Multi-Agent System Design Based on Security Policies

Zeus Andrade Zaldívar, Ma. De los Ángeles Junco Rey, Jorge Adolfo Ramírez Uresti, José Arturo Tejeda Gómez, Enrique David Espinosa Carrillo

Instituto Tecnológico y de Estudios Superiores de Monterrey campus Estado de México.
Atizapán de Zaragoza, México.
{zandrade, ajunco, juresti, jtejeda, enrique.espinosa} @itesm.mx

Abstract. Security plays a major role in modern computer systems and multi-agent systems are not an exception. Security and multi-agent systems have been related on several works but little work has been done in relating security with agent oriented software engineering. In this paper we discuss the use of security policies to guide the design process for multi-agent systems and describe extensions to the GAIA methodology for this purpose. The proposed extensions are illustrated with the design of a file sharing system based on the MLS security model.

1 Introduction

Nowadays multi-agent systems provide a good alternative for the development of large and complex systems. Multi-agent systems are particularly useful when the system aims to recreate a real world organizational structure and when the presence of autonomous, intelligent, proactive, learning capable and social entities is needed to interact in distributed systems.

The majority of modern systems operate on open environments where they are susceptible to attacks, this makes necessary to take security as one of the more important elements of any modern system. A lot of research has related security and multi-agent systems; however, most of it had been focused on the final stages of the system development process, especially when the system is already implemented or even deployed. Identifying security requirements based on the system behavior instead of controlling the system behavior according to the security requirements is a hard task and commonly leads to the introduction of vulnerabilities in the system [1], [2], [3], [4].

Little work has been done focusing on the incorporation of security through the whole process of the system development. Mouratidis in [5] proposes extensions to the TROPOS methodology to accommodate security. Following this approach, in this paper we propose extensions to the GAIA methodology [6], [7] to accommodate security restrictions and we present recommendations to guide the design process according to security policies.

2 Extending the GAIA methodology

Security is established through policies and mechanisms that ensure integrity, privacy and availability of the resources that can be provided, required or present in the system. Security policies for multi-agent systems could be seen as sets of rules restricting the relationships between the system elements in terms of privacy, integrity and availability. The more important system elements are: Agents, users and resources.

There may be a lot of different types of rules according to the kind of entities they restrict or the properties they are based on. Rules can force an entity to perform an action, make an action triggers another, set conditions for an action, control the way an action is performed, control the number of entities and resources, or restrict the system behavior in many other ways. Thousands of different kinds of rules can be defined according to the specific needs and requirements of a system.

It is not realistic to think in a design methodology that takes all possible kinds of rules. For this reason in this work we take under consideration a limited set of rule types:

1. Restrictions on the effective capabilities of entities. Conditions that must be fulfilled to enable an entity to perform an action. These restrictions cover any kind of action performed by an agent or user such as reading, writing, creating or consuming resources and create, suspend, resume, migrate, and terminate an agent.
2. Restrictions over the interaction between entities. Conditions that must be fulfilled during the interaction of two entities.
3. Restrictions on the properties or attributes of entities. Conditions that restrain the attributes of an entity and that must be always fulfilled.

This set includes rules to represent and model policies like the multi level security [8] and the commercial model [9]. These models are representative on computer security, so the set of rules to represent them are also representative.

To illustrate the use of security policies on the design of multi-agent systems we use as a case of study a hypothetical distributed file access system based on Bell and LaPadula multi-level security model [8] (military security model). This security model is summarized in the following section.

2.1 Multi level security

This model establishes rules for two kinds of entities: objects, which are passive entities and cannot perform any activities, and subjects, which are active entities performing actions and accessing resources (objects). The model can be represented by the following rules:

R1. Subjects have only one level.
R2. Objects have only one level.
R3. All subjects have a set of labels.
R4. All objects have a set of labels.
R5. For every object there is a set of permissions for writing and reading.
R6. Reading is allowed if the subject’s level is greater than or equal to the object’s level and the subject’s label set is a sub-set of the object’s label set.

R7. Writing is allowed if the subject's level is lesser than or equal to the object's level and the subject's label set is a sub-set of the object's label set.
R8. An authentication mechanism is necessary
R9 . All subjects must be authenticated before accessing the system.

The primary objective of this model is to assure privacy. Through the mandatory and discretionary controls the model guaranties that information can be accessed only by authorized entities. Authorization is contained on the levels and labels on every entity.

2.2 Designing the system

The system to be consists of a shared file system based on agents and the multi-level security model. Although this system does not reflect the real complexity of most modern multi-agent systems, it has been chosen as a case of study because its simplicity allows a simple illustration of the design process based on security policies.

In the following sections we present the process of design for this system according to the phases established by the GAIA methodology and incorporating extensions and guidelines to preserve the security requirements (the security policy).

2.3 The analysis

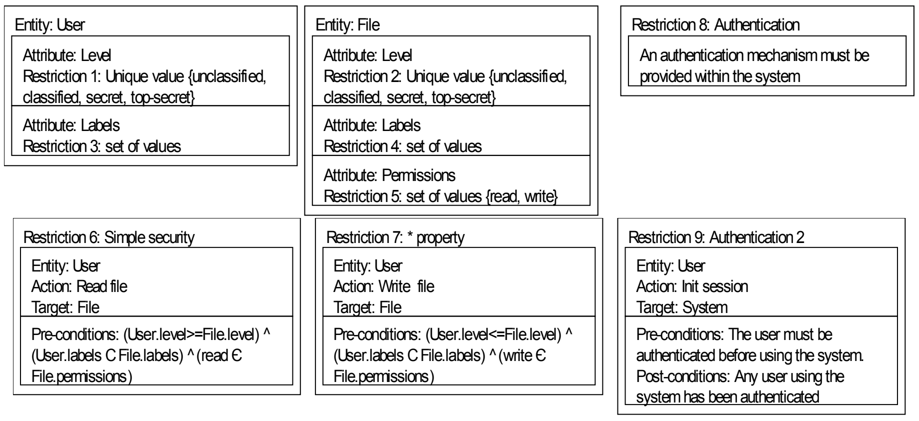
In this phase of the GAIA methodology, the system specification is modeled through the identification of the organizational structure and the specification of preliminary roles, environment and interaction models. The security policy is part of the system specification so we propose to model it on this phase.

The first stage of this phase is to establish an organizational structure for the agents on the system. In many cases the security policy will define explicitly or implicitly the organizational structure or structures that should be used. For the case of study we identify a single organizational structure.

The GAIA methodology uses only the system requirements to identify and design roles, protocols and the environment, the first extension we propose to this methodology is to use a security policy to identify those elements. This is done through two new types of models, a restrictions model, which represents the security policy, and a behavior model, which models the system behavior restricted by the security policy.

The security policy restricts entities, so it must be modeled before modeling entities into roles and resources. The restrictions model must present all the rules or restrictions of the policy and the relationship between the rules and the entities (active and passive) of the system. The restriction model for the case of study is shown on figure 1.

The restrictions model presents all entities, resources, actions and interactions restricted by the security policy but there may be other elements present in the system requirements which are not restricted. The behavior model should represent graphically all restricted and unrestricted elements. On figure 2 and 3 we present a graphical notation and a behavior model for the case of study.



Entity: User

Attribute: Level
Restriction 1: Unique value {unclassified, classified, secret, top-secret}

Attribute: Labels
Restriction 3: set of values

Entity: File

Attribute: Level
Restriction 2: Unique value {unclassified, classified, secret, top-secret}

Attribute: Labels
Restriction 4: set of values

Attribute: Permissions
Restriction 5: set of values {read, write}

Restriction 8: Authentication

An authentication mechanism must be provided within the system

Restriction 6: Simple security

Entity: User
Action: Read file
Target: File

Pre-conditions: (User.level>=File.level) ^ (User.labels C File.labels) ^ (read Є File.permissions)

Restriction 7: * property

Entity: User
Action: Write file
Target: File

Pre-conditions: (User.level<=File.level) ^ (User.labels C File.labels) ^ (write Є File.permissions)

Restriction 9: Authentication 2

Entity: User
Action: Init session
Target: System

Pre-conditions: The user must be authenticated before using the system.
Post-conditions: Any user using the system has been authenticated

Fig. 1. Preliminary restrictions model

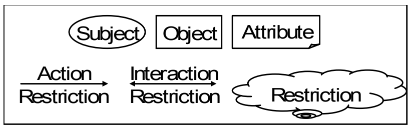


Fig. 2. Graphical notation for the behavior model

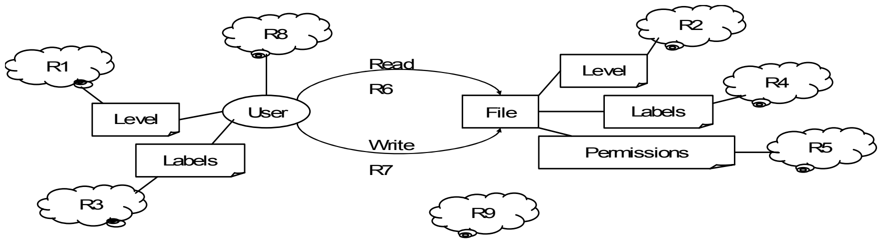


Fig. 3. Preliminary behavior model

Once we have established the restrictions and behavior models we can use them, along with the system specification, to generate the environment, roles and interaction models. Roles can be identified in the behavior model as subjects, resources as objects and interactions as double headed arrows connecting subjects.

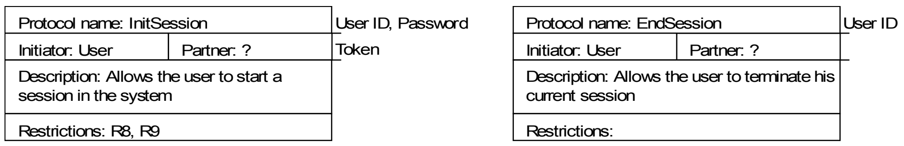
An environment model, as proposed in [7], for the case of study is presented below:

```
READS: File, Users, Sessions
CHANGES: File, Sessions
```

The role model, as defined on [7], presents roles along with their description, actions, protocols, permissions and responsibilities. Permissions allow the designer to express the effective capabilities associated to the role and the safety responsibilities allow express conditions (invariants) that must always be maintained. However these two elements of the role schema are not useful to describe restrictions over the actions that the entity can perform. The second proposed extension is to use references to the restrictions defined on the restriction model instead of using permissions and safety responsibilities to describe the effective capabilities of the role. This extension does not apply to roles that are not restricted by the security policy. The following schema represents the role for the case of study.

Role schema: User
Description: Access the files in the system.
Protocols and Activities: InitSession, ReadFile, WriteFile, EndSession
Responsibilities: Liveness: USER = InitSession.(ReadFile | WriteFile)*.EndSession
Restrictions: R1, R3, R6, R7, R9

The interaction model consist of a set protocols defined by the interactions between roles. The interactions can be identified on the restrictions and behavior models, these interactions must be modeled as protocols. Restrictions over interactions are referenced in the schema for the protocols. The third proposed extension is to aggregate restriction references to the protocol schemas as defined in [7]. The interaction model for the case of study is shown in figure 4.



Protocol name: InitSession		User ID, Password
Initiator: User	Partner: ?	Token
Description: Allows the user to start a session in the system		
Restrictions: R8, R9		

Protocol name: EndSession		User ID
Initiator: User	Partner: ?	
Description: Allows the user to terminate his current session		
Restrictions:		

Fig. 4. Preliminary interaction model

The security policy, captured with the restrictions and behavior models define the security requirements of the system. The environment, role and interaction models are build parting from the restrictions and behavior models, the protocol and role schemas include references to the restrictions applied to the element they represent.

The last stage of the analysis is to establish the organizational rules. We propose only one recommendation for this stage, security restrictions that are not concerned to entities, their attributes, actions and interactions can and should be modeled as organizational rules. Organizational rules for the case of study are presented below.

```
IntiSession (User)1
IntiSession (User) → ReadFile (User)
IntiSession (User) → WriteFile (User)
IntiSession (User) → EndSession (User)
```

The GAIA methodology establishes as output of the analysis phase the preliminary models of environments, roles and interactions; we propose to add the preliminary models of restrictions and behavior (called preliminary because they are to be updated in the following phase).

2.4 The architectural design

This phase is focused on the establishment of the final structure for the system. On this phase all preliminary models are refined until they become final models.

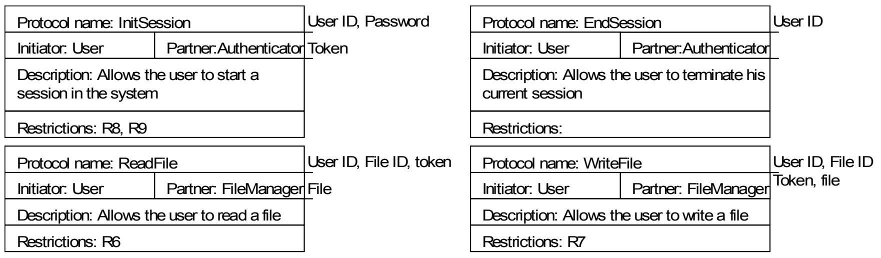
The first stage of this phase is to define the organizational structure for the system. Security requirements (policies) diverge greatly from one system to another so the use of organizational patterns, as described in [7], may be restricted to systems that adopt the same security policy. The organizational structure for the case of study is presented underneath.

$\forall i,\ \text{User} \xrightarrow{\text{depends on}} \text{Aunthenticator}$ $\forall i,\ \text{User} \xrightarrow{\text{depends on}} \text{FileManager}$

$\forall i,\ \text{User} \xrightarrow{\text{peer}} \text{User}$ $\forall i,\ \text{Authenticator} \xrightarrow{\text{peer}} \text{FileManager}$

The next stage on the architectural design is to complete and refine the role model. The GAIA guidelines should be used during this stage; this means that new roles (organizational roles) must be added following the organizational structure, the new roles must be evaluated to verify that all restrictions on the security policy are fulfilled. The fourth proposed extension is the definition of security roles, this kind of roles are not explicitly needed by the organizational structure or the security policy but that are needed to satisfy some restrictions. Security roles can be identified through the protocols for which one or both entities (initiator and partner) were not identified in the preliminary interactions model, as an example, in the case of study we defined the protocol InitSession, which is restricted, in the preliminary interactions model the partner for the entity user was not identified and this identifies the need for a security role. Also, restricted actions like ReadFile and WriteFile imply the existence a security role which will be in charge of verifying the fulfillment of the restrictions associated to these actions. Below we present the role schemas for the new roles, which along with those defined on the preliminary role model integrate the final role model.

Role schema: Authenticator Description: Authenticates users trying to acces the system and manages a list of current users Protocols and Activities: InitSession, AuthenticateUser, EndSession Responsibilites: Liveness: AUTHENTICATOR = ((InitSession.AuthenticateUser) | EndSession)$^{\omega}$ Restrictions: R8, R9
Role schema: FileManager Description: Controls the access to the files in the system. Protocols and Activities: ReadFile, WriteFile Responsibilites: Liveness: FILEMANAGER = (ReadFile | WriteFile)$^{\omega}$ Restrictions: R6, R7



Protocol name: InitSession		User ID, Password
Initiator: User	Partner:Authenticator	Token
Description: Allows the user to start a session in the system		
Restrictions: R8, R9		

Protocol name: EndSession		User ID
Initiator: User	Partner:Authenticator	
Description: Allows the user to terminate his current session		
Restrictions:		

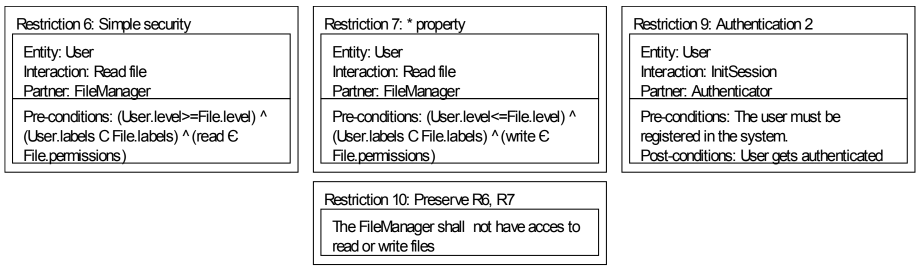
Protocol name: ReadFile		User ID, File ID, token
Initiator: User	Partner: FileManager	File
Description: Allows the user to read a file		
Restrictions: R6		

Protocol name: WriteFile		User ID, File ID Token, file
Initiator: User	Partner: FileManager	
Description: Allows the user to write a file		
Restrictions: R7		

Fig. 5. Interactions model

The next stage is to refine the interactions model. The GAIA methodology provides the necessary guidelines to perform this action. New protocols may be added during this stage and it must be verified that they comply with all security restrictions present on the policy. In figure 5 we present the protocol schemas that complete the interactions model.

The aggregation of new roles and protocols could introduce inconsistencies with the security policy, thus a review and refinement of the restrictions and behavior models is necessary. The refinement of those models requires to evaluate all security restrictions and to modify them or create new restrictions to obtain a new security policy equivalent to the original but including all new roles and interactions. The refined models of restrictions and behavior are shown in figures 6 and 7.



Restriction 6: Simple security
Entity: User
Interaction: Read file
Partner: FileManager
Pre-conditions: (User.level>=File.level) ^ (User.labels C File.labels) ^ (read Є File.permissions)

Restriction 7: * property
Entity: User
Interaction: Read file
Partner: FileManager
Pre-conditions: (User.level<=File.level) ^ (User.labels C File.labels) ^ (write Є File.permissions)

Restriction 9: Authentication 2
Entity: User
Interaction: InitSession
Partner: Authenticator
Pre-conditions: The user must be registered in the system.
Post-conditions: User gets authenticated

Restriction 10: Preserve R6, R7
The FileManager shall not have acces to read or write files

Fig. 6. Refinement of the restrictions model

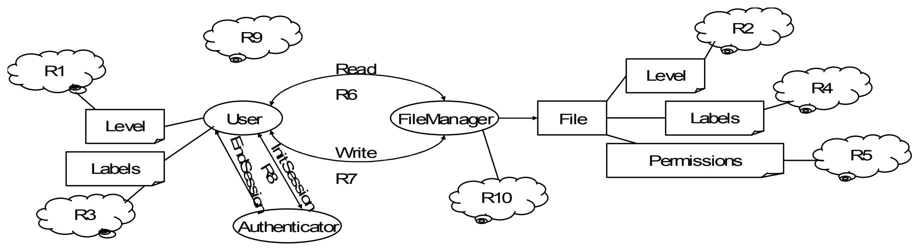


Fig. 7. Refinement of the behavior model

The design process must iterate on this phase until the roles, interactions, restrictions and behavior models become fully compliant with the system requirements and especially with the security policy.

2.5 The detailed design

On this phase the main objective is to find a match between agents and roles. Security restrictions play an important role during this phase because in many cases they define if it is possible for a single agent to play different roles. For our case of study, the agent model is presented below.

UserAgent $\xrightarrow{\text{plays}}$ User
FileSystem $\xrightarrow{\text{plays}}$ FileManager, Authenticator

The fifth proposed extension is the inheritance of restrictions from the roles to the agent classes. Agent classes are designed to play one or more roles, so an agent class should inherit restrictions form the roles it plays. It is necessary to verify the application all restrictions, this can be done through a table containing all entities (agents and objects), their restrictions and how the restrictions will be applied. Table 1 presents these elements for the case of study.

Entity	Agent	Restriction	Aplication
User	UserAgent	R1	The attribute level restriction must be applied to the agent.
		R3	The attribute labels restriction must be applied to the agent.
		R9	The agent must perform an authentication procedure.
Authenticator	FileSystem	R8	The agent must provide an authentication mechanism.
		R9	The agent must authenticate any UserAgent agents and deliver a token if the authentication is succesfull.
FileManager	FileSystem	R6	The agent must verify the restriction before delivering the file to the UserAgent.
		R7	The agent must verify the restriction before replacing the file obtained by the user.
		R10	This agent should not be able to read or write files by it self.
File		R2	The attribute level and it's restriction must be applied to the object.
		R4	The attribute label and it's restriction must be applied to the object.
		R5	The attribute permissions and it's restriction must be applied to the object.

Table 1. System restrictions table.

The next step on the detailed design is to build a services model, on this model all activities, actions and interactions are detailed. This model can be constructed following the guidelines provided by the GAIA methodology. For the case of study, the services model is shown in table 2.

Service	Pre-conditions	Post-conditions	Inputs	Outputs
StartSession	The user must be registered in the system	The user gets a token and starts a session	User id and a password	A token for user
ReadFile	The token must be correct, the file must exist and R6 must be fulfilled	The file is readed by the user	User id, file id and a token	The file to read
WriteFile	The token must be correct and R7 must be fulfilled	The file is written by the user	User id, file id, new file and a token	None
EndSession	The user must have an open session in the system	The user session is terminated and the token invalidated	User id	none

Table 2. Services model

The detailed design is the simplest phase of the methodology, however, the implementation of the systems relays completely on the output of this phase, this implies that if any of the restrictions were lost during the previous phases or in this one, the implementation of the system will not be compliant with the security policy.

It is necessary to keep on the output of this phase the restrictions and behavior models because they are useful to guide the implementation process towards the compliance with the security policy.

2.6 The designed system

The obtained design consists on the final models: agents, services, restrictions and behavior along with the system restrictions table, the organizational rules and the organizational structure.

The security policy is present in all models. In the system restrictions table, every agent class has a set of restrictions inherited from the roles it plays and that must be implemented on the system. In the same way, all protocols have a set of restrictions to fulfill. The organizational rules preserve the rest of the security restrictions not directly associated with a role or protocol.

2.7 Implementing the designs

The GAIA methodology does not deal with the implementation process and being the basis for this work, the extensions proposed are limited to the design process.

There are many different agent frameworks which allow the creation and deployment of multi-agent systems. A combination of GAIA with the Jade framework [10] has been proposed in [11] and [12]. Jade also provides a security add-on based on rules [13] which is suitable for the implementation of systems designed following the extensions proposed.

Agent classes in the GAIA agent model can be directly mapped to Jade agent classes and its actions, protocols and services as Jade behaviors to be adopted by agents.

The security add-on for Jade includes features like authentication, permissions and secure messaging between agents. Authentication allows users to authenticate themselves with the system, secure messaging allows the use of cryptography to sign or encrypt messages during agent interactions and permissions allow or deny users and agents to perform certain actions like creating, killing, pausing or terminating an agent, sending messages to certain agents, accessing certain java classes, etc.

In most of the cases, rules from the restrictions model can be directly translated into rules from the Jade security add-on.

A software tool named MASSD has been created to allow an easy way to implement systems designed as proposed here. The application allows the user to generate all models here described and finally to automatically generate Jade code to implement agents and services.

3 Conclusions and future work

Security is one of the major issues to address when developing a multi-agent system. The overall system behavior must be controlled according to the security requirements of the system. On this paper we proposed the use of security policies to guide the design process of a multi-agent system and we presented a set of guidelines, recommendations, models, schemas and extensions to the GAIA methodology for this effect. We also established a relationship between the security policy, its restrictions and the elements of multi-agent systems and the necessary elements for modeling security restrictions within the GAIA methodology.

A system designed following the guidelines proposed and using the extensions described should be compliant with the security policy used. This contributes to reduce vulnerabilities in the system architecture. A real world test case for the benefits obtained with this methodology is part of the future work.

The presented work is focused on a representative set of rules or restrictions that may be present on a security policy, a generalization of the restriction types, the incorporation of validation mechanism to probe the preservation of the security policy and the study of different security models to establish security organizational patterns are part of the future work.

The systems designed following this work can be easily implemented using the Jade framework along with its security add-on. A software tool for automatic Jade agent code generation has been developed.

The work is being applied to generate a secure system architecture for the @lis technet project [14].

References

1. Lhuillier, N., Tomaiuolo, M., Vitaglione, G. Security in Multiagent Systems, JADE-S goesdistributed. exp: in search for innovation, vol 3. no. 3. Septiembre de 2003.
2. Jim Tam, J., Titkov, L., Neophytou, C. Securing Multi-Agent PlatformCommunication. Department of Electronic Engineering, Queen Mary, University of London.
3. Noordende, G., Brazier, F., Tanenbaum, A. A security framework for a mobile agent system. Ámsterdam University.
4. Kagal, L., Finin, T., Joshi, A. Developing Secure Agent Systems Using Delegation Based Trust Management. University of Maryland.
5. Mouratidis, H., Giorgini, P., Manson, G. Modelling secure multiagent systems. Proceedings of the AAMAS conference. 2003.
6. Zambonelli, F., Jennings, N., Wooldridge, M. Developing multi-agent systems: the gaia methodology. ACM transactions on software engineering and methodology. 2003.
7. Wooldridge, M., Jennings, N. R., Kinny, D. The Gaia methodology for agent oriented analysis and design. Journal of Autonomous Agents and Multi-Agent Systems, 3 (3):285–312, 2000.
8. Bell, D. E. LaPadulla, L. Computer security model: Unified exposition and multics interpretation. Technical report ESDTR-75-306. 1975.
9. Clark, D. D. Willson, D. R. A comparisson of militar and commercial security policies. Proceedings of the IEEE sysmposium on security and privacy. 1987.
10. Java Agent Development Framework. http://jade.cselt.it
11. Moraitis, P., Spanoudakis, N. Combining Gaia and JADE for Multi-Agent Systems Development. 4th International Symposium "From Agent Theory to Agent Implementation" (AT2AI4), in: Proceedings of the 17th European Meeting on Cybernetics and Systems Research (EMCSR 2004), Vienna, Austria, April 13 - 16, 2004.
12. Moraitis, P., Petraki, E., Spanoudakis, N. Engineering JADE Agents with the Gaia Methodology. In R. Kowalszyk, et al. (eds), "Agent Technologies, Infrastructures, Tools, and Applications for e-Services", LNAI 2592, Springer-Verlag, 2003, pp. 77-91.
13. Java Agent Development Framework. Jade Security Guide. 2005.
14. @lis technet project. http://www.alis-technet.org.