Integrating Rule and Agent-Based Programming to Realize Complex Systems

ALESSANDRO BENEVENTI, AGOSTINO POGGI, MICHELE TOMAIUOLO, PAOLA TURCI
Dipartimento di Ingegneria dell'Informazione
Università degli Studi di Parma
Viale delle Scienze, 181A – 43100 - Parma
ITALY

Abstract: - Current agent-based frameworks are used for the realization of flexible distributed systems, but rarely they provide the support for building really adaptive and intelligent applications. This paper presents a framework integrating a scripting engine and a rule-based system inside a FIPA compliant agent framework. Both tasks and rules can be moved among the deployed agents. The security issues are analyzed and proper authentication and authorization mechanisms are deployed. Application areas range from e-learning to e-business, from service composition to network management. In our experiences, the system proved particularly suitable for building advanced service-based applications to be deployed in open and evolving network environments.

Key-Words: Agent-based systems. Rule engine. Adaptive agents. Security.

1 Introduction

The main motivation behind this work is the need to fill the existing gap between multi-agent systems as environments to build distributed applications, and their missed promise to pave the way for really adaptive and smart applications. The solution presented in the paper is founded on the integration of a rule-based framework, Drools [3], and a scripting engine, BeanShell [2], inside JADE [12], a widespread, FIPA compliant [5], distributed multi-agent system.

The added value of the proposed solution is twofold. The first evident advantage is that it allows the creation of rule-based agents, whose actions are based on the activation of different rules according to the evolving perception of the world. Another important result is the possibility to move rules from one agent to another, allowing them to share learned behaviours, evolving their state and code, in order to achieve their users' needs and to best fit their environment.

Of course, these social interactions, i.e., the ability to send and receive new behaviours, require security issues to be carefully analyzed. The paper tackles these issues and describes some mechanisms to define precise policies, to limit the access to critic resources only to authenticated and authorized principals.

In particular, Section 2 describes existing technologies and theories about related areas, as rule-engines and code mobility. Section 3 deals with the integration of the Drools engine and the BeanShell interpreter into JADE. Section 4 describes a concrete e-learning application where we're exploiting rules and code mobility, and then some more application areas we're going to apply these features to. Finally some overall conclusions are drawn, and possible future development lines are traced.

2 Adaptive rule-based agents

The advantages or rule-based systems over procedural programming environments are well known and widely exploited, above all in the context of business applications. Working with rules helps keeping the logic separated from the application code; it can be moved outside the code and modified by non-developers. Another important advantage is that the logic is not scattered around the whole code, but is centralized in one point, where it can be analyzed and validated. Finally, rule-based engines are often well optimized, and they are able to efficiently reduce the number of rules to match against the updated knowledge base.

Among the different mechanisms to implement a rule-engine, mainly thanks to the high degree of optimization that can be obtained, the Rete algorithm [6] has gained more and more popularity.

Currently, a number of different rule-engines are available, some of which implement the Rete algorithm or one of its variants. Probably the best know of them is Jess [7], developed at Sandia National Laboratories in late 1990s. Jess has always been widely adopted by the JADE community to realize rule-based agent systems, too, and examples of use can be found in the JADE official documentation [12]. But, since this framework is no more licensed as a free open-source package, the necessity to have low cost alternatives is becoming more and more impelling. Our evaluation focused on Drools [3], a well known, freeware tool that implements the so-called Rete-OO algorithm.

Apart of its open-source availability, one of the main advantages of Drools is exactly the fact that it's not just a literal implementation of the Rete algorithm, but rather an adaptation for the object-oriented world. This greatly eases the burden of integrating the rule-engine and the application rules with the existing external objects. In Drools, asserted

facts are simple Java objects, that can be modified through their public methods and properties. Where Jess requires hundreds of lines of code, for example to simply access an ACL message mapped into a Java object, Drools rules can obtain the same result in a dozen of easy-reading code lines. Drools rule can bee specified through an xml file, and they can be expressed using different scripting languages, as Python, Groovy and Java. They can even be added dynamically to the engine through the Semantic Module Framework. Instead Jess only accepts rules written in the CLIPS language. This could require developers to learn a new Lisp-like language and deploy additional efforts to adapt it to their object-oriented development environment.

One of the main advantages of Jess over Drools is the control it provides on the handling of active rules. But this difference is going to disappear, as the last version of Drools added a customizable Conflict Resolution Strategy framework, exactly to fill this gap: the new APIs allow, for example, to select the rules to execute first according to their priority, or to the number of conditions they express.

A lot of work has been done about the use of rules to realize agent systems. On the one hand, rules have been shown suitable to define abstract and real agent architectures and have been used for realizing the so called "rule-based agents", that is, agents whose behaviour and/or knowledge is expressed by means of rules [17,15,10,16]. On the other hand, given that rules are easy and suitable means to realize reasoning, learning and knowledge acquisition tasks rules have been used into the so called "rules-enhanced agents", that is, agents whose behaviour is not normally expressed by means of rules, but that use a rule engine as additional component to perform specific reasoning, learning or knowledge acquisition tasks [9,13].

Both the approaches have some advantages and disadvantages. Rule-based agents provide all the advantages of rule-based systems and a uniform way to program them, but their performance is inadequate for some kinds of applications. Rule-enhanced agents allow the use of different programming paradigms; therefore, it is possible to use the most appropriate paradigm for the realization of the different tasks both to simplify the development and to satisfy the performance requirements, but there is an additional cost for the management of the integration /synchronization of such heterogeneous tasks.

In our system, the rule-engine is integrated into an agent as a JADE behaviour. This approach guarantees both the advantages of full rule-based agents and the ones of rule-enhanced agents. In facts, both procedural and rule-based behaviours can be seamlessly added to each deployed agent, according to the application features and requirements.

Mobile code prove useful in many context [8], thanks to its ability to overcome network latency, reduce network load, allow asynchronous execution and autonomy, adapt dynamically, operate in heterogeneous environments, provide robust and fault-tolerant behaviours. Anyway, a wide range of different technologies are currently available, and all claims to be founded on mobility of code. These technologies vary from applets and other dynamic code downloading mechanisms, to full mobile agent systems, adhering to models as code on demand, remote evaluation, mobile agents.

In our system, two different cases are possible: asking a remote agent to execute a task, or to apply a new rule to its knowledge base. While mobile rules falls into the class of asynchronous requests with deferred execution, instead mobile tasks fall into the synchronous class. In both cases the moved entity is a fragment of code, to be interpreted by a scripting engine on the target agent, and not a complete thread of execution.

Lots of research work has been devoted to analyze the different security threats that a mobile code system could face, and the relevant security countermeasures that could be adopted. In [11] two different classes of attacks can be identified, depending on their target: the ones targeting the executing environment of mobile code, and the ones targeting the code itself.

While the fact that mobile code could pose threats to its hosting environment is widely accepted, instead often the possibility to face threats against the hosted code is not taken into consideration. This is certainly due to a lack of effective countermeasures to prevent the hosting environment from stealing data and algorithms from the mobile code, from executing it too slowly to be effective, altering its execution flow, or stopping its execution. Experimental algorithms exist to at least detect "a posteriori" this type of threats, including partial result encapsulation, mutual itinerary recording, itinerary recording with replication and voting, execution tracing. Some algorithms even try to prevent some types of attacks to the code hosted in malicious environments, but their real effectiveness has yet to be proved; these include environmental key generation, computing with encrypted functions, and obfuscated code (sometimes called time limited blackbox).

Our main effort has instead been devoted in protecting the executing environment hosting mobile code. Potential threats posed by hosted code include masquerading, denial of service, eavesdropping, and alteration. Available security countermeasures to protect the execution environment against potentially malicious mobile code often rely on algorithms to prevent attacks, like software-based fault isolation, safe code interpretation, authorization and attribute certificates, proof carrying code. Other techniques are focused on detecting attacks to the environment and tracing them to their origin; these include state appraisal, signed code, path histories.

In particular, in our system we leveraged on the security means provided by Java, and extended them to allow the definition of precise protection domains on the basis of authorization certificates [14]. These certificates, attached to mobile code, list a set of granted permissions and are signed by trusted authorities according to customizable policies.

The authorization certificates owned by the agents can also be used to delegate access rights to other agents, to allow them to complete the requested tasks or to achieve delegated goals [18]. Finally, masquerading and alteration threats are prevented by establishing authenticated, signed and encrypted channels between remote components of the system.

3 JADE, Drools, BeanShell

The concrete implementation of the proposed system is the direct result of the evaluations exposed in the preceding sections. In particular, we decided to not start from scratch, from the development of a totally new agent platform, but instead we judged that existing solutions demonstrated during the time to be a sound layer on which more advanced functionalities should be added.

The chosen system was JADE. Past experiences in international projects, proved it to be preferable to other solutions, thanks to its simplicity, flexibility, scalability and soundness. As already argued, to the integration of JADE with Jess, yet valid in some contexts, we preferred instead the integration with an open source, object-oriented software, as Drools. To the rich features of Drools, we added the support for communications through ACL messages, typical of FIPA agents. Drools rules can reference ACL messages in both their precondition and consequence fields, which are expressed in the Java language and executed by the embedded BeanShell interpreter. Moreover, a complete support was provided, to manipulate facts and rules on Drools agents through ACL messages.

Inside the Drools environment a rule is represented by an instance of the Rule class: it specifies all the data of the rule itself, including the declaration of needed parameters, the extractor code to set local variables, the pre-conditions making the rule valid, the actions to be performed as consequence of the rule. Rule object can be loaded from xml files at engine startup, or even created and added to the working memory dynamically.

Rules contain scripts in their condition, consequence and extractor fields. The scripts can be expressed using various languages, fore example Python, Groovy and Java. In this last case, the script is executed by the embedded BeanShell engine. When a rule is scheduled for execution, i.e. all its preconditions are satisfied by asserted facts, Drools creates a new instance of a BeanShell namespace, set the needed variables inside it and invokes the BeanShell interpreter to execute the code contained in the consequence section of the rule.

Drools agents expose a complete API to allow the manipulation of their internal working memory. Their ontology defines AgentAction objects to add rules, assert, modify and retract facts. All these actions must be joined with an authorization certificate. Only authorized agents, i.e. the ones that show a certificate listing all needed permissions, can perform requested actions. Moreover, the accepted rules will be confined in a specific protection domain, instantiated according to their own authorization certificate.

Finally, we decided to provide direct access to the BeanShell interpreter, too. BeanShell agents can receive tasks, submitted through ACL messages. As the requests to perform actions contain the code of the task, expressed in the Java language, it is possible to use this feature to implement applications adhering to the remote evaluation model. Moreover, the future integration of advanced grid features, as transparent and reconfigurable load balancing functions, could pave the way for the development of distributed computing environments founded on networks of FIPA agents and platforms.

BeanShell is an application written by Pat Niemeyer that allows to use Java as a scripting language [2]. Usually, the main difference between a scripting language and a compiled one, lies in the handling and control of types. In this sense, BeanShell is a new type of scripting language: it allows the developer not to renounce to type control. In this way, it is possible to write BeanShell scripts that look like Java applications under every degree. But BeanShell allows to relax the type control to different extents, too, making the code more similar to a traditional scripting language. The advantage of BeanShell is therefore to not impose any sort of syntactic barrier between its scripts and real Java code. All this is allowed by the use of the Java Reflection API. In facts, as BeanShell is executed into the same Virtual Machine where the embedding application is executed, programmers are free to work with true Java objects, inserting and extracting them freely from the scripting environment of BeanShell.

In particular, we integrated the scripting engine inside a JADE agent, and provided an API for interacting with it through ACL messages. The FIPA request protocol is used to submit tasks. A specific ontology describes the new AgentAction objects which can be used to submit tasks and to manipulate variables in the BeanShell environment. The code to perform a submitted task is contained into the AgentAction object, in the form of Java statements. If proper permissions are owned, the code will be executed by the embedded scripting engine of the BeanShell agent.

While mobility of rules and code among agents paves the way for real adaptive applications, it cannot be fully exploited if all the security issues that arise aren't properly addressed. The approaches to mobile code security are different, depending on the particular threats that should be faced. In the context of our applications, we decided to leave out the problem of threats of hosting environments against received code. These issues are harder to face, and solutions often rely on detection means, more than prevention ones.

In our work, instead we focused on the problem of receiving potentially malicious code, that could harm the hosting agent and its living environment. For this purpose,

we leveraged on JadeS [14], the security framework that is already available for JADE, to implement two different layers of protection.

The security means we implemented in our system greatly benefit from the existing infrastructure provided by the underlying Java platform and by JADE. The security model of JADE deals with traditional user-centric concepts, as principals, resources and permissions. Moreover it provides means to allow delegation of access rights among agents, and the implementation of precise protection domains, by means of authorization certificates issued by a platform authority.

In the security framework of JADE, a principal represents any entity whose identity can be authenticated. Principals are bound to single persons, departments, companies or any other organizational entity. Moreover, in JADE even single agents are bound to a principal, whose name is the same as the one assigned by the system to the agent; with respect to his own agents, a user constitutes a membership group, making thus possible to grant particular permissions to all agents launched by a single user.

Resources that JADE security model cares for include those already provided by security Java model, including local file system elements, network sockets, environment variables, database connections. But there are also resources typical of multi-agent systems that have to be protected against unauthorized accesses. Among these, agents themselves and agent execution environments must be considered.

A permission is an object which represents the capability to perform actions. In particular, JADE permissions, inherited from Java security model, represent access to system resources. Each permission has a name and most of them include a list of actions allowed on the object, too.

To take a decision while trying to access a resource, access control functions compare permission granted to the principal with permission required to execute the action; access is allowed if all required permissions are owned.

When an agent is requested to accept a new rule or task, a first access protection involves authenticating the requester and checking the authorization to perform the action; i.e.: can the agent really ask to add a new rule, or to perform a given task on its behalf? To perform these tasks, the requester needs particular permissions, i.e. instances of the DroolsPermissions and BshPermission classes. A DroolsPermission object can authorize the execution of requests as add or remove rules or add, remove and manipulate facts. A BshPermission object can authorize the execution of requests as submit a task or remotely set or cancel a variable.

So, only authenticated and authorized agents can successfully ask another to accept rules and tasks. But till this point the security measures don't go further than what other technologies, like ActiveX, already offer. In facts, once the request to perform a given task is accepted, then no more control on the access to protected resources can be enforced. The agent can choose to trust, or not to trust. But, if the request is accepted, then the power of the received code cannot be limited in any way.

Instead, to deploy the full power of task delegation and rule mobility, the target agent should be able to restrict the set of resources made accessible to the mobile code. The agents should be provided means to delegate not only tasks, but even access rights needed to perform those tasks. This is exactly what is made possible through the security package of JADE, where distributed security policies can be checked and enforced on the basis of signed authorization certificates.

In our system, every requested action can be accompanied with a certificate, signed by a known and trusted authority, listing the permissions granted to the requester. Permissions can be obtained directly from a policy file, or through a delegation process. Through this process, an agent can further delegate a set of permissions to another agent, given the fact that it itself can prove the possession of those permissions.

The final set of permissions received through the request message, can finally be used by the servant agent to create a new protection domain to wrap the mobile code during its execution, protecting the access to the resources of the system, as well as those of the application.

4 An Application

A first exploitation of the implemented framework has to do with the development of a multi-agent system to support the automatic generation of courses. In the following sub-sections a brief description of the system and of its future evolution are reported.

The development of electronic course material and the consequent need to keep the content up to date takes much effort and is time consuming. Therefore it is getting vital to provide a valuable support for teachers. Learning objects, on the other hand, appear to have significant potential for creating highly personalized learning programs and easily updated courses.

The aim of our application - learning objects on demand, is to provide an infrastructure to support the automatic search of learning objects. This is accomplished by exploiting the rule-based agent, described in the previous sections, acting as an intermediary agent having the role of a broker, in order to support a matching between the learning objects, appropriately annotated with metadata, and the users' preferences. By means of this approach, we intend to achieve a high degree of reusable course content as well as a reduction in costs for courses development.

This application is part of the TechNET [19] project, funded by the European Community and started in September 2003. This project addresses the key area of Education and Cultural Heritage within the @LIS call, demonstrating a highly innovative teaching and experimentation environment spanning across 8 countries.

The environment functions as a live continuously running network – enabling students, learning professionals and researchers to gain hands-on experience of using cutting edge Web/Internet technologies to create complex dynamic on-line applications. In particular the demonstrator will include a lead application example that will provide personalized wire line and wireless teaching services spanning 5 Latin American and 3 European countries.

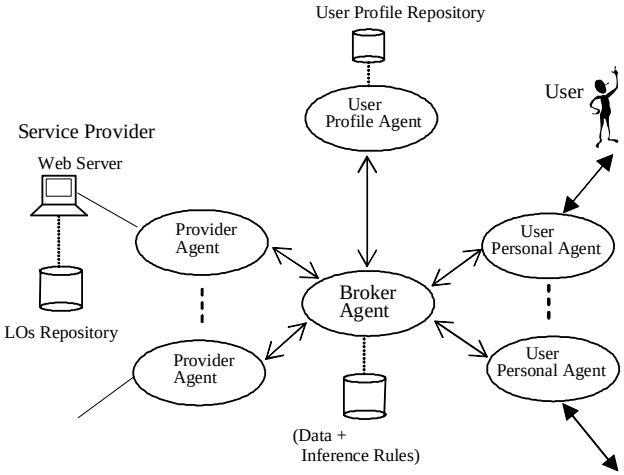

Fig. 1. "Learning Objects on Demand" system architecture

The project is in its first phase of a multiphase effort. Our aim, in this first phase, is to develop a simplified version of which will be the final system

The model expects that several components of the same type can coexist in the system. The current version of the demonstrator is based on a single Intermediary, a small number of Service Providers and Personal Agents. Obviously users in turn can play the role of services providers and vice versa, but in this initial basic version of the demonstrator this eventuality has not been considered.

The core of the system is represented by the Broker Agent, a rule-based agent which is in charge of implementing a middle layer able to match properties of the learning objects with interests and demands of the users.

The Service Provider is responsible for providing the description of each LO. In this first phase much effort has been spent in conceptualize and design LOs. Since the broad acceptance of SCORM [1] as a de facto standard for content creation and distribution, each LO has been annotated according to the SCORM Meta-data Information Model. We decided to use six of the nine categories of meta data elements; more specifically the following categories: general, technical, educational, relation, classification and finally rights. The latter is of particular importance in order to guarantee the intellectual property and to regulate the conditions of use for the resource.

Each service provider is represented into the agent community by its Provider Agent. We have implemented a Provider Agent that represents its LO service. It is connected through the Internet to a Web server that allows querying the Database with the description of the LO.

LO metadata are described and published to the Broker Agent, by sending it a FIPA ACL message that request to register an object with the Broker Agent. In the registration message it is mandatory to declare values for the following three attributes: identifier, title, description and keywords, where the semantics of this last attribute is a list of keywords that help classifying the LO. Optionally, other attributes can be registered according to the chosen SCORM categories. If no meta-data elements belonging to the "rights" category are specified, this means that no copyright or other restrictions apply to the use of this resource.

The intermediary is a third party between users and providers. In the demo, it provides a personalization service to the user through the learning of its profile (User Profile Agent) and the matchmaking between user preferences and provider capabilities (Broker Agent). Overall, this functionality is implemented through the collaboration between two agents.

The rule-based engine, that is the core of the project, has the ultimate goal to help the user to retrieve the information he is looking for. Therefore, at a first guess, there should be no need for a relevance measure, in the sense of classical keyword-based search engines, to filter out unwanted results. However the user may provide generic preferences, which will have as outcome a large amount of information with no possibility of saying that some results satisfy better the search criteria than others, so that we could sort the list of answers. It is obvious that objective relevance measures (that do not take into account the user's context), like the keyword frequency in keyword-based search engines, will not be of much help in our case. In this case, an interactive negotiation with the user will take place to refine the set of answers before sending them back, or a subjective relevance measure may be used to rank the results.

The User Profile Agent, in charge to carry out this task, gives to the intermediary very powerful tools that enable monitoring of changes in users preferences and adaptation of offers. It maintains a profile of each registered user and by observing the behaviour of the user, the agent filters the LOs according to the learning user preferences.

The User Profile Agent has been designed to function both in the "pull" and "push" mode. In the pull mode, it acts as a mediator in the query processing cycle, using user and context dependent information to filter and sort the results. While in the push mode, as soon as the confidence on ranking of documents is high enough, the same engine proposes the new LOs to the user for which the predicted ranking passes above a certain threshold.

The user is able to switch off the adaptive behaviour, that is he can render inoperative the activity of the UPA, and return to a static model. If he goes for the adaptive behaviour, the Broker Agent will receive two kinds of rules: "temporary rules", sent by the Personal Agent and the User Profile Agent in order to look for interesting LOs, and which will be discarded at the end of the search process; "permanent rules", sent by the User Profile Agent as a

consequence of the user registration. The last kind of rule is concerned with the general interest of the user and then they will be removed when the user decides to deregister himself/herself. In the first version of the system two kinds of users, with different rights, have been considered, that is teachers and students.

The JADE Agent Platform provides the middleware necessary to manage the agents, distribute them on several hosts, and implement the communication mechanism. JADE enables full scalability of the multi-agent system and several different configurations can be selected, as necessary.

For the successive phases of the project, in order to improve our system in terms of effectiveness, scalability and better distribution of the workload, we envisage a network of distributed broker agents, each dedicated to specific interest areas, possibly having common characteristics. One of the most challenging aspects of this evolution is to ensure that the work carried out by different brokers results in a coherent whole. To address this, we will consider a different role for the user personal agent, which will be in charge of collaborating with several broker agents in order to find the needed teaching materials. We identified goal delegation [18] from the personal agent to the broker agents or from the UPA to the Broker Agent (if the adaptive behaviour is chosen), as the key mechanism to reach a complex goal such as the organization of a courseware. The PA/UPA will decompose the global goal into subgoals and will assign them to the broker agents. The goal for the broker will be defined in terms of rules.

5 Conclusion

This paper described the integration of BeanShell, a scripting engine for the Java language, and Drools, an object-oriented rule-engine, with JADE, a FIPA-compliant agent development framework. The resulting system joins the soundness of JADE as a platform for distributed multi-agent systems, with the expressive power of rules and the ability to adapt to changing conditions granted by mobile code.

Of course, the development of real world applications poses serious security requirements, which can be faced by means of detailed security policies and delegation of authorizations through signed certificates. Application areas include, but certainly are not limited to, e-learning, e-business, service-composition, network management.

The development of advanced grid features, as transparent and dynamic load balancing functions, will add even greater value to the system, making it suitable to realize smart and distributed computing environments..

References:

[1] ADL SCORM, Advanced Distributed Learning Initiative, Sharable Courseware Object Reference Model (version 1.2). Available at ADLNet http://www.adelnet.org.

[2] Beanshell Home Page. 2004. Available from http://www.beanshell.org.

[3] Drools Home Page. 2004. Available from http://www.drools.org.

[4] Eberhart. OntoAgent: A Platform for the Declarative Specification of Agents, In Proc. of ISWC 2002, Cagliari, Italy, 2002.

[5] Foundation for Intelligent Physical Agents Specifications. Available from http://www.fipa.org.

[6] Forgy, Charles L., "Rete: A Fast Algorithm for the Many Pattern / Many Object Pattern Match Problem", Artificial Intelligence 19(1), pp. 17-37, 1982.

[7] E.J. Friedman-Hill. Jess, the Java Expert System Shell. Sandia National Laboratories. 2000. http://herzberg.ca.sandia.gov/jess.

[8] A. Fuggetta, G.P. Picco, G. Vigna, Understanding code mobility, IEEE Transaction on Software Engineering 24 (5):342–362, 1998.

[9] O. Gutknecht, J. Ferber, F. Michel. Integrating tools and infrastructures for generic multi-agent systems. In Proc. of the Fifth Int. Conf. on Autonomous Agents. Montreal, Canada, 2001

[10] K.V. Hindriks, F.S. de Boer, W. van der Hoek, J.C. Meyer. Control Structures of Rule-Based Agent Languages. Proc. ATAL-98, Paris, France, 1998.

[11] W. Jansen, T. Karygiannis. Mobile agent security. NIST Special Publication 800-19.

[12] JADE Home Page, 2004. Available from http://jade.tilab.com.

[13] E.P. Katz. A Multiple Rule Engine-Based Agent Control Architecture Technical Report HPL-2001-283. HP Laboratories Palo Alto - Software Technology Laboratory. 2002.

[14] A. Poggi, G. Rimassa, M. Tomaiuolo. Multi-user and security support for multi-agent systems. In Proc. WOA 2001: 13-18. Modena, Italy. 2001.

[15] A.S. Rao. AgentSpeak(L): BDI Agent Speak Out in a Logical Computable Language. In W. van der Velde ansd J.W. Perram (eds.), Agents Breaking Away, pp.42-55, 1996.

[16] M. Schroeder, G. Wagner. Vivid agents: Theory, architecture, and applications. Int. Journal for Applied Artificial Intelligence, 14(7):645-676, 2000.

[17] Y. Shoham. Agent-oriented programming. Artificial Intelligence, 60(1):51-92. 1993.

[18] M. Somacher, M. Tomaiuolo P. Turci. Goal Delegation in Multiagent System. AIIA 2002. Siena, Italy. 2002.

[19] TechNET project. Home Page available at http://www.alis-technet.org