MULTIAGENT SYSTEM FOR SOFTWARE MONITORING AND USERS' ACTIVITIES IN A NETWORK EQUIPMENT

RESUMEN

This article presents a multiagent system (MAS) applicable to software monitoring and to the activities carried out by the users on the equipment of a network. The concept of distribution and on the use of mobility; the latter being an agents' property which provides a wide range of alternatives to solve or do tasks at the exact place where they are needed. *The development of a MAS is based on a GAIA methodology and AUML language.*

KEYWORDS: Agents, MAS, software monitoring, mobility, GAIA

ABSTRACT

Este artículo presenta un sistema multiagente (SMA) que permite monitorear las actividades realizadas por los usuarios en los equipos de una red de área local. El concepto de distribución y el uso de la movilidad, son propiedades de los agentes que proporcionan una amplia serie de alternativas para solucionar o hacer tareas en el lugar exacto donde se requieren. Para el desarrollo de este SMA se utilizó la metodología GAIA y lenguaje AUML.

PALABRAS CLAVES: Agentes, SMA, Software de monitoreo, movilidad, metodología GAIA

OSCAR H. FRANCO
Ingeniero de Sistemas, M.Sc
Docente Investigador
Universidad Autónoma de Manizales
oscarhf@autonoma.edu.co

LUIS F. CASTILLO
Ingeniero de Sistemas, Ph.D. (c).
Docente Investigador
Universidad Autónoma de Manizales.
lfcastil@autonoma.edu.co

JUAN M. CORCHADO
Ingeniero Informático, Ph.D.
Profesor titular
Universidad de Salamanca
corchado@usal.es

CARLOS ANDRES LOPEZ
Ingeniero en Computación
Universidad Autónoma Manizales
andresbox@gmail.com

1. INTRODUCTION

The Multiagent system (MAS) paradigm is based upon the idea on overcoming the restrictions inherent to any intelligent system, be it natural or artificial, thus generating associations of simple systems so that they can share knowledge and capabilities in the same way as people overcome their individual restrictions by constituting societies. Each of these simple systems that is grouped would constitute and agent, and the community, globally considered, would be a multiagent system [1]. A description of the set of properties displayed by a MAS can be found in the technical report prepared by Iglesias [2], which states that a MAS must allow as, a minimum, the cooperation and communication among agents. As proposed by García and Pavón [3] a wide series of real applications in which the use of agent systems may facilitate widely the design and development efforts can be numbered.

In this article, emphasis will be made on the concept of distribution and on the use of mobility; the latter being an agents' property which provides a wide range of alternatives to solve or do tasks at the exact place where they are needed. This contributes to the distribution of processing times and to the construction of more natural, or at least closer- to-real-life computing solutions. Specifically in this case of study, mobility is applied to "visit" network equipment and carry out the programmed monitoring, with the help of a Windows service.

One of the major problems in the development of an architecture based on multiagent systems is that there are currently no clear standards or well-developed methodologies for defining the steps of analysis and design that need to be taken. There are at present a number of methodologies: Gaia [4], AUML [5,6], Ingenias [7], Tropos [8], Message [9]. For this study, we have taken GAIA for our Multiagent System. GAIA is a simple methodology that allows us to carry out a preliminary analysis and design with which to confront the problem at a general level. The great advantage is that we can carry out a rapid, broad study. We are able to obtain a generalized vision of the problem in terms of organization, which helps enormously in the development of such a research project.

2. CONTENT

The problem of monitoring applications in a computer and the applied MAS will be explained in the following section. The implementation of the case with Distributed JADE will be explained in the fourth section, and the results from the execution of the System will be presented in the conclusions.

Fecha de Recepción: 31 de agosto de 2006
Fecha de Aceptación: 12 de Febrero de 2007

3. STUDY CASE AND MAS

It is very important to monitor the equipment that makes up a network, the software that has been installed in each computer and its frequency of use when managing the resources of a computer network. Although this could be restricted with permissions assigned to the users with reference to the installation and execution of programs, and thanks to social hacking, the agents may become an interesting tool that can be used to monitor, in a distributed manner and thanks to their mobility, the management of the computer network of an enterprise. To meet this need, agents were used according to their distributed nature and their characteristic mobility, which was well-documented but scarcely implemented in the MAS due to security aspects.

An organizational design that favors the feasibility of the programming, retrieval and consolidation of logs is then presented. This structure consists of an agent-user that represents the real user in the system, and administrator that facilitates the capturing, updating and consultation of the persistant data, a coordinator that controls the start, collection, ending and analyses of logs, a agent-visitor that goes to every computer and executes the procedures necessary to generate the information required in every monitoring. This step is carried out with the help of a Windows service and an agent-auditor that analyzes the information that is retrieved and triggers some warnings which, finally, allow the real users to interact with the system, to detect when an aspect to be monitored must be revised

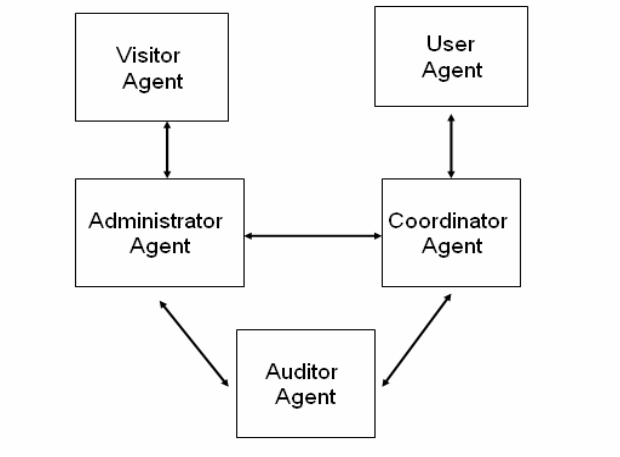

Figure 1. GAIA Familiarity Model

With the schema shown above (Figure 1), an effective solution combining mobility and distribution of multiagent systems was provided. Thus, the design of the system turns into the construction of a team work (as we know it in real life) where every member has some specific functions which contribute to the achievement of the aim. The implementation process is described below.

4. METHODOLOGY

Due to the special characteristics of the multiagent systems, the analysis and design of the project were achieved by using GAIA methodology and the AUML language. The latter was used due to the fact that GAIA does not generate a design concrete enough to be implemented directly.

4.1 Gaia

GAIA was developed by Woolridge, Jennings and Kinny in 2000 and consists basically of two phases (Figure 3): an analysis and a design one. These authors state that ”GAIA centers itself around the idea that the construction of agent-based systems constitutes an organizational design process” [4].

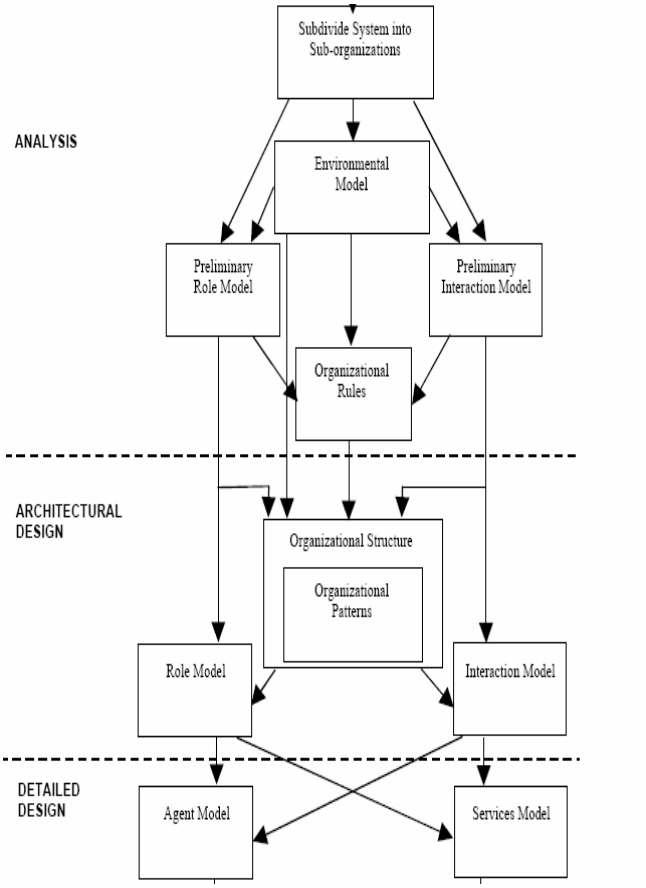

Figure 2. GAIA PHASES

The analysis phase deals with the comprehension of the system and its structure. GAIA systems differentiate from the following models:

Interaction model: Dependence and relationships in a role group are defined.

Role Model: After identifying the system roles and having a clearer idea about the interactions that take place among them, this phase aims at consolidating a thoroughly elaborated model where each role is documented.

The GAIA design phase consists of three models: the agent model that identifies the types of models included in the system and their corresponding instances; the service model that identifies the services required for the agents' roles, and, finally, the familiarity model that stands for the links existing among the agents that structure the system.

4.2 Auml

AUML language is of importance to complete and refine the outcome from GAIA, which is a little concrete to be implemented directly. This language centers around the use of already existing AUML development tools, thus adapting them to the agents' specific characteristics. An Odell's proposal suggesting the representation of the three layers known as AIP (Agent Interactive Protocol) was used. According to Odell "AIP describes communication patterns that include: a permitted sequence of messages among agents with different roles; obligations about the contents of the messages; and semantics consistent with the communication acts within a communication pattern." [5]

Protocol layer: Communication protocols are represented in this layer.

Interaction layer: The interactions among agents are represented here. This layer may use two types of diagrams, each one with different characteristics. Accordingly, it is necessary to select the protocol that best adapts itself to the interaction to be modeled; these diagrams may be sequence or collaboration ones. Due to the distribution characteristics, collaboration diagrams were used in this work (Figure 3).

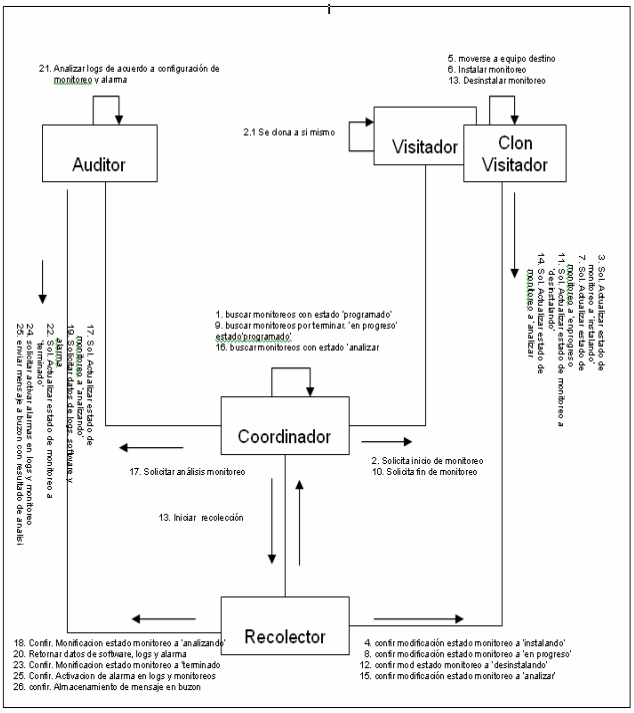

Figure 3. Colaboration Diagram

5. STUDY CASE IMPLEMENTATION

A four-tier architecture was used in the implementation of the system:

1. A layer of processing agents: the agents and their functionality were defined.
2. A presentation layer: in charge of managing the interface and interaction of the system users and the MAS.
3. A data layer: to administer the database and the routines to have access to the data.
4. A client layer: a Windows service to be installed in every computer to achieve the monitoring.

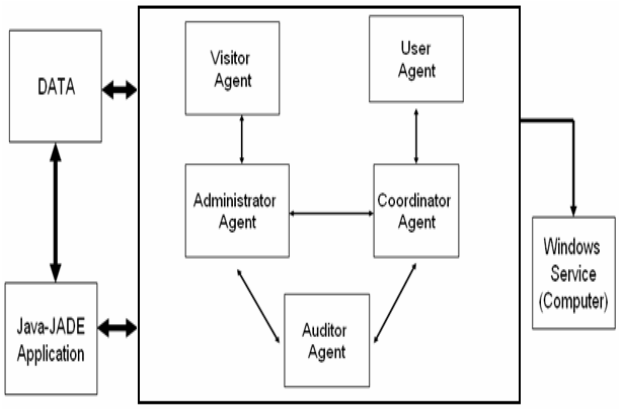

Figure 4. Four-tier architecture

JADE [10] was used in the implementation of the system to administer the agents. This platform allows the

management and follow-up of the MAS agents; the database was developed in Mysql, the Windows service with .NET and the presentation layer (system interface) was made in JAVA. The system basically functions as follows: it provides an interface that allows the storage and updating of the data necessary to program monitoring such as organizations, groups, equipment, software, warning logs, etc. See Figure 5.

Figure 5. Presentation layer

Once a monitoring with the specifications on what is to be done in the end computers is programmed, an agent goes to the computer and installs Windows services. The Windows service generates text files (Logs) of configured information and, once the monitoring is achieved, the agent-visitor retrieves and stores the information generated, then, an agent-auditor analyzes this information and presents the results from it (Figure 6).

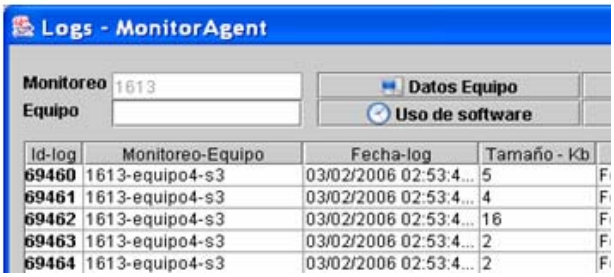

Figure 6: Results from monitoring

An important feature of the system is the customization of the information to be monitored and of the warnings that are to be revised in each monitoring. As a matter of fact, it is possible to program monitoring for particular cases and evaluate them according to the conditions and the time desired. As it is shown in the main interface (Figure 5), the user is constantly notified about his monitoring. This is achieved by displaying a list of what he has programmed in its present status, and this list also shows him every thing that happens with his monitoring in a message form: whether the installation was successful, there was a failure in any computer or there was a monitoring that violated any condition on the associated risk. From this explanation, the interpretation that the most relevant characteristic was the agents' mobility was drawn. When referring to JADE mobility, it is necessary to take into account that it implies a serialization of part of the software (in this case the agent-visitor) in a specific place (the server), the sending of the data through a canal (LAN) and the de-serialization in another destination place (PC-Objective). It is also important to make clear that the aim is not to instantiate a new object in another place, but to transfer that software piece with code and data status. This means that if the agent-visitor had declared a variable X=50 at destination A, when variable X is displaced to a destination B, it will keep its value 50 and so the other instances and variables that the class may contain.

6. PROOF DOCUMENTATION

The system described above was tested in the computer network at Autonoma University in Manizales, Colombia, and various characteristics of its functioning were analyzed. The number of computers being monitored concurrently and on which relevant information was consulted and updated was taken into account. The results were the following among others:

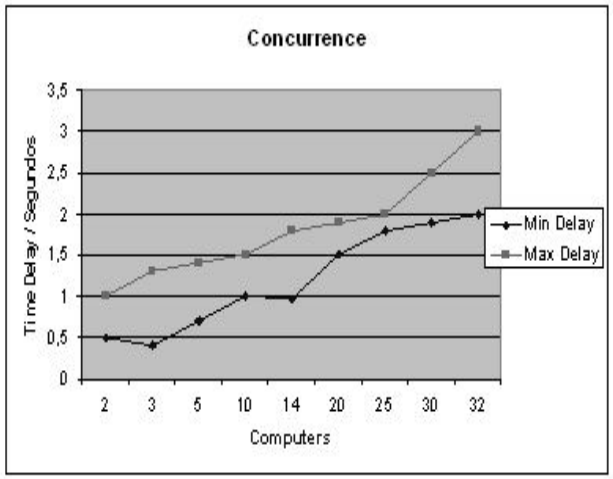

Figure 7. Results

The response time for updatings range from 0.5 to 1.5 seconds; in the case of consultations it depends on the data to be retrieved. The slowest consultation takes 3 seconds approximately. The performance of the server is not affected when the agents are cloned and the monitoring data are received. The performance of the PCs under analysis decreases when the files are initialized and verified, this process being narrowly linked to the amount of files in the computer and on its physical capacity.

7. CONCLUSIONS

The development of multiagent systems, given their distributed nature, favors the appearance of characteristics such as: location transparency, access transparency, tolerance, concurrence, replication and mobility. It is important to implement this kind of systems in organizations where a high degree of control on the technological platform owned is required. Thus, a useful tool is provided for the management of system administrators and computing auditing groups in their control activities, and an information-providing application to the organizations with reference to the events in their network components.

GAIA is a methodology through which it is possible to make the analysis and design of a multiagent system, but it displays some weaknesses when the multiagent system interacts with other type of components. Accordingly, big lacks to model the parts of the system, which do not function as agents, were found, such as the Windows service. It was then necessary to turn to diagrams to model these aspects in an approximately effective way.

The use of AUML with sequence, activity and collaboration diagrams allowed to increase the detail level in the design phase. This aspect favored the implementation of the agents' behaviors with JADE.

The multiagent systems may be used to solve different types of problems, especially those involving distributed systems, which process a great deal of information and facilitate a collaborative solution. However, before they are applied to the solution of a problem, the requirements and characteristics of each specific case must analyzed in detail.

8. ACKNOWLEDGEMENTS

Mr. Oskar Llano because of the translation of the article

9. BIBLIOGRAPHY

[1] Wooldridge, M. The Logical Modelling of Computational Multi-Agent Systems. Dissertation, Department of Computation, UMIST, Manchester, UK, October 1992.

[2] Iglesias C.A. Definición de una metodología para el desarrollo de sistemas multiagente. 1998. Tesis doctoral. Universidad Politécnica de Madrid, E.T.S.I. Telecomunicación, Departamento Ingeniería de Sistemas telemáticos. P. 193-202.

[3] Garcia Alonso, D. y Pavón Maestras J. Introducción al estándar FIPA. En: Informe técnico UCM-DSIP 98-00. Departamento de Sistemas informáticos y programación. Universidad Complutense de Madrid. 2000.

[4] Wooldridge, M. and Jennings, N. R. and Kinny, D.: The Gaia Methodology for Agent-Oriented Analysis and Design. Journal of Autonomous Agents and Multi-Agent Systems, 3 (3). pp. 285-312. (2000)

[5] Bauer, B. and Huget, M. P.: FIPA Modeling: Agent Class Diagrams (2003)

[6] Odell, J. and Huget, M. P.: FIPA Modeling: Interaction Diagrams (2003).

[7] Gomez, J. and Fuentes, R.: Agent oriented software engineering with INGENIAS. In Proceedings of 4th Iberoamerican Workshop on Multi-Agent Systems (Iberagents'02), Malaga, Spain. (2002)

[8] Mylopoulos, J., Kolp, M., and Castro, J.: UML for agent-oriented software development: The TROPOS proposal. In Proc. of the 4th Int. Conf. on the Unified Modeling Language UML'01, Toronto, Canada. (2001)

[9] EURESCOM MESSAGE: Methodology for engineering systems of software agents. Technical report P907-TI1, EURESCOM (2001)

[10] http://jabe.tilab.com june 26, 2006.

10. CONSULT

[1] BATES, J., Bryan Loyall, A., and Scott Reilly, W. An architecture for action, emotion, and social behaviour. Technical Report CMU–CS–92–144, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.1992

[2] BATES, J., Bryan Loyall, A., and Scott Reilly, W. Integrating reactivity, goals, and emotion in a broad agent. Technical Report CMU–CS–92–142, School of Computer Science, Carnegie-Mellon University, Pittsburgh, PA.1992

[3] SHOHAM, Y. Agent-oriented programming. Artificial Intelligence, 60(1):51–92.1993

[4] CORCHADO J.M., Corchado E. y Pellicer M.A.: Design of cooperative agents for mobile devices. Proceedings of the International Conference on Cooperative Design, Visualization and Engineering - CDVE2004. Luo Y. (Ed.) LNCS 3190, Springer Verlag. ISSN 0302-9743. ISBN 3-540-23149-8 pp. 205-212. (2004)

[5] CORCHADO J. M., Pavón J., Corchado E. and Castillo L. F. : Development of CBR-BDI Agents: A Tourist Guide Application. 7th European Conference on Case-based Reasoning 2004. LNCS 3155, Springer Verlag. pp. 547-559 (2005)