Scalable Computing: Practice and Experience
Volume 7, Number 1, pp. 49–58. http://www.scpe.org
ISSN 1895-1767

SOFTWARE AGENTS IN SUPPORT OF STUDENT MOBILITY

MARIA GANZHA*, WOJCIECH KURANOWSKI†, AND MARCIN PAPRZYCKI‡

Abstract. Autonomous software agents are often claimed to become a new generation of tools facilitating efficient management of information. While a number of possible agent application areas can be found in the literature, support for "academic mobility" is not one of them. At the same time student mobility is one of the important objectives within the European Union and, as we argue in this paper, software agents could be used to streamline administrative processes involved in setting up student participation and help students that are interested in it as well as administrative units that have to support it. In this paper we introduce an agent system designed to facilitate student mobility, present UML diagrams of agents of that system and discuss an initial implementation of a system-skeleton.

Key words. Multi-Agent System, Agent mobility, JADE agent environment

1. Introduction. One of the more important current goals that the European Union is striving at achieving (with only limited success) is social mobility. In this context, one of promising ways of achieving future social mobility is through various forms of "academic mobility" involving students and faculty members of EU-located institutions of higher learning visiting other such institutions. Mobility of "academicians" is supported financially through a number of Marie Curie Mobility Programs. There, programs like Socrates and Mundus are designed, among others, to allow students to visit universities in other EU countries and spend there one or two semesters, while obtaining a living stipend from the EU. Such a visit is possible when: (a) two universities have a bilateral agreement and (b) student applies to the program and is accepted (if there are more interested students than the agreed number of exchanges, wins a competition). Note that faculty members can be also a part of Socrates/Mundus agreements and therefore results presented here can be extended into support of faculty mobility, but they are outside of scope of our current interest.

Obviously, arranging a student visit involves a large number of administrative steps and further steps are also required post completion of a visit. Fulfillment of all necessary requirements is a tedious task and takes a lot of energy on the part of the student and resources on the part of the University.

As it was suggested in [8, 12, 16] autonomous software agents are one of best possible approaches to manage and deliver personalized information in large complex environment. Recently a few research projects have attempted at pursuing this suggestion. One of important projects in this area is the EU-funded—Pellucid [10, 18, 13, 14, 15]. Pellucid attempted at tackling management of experience in public organizations, particularly those aspects of experience management related to organizational mobility (e.g. movement or circulation of staff from one unit to another—within an organization). The basic metaphor for experience management is that of an intelligent assistant that looks over oneâĂŹs shoulder and answers questions one might have at a particular point of work. Such an assistant detects that an employee is working in a particular context, offers knowledge resources that facilitate her work. To this end, the Pellucid platform integrates technologies such as autonomous cooperating agents, organizational memory, workflow and process modeling, and metadata for accessing document repositories [11]. In the context of our work, ubiquitous intelligent access to document repositories and document flow modeling are of particular interest. Results obtained within the Pellucid project were somewhat similar to research on utilizing software agents in document flow reported in [1, 2]. Finally, in [9] a schema of an architecture of the X-DoC WFMS project, which involves conceptualization and implementation of a workflow management system in Graduate Admission Process, was presented.

Following these suggestions we have decided to develop an agent system that would facilitate and support a different aspect of "student management"— SOCRATES-type mobility program(s). Results presented here are an extension of work reported in [3, 4].

We proceed as follows. In the next section we summarize steps that have to be undertaken by a student who would like to participate in a mobility program. We follow with the description of the design of an agent system and details of its implementation and, in Section 4, discuss the performance of the system. We complete paper with a brief description of our future research directions.

*Department of Administration, Elbląg University of Humanities and Economy, ul. Lotnicza 2, 82-300 Elbląg, Poland (ganzha@euh-e.edu.pl),
†Wirtualna Polska, Software Development Department, ul. Traugutta 115C, 80-226 Gdańsk (wkuranowski@wp-sa.pl),
‡Computer Science Institute, SWPS, ul. Chodakowska 19/31, 03-815 Warszawa, Poland (marcin.paprzycki@swps.edu.pl).

2. Student mobility—what has to happen? Let us consider two EU-based institutions of higher learning that are to be involved in a Socrates-type student exchange program. While there exists a number of possible names for such institutions (e.g. college, academia, university etc.), hereafter we will use a name *university* to simplify the description. The first thing that is happen is that two, or more, universities have to sign a bilateral agreement and report it to the "central-Socrates-agency" (in Brussels). It is only after this agreement is officially registered with such an EU-agency when students can be accepted into the program. Since the agreements are typically signed by International Offices of each university, they are in some ways outside of bound of our system (for more details see below).

Administrative steps that lead to student participation in the program involve a number of administrative units within both universities. Nowadays, even in countries like Poland or Romania, we can observe fast increasing role of electronically stored and processed data within universities (e.g. student records). Furthermore some universities already provide an interface that allows students to check items like: course-schedule, upcoming exams, earned credits etc. Finally, almost all students and most faculty members and administrators communicate using e-mail (to a lesser or greater extent). Thus there exist basis for developing system like the one outlined here. Let us now conceptualize situation when a student from an EU-located university wishes to participate in a Socrates-type student exchange program. We assume here that her home university has already a number of bilateral Socrates-agreements signed and registered with the central agency. In this case the following steps have to be completed (see also Figure 2.1):

- before departure
  1. Selecting foreign university
  2. Applying to the program
  3. Being accepted to a particular exchange
  4. Delivering all necessary data appropriate administrative units both at the local and the foreign university
  5. Organizing a place to live at the foreign site
- after arrival at foreign university:
  6. Contacting appropriate department at the host university
  7. Arranging the schedule of courses to be taken
  8. Managing courses and credits required to meet the requirements of the exchange program
- after returning to the home-university:
  9. Completing a survey or delivering a report to the home-site coordinator.

In current practice, the first four steps involve mostly interactions between the student and the Dean's Office at her local university, as well as an information exchange with the local exchange program coordinator. Let us note here, that the situation when multiple students are interested in a limited number of openings within an exchange program is handled in (3) "being accepted to a particular exchange". There a "competition" takes place and an appropriate number of students are selected. What is particularly interesting from our perspective is answer to the question, what happens to these students who did not qualify to a given exchange. As it becomes clear below, our proposed system allows such students, in a very natural way, becoming involved in subsequent "competitions" (if any available exchanges remain unfilled). Step (5) is often completed "automatically" by an office at the host institution that receives information about incoming exchange students as a part of the document circulation involved in steps (1)–(4). Otherwise, student has to search a flat or to communicate with a separate organization which supervises dormitories/apartment rental. After arriving at the chosen university student has to contact the host department to arrange the course schedule in such a way to fulfill the requirements of the program (e. g. to accumulate a required number of credits, to study subject areas that were covered by the bilateral agreement etc.).

In all universities, appropriately prepared to handle exchange students, steps (1)–(4), (6)–(8) or (9) don't present problems when considered independently (even if they are not supported by electronic means of communication and thus unnecessarily tedious). Problems materialize when all steps have to be completed "together" and thus, when various documents have to circulate between different units within university; between different units in different (foreign) universities and, finally, between these units (both local and foreign) and the student. Moreover, since not every university supports electronic data management to the same extent (and some universities in countries like Bulgaria, have only a very minimal IT support in administration), it is often the case that an extremely large number of documents have to be transferred "manually". This involves sending letters, faxes, receipts (in case of organizing a flat) and/or numerous telephone calls. In it particularly in this

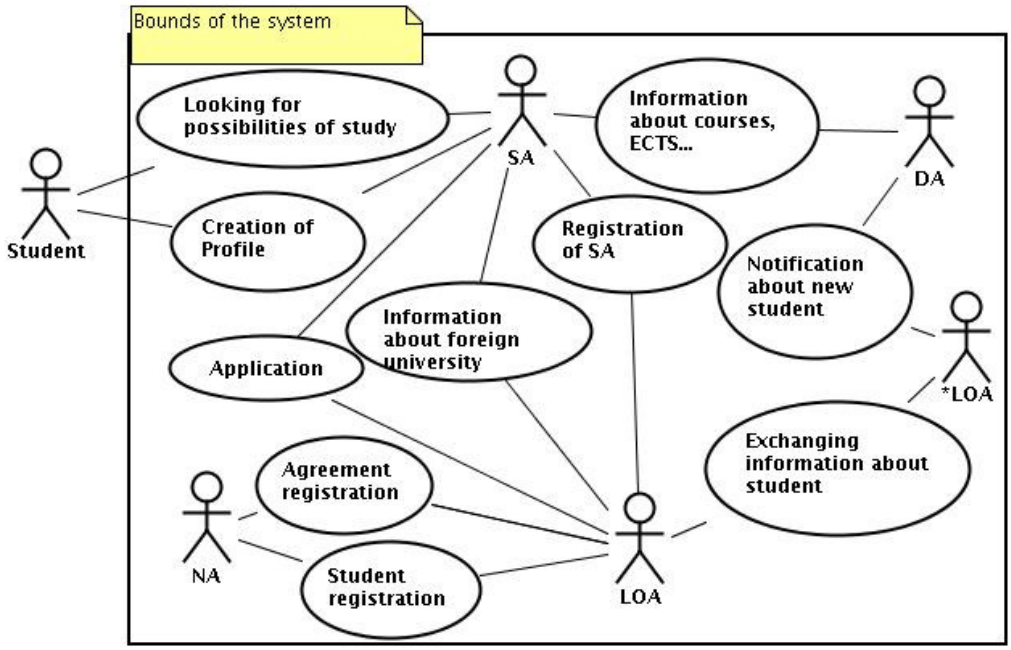

FIG. 2.1. *System Summary; interactions between agents*

regard that the proposed system, described in the next section, is expected to be particularly helpful.

3. Student mobility — proposed agent system. The main idea of our project is to develop a solution which would make formalities of taking part in a student exchange program simpler, and also reduce number of issues that presently have to be dealt with "face-to-face." We propose a system that would facilitate semi-automatic (and possibly even automatic) decision making and enable fully automatic flow of information required to establish participation in an exchange program. Furthermore, as the system develops, it could completely remove humans from the process (other than the student expressing a desire to participate in it). Let us start from summarizing (in Figure 2.1) the proposed flow of activities. Here, we have divided the functionalities into the following agents:

Student Agent (*SA*) is an interface between the student and the system and is also students' "representative." Following the line of reasoning presented in [3, 4, 10] it is assumed that in the university of the future the *SA* will be able to organize or provide view of students' schedule, check the total number of credits acquired thus far, make an appointment with a professor and/or advisor etc. In this way the *SA* is a limited case of well-known (in agent literature) paradigmatic concept of a "personal agent" [12]. In our current system, the *SA*, plays an even more limited role and represents the student only in organizing his participation in the student-exchange program. After the student is accepted and arrives at the foreign university, the *SA* communicates with the *Department Agent* at the host university and supplies the exchange student with all required information. Among others, it helps student to arrange his course schedule. While in Figure 2.1 we can see the top level view of all interactions with other agents that the *SA* is involved in, in Figure 3.1 we present the complete UML state diagram of this agent. The MCDM stands for Multicriterial Decision Making (the same demarcation is used also in the case of the Local Office Agent) and denotes the fact that in a full-blown, mature system implementation this step of agent operation involves an optimization procedure that leads to a decision. In the case of the *SA* the decision where to study could involve a very large number of criteria such as, geographical location (e. g. student wants to go where climate is warm), particular country (e.g. student does not want to go to France), program of study (e. g. student is interested in e-commerce and not in theoretical foundations of computer science) etc. Note that the blue (grey) box Studying involves a large number of steps (the same notation is used across the paper). Inside of the Studying box, one more MCDM is enclosed. This one involves selection of class schedule. Here, among others, decisions balancing interest in subject with willingness to wake up at 7 AM could be made.

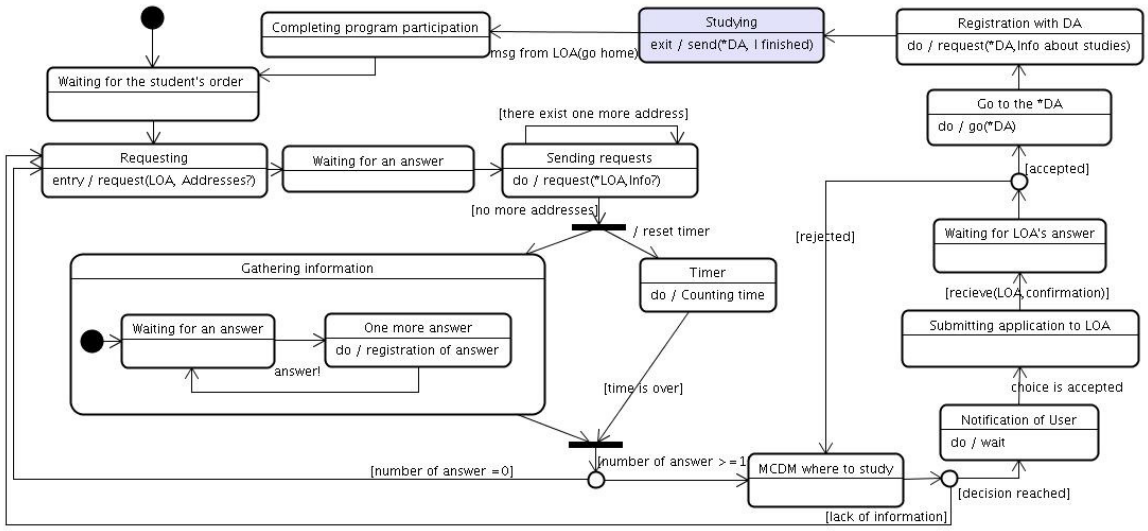

FIG. 3.1. *Student Agent State diagram*

In our system, the *Local Office Agent* (*LOA*) acts as a coordinator of the Socrates program (and even its diagram shows this by indicating that in most part the *LOA* "services" received messages). *LOA* stores information about universities that currently have bilateral agreements with its university. This list is constructed on the basis of messages obtained from the Notification Agency agent. Here we have to recall that signing bilateral agreements is the domain of International Offices. The way our system works, these offices have to notify the Notification Agency agent first and that agent has to notify the *LOA* that it is ready to accept students within the purview of a given exchange (such a notification contains also all necessary information, including appropriate deadlines). Otherwise it would be possible that the *LOA* would accept students to the program that the Notification Agency would not yet be ready to service. The *LOA* exchanges appropriate messages required to set up departure of a student to another university and handles student returning back home form an exchange. In Figure 3.2 we present the complete UML state diagram of this agent. Note the Considering Applications, box (appearing within that Figure). In the proposed system Student applications are accepted until a certain deadline. When the deadline passes, they are pre-processed first to eliminate students who do not satisfy initial selection criteria (e.g. at a given University students who have not completed successfully previous semesters may not be allowed to participate in the exchange program). The remaining applications are considered using an MCDM, the details of which are likely to be institution dependent (e. g. at a given University, the Grade Point Average (GPA) in the core courses may be more important than the overall GPA).

Notification Agency (*NA*) represents offices ("in Brussels") that supervise the student exchange program (including the financial matters). In our system, the *NA* has two functions: (1) the above described bilateral agreement management; each such agreement has to be registered with the *NA* that in turn notifies the *LOA* and the $*LOA$ that it is ready to service it, and (2) student participation management. Specifically, the *NA* has to be notified that a given student is to participate in an exchange program. In response the *NA* validates the proposal to assure that it adheres to the rules of the program (also to check if the limits of participation in a given program have not been somehow breached). When a given proposal has been positively validated (1) one of the spots available in the negotiated bilateral agreement is taken and (2) a given student will be funded by the Socrates scholarship. In Figure 3.3 we present a complete UML diagram of the *NA* agent.

Department Agents (*DA*) may be conceptualized as a virtual combination of a department head and secretary. One such agent is created for each individual departments of each university. These agents are envisioned to be responsible for courses offered during a given semester, course schedules, and calculation of ECTS, etc. Since most of functionality of this agent is related to the functioning of the university rather than to our system and falls mostly beyond the scope of our work, we have decided to omit its detailed UML-based conceptualization.

Finally, the $*$*Local Office Agent* ($*LOA$) is the *LOA* counterpart at the foreign host institution. In other words, the $*LOA$ is the *LOA* of the foreign university.

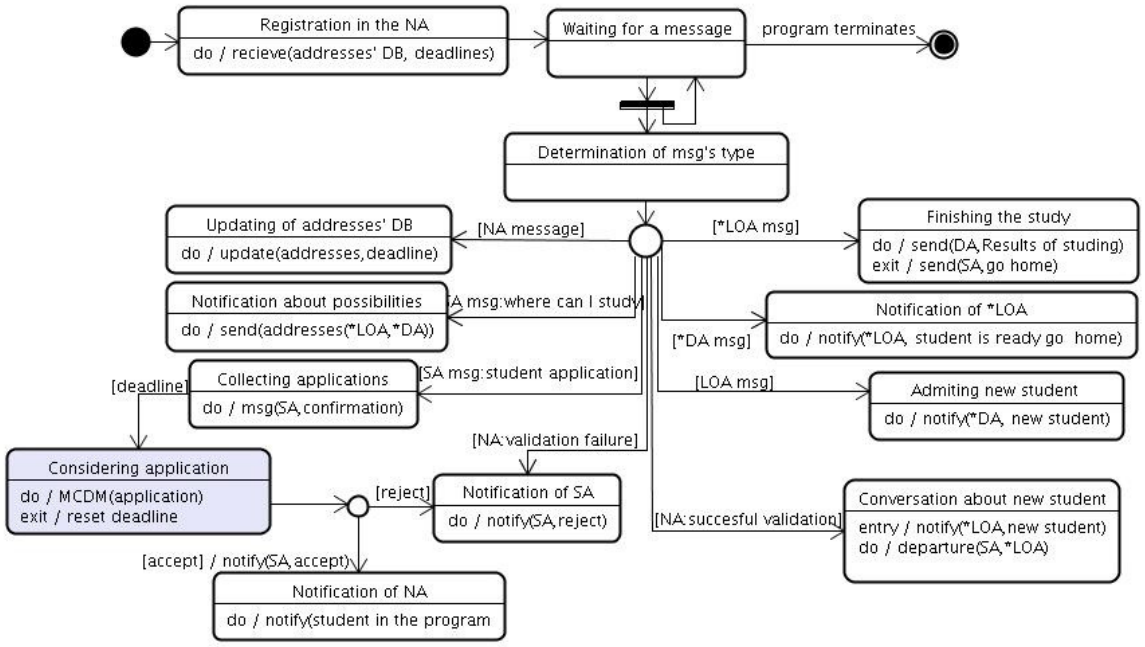

Fig. 3.2. *Local Office Agent State diagram*

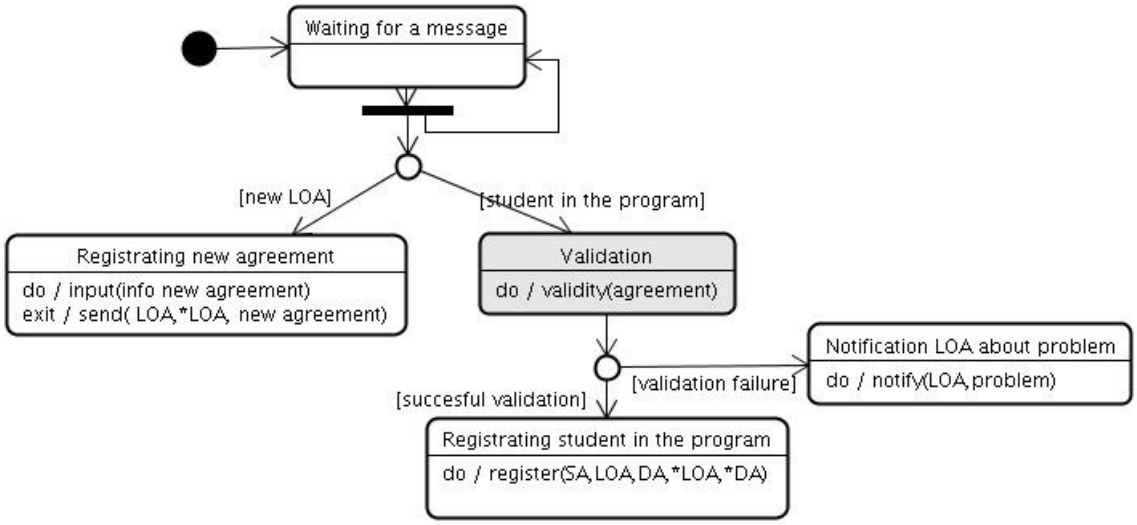

Fig. 3.3. *Notification Agency state diagram*

3.1. Agent Interactions. Let us now list interactions between agents that take place when the *SA* attempts at arranging the exchange program for the student (see Figure 2.1). We assume that the system has been initialized, that the *NA* has send the list of confirmed bilateral agreements to the *LOA´s* residing in the system etc. In other words, the system is ready to service students. In this stage, student has communicated with her *SA* and established the selection criteria (e.g. country, subject area, etc.). Then, the system performs the following actions (working autonomously — as we assume that when student specifies requirements, agents make all decisions). Note that communication between agents is achieved through exchange of standard ACL messages.

1. *SA* sends search request to the *LOA* to get addresses of all foreign universities that her *LOA* has bilateral agreements signed with (in the specified field of study)

2. upon reception of the address list, the *SA* sends messages to all of them, requesting information about local requirements/arrangements/possibilities
3. foreign *DAs* (*∗DAs*) reply providing requested details
4. *SA* performs multicriterial optimization (MCDM) and selects one or more of available universities as the place where the student should go for the exchange
5. *SA* informs the student about possibilities and suggests which one to choose (to be able to run the system automatically we have removed this step and replaced it with a fully automatic selection process)
6. *SA* sends to the *LOA* an application to the selected university and upon receiving confirmation that the application has been received suspends itself until a decision is reached (the *LOA* is assumed to process applications in batches after certain deadlines)
7. *LOA* informs the *SA* if student qualified for the exchange—if student did not qualify, the *SA* goes back to 5-above and the process repeats
8. *LOA* informs the NA that a given student was selected to participate in a given student exchange and waits for confirmation
9. when the *NA* validates the request it confirms it by sending message back to the *LOA*
10. *LOA* sends all of the necessary documents for the student to become a part of the exchange program to the (host) *∗LOA* and obtains confirmation
11. *∗LOA* registers an incoming exchange student (her/his *SA* is also registered with the local system)
12. *SA* moves to the foreign host
13. *SA* contacts appropriate *∗DA*
14. *∗DA* informs the *SA* about courses available
15. *SA* performs multicriterial optimization and on the basis of knowledge of student preferences and selects courses that match them
16. *SA* informs the *∗DA* that student completed scheduled courses (currently, to test the system, we have implemented a simple automatic selection, but a realistic system should involve student in the decision-making process; both possibilities are covered by the Studying box in Figure 3.1)
17. *∗DA* informs the *SA* and the *∗LOA* how many ECTS student accumulated
18. *∗LOA* "allows" the *SA* to go home
19. *SA* moves to its home container
20. *∗LOA* informs *LOA* about results of student exchange program participation (grades, ETCS, etc.)

Obviously, at this stage of the project the multicriteria decision making processes, mentioned above in points (4) and (14), have been replaced with a set of very simplistic selection procedures. However, delving into decision making was not of our current interest and is definitely outside of the scope of this paper. What we were interested was to develop the system skeleton and illustrate experimentally that it works. To show that agents communicate accordingly to the specification and that agent mobility is appropriately utilized to work in unison with proposed student mobility. As illustrated in the next section, we have fully achieved this goal.

4. System implementation and operation. The proposed system has been implemented in JADE 3.3 [7]. In a JADE based agent system, all agents exist within a *platform* that can be spread among multiple computers. Within a platform, agents reside in and move between *containers*. In our experimental setup, every container represents one university. We have inserted *LOA*s and *DA*s into each container (recall that a *LOA* can play a role of a *∗LOA* depending on the direction of the proposed student exchange). Additionally an *NA* is created in the Main-container (the Main-container is the name used by JADE for the "system" container that is created when JADE platform is started for the first time). After the system is initialized in this way we can create as many *SA*s as we need.

A "single" system run involves an *SA* performing all necessary steps to organize the exchange program for its student-master. As noted, in our current implementation we use very simple selection criteria, i. e. the place where the exchange program was to take place was selected on the basis of only two student preferences: field of study and number of ECTS credits she gathered thus far. An example of a system run is represented in Figures 4.1-4.3 (here the, JADE provided, Sniffer Agent which "records" all messages incoming to and originating from agents, it was told to "sniff," was used to indicate the operation of the system).

In the experiment we observe a sample scenario involving five universities (located at five separate computers): UNIV1, UNIV2, UNIV3, UNIV4, UNIV5. At the UNIV1, *DA*s representing IT and Biology departments have been created. Similarly, at the UNIV2 we see departments of IT and Chemistry, at the UNIV3 depart-

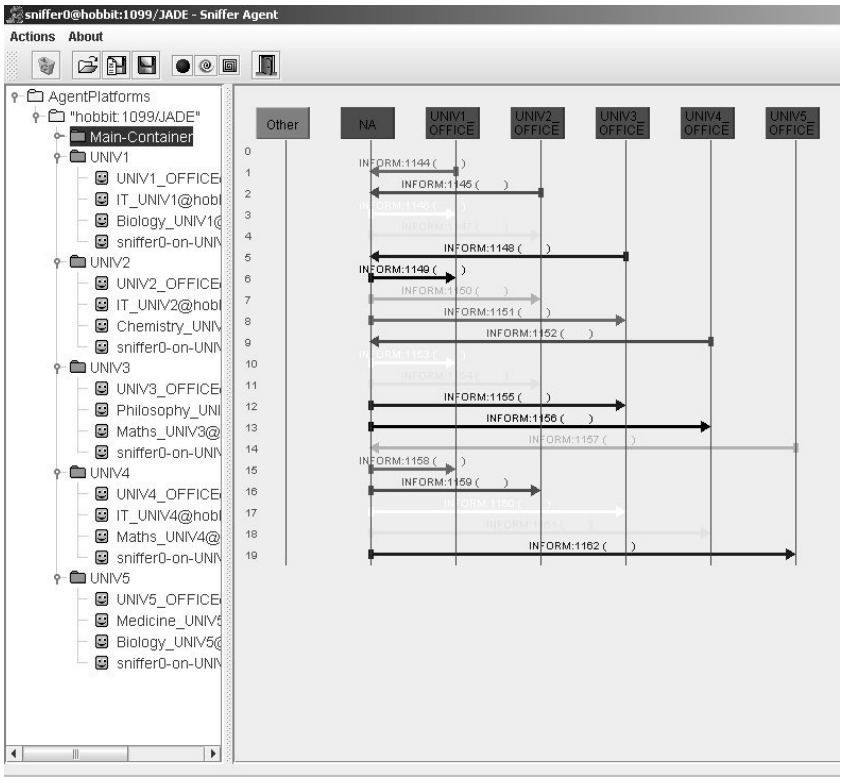

FIG. 4.1. *Sniffer Agent report for the Initial Part of Experiment 1*

ments of Philosophy and Mathematics, at the UNIV4 departments of IT and Mathematics, while at the UNIV5 departments Medicine and Biology. In Figure 4.1 we see the Initial Part of the experiment, where the *LOA* agents register with the NA. Furthermore, an *SA* was created within container representing the UNIV1 university. This agent registers with its *LOA* and later requests addresses of available exchange programs that are of possible interest to its student-master. This process is depicted in Figure 4.2

Finally, in Figure 4.3 we observe the moment when the *SA* arrives at the UNIV4 university. The main point of this scenario is for an IT student at the UNIV1 to arrange (and complete) an exchange with the IT department at the UNIV4 and this mission is accomplished.

In a separate experiment, using the psexec scripting program [17] we have created 22 containers representing 22 universities (located in 22 countries), on 20 separate, networked computers. We have then placed "random" departments on each one of them and successfully run experiments with "students" (*SA*s) seeking exchange programs among all of these university departments (computers). A sample screen representing this experiment is presented in Figure 4.4. Finally we have experimented with a "mixed environment." For instance we have run the Main-container on a Linux-based laptop, while the remaining computers have been running Windows. We have observed no problems in any of trial runs. More details of these experiments (involving an earlier, somewhat less sophisticated version of the system) can be found in [3, 4].

5. Concluding remarks. Our project, in its current stage, illustrates the most important (from the point of view of agent system design and implementation) features of system that would enable student mobility automation. Those are: mobility, communication, registration, searching etc. Furthermore, the system skeleton has been implemented and shown experimentally to work (even though, we have to admit, utilizing an extremely simplified sets of rules for decision making, selection etc.). We were able to run experiments on a single network, utilizing up to 20 computers, including mixed Linux-Windows setup and found no problems. One of the important issues that have to be considered when constructing agent systems is that each such a system has to reflect the real world. Our example shows potential of software agents to automate an existing real-world scenario. In the next steps of the development of this system, we will attempt at making it to resemble the reality even more, by focusing on developing and implementing the following features:

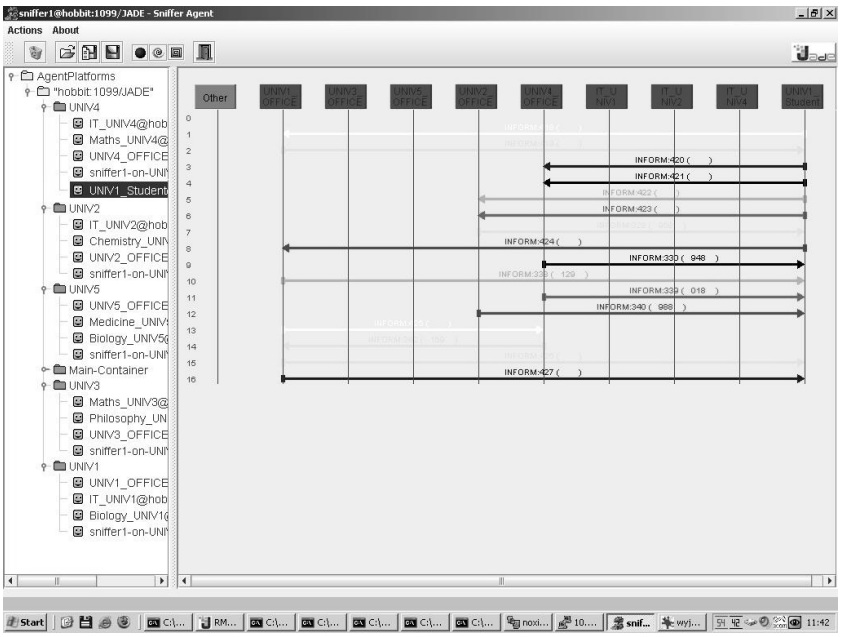

FIG. 4.2. *Sniffer Agent report of* SA *creation (Experiment 1)*

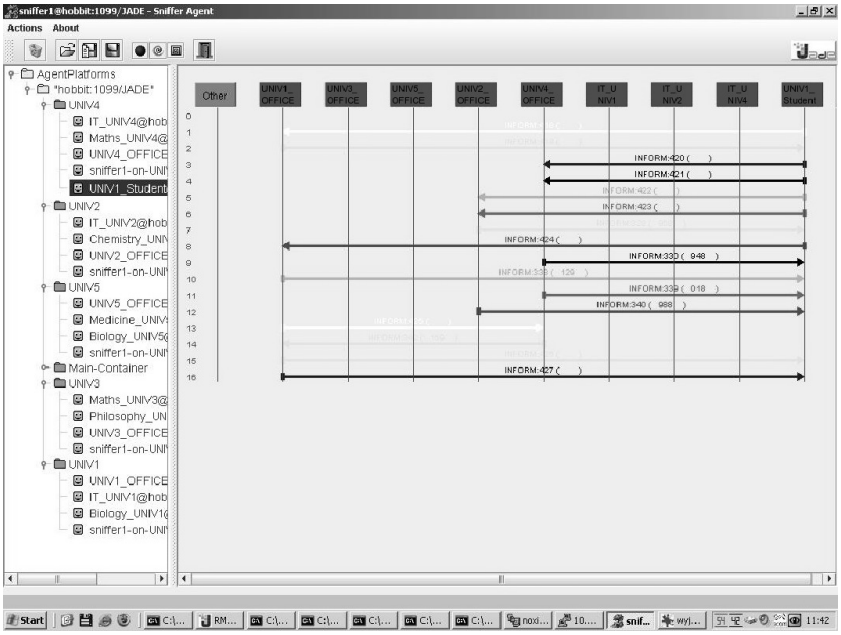

FIG. 4.3. *Sniffer Agent report* SA *arrives at the UNIV4 university (Experiment 1)*

1. Student Agent personalization (agent that actually knows what its student-master really "wants" and is able to truly represent her interests). In this context, we will have to find a way to represent user profile and this representation will have to be tied to the ontologies of "world of academia" that will have to be developed (see 4. below). A proposal how to tie ontologies and user profiles has been recently put forward in [5, 6].
2. Adding functions to the Department Agent that would extend the communication between the *DA* and the *SA* and facilitate possibility of developing the MCDM module that is to select the student-optimal course schedule.

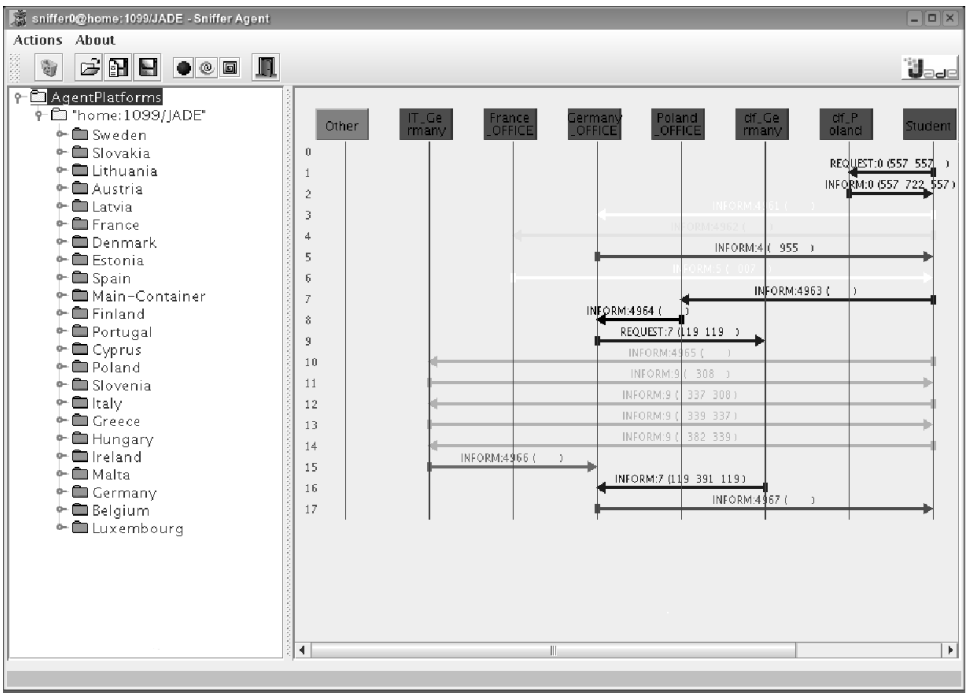

FIG. 4.4. *System run representing 22 countries (1 university per country); partial report form the sniffer agent*

3. Adding more intelligent decision making components (MCDM modules), so that selections are based on a realistically selected set of criteria. We do not assume that our goal has to be to develop fully-functional modules, but rather establish which technology should be used to seamlessly integrate it into the system under development.
4. Making communication between agents more realistic by developing and/or utilizing existing ontologies and negotiation protocols concerning various aspects of "academic life".
5. Moreover, performing international tests (computers located in different countries) is compulsory as what we want to achieve is globally working system.

We will be reporting on our progress in the near future.

REFERENCES

[1] D. HANDL, H.-J. HOFFMANN, *Workflow agents in the document-centred communication in MALL2000 systems*, http://www.aois.org/99/handl.html

[2] A. DOGAC, Y. TAMBAG, A. TUMER, M. EZBIDERLI, N. TATBUL, N. HAMALI, C. ICDEM, AND C. BEERI, *A Workflow System through Cooperating Agents for Control and Document Flow over the Internet* In: Cooperative Information Systems, 7th International Conference, CoopIS 2000

[3] M. GANZHA, W. KURANOWSKI, M. PAPRZYCKI,*Socrates Agents — in Support o Student Mobility* In: Proceedings of the 17th Mountain Conference of the Polish Information Society, Szczyrk, Poland (to appear)

[4] M. GANZHA, W. KURANOWSKI, M. PAPRZYCKI, S. RAHIMI, M. SZYMCZAK, *Designing an Agent-based Student Mobility Support System* In: In: W. Essaidi, N. Raissouni (ed.) Information and Communication Technologies International Symposium Proceedings, Al Khalij Al Arabi, Tetuan, Morocco, 2005, 44-50

[5] M. GAWINECKI, Z. VETULANI, M. GORDON, M. PAPRZYCKI, *Representing Users in a Travel Support System*, in: Proceedings of the ISDA 2005 Conference (to appear)

[6] M. GORDON, M. PAPRZYCKI, *Designing Agent Based Travel Support System*, in: Proceedings of the ISPDC 2005 Conference, IEEE Computer Society Press, Los Alamitos, CA, 2005, 207-214

[7] *JADE. Java Agent Development Framework* See: `http://jade.cselt.it`

[8] N. R. JENNINGS, *An agent-based approach for building complex software systems*, Communications of the ACM, 44 (4), 2001, 35-41

[9] R. KRISHNAN , L. MUNAGA, K. KARLAPALEM, *XDoC-WFMS: A Framework for Document Centric Workflow Management System*, In: Data and Semantics of Web Information Systems Workshop (DASWIS) Japan 2001

[10] S. LAMBERT, ET. AL., *Knowledge Management for Organisationally Mobile Public Employees*, in: Wimmer M. A. (Ed.), KMGov 2003, LNAI 2645, 2003, 203-212

[11] S. Lambert, et. al., *A Framework for Experience Management in e-Government*, in: Proceedings of the 4th Eropean Conference on e-Goverment, 2004, pp. 451-460
[12] P. Maes, *Agents that Reduce Work and Information Overload*, Communications of the ACM, 37(7), 1994, 31-40
[13] Nguyen T.G., Hluchy L., Laclavik M., Balogh Z., Budinska I., Dang T.T. *Pellucid - Platform for Organisational Public Emploees.* International Conference on Emerging Telecommunication Technologies and Application - ICETA'2004, pp. , IEEE Computer Society, ISBN . September 2004, Kosice, Slovakia.
[14] Dang T.T, Hluchy L., Nguyen T.G., Budinska I., Laclavik M., Balogh Z *Knowledge Management and Data Classification in Pellucid* In: Intelligent Information Systems - IIS 2003: New Trends in Intelligent Information Processing and Web Mining, pp. 563-568, Springer-Verlag, Series: Advances in Soft Computing, ISBN 3-540-00843-8, June, 2003, Zakopane, Poland
[15] M. Laclavik, Z. Balogh, L. Hluchy, G. T. Nguyen, I. Budinska and T. T. Dang*Pellucid Agent Architecture for Administration Based Processes*In: International Conference on Intelligent Agents, Web Technology and Internet Commerce - IAWTIC 2003, in CDs. February 2003, Vienna, Austria.
[16] H. Nwana, D. Ndumu, *A perspective on software agents research*, The Knowledge Engineering Review, 14 (2), 1999, 1-18.
[17] *PS Tools* See: `http://www.sysinternals.com`
[18] Słota, R., Krawczyk, K., Dziewierz, M., Kitowski, J., Lambert, S., *Agent paradigm for accessing document repositories in Pellucid platform*, EuroWeb 2002 conference : the Web and the GRID: from e-science to e-business : Oxford 17–18 December 2002 / eds. Brian Matthews, Bob Hopgood, Michael Wilson. — Swindon : The British Computer Society, 2002. pp. 192–194.

Edited by: Shahram Rahimi, Raheel Ahmad
Received: October 11, 2005
Accepted: March 19, 2006