ANEMONE - A Network of Multi-Agent Platforms for Academic Communities

G. Armano, P. Baroni, G. Cerchi, M. Colombetti, A. Gerevini, M. Mari, A. Poggi,
C. Santoro, E. Tramontana, M. Verdicchio

***Abstract*—This paper presents ANEMONE, a multi-agent platforms network that provides services for the academic community implemented by using the JADE agent development framework. In particular, ANEMONE provides a set of services to support i) academic people in some of their recurrent activities (fix an appointment, organize a meeting and search documents on the Web, ii) students in getting information about courses and iii) information technology people (including students) in getting information on documents and people that may help them to solve their programming problems. Moreover, it also provides a set of system-oriented services for the management of agent platforms and services and for the realization of new types of service.**

***Index Terms*—Multi-agent systems, cooperative systems, user-oriented services**

I. Introduction

One of the main reasons to use autonomous software agents is their ability to interact to show useful social behaviors rapidly adapting to changing environmental conditions. But the most interesting applications require that large and open societies of agents are in place, where collaborating and competing peers are able to interact effectively. In a context where a number of possible partners or competitors can appear and disappear, agents can highlight their ability to adapt to evolving social conditions, building and maintaining their networks of trust relations within a global environment.

The first effort to create such a large and open society of autonomous software agents was Agentcities [1]. This project developed a network of agent platforms spanning over the whole globe and a number of complex agent-based applications were deployed on the network. OpenNet is an evolution of Agentcities whose goal is to integrate agent platforms with Web Services / Semantic Web platforms [2]. In this paper, we present ANEMONE, an agent platform network developed inside the OpenNet initiative.

II. Anemone

ANEMONE is a multi-agent platforms network that provides services for the academic community (professors, researchers and students) implemented by using the JADE agent development framework [3].

ANEMONE offers both system-oriented and user oriented services. System oriented services allow the management of the network and the realization of new user-oriented services through their extension and composition. User-oriented services have the goal of helping academic users and can be used through a simple Web browser.

III. System-Oriented Services

System-oriented services have the goal to provide support to the management of agent platforms and services and provide reusable components to realize new types of service.

A. Platforms and Services Management

Platform and service management services are based on the services provided by the JADE agent development software and a set of services to register and search platforms (Agent/Service Platform Directory Services), agents (Agent Directories) and services (Service Directories) in an open network of agent platforms. Using these services, it is possible to connect a new platform to the openNet network, making it visible to others, and deploy own naming, directory and monitoring services.

Moreover, a set of monitoring services (Agent/Service Platform Monitoring Services) allow to monitor the platform's status and its ability to communicate with others.

Manuscript received November 3, 2005. This work is partially supported by the "Ministero dell'Istruzione, dell'Università e della Ricerca" through the COFIN project ANEMONE.

M. Mari and A. Poggi are with the Dipartimento di Ingegneria dell'Informazione, University of Parma, Italy (e-mail: mari@ce.unipr.it, poggi@ce.unipr.it).

P. Baroni is with the Dipartimento di Elettronica per l'Automazione, University of Brescia, Italy (e-mail: baroni@ing.unibs.it).

G. Armano and G. Cerchi are with the Dipartimento di Ingegneria Elettrica ed Elettronica, University of Cagliari, Italy (e-mail: armano@diee.unica.it).

C. Santoro is with the Dipartimento di Ingegneria Informatica e delle Telecomunicazioni, University of Catania, Italy (e-mail: csanto@diit.unict.it).

E. Tramontana is with the Dipartimento di Matematica e Informatica, University of Catania, Italy (e-mail: tramontana@dmi.unict.it).

M. Colombetti and M. Verdicchio are with the Dipartimento di Elettronica e Informazione, Politecnico of Milano, Italy (e-mail: colombet@elet.polimi.it, verdicch@elet.polimi.it).

B. Agent Interaction

As we introduced above, the platforms in the ANEMONE network are developed by using JADE. Agent interaction in JADE systems is based on message exchange. Thus, the adopted agent communication language has a crucial role. JADE agents' messages follow the standard proposed by the Foundation for Intelligent and Physical Agents (FIPA [10]), which is the most complete proposal to date.

Still, the FIPA semantics shows some shortcomings that inevitably affect the systems whose communication is based on such standard. Our aim is to provide a different semantics to tackle these problems. The FIPA proposal and ours share the assumption that agent communication should be dealt in terms of communicative acts, a special action type aiming at allowing information interchange. We part from FIPA guidelines when it comes to defining the semantics of such acts. The FIPA semantics exploits the Belief, Desire, Intention (BDI, [5]) model, which views communicative acts as events that change agents' mental states. Instead of analyzing changes in the state of the internal architecture of agents, our approach focuses on the external social state holding among agents. We describe communicative acts as actions performed by agent to change their commitments towards the others. We rewrote the FIPA Communicative Act Library according to our perspective to have a benchmark for the two approaches. The advantages of dealing with social states rather than mental ones have surfaced in the analysis of the Contract Net protocol, in which an agent, in a need for a specific service, issues a call for proposal to other agents. To check whether the contract has been fulfilled, the current FIPA protocol prescribes an inform message from the service provider itself. This procedure is effective only under very strict assumptions about the sincerity of the agents, which we cannot afford when we deal with open multi-agent systems. In a commitment-based approach, on the contrary, each message exchange of the Contract Net protocol leads to changes in the social dimension of the multi-agent system, in that, commitments are proposed, and such proposals are accepted or rejected. Commitments are public and reflect an objective state of affairs between agents. They can be stored for further reference and thus they offer an effective way to check whether agents have fulfilled their commitments. The FIPA communicative act library in terms of commitments and the results of the analysis of the Contract Net protocol are formally presented in [6].

We provide the ANEMONE network and, more generally, every JADE-based multi-agent system such commitment-based communication system in the form of an agent that is called Notary. The Notary is responsible for examining the content of the messages that are exchanged over the system and creating public structured data items reporting the relevant commitments between agents. The Notary makes use of witness agents the communicating agents have agreed upon to check whether the commitments have been fulfilled or violated. The witnesses reply to queries by the Notary, and thus increase its knowledge base. The Notary exploits a JESS (Java Expert System Shell, [7]) inference engine to reason about its own knowledge base and verify whether a commitment has been fulfilled or not. To support the commitment generation and manipulation processes, the exchanged messages need to be carrying more information than as prescribed by the usual FIPA standard. To maximize backward compatibility, we have chosen not to add fields to the original FIPA message structure, but to enrich and standardize its content field by means of XML. The Notary is provided with XML parsing capabilities thanks to a SAX (Simple API for XML, [8]) module.

The Notary-enhanced communication system introduces significant overhead in the message interchange process, which may not suit the needs for lightweight application in such environments like PDAs or mobile phones. This service is offered as an option when agents need a trusted third-party to guarantee for their communication process, e.g. in electronic auctions or business transactions. Agents only need to put the Notary among the messages' addressees to obtain its service.

C. Automated Reasoning

Domain-independent automated reasoning services, based on stand-alone software tools previously developed in the context of other research activities, are made available to the community by a wrapper agent, which is in charge of receiving requests from other agents specifying reasoning tasks to be carried out, of exploiting the suitable software system to produce the relevant solutions, and of returning them to the requestor agents.

The wrapper agent and the related agents devoted to registration and brokering of the available reasoning services are implemented according to the FIPA specification "Agent software integration" [10].

Two reasoning services are currently being integrated into the ANEMONE network, namely an argumentation system and a planning system.

Argumentation theory is a framework for practical and uncertain reasoning, where arguments supporting conclusions are progressively constructed in order to identify the set of conclusions that should be considered justified according to the current state of available knowledge. The use of argumentation has been advocated both at the level of interaction among agents to support dialogue and negotiation and at the level of an agent's internal reasoning (see [14] for a survey).

Since the construction of arguments proceeds by exploiting incomplete and uncertain information, conflicts between them may arise: the conflict relations between arguments are formally represented by a structure called defeat graph. The core problem is then to compute the "defeat status" of the arguments, namely to determine which arguments emerge undefeated from the conflict: several semantics have been proposed to this purpose in the literature.

A reasoning task in this case consists in the specification of a defeat graph and the solution provided is the defeat status assignment for the arguments included in the graph. The solution may be produced according to the well-known grounded [13] semantics or to the recently introduced CF2 [9] semantics.

As to the planning system, this reasoning service receives requests to solve plan generation problems specified using the recent standard PDDL2.2 language [12], and computes plans solving such problems (assuming they are solvable and not too hard for the integrated planner). PDDL2.2 is an expressive planning language supporting the representation of domains involving numerical quantities, actions with durations, predictable exogenous events and domain axioms. The integrated planning system is LPG [11], an efficient, state-of-the-art, fully-automated planner which received two awards at the last International planning competition.

IV. User-Oriented Services

ANEMONE provides a set of services to support i) academic people in some of their recurrent activities (fix an appointment, organize a meeting and search documents on the Web, ii) students in getting information about courses and iii) information technology people (including students) in getting information on documents and people that may help them to solve their programming problems.

A. Agenda Management

An agenda management system called MAgentA (Multi-Agent Agenda) has been developed. The system, besides managing users' personal agendas, provides a specific support to meeting organization: through a process of automated negotiation agents are able to determine the temporal location of a meeting which best fits the preferences of their owners, while satisfying some constraints specified by the meeting proposer.

The MAgentA system, implemented using the JADE agent development environment, consists in the following agents:

- a user management (UM) agent, in charge of managing the authentication of authorized users;
- a meeting management (MM) agent, in charge of coordinating the negotiation of a meeting among users' agents and of managing a database of meetings;
- a set of personal agenda (PA) agents, which represent individual users and maintain information about their scheduled activities and their preferences over their possible temporal allocation. PA agents are expected to be continuously running and available to receive meeting organization requests from other agents;
- a set of GUI agents, in charge of managing the interaction with the MAgentA users through a graphical interface. A GUI agent is activated only when necessary, i.e. during a user working session.

In a typical use scenario, a user, after authentication, interacts with a GUI agent to express her/his preferences about temporal locations of requested meetings and possibly to insert some personal scheduled activities within her/his agenda. The GUI agent communicates this information to user's PA agent which will use them when negotiating the organization of a meeting.

Moreover, using the GUI, a user may initiate the organization of a meeting by specifying:

- some temporal constraints about the temporal location of the meeting;
- the minimum and maximum duration of the meeting;
- a list of expected participants, partitioned into necessary participants and optional participants.

Once a request of a meeting organization has been formulated by the initiator user, it is submitted to the MM agent which tries to identify a solution, namely a suitable temporal location of the meeting, through a negotiation process consisting in the following steps.

First of all, using the FIPA contract-net protocol, the PA agent of every participant is solicited to propose a set of possible solutions compatible with the meeting temporal constraints, and to specify the user's preferences about the proposed solutions.

Then the MM agent verifies whether there exists a temporal location where all participants are available and, if one or more of them exists, it proposes them to the initiator user, ordering them on the basis of participants' preferences. Otherwise, the solutions where at least the necessary participants are available are searched for. Again, if these "weaker" solutions are found, they are proposed in an order consistent with the preferences specified in the agendas of the involved users; otherwise, a final search for (possibly less-satisficing) solutions is carried out, where personal activities included in participants' agendas are ignored. The list of the solutions found or a message of failure is then provided to the initiator user.

If a list of solutions has been found, the initiator user selects and confirms one of them: a notification is then sent to the MM agent and to the involved PA agents, which add the meeting to their databases.

In case of failure, it is up to the user to define a new request with different constraints and to initiate a new negotiation process.

B. Supporting Students in their University Activities

DIEE has developed an e-service devised to support graduated and undergraduated students in their activities. It is built upon a generic multi-agent architecture, designed to support the implementation of applications aimed at: (i) retrieving heterogeneous data spread among different Internet sources (i.e., generic web pages, news, and forums), (ii) filtering and organizing information according to personal interests explicitly stated by each user, and (iii) providing adaptation techniques to improve and refine throughout time

the profile of each selected user. The generic architecture has been called PACMAS, standing for Personalize, Adaptive, and Cooperative MultiAgent System, and encompasses four main levels (i.e., information, filter, task, and interface), each being associated to a specific role that agents can play. The communication between adjacent levels is achieved through suitable middle agents, which form a corresponding mid-span level. Each level is populated by a society of agents, which are autonomous and flexible, and can be personalized, adaptive and cooperative depending on the role they assume in the implemented application. PACMAS agents belong to one of the following categories:

- information agents, which access information sources, and are able to collect and manipulate such information [19];
- filter agents, able to process information according to user preferences [16] ;
- task agents, which help users to perform tasks by solving problems and exchanging information with other agents [17];
- interface agents, devised to facilitate the interaction between the user and other agents [18];
- middle agents, which are in charge of establishing communication among requesters and providers.

Let us consider a typical University Department. It generally makes available the information about courses, seminars, exams, professors, and students on different areas: web sites, forums, and news (NNTP) servers. All relevant information is not directly available but it is usually spread on the department portal, on the web site of each course, and on the personal page of each professor. Furthermore, each professor might activate her/his news and forum service. Some of the information potentially interest all students, such as lesson timetables, exam dates, taxes, and student tutoring. On the other hand, students belonging to different courses are interested in different lessons and exams. Typically, a student in search of relevant information about her/his University activities browses web sites, and reads announcements from forum and news services. This is often a repetitive and boring task that can be automated. From our perspective, personalization and adaptation represent the added value of such an automated system.

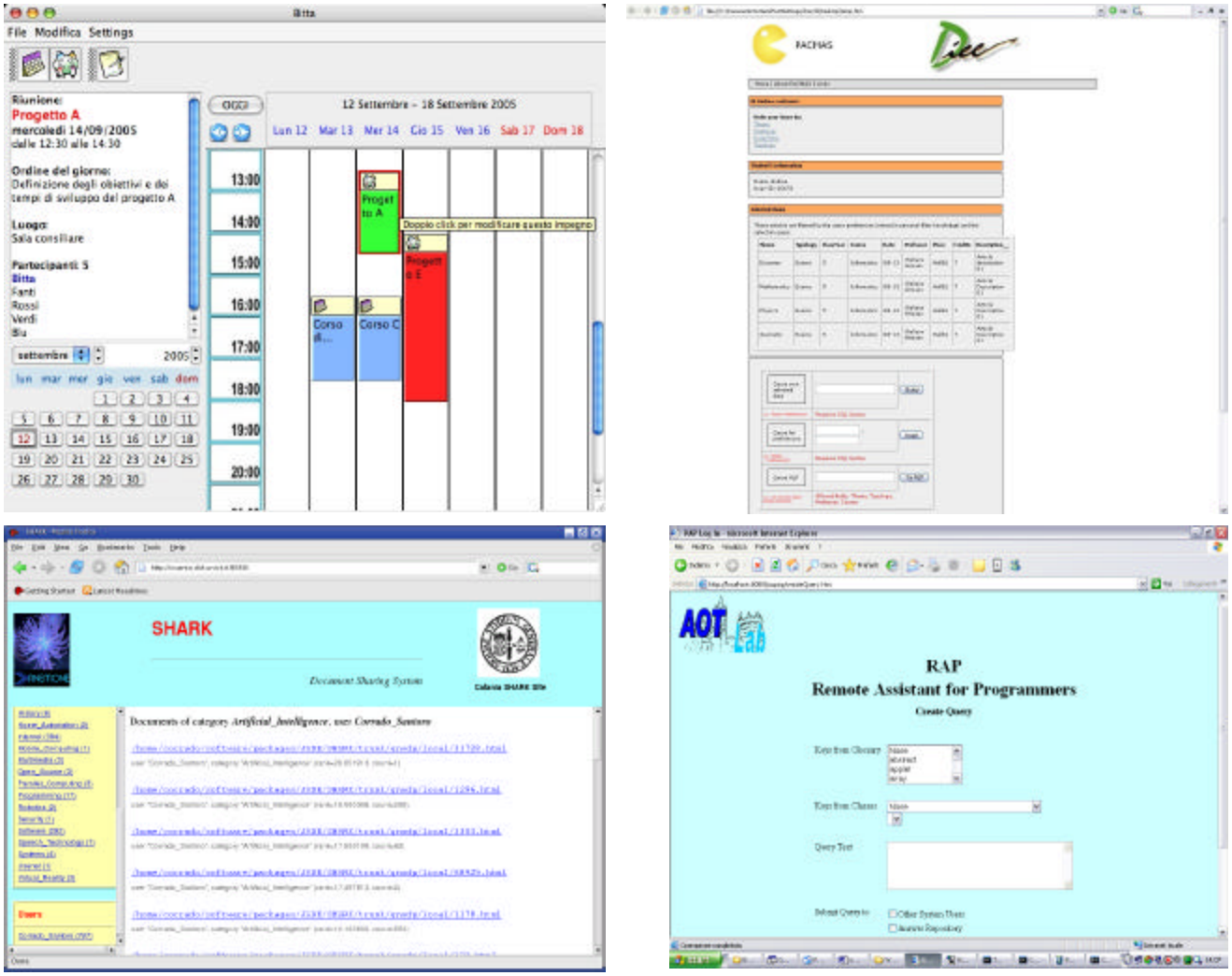

Figure 1. The GUIs of the four user-oriented

To provide an e-service able to support students in their activities, a prototype based on the PACMAS architecture has been implemented, using JADE [3] as the underlying framework. Supporting students involves several activities: information extraction, information retrieval and filtering, information processing, and result presentation. Each activity corresponds to a suitable level of the PACMAS architecture. Information extraction is carried out at the information level by information agents that play the role of wrappers, specialized for dealing with a specific information source. Information retrieval and filtering is carried out at the filter level, populated by two kind of agents: generic and personal. Generic filters are specifically aimed at removing all non relevant information retrieved from the involved information sources, whereas personal filters are devoted to select the information according to the personal needs, interests, and preferences of the corresponding user. Information processing is carried out at the task level, where agents are customized for a specific task (e.g. lesson timetable, seminars, and exams scheduling). Result presentation is carried out at the interface level, through agents that interact with the users. A suitable graphical interface - personalized for each user - that can run on a web browser, is available to allow communication among interface agents and the user.

The prototype has been tested on the information system of the Department of Electrical and Electronic Engineering (DIEE) at the University of Cagliari.. The system is able to learn specific user's interests to retrieve, filter, and show only the information deemed relevant by her/him. A beta version of the web service is available at: http://iascw.diee.unica.it/PacmasWWW.

C. Documents Search

SHARK is a multi-agent P2P document sharing system aiming to provide users with a more effective tool to find documents and promote collaborations among them [20]. Each SHARK agent (such as Categoriser, Searcher, UserProfiler, etc.) autonomously performs a small task, such as document categorisation, finding, user profiling, etc. and communicates its results to other agents.

The hosts in a SHARK network are given different roles. A client host provides users with a few services, such as user profiling and document analysis, and allow users to log in to an AgentCities [21] host. AgentCities hosts are servers, connected to each other, each running a FIPA-compliant agent platform and handling data related to SHARK users and their documents.

1) SHARK Agents

n the following we describe the agents that constitute SHARK. Agent Cruncher analyses shared documents and extracts from each a set of keywords. For this, Cruncher uses filters to recognise and remove HTML, LaTex, RTF and PDF tags that are used only to format the text; then it removes the stop words, and, for all the remaining words, it extracts the appropriate stems. The output is a list of word stems ranked by the number of occurrences found [22][23].

Agent Categoriser, on the basis of extracted word stems, associates categories to documents. Categoriser holds a knowledge base, containing, for each known category: its name, the list of keyword stems and the respective frequency. Taking as input a list of word stems, Categoriser calculates the "distance" between such a list and the known categories, by using the dot product. The category that minimises the distance is chosen as the category to which the document belongs.

Agent UserProfiler detects the activities that a user operating with a web browse performs, and analyses the shared documents in order to continually update his/her profile. The user profile consists of the list of categories corresponding to shared documents or visited web pages. Each category is associated with a score, which reflects the degree of interest, measured on the basis of the number of shared documents and visited web pages.

Agent Searcher runs on an AgentCities host and holds the list of categories identified for the local shared documents, for each category the list of documents and the user providing each document. Given a user-provided query (as a list of keywords), Searcher looks for matching categories and returns the list of corresponding documents with the user providing each. The query is then propagated to the other AgentCities hosts, where local Searchers will perform analogous activities.

Agent Correspondent handles document download requests originating from other users.

Agent Advertiser periodically checks user profiles in order to find a partial match. Whenever the matching degree is above a given threshold, the users with common interests are notified with an email message. It is then up to the users to find the opportunity for a collaboration.

The instances of the agent classes described above run on different hosts. The user host is equipped with Cruncher, UserProfiler and Correspondent; AgentCities servers host Categoriser, Searcher and Advertiser.

2) Using SHARK

Users interact with SHARK by means of a web interface, of which we highlight here two important features. The first one is the searching facility: once a user has performed a query, by typing a set of keywords into a web form, this is sent to the Searcher on the AgentCities server the user is connected with. The results of Searcher are sorted so that the more relevant document is that exhibiting the highest frequency (in percentage with respect to all the document's keywords) of the keyword queried-if only one keyword is provided. If more than one keyword is given, the total relevance is computed as the average of each single keyword relevance.

The second feature is the collaboration facility. This is connected with searches and consists of providing a list that reports the name of the users who have, in their user profile, the keyword(s) queried. Names are ranked according to the

relevance of the user profile with respect to the keyword(s) queried. Relevance is computed using the same method employed for documents.

D. Software Development

RAP (Remote Assistant for Programmers), is a Web and multi-agent based system to support remote students and programmers during common projects or activities based on the use of the Java programming language [24].

1) RAP Agents

In this section we describe the agents that compose the RAP system. Personal Agents allow the interaction between the user and the different parts of the system and, in particular, between the users themselves. Moreover, these agents are responsible of building the user profile and maintaining it when the user is "on-line". User-agent interaction can be performed in two different ways: through a Web based interface or through emails (if the user is not on-line). User Profile Managers are responsible of maintaining and updating the profile of system users. Answer Managers maintain the answers provided by users during the life of the system and they find the appropriate answers to the new queries of the users. Besides providing an answer, these agents update the score of the answer and forward the vote to the User Profile Manager for updating the user profile. Document Managers find the appropriate documents to answer the queries submitted by system users. E-mail Managers are responsible of the communication between the system and the off-line users. Starter Agents are responsible for activating a Personal Agent when either a user logs on or another agent requests it. Directory Facilitators are responsible to inform an agent about the address of the other agents active in the system (yellow pages service).

2) Profile Management and Open Communities

The management of user and document profiles is performed in two different phases: an initialization phase and an updating phase. In order to simplify and reduce the possibility of inaccuracy due to people's opinions of themselves and to incomplete information, we decided to build the initial profile of the users and documents in an automated way. Profiles are represented by vectors of weighted terms whose values are related to the frequency of the term itself in the user's documents. Document and user profiles are computed by using "term frequency inverse document frequency" (TF-IDF) [24] algorithm. Each user profile is built by user's Personal Agent through the analysis of the software she/he wrote. This is only the initial user's profile, it will be updated when the user writes new code or interacts with the system answering some queries.

An important requirement that has guided the design of RAP has been the support for open and distributed communities. RAP structure is open, since new users can register and access the system, and a registered user can acquire new skills or produce new software. The community beneath RAP is distributed: the whole system can consist of a dynamic group of local communities. Each community can operate isolated, but can also decide to join a group of communities, sharing experts and documents repositories.

The open and distributed nature of the system entails some significant problems in the evaluation of information: the evaluation of both experts and documents is strongly dependent on the actual composition of the community group. For example, if a user is rated as the maximum expert to answer a query, he is rated considering only the users registered in the system at that moment. As a matter of fact, TF-IDF algorithm can be easily used in a centralized system where all the profiles and the data are managed, while our context is more complex. For these reasons, each profile component of RAP is associated with two elements: an absolute element and a TF-IDF weighted element. The absolute one depends only on the user (or document) profile, instead the TF-IDF element is related to both the user profile and the whole community profiles. Moreover, while the absolute element is stored in a database, the weighted one is maintained in memory and it is recalculated when necessary.

V. CONCLUSION

In this paper, we presented ANEMONE, a multi-agent platforms network that provides services for the academic community (professors, researchers and students).

The ANEMONE network and services are the result of a project involving five Italian universities (University of Parma, University of Brescia, University of Cagliari, University of Catania and "Politecnico di Milano" Technical University) and the realized network is composed of five nodes deployed in the different universities. However, the ANEMONE network can interoperate with agent platforms deployed in different parts of the world. In fact, ANEMONE project takes part of the OpenNet initiative [2] that is a project dedicated to facilitating collaboration between research projects developing, applying and above all deploying Agent, Semantic Web, Web Services, Grid and similar networked application technologies in large-scale open environments such as the public Internet. In particular, the core partners of this initiative deployed a backbone network of agent platforms, including a platform at the University of Parma. This backbone network has the goal to be the interconnection network among the systems and prototypes belonging to the initiative (currently different projects are running in different part of the world and different tens of agent platform are active).

REFERENCES

[1] "Agentcities: A Worldwide Open Agent Network" Steven Willmott, Jonathan Dale, Bernard Burg, Patricia Charlton and Paul O'brien. Short article in Agentlink News Issue 8, November 2001.

[2] OpenNet initiative Home Page. Available from http://x-opennet.org/.

[3] F. Bellifemine, A. Poggi, G. Rimassa, Developing multi agent systems with a FIPA-compliant agent framework. Software Practice & Experience, 31:103-128, 2001
[4] Foundation for Intelligent Physical Agents, “Agent Communication Language Specifications”, http://www.fipa.org/repository/aclspecs.html, 2002
[5] M. Wooldridge, “Reasoning about rational agents”, MIT Press, 2000
[6] M. Verdicchio, M. Colombetti, “A Commitment-based Communicative Act Library”, Proceedings of the Fourth International Conference on Autonomous Agents and Multiagent Systems (AAMAS 05), vol. 2, p755-761, Utrecht, 2005
[7] Ernest Friedman Hill, “JESS in Action”, Manning Publications, 2003
[8] D. Megginson, “SAX”, http://www.saxproject.org, 2004
[9] P. Baroni, M. Giacomin, G. Guida, SCC-recursiveness: a general schema for argumentation semantics, accettato per la pubblicazione su Artificial Intelligence, 2005
[10] FIPA Specification 00079, Agent Software Integration, 2001, http://www.fipa.org/specs/fipa00079/
[11] A.Gerevini, A, Saetti, I. Serina, Planning through Stochastic Local Search and Temporal Action Graphs in LPG, Journal of Artificial Intelligence Research (JAIR), 20, 2005, 239-290
[12] J. Hoffmann, S. Edelkamp, The Deterministic Part of IPC-4: An Overview, to appear in Journal of Artificial Intelligence Research (JAIR), 2005
[13] J. Pollock, How to Reason Defeasibly, Artificial Intelligence, 57(1), 1992, 1-42
[14] H. Prakken and G. A. W. Vreeswijk, Logics for Defeasible Argumentation, in Dov M. Gabbay and F. Guenthner Eds., Handbook of Philosophical Logic, Kluwer, 2001
[15] K. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI 97), pages 578.583, 1997.
[16] A. Falk and I. Josson. Paws: An agent for www-retrieval and filtering. In Proceedings of Practical Application of Intelligent Agents and Multi-agents Technology (PAAM-96), pages 169.179, 1996.
[17] J. Giampapa, K. Sycara, A. Fath, A. Steinfeld, and D. Siewiorek. A multi-agent system for automatically resolving network interoperability problems. In Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, pages 1462.1463, 2004.
[18] H. Lieberman. Autonomous interface agents. In Proceedings of the ACM Conference on Computers and Human Interface (CHI-97), pages 67.74, 1997.
[19] P. Maes. Agents that reduce work and information overload. Communications of the ACM, 37(7):31.40, 1994
[20] A. Di Stefano, G. Pappalardo, C. Santoro, E. Tramontana. “SHARK, a Multi-Agent System to Support Document Sharing and Promote Collaboration”. In Proceedings IEEE Hot P2P Workshop. Volendam, Holland. October, 2004.
[21] WWW. www.agentcities.net, 2005.
[22] H. Lieberman. Letizia: An Agent That Assists Web Browsing. In International Joint Conference on Artificial Intelligence, Montreal, August 1995.
[23] H. Lieberman, P. Maes, and N. Van Dyke. Butterfly: A Conversation-Finding Agent for Internet Relay Chat. In International Conference on Intelligent User Interfaces, Los Angeles, January 1999.
[24] L. Lazzari, M. Mari, A. Negri, A. Poggi: Support Remote Software Development in an Open Distributed Community. In AAMAS05 workshop Agent-Based System for Human Learning (ABSHL), Utrecht, 2005.
[25] Salton, G.: Automatic Text Processing. (1989), Addison-Wesley.