Open access • Proceedings Article • DOI:10.1109/ENABL.1999.805175

Using agents for distributed software project management — Source link

Rory V. O'Connor, John Jenkins

Institutions: Dublin City University, Middlesex University

Published on: 16 Jun 1999 - Workshops on Enabling Technologies: Infrastracture for Collaborative Enterprises

Topics: Software project management, Software system, Resource-oriented architecture, Software development and Software construction

Using Agents for Distributed Software Project Management

Rory O'Connor
School of Computer Applications
Dublin City University
Ireland
roconnor@compapp.dcu.ie

John Jenkins
School of Computing Science
Middlesex University
UK

Abstract

The paper explores the role of artificial intelligence techniques in the development of an enhanced software project management tool, which takes account of the emerging requirement for support systems to address the increasing trend towards distributed multi-platform software development projects. In addressing these aims this research devised a novel architecture and framework for use as the basis of an intelligent assistance system for use by software project managers, in the planning and managing of a software project. This paper also describes the construction of a prototype system to implement this architecture and the results of a series of user trials on this prototype system.

1. Software Project Management

The key issue in project management is decision making. Software project managers make many decisions every day, ranging from the relatively inconsequential to the significant. Ceteris paribus, good outcomes from those decisions are more desirable than bad outcomes. Project managers make decisions based on a combination of judgement and information from staff, clients, research literature and current market forces, as well as knowledge gained from previous projects. Ideally, all relevant information should be brought together before judgement is exercised. The quality of a decision depends on the adequacy of the available information, the quality of the information, the number of options available at the time of the decision and the ability of the people involved to interpret this information.

Software-intensive projects often fail because the project managers lack knowledge of good practices and effective processes which can reduce risk and increase the likelihood of success. Managers of projects need to know how to establish a set of processes which are tailored to a project's requirements in terms of functionality, time, cost, quality and their associated risks.

Constructing, maintaining and extending large complex software systems pose many problems of managing all the people, systems and agencies involved. Although many project management systems are currently available, the enormous scope and complexity of current software systems means moving beyond the current state of practice (for example, PERT charts or Microsoft Project). Managing large-scale projects requires facilities for coordinating independent activities and managing the project plans themselves.

To support project managers, organisations have sought to develop tools to assist with various aspects of the management of their software processes. Many tools exist in the market today that assist a project manager in addressing some of these objectives. These tools fall into three main categories:

- Project planning - tools which are concerned with the scheduling aspects of planning a project, and pay less attention to organisation and methodological aspects of management.
- Process management - which support the framework and rules of management of the project's process.
- Risk analysis - tools used at specific stages during a project to assess risk.

However, most of these systems fall short of supporting the project manager in their decision making processes and do not offer assistance in representing knowledge about plans and designs, or provide mechanisms for reasoning about plans and designs in flexible ways. Further aspects to supporting the software project manager which are not addressed by today's support systems are the distributed and cross-platform nature of systems development.

This research is motivated by the assertion that users of existing software project management systems could benefit greatly from the inclusion of intelligent assistance techniques in such tools. In addition, such new support systems should provide for the distributed cross-platform nature of modern client-server development.

Therefore the objective of this research was to address this shortfall and provide a support tool which will increase the likelihood of success by helping the project manager who

has to make decisions on these issues. Such a tool will encapsulate expert knowledge and make it available to all users. Some of the potential benefits of this approach as applied to the to decision-making process in the domain of software project management are:

- Suggestions are made which help the user balance cost, quality and time in making decisions about the use of project resources.
- Knowledge is shared about different lifecycle models and why one or another may be most suitable for the users projects.
- Measurements are suggested which will enable the user to see how well the project is reaching greater organisational goals and re-plan the ways to reach these goals, if necessary.

Even the most experienced project manager may have difficulty knowing the best planning options, even if the critical input parameters of resources, constraints and requirements are known.

2. The Role of an Intelligent Assistant

The notion of an intelligent assistant is not new. Indeed, as far back as 399 BC Socrates claimed to have an intelligent assistant, although not in the strictest sense of course. But Socrates did claim to have a non-human companion, which he called a Daemon. Intelligent and always ready to offer good advice, Socrates daemon could be trusted to act without prompting. Real, hard-coded, linguistic and symbolic links abound between Socrates daemon and today's notion of an intelligent assistant.

A software system designed to act as an intelligent team member (or Daemon) could help in the planning and execution of a project. Such an intelligent project assistant could help to preserve knowledge about tasks, to record the reasons for decisions and retrieve information relevant to new problems. They could function as co-workers, assisting and collaborating with the design or operations teams for complex systems. They could also supply institutional memory. They could recall the rationale of previous decisions and, in times of crisis, explain the methods and reasoning previously used to handle that situation.

Significant design projects are typically accomplished by teams. An intelligent project assistant could act as design associate [15]. Designs are almost always redesigned; effective redesign requires an understanding of why previous design choices were made and of how these choices achieved or compromised the desired goals; all are vulnerable to loss of important information from changes in design-team membership.

In software development projects in particular, an intelligent project assistant can keep track of specifications, design proposals, and implementations for a software project throughout its life cycle. It can record the design decisions of a constantly changing team and also be a repository of solutions and components for new projects. Reasoning techniques can be used to track the (mis)match between specifications and implementations, while analogy techniques can be used to look for existing specifications, components or implementations that match some new requirement.

An intelligent project assistant can additionally be of benefit when training new personnel. For many tasks, on-the-job training is extremely effective, providing the trainee with the chance to make real, on-the-spot decisions and see the consequences. On-the-job training is impossible, however, when a bad decision can be disastrous - as in the management of a large complex software development project. Simulations of the project management process, would enable the development of training systems for such situations [5]. These same simulation capabilities are also important when the cost of assembling large groups of people for training is prohibitive.

As part of this research a survey of software project management tool users was conducted to obtain an appreciation of the actual state-of-practice by project managers in relation to tool usage, i.e. what do they actually use these tools for and is this consistent with the tool vendors intended usage [13]. In addition, participants were also asked to consider the aspects of intelligent assistance previously discussed and comment on the possible benefits of incorporating this into a project management support system. All of the project managers considered the notion of an intelligent project assistant as a useful addition to the existing range of features in project management tools. In particular, they supported the notion of a tool which could intelligently manage project knowledge, and capture knowledge and lessons learned about projects into a project knowledge base. Apart from the intelligent assistance aspects of this research, the problems associated with organisations having distributed project teams, coupled with multiple hardware platforms was identified by the project managers surveyed, thus highlighting the need tools to operate in a distributed multi-platform environment.

3. Implementing Intelligent Assistance

Many solutions have been proposed to the notion of intelligent assistance over the years. These fall under four major categories, Decision Support Systems, Expert Systems, Expert Critiquing Systems and Blackboard systems. In addition, the concept of Intelligent Agents [16] has recently emerged as a potential fifth category. It is proposed that in the complex domain of software project management, a useful tool to support the project manager in the decision making process is likely to be a hybrid of a number of techniques, including DSS, ES, ECS and the blackboard model. It has therefore been proposed to incorporate the information gathering and analysis techniques of DSS, with the ability of ES to propose possible solutions using expert knowledge and best practices and the power of ECS to critique the possible solutions, thus providing the project manager with every facility to make an informed and quality decision [11] [12].

It is considered that an agent based system will provide for an approach which enables the inter-working of a variety of well understood techniques within a single underlying framework - that of agent-orientated system. We therefore propose a system composed of a library of intelligent software agents - where each agent would play the role of a 'mini-expert system' or 'mini-critiquing system', each with an associated knowledge base. These agents would utilise the blackboard model of problem solving to converge on possible solution states and examine those states to assess their suitability given current conditions. This agent-orientated system would operate within the overall framework of a decision support system, which would provide for the gathering and analysis of data regarding a project and the development of models of the project with the aid and critique of the agents.

The major benefits perceived of this approach are facilitating and improving the quality of decision making by a software project manager by reducing information overload and augmenting the cognitive limitations and bounds of the decision maker. This hybrid method of assistance coupled with the architectural properties of intelligent agents (dynamic and distributed objects) present an ideal strategy to implement intelligent assistance systems in the domain of software project management.

In addition to the properties outlined above, an agent-orientated architecture is a natural choice to address the issue of heterogeneous client-server systems development. Recent research in Java-based agents [14] [2] and mobile Java-based agents [9] have concluded that they are a viable technology on which to establish a platform independent agent-orientated architecture. To address the distributed client-server issue, research conducted at San Jose State University [10] successfully used CORBA (Common Object Request Broker Architecture) as a basis for developing platform independent client-server systems, including agent-orientated systems. To provide the necessary flexibility for the proposed system and to tackle the issues above, it is considered that both Java and CORBA provide an appropriate framework on which to base our proposed system.

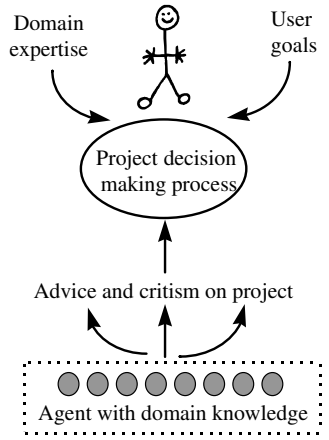

Figure 1 Decision Making Paradigm

The decision making paradigm of this agent-orientated approach is illustrated in Figure 1, where two entities, agents and a user, working together to contribute what they know about the domain to solving some problem and hence make a quality informed decision. The users primary role is to generate and modify solutions; the agents is to analyse those solutions and produce a critique and advise on a possible solution for the human to apply in the next iteration of this process. In this scenario agents constantly 'watch' the actions of the user by way of monitoring project parameters as the user developing project plans and inputs actual project data. When an agent has all the information it needs, it will proceed with its analysis and produce its conclusions. These conclusions would be given to the user in terms of advice/criticism on the current/predicted future project situation, and also may be used as input data to other agents. For example, there may be a number of agents who specialise in the selection of the most appropriate process model (lifecycle) for a particular project. The agents could have a set of criteria based on certain attributes of the project such as: the problem domain, product, available resources, personnel, and organisational attributes. These attributes would be examined and for each process model a comparative rating produced, indicating the most appropriate choice of process model. The agent would then communicate these findings to the user and other agents.

4. System Architecture

Software architecture focuses on three aspects of software design [1];

- **Partitioning** - The functional partitioning of software modules.
- **Interfaces** - The software interfaces between modules.
- **Connections** - The selection and characteristics of the technology used to implement the interface connections between software modules.

This 'partitioning' approach was taken to the development of a suitable architecture to support the proposed intelligent assistant system. In this sections we will describe the system architecture from a high level and examine some of the modules and their connections.

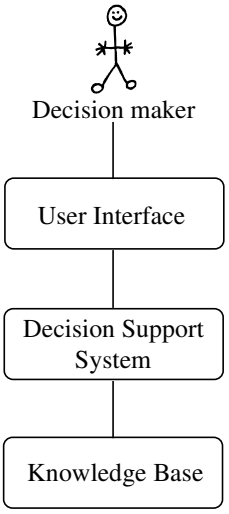

Figure 2 System Architecture

A high-level (user level) view of the system architecture is illustrated in Figure 2 and consists of the user interface to the system, the decision support system itself and the underlying knowledge base which contains the expertise and knowledge which will be used to assist the project manager in the planning, managing and execution of a software development project.

The component modules of the system architecture are illustrated in Figure 3, and are described below:

- **User Interface** - This component handles the management of all the screen elements (menus, dialog boxes, etc.), validates data entered by the user and passes on clear functional messages to the rest of the system.
- **System Kernel** - This is the core component of the system and handles all the processing and storage of user entered data. It manages all aspects of project plans and channels advice from the agents to the user.
- **Data Manager** - This component manages all aspects of the mapping from the logical view of data to its physical storage and maintenance. This module is under the control of the System Kernel and all requests for data must be channelled through the Kernel.
- **Agent Controller** - This module acts as a controller (or supervisor unit) over the agent community and manages the scheduling and execution of agents, as well as governing write access to the Blackboard.
- **Blackboard** - This represents the global problem solving state of the system. Over time, agents produce changes to the Blackboard which lead incrementally to advice on the project under consideration. The Blackboard is under the control of the Agent Controller and all requests for data read / write must be channelled through the Agent Controller.
- **Agent Library** - All agents are contained in an agent library, but remain under the control of the Agent Controller. The purpose of the Agent Library is to manage the physical agents themselves and to service requests for agent interactions from the Agent Controller.

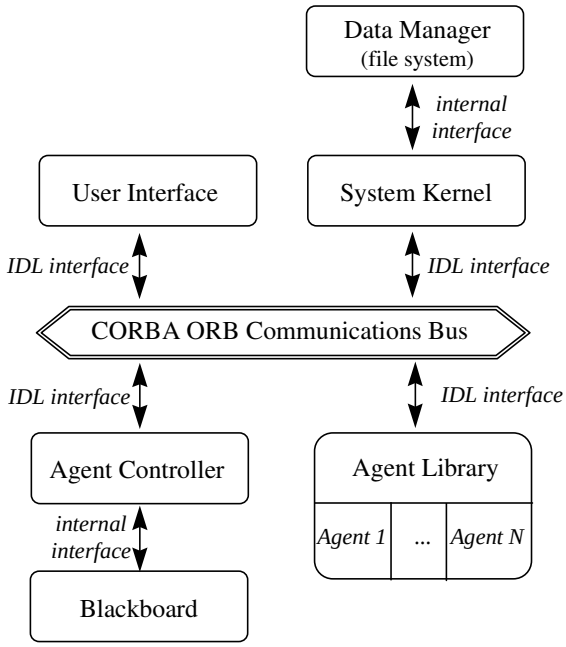

Figure 3 Component Architecture

CORBA [3] was chosen as the interface bus on which to implement message passing between each of the modules in the system architecture. The CORBA bus allows transparent access to distributed objects over a

heterogeneous network of machines and operating systems. CORBA distributes messages via its Object Request Broker (ORB) transparently between registered objects. The ORB receives requests from a 'client' to send a message to an object. The broker locates the object referred to by the client and delivers the message to that object. This style of architecture combined with the flexibility of CORBA provides a unique solution to the requirements of independence of implementation language, capacity for evolution and interfacing ability.

The use of CORBA allows us to maintain the tool kernel on a typical server while porting the GUI to a client machine, with both the agent library and the project database located physically anywhere on the network. With this in mind, possible alternatives would be:

- A 'thin client' - possibly web based or implemented in Java, which would therefore be platform independent, while still providing a multimedia oriented GUI.
- Classical server-side application containing the kernel, agent supervisor and database.
- An agent library located anywhere on the LAN or an intranet/internet, which may be implemented in any language and accessed via CORBA.

The main advantages of this approach are:

- The 'light-weight' clients are platform independent, thrifty in resources, and easily upgradable.
- The application has a powerful classical kernel, while retaining the advantages of client-server computing.
- It provides a facility for the tool to evolve new services, which can be added as new modules.
- The use of a CORBA bus provides interfacing capabilities with other (future) CORBA complaint

Various research projects have investigated languages for implementing intelligent agents in recent years [16] [17]. In the early stages, agents were local to individual projects and their languages were mostly idiosyncratic. As a result there are a large number of representation languages, each with their own particular characteristics, which do not have inter-agent communication capabilities. An obvious solution is to have a lingua franca, where ideally all agents that implement the same lingua franca would be mutually intelligible [8]. However, the agent community is still a long way from attaining this goal.

An example of a popular intelligent agent language is JESS (Java Expert System Shell) [14] which is a clone of the core of the CLIPS (C Language Integrated Production System) [4] expert system shell developed. JESS contains the main features of CLIPS and is downward compatible with CLIPS, in that every valid JESS script is a valid CLIPS script. The primary representation methodology in both CLIPS and JESS is a forward chaining production rule language based on the Rete algorithm.

Currently, only a small number of Java based products for the development of expert system tools exist in the market place. For example, [6] [7] reviews five commercially available tools: Advisor/J (Neuron Data Inc.), Ilog Rules for Java (Ilog Inc.), CruXpert (Crux Inc.), Selectica SRx Selection Engines (Selectica Inc.) and JESS (Sandia National Laboratories). The JESS system was chosen as the primary knowledge representation system for a number of reasons, with the primary motivation that (from an architectural perspective) it is based on an open, mature and portable knowledge representation language, i.e. CLIPS.

5. Prototype Implementation

The development platform chosen was a standard Intel Pentium PC running the Windows NT 4 operating system connected via a LAN to several Intel Pentium PC servers and Sun Microsystem servers and workstations running the Solaris operating system. For the development of this system, two main tools were used: a Java compiler - in this case Sun Microsystems JDK - and a Java implementation of a CORBA ORB - in this case Iona's OrbixWeb.

An early prototype of the proposed system was developed for a number of reasons: Firstly as a proof of concept - the prototype assisted in highlighting any flaws in the proposed architecture and investigated the feasibility of the distributed architecture based on CORBA and Java. It also assisted in identifying possible communication, data storage and knowledge representation problems that may occur.

The prototype consisted of a number of servers in the agent architecture - Agent Controller, GUI, and the Agents themselves. The advantage here is that each of these servers are asynchronous from each other and from the clients that call them, making them almost completely independent. Thus on the majority of occasions when the client call is made it can continue with its own process, the call is one-way and so does not have to wait for the calls completion.

This initial prototype system was successfully developed and deployed in a distributed manner over a number of networked machines. For example, the expert agent server was run on a Windows NT server, with the client (GUI) deployed on Windows 95, with agent communication taking place (via CORBA) over the network using TCP/IP.

Following from this initial prototype a series of four further prototype systems were constructed over an extended period of time, with a phased approach to the evolution of system services being adopted. The final prototype consisted of a set of 30 expert advisory agents in the areas of project planning/re-planning, risk management and metrics and a full set of basic services including the ability to produce multiple scenarios based on the 'current' status of the project. As the end of each prototype development stage the current system was demonstrated to a representative group of software project managers as outlined in section 6. Currently the prototype system is undergoing a commercialisation phase, with a view to launching a product in the tool market place.

6. Trial Usage

There are several reasons for conducting user trials of the prototype system; Firstly it exposes the system to 'real world' project managers and obtains feedback from them. In addition it provides a mechanism to elicit opinion from users as to the added value of the system as compared to traditional project management systems. The trials were conducted using twelve project management staff from two organisations. These staff ranged in experience from novice to highly experienced, with the projects under their control ranging from small scale to large, highly complex systems. The actual user trials were conducted in four distinct stages, one for each of the major prototype releases (not including the first prototype, as it was an architectural proof):

- **Trial 1** - was conducted using the second prototype and had as its purpose the aim of testing the user interface with respect to data capture.
- **Trial 2** - was conducted using the third prototype and had the dual purpose of the testing of scenarios and generated advice associated with them.
- **Trial 3** - was mostly concerned with testing the total functionality of the system.
- **Trial 4** - was conducted using the final prototype. This trial concentrated on advice produced by the agents.

The main output of these trials was a set of four review documents - one for each trial - which detailed the comments and opinions of the users involved in each trial. The following are some of the main finding of these user trials:

- **Operation** - The prototype system was successfully operated by a number of users on a variety of machines. This was an indication that the prototype system was capable of being executed in a commercial environment, although the slow speed of execution was an important issue. However, users acknowledged that the speed issue was not of great importance for a research prototype, but would be for a commercial version.
- **Decision support** - The general feeling of users was that the prototype system demonstrated that the notion of intelligent assistance for software project management was feasible. In addition, they considered that the prototype implementation provided a suitable framework for supporting decision making and had the potential to be of use in a commercial setting.
- **Project descriptions** - The general opinion of users was that the mechanisms of describing projects (via models and scenarios based on a model) was an appropriate and useful device to capture information about a project. In particular the notion of multiple scenarios to examine multiple views (with corresponding advice) of a project was useful.
- **Advice** - Of paramount interest in these trials was user feedback in relation to advice produced by agents. The overall trend was that novice users considered the advice appropriate and useful as either a reminder of a particular aspect of management, or as an indicator of which direction to consider. However, more experienced project managers expressed the desire for more specific and quantitative advice.
- **Training tool** - A suggestion put forward by a number of users was the possibility of a repositioning of the system for use as a training tool, in which users could develop a model of a fictitious project and thus practice project management skills on a 'virtual project'.

The most difficult issue to tackle which arose during the user trials was the request for advice which was more quantitative in nature. This had proved difficult for two reasons; Firstly, little suitable source material was available which contained quantitative data / results that could be used as the basis for agents. Secondly, it is difficult for humans to discern the differences between quantitative values at a fine grain level with domains such as software project management. For example, there is no appreciable difference between the values of 70% and 75% if they were expressed as a measure of suitability for a given lifecycle model. However, it is worth noting that this quantitative issue - while an important issue in its own right - is not a central issue to the proposed architecture of this thesis. It is however an indicator of the nature of advice users perceive to be useful in addition to the advice already produced.

The comments received from users were based on a series of research prototypes and indicate the proposal of an intelligent assistant system for software project management is a viable notion.

7. Conclusions

This paper has set forth a proposal for an intelligent assistant system for use by software project managers. Such an intelligent project assistant could help to preserve knowledge about tasks, function as co-workers, assisting and collaborating with the design or operations teams for complex systems.

This research reported in this paper has proposed a novel architecture for the development of the above intelligent assistant system. This approach is a fusion of a number of techniques within a multi-agent framework which aims to improve the quality of the decision making process in the less well understood domain of software project management. This framework incorporates the information gathering and analysis techniques of a Decision Support System with the ability of an Expert System to propose possible solutions using expert knowledge and best practices and the power of an Expert Critiquing System to critique the possible solutions, thus providing the project manager with every facility to make an informed and quality decision. This novel approach enables the inter-working of a variety of well understood techniques within a single underlying framework. An important characteristic of this approach is the combination of these techniques in an open distributed environment with the potential for continuous evolution.

To assist with validating the proposed architecture, a prototype system was developed as part of this research and a series of trials conducted in a commercial environment using software project managers. The conclusion of these trials was that the prototype system demonstrated that the notion of an intelligent assistant system for software project management was a viable commercial concept. Further, the prototype system demonstrated that the proposed architecture provided a suitable framework for supporting decision making and had the potential to be of use in a commercial setting.

One of the significant drawbacks in relation to the evaluation of the system described in this paper - or indeed any software engineering tool - is that a comprehensive evaluation study requires an extended period of time with access to a large group of potential users. However, in this is a luxury not afforded to most academic research projects. Notwithstanding the foregoing, it is considered that the research reported in this paper provides a significant step forward in the development of a new generation of intelligent assistant systems for software project management.

References

[1] W.Brown, R.Malveau, H.McCormick and T.Mowbray, “Anti Patterns - Refactoring Software Architectures and Projects in Crisis”, Wiley, 1999.

[2] A.Caglayan and C.Harrison, “Agent Sourcebook”, Wiley, 1997.

[3] “CORBA: Architecture and Specification”, Object Management Group, 1996.

[4] J.Giarratano and G.Riley, “Expert Systems - Principles and Programming”, PWS Publishing Company, 1994.

[5] B.Grosz and R.Davis (Eds.), “A Report to APRA on Twenty-First Century Intelligent Systems”, American Association for Artificial Intelligence, 1994.

[6] C.Hall (Ed.), “Intelligent Software Strategies”, Cutter Information Corp., Summer 1997.

[7] C.Hall (Ed.), “Intelligent Software Strategies”, Cutter Information Corp., Fall 1997.

[8] M.Huhns and M.Sing, “Conversational Agents”, IEEE Internet Computing, Vol. 1, No. 2, 1997.

[9] D.Lange and M.Oshima, “Programming Mobile Agents in Java - with the Java Aglet API”, technical Report, IBM Research, Japan, 1997.

[10] R.Orfali and D.Harkey, “Client/Server Programming with Java and CORBA”, Wiley, 1997.

[11] R.O'Connor, T.Renault, C.Floch, T.Moynihan and A.Combelles, “Prompter - A Decision Support Tool using Distributed Intelligent Agents”, In Proceedings of EXPERSYS-97, 1997.

[12] R.O’Connor and T.Renault, “Designing an Internet Enabled Decision Support Tool in the Domain of Software Project Management”, In Proceedings of EIS-99, 1999.

[13] R.O’Connor and J.O.Jenkins, “Supporting Effective Software Project Management and Control by the use of Intelligent Knowledge-based Guidance”, In Proceedings of 9th European Software Control and Metrics conference (ESCOM), pp. 143 - 151, Rome, Italy, 1999

[14] M.Watson, “Intelligent Java Applications”, Morgan Kaufmann, 1997.

[15] D.Weld (Ed.), “The Role of Intelligent Systems in the National Information Infrastructure”, AI Magazine, Fall 1995.

[16] M.Wooldridge and N.Jennings, “Intelligent Agents: Theory and Practice”, Knowledge Engineering Review, Vol. 11, No. 2, 1995.

[17] M.Wooldridge, J.Muller, M.Tambe (Eds.), “Intelligent Agents II: Agents Theories, Architectures and Languages”, Lecture Notes in Computer Science 1137, Springer Verlag, 1995.