide range of opportunities for expansion and introduction of innovations in the course structure and organization.

As authors of the paper have been cooperating in research and educational activities for more than a decade, we managed to approach joint conclusions and advantages of delivering such kind of courses. Regardless of the fact that one course is mandatory (DNAD) and the other one is elective (AT), some common conclusions follow:

- If students are highly motivated and eager to enhance their knowledge in new, contemporary topics in ICT and CS, then they are ready for hard work and extra activities in order to successfully accomplish rather demanding tasks.
- Successful realization of courses on Distributed Systems and Agent Technology, based on high exam passing rate for motivated students, confirms that these topics should be a part of modern ICT and CS curricula.
- If it is possible to use an in-house developed software system for theoretical but predominantly for practical aspects of a course, then such software offers significant additional value.
- Organization of students' practical projects based on team work is motivational and inspiring for the majority of students. It offers them experience they will face in their future work and jobs in companies.

We believe that material, experiences and lessons learned from delivering these rather innovative courses presented in the paper could also be useful for other teachers and educational institutions.

References

[1] M. Ivanović and A. Klašnja Milićević, "Big data and computational intelligence", *International Journal of High Performance Computing and Networking* (in print).

[2] C. Bădică *et al.*, "Role of agent middleware in teaching distributed network application development", in *Proceedings of the 8th International Conference KES-AMSTA*, vol. 296, 2014, *Advances in Intelligent Systems and Computing*, pp. 267–276, 2014, Springer International Publishing. https://doi.org/10.1007/978-3-319-07650-8_27

[3] J. Kesaniemi and V. Terziyan, "Agent-environment interaction in mas - introduction and survey". In: F. Alkhateeb (Ed.), Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies, pp. 203–226, 2011, IntechOpen. doi:10.5772/15145.

[4] M. Beer *et al.*, "Multi-agent systems for education and interactive entertainment: design, use and experience", IGI Global, 2011. https://doi.org/10.4018/978-1-60960-080-8

[5] F. Stonedahl *et al.*, "MAgICS: Toward a multi-agent introduction to computer science", In. *M. Beer, M. Fasli, & D. Richards (Eds.) Multi-Agent Systems for Education and Interactive Entertainment: Design, Use and Experience*, pp. 1–25, 2011, Hershey, Information Science Reference.

[6] I. Sakellariou *et al.*, "An intelligent agents and multi-agent systems course involving NetLogo", in *Multi-Agent Systems for Education and Interactive Entertainment: Design, Use and Experience*, pp. 26–50, 2011, Hershey, Information Science Reference.

[7] I. Stamatopoulou *et al.*, "Teaching, learning and assessment of agents and robotics in a computer science curriculum", in. *Proc. of the 11th International Symposium on Intelligent Distributed Computing - IDC 2017*, *Studies in Computational Intelligence 737*, Springer, pp. 321–332, 2018, ISBN 978-3-319-66378-4C.

[8] J. Wiens and D. Monett, "Using BDI-extended NetLogo agents in undergraduate CS research and teaching", in *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS)*, pp. 396–402, 2013, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), CSREA Press U.S.A.

[9] D. Mitrović *et al.*, "The Siebog multiagent middleware". *Knowl.-Based Syst.*, vol. 103, pp. 56–59, 2016. https://doi.org/10.1016/j.knosys.2016.03.017

[10] C. Bădică *et al.*, "Software agents: languages, tools, platforms", *Computer Science and Information Systems*, vol. 8, no. 2, pp. 255–298, 2011. https://doi.org/10.2298/CSIS110214013B

[11] F. L. Bellifemine *et al.*, *Developing multi agent systems with JADE*, John Wiley & Sons Ltd, Chichester, West Sussex, England, 2007.

[12] Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM) and IEEE Computer Society. Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. ACM, New York, NY, USA. Accessed on: Oct. 15, 2016.
http://dx.doi.org/10.1145/2534860

[13] G. Coulouris *et al.*, *Distributed Systems. Concepts and Design (Fifth Edition)*, Addison Wesley, 2011.

[14] M. Wooldridge, *Introduction to MultiAgent Systems*, Second Edition, John Wiley and Sons, 2009.

[15] D. Ince, *Developing Distributed and E-Commerce Applications*, Second Edition, Addison-Wesley, 2003.

[16] J. Graba, *An Introduction to Network Programming with Java*, Springer, 2007.

[17] N. Santoro, *Design and Analysis of Distributed Algorithms*, John Wiley & Sons, 2007.

[18] J. R. Albrecht, "Bringing big systems to small schools: distributed systems for undergraduates", *SIGCSE Bull.*, vol. 41, no. 1, pp. 101–105, 2009, ACM, New York, NY, USA.
https://doi.org/10.1145/1539024.1508903

[19] C. Bădică *et al.*, "Agent reasoning on the web using web services", *Computer Science and Information Systems*, vol. 11, no. 2, pp. 697–721, 2014.
http://dx.doi.org/10.2298/CSIS140301038B

[20] C. van Aart, *Organizational Principles for Multi-Agent Architectures*, Birkhäuser Verlag, Whitestein Series in Software Agent Technologies, 2005.

[21] C. Bădică *et al.*, "Distributed agent-based online auction system", *Computing and Informatics*, vol.33, no.3, pp. 518–552, 2014.
http://www.cai.sk/ojs/index.php/cai/article/view/2216

[22] A. Mocanu *et al.*, "Ubiquitous multi-agent environmental hazard management", in *Proc. of the 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (SYNASC), pp. 513–521, 2012, IEEE Computer Society.
https://doi.org/10.1109/SYNASC.2012.72

[23] D. Mitrović *et al.*, "Extensible Java EE-based agent framework in clustered environments", LNCS, vol. 8732, pp. 202–215, Springer, 2014.
https://doi.org/10.1007/978-3-319-11584-9_14

Received: October 2018
Revised: January 2019
Accepted: January 2019

Contact addresses:

Costin Bădică
Department of Computers and Information Technology
Faculty of Automation, Computers and Electronics
University of Craiova
Romania
e-mail: cbadica@software.ucv.ro

Milan Vidaković
Faculty of Technical Sciences
University of Novi Sad
Serbia
e-mail: minja@uns.ac.rs

Sorin Ilie
Department of Computers and Information Technology
Faculty of Automation, Computers and Electronics
University of Craiova
Romania
e-mail: ilie_sorin@software.ucv.ro

Mirjana Ivanović
Faculty of Sciences
University of Novi Sad
Serbia
e-mail: mira@dmi.uns.ac.rs

Jovana Vidaković
Faculty of Sciences
University of Novi Sad
Serbia
e-mail: jovana@dmi.uns.ac.rs

Costin Bădică is a Professor at the Department of Computers and Information Technology, Faculty of Automation, Computers and Electronics, University of Craiova, Romania since 2006. He obtained his PhD degree. in Automatic Control in 1999. The research of Prof. Bădică has been the intersection of artificial intelligence, software engineering and distributed systems. He has contributed to the fields of applied formal representations and reasoning, as well as intelligent distributed multi-agent systems, platforms and programming. Prof. Bădică co-initiated the "Intelligent Distributed Computing – IDC" series of conferences, and is a Founding Member of Romanian Association of Artificial Intelligence. As guest editor, he has organized many international journal special issues and managed many national and international research projects.

Milan Vidaković was born in Novi Sad, Yugoslavia (nowadays Serbia), in 1971. He received his BSc degree in electrical engineering from the University of Novi Sad, Novi Sad, Yugoslavia, in 1995, and the MSc and PhD. degrees in Electrical Engineering from the University of Novi Sad, in 1998 and 2003, respectively. In 1995, he joined the Department for Computing and Control of the Faculty of Technical Sciences, University of Novi Sad as a teaching assistant. In 2014, he became full professor at the same university. His current research interests include agent technology, distributed computing, object-oriented programming, web programming, and internationalisation and localisation.

Mirjana Ivanović holds a position of full professor at the Faculty of Sciences, University of Novi Sad, Serbia, since 2002. She is a member of the University Council for Informatics. She has been a member of Program Committees of more than 250 international conferences and General Chair and Program Committee Chair of numerous international conferences. She has also been invited speaker at several international conferences and visiting lecturer in Australia, Thailand and China. As leader and researcher, she has participated in numerous international projects. Currently, she is the Editor-in-Chief of Computer Science and Information Systems Journal. Her research interests include multi-agent systems, e-learning, and applications of intelligent techniques (CBR, data and web mining).

Sorin Ilie has been a teaching assistant at the Department of Computers and Information Technology, University of Craiova, since 2009. He obtained his PhD degree in Computers and Information Technology in 2013. Dr. Ilie is interested in research topics along the lines of intelligent distributed applications and agent based simulations. He is an active member of the Intelligent Distributed Systems research group led by professor Costin Bădică.

Jovana Vidaković was born in Novi Sad, Serbia, in 1975. She received her BSc and MSc degrees in Computer Science from the Faculty of Sciences, University of Novi Sad, Serbia, in 1999 and 2003, respectively. She received her PhD degree from the Faculty of Technical Sciences, University of Novi Sad, Serbia, in 2015. Currently, she works as an Assistant Professor at the Faculty of Sciences, University of Novi Sad, where she lectures in several Computer Science and Informatics courses. Her research interests are related to database systems and information systems.