MAMSY: A management tool for multi-agent systems*

Victor Sanchez-Anguix, Agustin Espinosa, Luis Hernandez and Ana Garcia-Fornes

1 Introduction

Over the last few years multi-agent systems have proved to be one of the most productive areas for Artificial Intelligence researchers. There are still several subjects that are under research, i.e security, agent programming languages and core enhancements for efficient and scalable systems. Although some advancements have been made in this area, most multi-agent systems still remain under the *closed multi-agent system* category, where the whole system is owned by a private user or organization. One of the main goals in multi-agent research is producing *open multi-agent systems*. According to [15], an *open multi-agent system* is a system where elements can be removed or can be added during the runtime. We are more interested in a specialization of such systems: *open multi-agent systems* where the whole system may not be owned by a private user or organization. From now on we will refer to this system category.

As people extend the system, they will increasingly become interested in monitoring and controlling their part of the system. This is where the term *management* comes in, having been commonly used in software engineering where management systems are the most frequent type of application. The term *management* acquires a special meaning when it is applied to multi-agent systems. In multi-agent system management, the items that are managed are resources. To our knowledge extent, there is no formal definition for the term *resource* in multi-agent systems thus we decided to provide one.

Definition 1. A resource is a physical or virtual object which has an owner, and therefore needs to be administered according to its owner's goals.

e-mail: sanguix@dsic.upv.es, aespinos@dsic.upv.es, lhernand@dsic.upv.es, agarcia@dsic.upv.es
Universidad Politecnica de Valencia, Departamento de Sistemas Informaticos y Computacion, Grupo Tecnologia Informatica Inteligencia Artificial, Valencia Camino de Vera 46022, Spain

* This work was supported by CONSOLIDER-INGENIO 2010 under grant CSD2007-00022 and Spanish goverment and FEDER funds under TIN2008-04446 project.

Physical resources are mainly *node machines or nodes* where the multi-agent system is being is executed. Obviously, administrators are interested in seeing how their machines are being used, which agents are being executed, current system overload and so forth. Administrators may want to modify their state when it is not desirable. It is also normal practice to associate the term *resource* with something physical but in multi-agent systems it is also possible to find resources which are not physical. These are called virtual resources. From the administrator's point of view, agents are virtual resources that he himself has placed in order to achieve certain goals.

Since research into multi-agent system management is still in its infancy stages, there is neither an agreed definition for the term *multi-agent system management*. It is possible to find some works about management related issues at [12], but no formal definition has been given. In fact, to our knowledge extent, [12] is the only work related to multi-agent management. In accordance with our research and in order to help future works we define the term *multi-agent system management*.

Definition 2. Multi-agent system management consists in the visualization and control of the physical and virtual resources located in a multi-agent system.

As we stated before, resources in an *open multi-agent system* may have different owners with perhaps different objectives and interests. It is obvious that owners will be interested in what actions can be performed remotely over their resources by other entities or users. A user is an owner who places virtual or physical resources into the multi-agent system. Since our goal is *open multi-agent systems* it is possible to find in the system users with opposing goals and even untrusted users which only goal is to attack the system. It is not advised to deploy an open multi-agent system which has not treated properly these issues. Serious problems arise with the authentication of users in these highly dynamic and open systems, so associating authenticated owners to resources is still a problem. In fact, authentication is one of the main lines of research currently available in multi-agent systems. Once we have authenticated owners and resources, it is necessary to set permissions for these resources, always bearing in mind the highly variable set of users we may find in a multi-agent system.

But permissions and security are not the only problems management in multi-agent systems may face. For instance, the amount of information and data in *open multi-agent systems* may vary from a small set to very large sets of data. Management tools should show data in a friendly and clear way so users take full advantage of the developed tools and perform their tasks efficiently. User interface research has already provided solutions for most of these inconveniences and their application here is highly encouraged.

MAMSY is a management tool for multi-agent systems hosted by Magentix[23][5] platforms. In the following sections we will will analyze current management tools briefly in order to point their common features. After that we will describe Magentix platform briefly and then give a more detailed description of MAMSY architecture and its main features. Finally, we will give some conclusions and describe our future lines of work in management for multi-agent systems.

2 Current management tools

For the purpose of developing MAMSY, a previous study was carried out on current management possibilities in available platforms. The targets of the study were 3APL[10], AgentBuilder[4], AgentScape[8], Aglets[18], AGlobe[22], Comet-Way Agent Kernel (AK)[1], Cougaar[14], Decaf[17], Diet[16], Hive[19], Jade[7], Jadex[6], Madkit[13], Open Agent Architecture[9], Semoa[2], Tryllian Agent Development Kit[3] and Zeus[20]. The objective of the analysis was to detect which functionalities were implemented in current management tools and to take into account clear visualization methods. We designed an ideal taxonomy of services which should be included in every multi-agent management tool. This taxonomy and results of the analysis on current management tools can be observed in Table 1.

It was observed that some management tools [1][14][16][3] do not include modifying functionalities for platform current state. They usually only offer a way of configuring the multi-agent system before it is started or visualization functionalities. Management tools that are able to perform modifying functionalities offer a heterogeneous set of actions which deal with nodes and agents in the current system. Although some functionalities are common, they differ in the way they are implemented. The largest set of actions avaliable was found in [7] and related tools like [6].

The visualization methods are similar but they are not prepared to handle big volumes of data since most of them do not offer search and filter mechanisms, [14] being the only management tool that had such services. Although current state management offers a large set of actions, current management tools do not offer services for looking through platform past. In [11] it is described the problems a human faces when observing and monitoring a set of distributed concurrent processes. It is also pointed that global-time-stamped information is the only realiable picture for such systems. We are convinced that this past information could be used in debugging tasks in the way [11] proposed.

3 Magentix: A Multiagent platform integrated in Unix

Magentix is a multi-agent platform developed by GTI-IA, a research group located at Universidad Politecnica de Valencia . It has been developed entirely using the C programming language and makes considerable use of Unix's core calls. So the platform is currently only available on Unix-like systems. One of the biggest problems faced in multi-agent systems is platform core performance when the number of agents and nodes is large. Through the use of a non-interpreted language like C and Unix's core calls, Magentix aims for a multi-agent platform that solves scalability and performance issues.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
3APL						x		x	x	x			x		x
AgentBuilder						x				x					x
AgentScape	x					x		x	x						
Aglets						x		x	x	x	x	x			
AGlobe						x		x	x						x
AK															
Cougaar	x	x	x	x		x	x							x	
Decaf						x			x				x		x
Diet						x								x	
Hive	x			x		x		x	x						
Jade	x		x	x		x		x	x	x	x	x	x		x
Jadex	x		x	x		x		x	x	x	x	x	x		x
Madkit	x			x		x		x	x					x	x
OAA	x					x			x						
Semoa	x					x		x	x						
Tryllian															
Zeus						x								x	x
MAMSY	x		x	x		x	x	x	x					x	x

Table 1 Current management tools analysis

1:Visualize nodes: List instances of platform kernel in our system and related information – **2:Filter/Search nodes:** Narrow the list of nodes according to our interests – **3:Add node:** Add a new node to the current system – **4:Remove node:** Stop the activity of a system node and delete it from the system – **5:Stop/Restart node:** Stop the activity of a system node and its agents/Restart the activity of a stopped node, recovering the state it had before it was stopped. – **6:Visualize agents:** List agents in our system and related information – **7:Filter/Search agents:** Narrow the list of agents according to our interests – **8:Add agent:** Add a new agent to our system – **9:Remove agent:** Delete an agent from our system – **10:Stop/Restart agent:** Stop the activity of an agent/Restart the activity of a stopped agent, recovering the state it had before it was stopped – **11:Migrate agent:** Stop an agent and restart its execution in another node – **12:Clone agent:** Copy an agent and start its execution in a node – **13:Send message to agent** – **14:Organizational support:** Visualization of agent organizations and operations that deal with organizations – **15:Communication analyzer tools:** Tools that let us observe messages exchanged between agents

3.1 Agent messaging

Older versions of Magentix [23][5] used plain text as a way of exchanging information between platform agents. Recently, a new version of Magentix was developed using RDF as a way of structuring messages. RDF is a resource description language recommended by W3C which expresses information as a triplet. Each triplet has a subject, a predicate and an object. Subjects and predicates content can only be expressed as a URI (Universal resource identificator) while objects may be a URI or a literal string. Knowledge graphs can be formed if we consider subjects and objects as nodes and predicates as named links. Using RDF graphs as a way of structuring information generates an extra overload of data in communications but in exchange it allows developers to use richer and more structured knowledge representation. Magentix messages are RDF graphs which are serialized and then sent to agents. A

Magentix simplified agent message can be observed in Fig.1. Round graph nodes symbolize URIs whereas box graph nodex symbolize literal values. Those round graph node which content in enclosed by parentheses symbolize a special unique URI node which is called *blank*. The figure shows a *Hello world* message in Magentix which is sent to agent *mgx_history*.

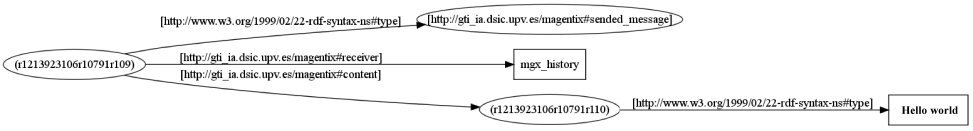

Fig. 1 Hello World! sample message

4 MAMSY: A tool for multi-agent system management

MAMSY stands for *Magentix Management System* and is a tool designed to help system administrators monitor and control their resources in a Magentix multi-agent system. Furthermore MAMSY aims to solve some of the management problems explained in the introductory section.

Functionalities offered to users are a subset of our ideal service taxonomy, but since MAMSY and Magentix are still in the development phase, more services are expected to be offered to system administrators. However, we include a new type of management called *past time management* which allows users to analyze and observe past events and messages.

Due to the fact that one of MAMSY's goals is to be user friendly, the design team decided to enhance the tool with a desktop graphical interface. Selected libraries were Trolltech's Qt C++ APIs.

4.1 Functionalities

MAMSY's services and functionalities may be classified into two types, according to the time period they act on or visualize. Firstly we have present time functionalities, services which allow users to visualize and modify the platform's current state. Lastly, we can also find past time functionalities that are focused on visualizing events and messages in a selectable time range.

4.1.1 Present time functionalities

Three different resources may be managed in current state: Nodes, agents and units. As we stated, nodes are physical hosts that form the multi-agent system, each one executing a magentix core instance. Agents are the main subject in MAMSY's management capabilities and units only allow visualization since unit management is relegated to agents. We will describe these functionalities briefly below. A short matching between our proposed taxonomy of present services and MAMSY's current services can be found in Table 1. We can observe MAMSY showing present time information in Fig.2.

Node managing

-Add host: Adds a new node to the current Magentix system. It requires that Magentix is installed on the target host.
-Remove host: A host and its agents are deleted from the system.
-Stop Platform: The system permanently stops its activities.
-Node visualization: Nodes appear in a hierarchical view with their agents nested.

Agent managing

-Add agent: An agent may be added to any node.
-Remove agent: A local or remote agent is deleted from the system.
-Search agent: A search engine is included so users can search agents by name.
-Agent visualization: Agents can appear along with the node they are executing in or with the organizations they are a member of.

Agent Organization managing

-Search organization: A search engine is included so users can search for organizations by name.
-Organization visualization: Units only appear in 'Logical View', showing what members they have or even which organizations they are a member of.

4.1.2 Past time functionalities

One of the most important features implemented in MAMSY is past time related functionalities. No similar features were found when analyzing current management tools. Whereas current systems just deal with the very near past, more specifically a very limited time window, our management tool can store a wide time range for later visualization. The information we are interested in storing for later visualization is platform events, i.e agent creation/deletion, new host, host deletion, unit creation/deletion, unit member addition/deletion and messages exchanged between users. We will review in section 4.2 who is in charge of storing and distributing this information.

We used a sniffer-like visualization method to show past messages and events to system the administrator, due to the fact that it is a commonly spread application in the world of multi-agents. Since the amount of data may be quite large, we

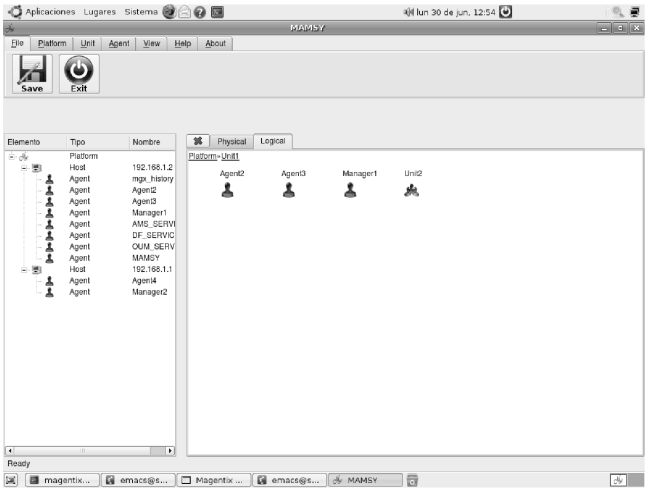

Fig. 2 MAMSY showing present information

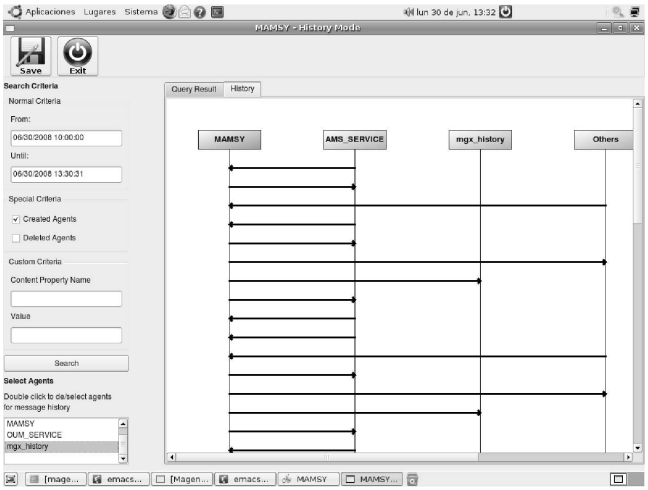

Fig. 3 MAMSY showing past information

decided to include several filters that allow users to narrow the information they are interested in. More specifically, we allow through an user interface to filter agents created/deleted in a time range, agents present in a time range and agents that sent/received messages with a concrete property and value specified by the user. This visualization method and filters can be observed at Fig.3.

Information gathered and showed to users was considered extremely interesting to system administrators because of its applications to error detecting tasks and debugging. Furthermore, stored information also contains messages exchanged be-

tween agents so further analysis may highlight interesting information regarding agent behaviour, agent likes and dislikes and so forth.

4.2 Architecture

One of the key factors in the design of MAMSY was its placement in the multi-agent system. The final design determined that MAMSY would consist of two main components. Each of these components was placed as a special agent in the multi-agent system.

The first agent is commonly referred to as a history agent and its main task consists of capturing events and messages produced in the system. The second agent is called the MAMSY agent and its task consists of capturing user requests and showing requested information to the user. However MAMSY is really the set of the two types of agents. When the history agent is present in the system, platform services send a special RDF notification to the history agent with the event that has been produced. Then the history agent looks for the MAMSY agent instances and sends them RDF notifications with the event produced, thus they can update their state and look. Arriving events and messages are properly stored into a database once they have been processed. Magentix RDF graphs are stored into a database using Redland API. The chosen database was open source PostgreSQL although it is possible to use other free database engines like MySQL and SQLite. When MAMSY agents desire information about the past or history agent needs to query for information, they send a SparQL query, a RDF query language specified by the W3C, which is executed on history agent database. MAMSY agent users do not input the SparQL query, but instead they fill a form which is translated by the MAMSY agent to SparQL once the user has decided to execute the query.

4.3 Security features

It has been mentioned that one of the main problems in multi-agent management is security, more specifically user authentication and permission policies. MAMSY solves part of these inconveniences using SSH protocol as a secure channel for sending commands to be performed over resources. Using SSH forces users to have an appropiate username and password for the host machine they want to act on. Consequently, when a user performs an action on a Magentix resource it is almost certain that they are trusted users.

5 Conclusions and future work

In this paper we have described some problems faced when dealing with multi-agent system management. Even though security is a big problem, tool designers should not forget good user interface practices due to the large amount of data a multi-agent system may generate.

One of the conclusions extracted from the study of current management tools is that they provide a heterogeneous set of management actions. We also detected the lack of visualization about past time information which can be useful for a wide range of applications.

In this paper we presented MAMSY which is our first attempt at addressing management of Magentix multi-agent systems. We have described its current functionalities, its architecture and security features. It offers services for current state management and a new service for visualization of past messages and events. Our current tool has been used in order to help the development of a Tourism Application[21].

However MAMSY and Magentix are not finished works. Both of them are still in development phase and gradually including new features. Some of the most important functionalities in future MAMSY versions will include:

-Add more actions for current state management
-Explore new security models which enhance authentication and permissions issues: Unfortunately, the current model still lacks flexibility due to the fact that there are situations where SSH is not suitable. For instance, in an ideal system there may be hosts whose owners let other users execute their agents freely. This is not accomplished using only the SSH security model.
-Explore new visualization methods for past events: More visualization methods which give a clearer view about platform past.
-Intelligent and autonomous management: Avoid system overloads through autonomous and intelligent management.

References

1. Cometway agent kernel. URL http://www.agentkernel.com/
2. Semoa. URL http://semoa.sourceforge.net/
3. Tryllian agent development kit. URL http://www.tryllian.org/
4. Acronymics, Inc.: AgentBuilder User's Guide (2004)
5. Alberola, J.M., Mulet, L., Such, J.M., Garca-Fornes, A., Espinosa, A., Botti, V.: Operating system aware multiagent platform design. In: Proceedings of Fifth European Workshop On Multi-Agent Systems (EUMAS 2007), pp. 658–667. Association Tunisienne D'Intelligence Artificielle (2007)
6. Alexander Pokahr Lars Braubach, W.L.: Jadex: Implementing a bdi-infrastructure for jade agents. EXP - In Search of Innovation (Special Issue on JADE) **3**(3), 76–85 (2003)
7. Bellifemine, F., Poggi, A., Rimassa, G.: JADE - a FIPA-compliant agent framework. In: Proceedings of the Practical Applications of Intelligent Agents (1999)

8. Brazier, F.M.T., van Steen, M., Wijngaards, N.J.E.: Distributed shared agent representations. In: B. Krose, M. de Rijke, A.T. Schreiber, M. van Someren (eds.) Proceedings of the 13th Dutch-Belgian AI Conference, p. 77 (2001). Extended abstract of AEMAS2001 publication
9. Cheyer, A., Martin, D.: The open agent architecture. Autonomous Agents and Multi-Agent Systems **4**(1-2), 143–148 (2001)
10. Dastani, M., Birna, M., Dignum, R.F., jules Ch. Meyer, J.: A programming language for cognitive agents: Goal directed 3apl. In: In Proc. ProMAS03. 2003, pp. 111–130. Springer (2003)
11. Garcia-Molina, H., Jr., F.G., Kohler, W.H.: Debugging a distributed computing system. IEEE Trans. Software Eng. **10**(2), 210–219 (1984)
12. Giampapa, J.A., Juarez-Espinosa, O.H., Sycara, K.P.: Configuration management for multi-agent systems. In: AGENTS '01: Proceedings of the fifth international conference on Autonomous agents, pp. 230–231. ACM, New York, NY, USA (2001)
13. Gutknecht, O., Ferber, J.: The madkit agent platform architecture. In: In Agents Workshop on Infrastructure for Multi-Agent Systems, pp. 48–55 (2000)
14. Helsinger, A., Wright, T.: Cougaar: A robust configurable multi agent platform. Aerospace Conference, 2005 IEEE pp. 1–10 (2005)
15. Hewitt, C.: Open information systems semantics for distributed artificial intelligence. Artif. Intell. **47**(1-3), 79–106 (1991)
16. Hoile, C., Wang, F., Bonsma, E., Marrow, P.: Core specification and experiments in diet: A decentralised ecosystem-inspired mobile agent system. In: Proc. 1st Int. Conf. Autonomous Agents and Multi-Agent Systems (AAMAS2002), 2002, pp. 623–630. ACM Press (2002)
17. J.R. Graham K.S. Decker, M.M.: Decaf - a flexible multi agent system architecture. Autonomous Agents and Multi-Agent Systems **7**, 7–27(21) (July 2003)
18. Karjoth, G., Lange, D., Oshima, M.: A security model for aglets. IEEE Internet Computing **1**(4), 68–77 (1997)
19. Minar, N., Gray, M., Roup, O., Krikorian, R., Maes, P.: Hive: Distributed agents for networking things. Agent Systems and Applications, International Symposium on / International Symposium on Mobile Agents **0**, 118 (1999)
20. Nwana, H.S., Ndumu, D.T., Lee, L.C., Collis, J.C., Heath, M.: Zeus: A collaborative agents tool-kit
21. Palanca, J., Aranda, G., Garcia-Fornes, A.: Building service-based applications for the iphone using rdf: a tourism application. In: Proceedings of the International Conference on Practical Applications of Agents and Multiagent Systems (2009)
22. Rollo, M., Pchouek, M.: A-globe: Agent platform with inaccessibility and mobility support. In: Cooperative Information Agents VIII, number 3191 in LNAI. Springer-Verlag (2004)
23. Such, J.M., Alberola, J.M., Garca-Fornes, A., Espinosa, A., Botti, V.: Kerberos-based secure multiagent platform. In: Sixth International Workshop on Programming Multi-Agent Systems (ProMAS'08), pp. 173–186 (2008)