Intelligent Management of Computer Networks: The GIR Proposal[*]

Alceu Britto Jr., Emerson Paraiso, Edson Scalabrin, Celso Kaestner, Bráulio Ávila
LASIN[†]— PPGIA[‡]
PUC PR — Pontifical Catholic University of Paraná
R. Imaculada Conceição, 1155
80215-901 Curitiba - PR, Brazil
{alceu, paraiso, scalabrin, kaestner, avila}@ccet.pucpr.br

Abstract

This paper presents a system proposed for the intelligent management of computer networks. The system is based on the use of Artificial Intelligence techniques — data mining, expert systems and multi-agent systems. The work is based on the following lines of actions: (a) the use of distributed agents for the intelligent search of information in the network, supplying it in a more abstract way, adapted to the decision-making task; (b) the use of machine learning and data-mining techniques that, starting from the log files which register previous problems and their solution, allow the use of experience thus obtained in the solution new problems; and (c) the use of heuristics and conduction rules supplied by experts, through a decision support system, as an advisor element to network operators. This paper presents a discussion about the techniques employed along these three research lines and the results already obtained.

1. Introduction

The migration from centralized to distributed systems is a reality. Indications of this new reality are the popularization of network technology, the increased reliability of servers and the enhanced processing capacity of workstations. In large companies this behavior may be easily evidenced by the acceleration of the downsizing process and the massive use of distributed processing systems.

[*]This system integrates the "GIR — Gerência Inteligente de Redes" (Intelligent Management of Computer Nets) Project, carried out under angreement between PUC PR and Siemens Telecomunications.

[†]Intelligent Systems Laboratory

[‡]Graduate Program in Applied Informatics

If on the one hand these technologies bring unquestionable benefits, the management of large computer networks is not a trivial task. Some problems may be readily identified, particularly those concerning the dynamic access to the resources of a system, such as servers, routers, etc., which are distributed in the new model. Manager or operator support systems typically offer limited help at the operational level, most of them consisting in event monitoring, communication and logging.

The development of efficient and efficacious tools is called for to help in the management of the resources available in the network. A few quite complex tools were developed for this purpose, such as OpenView^{TM} [12] and Optivity^{TM} [15], for HP^{TM}, SUN^{TM} and other platforms. The main problem inherent to such tools is the overload they impose on the operator, both in operational terms, when trying to manage resources on-line, and in terms of knowledge handling - cognitive overload, as a consequence of the quantity of information handled at each decision-making [7], [11].

The GIR (Gerência Inteligente de Redes de Computadores — Intelligent Management of Computer Networks) system is proposed in this paper to monitor and manage computer networks, relying on the use of Artificial Intelligence Techniques. The main techniques employed by GIR involve the application of Multi-Agent Systems [29] [28] [10] [16] [26] [4] [1] Machine Learning and Data-Mining [21] [22] [8] [2] [9] [3] and Expert Systems [24] [19] [27].

The paper is structured as follows: the next section presents an overview of the proposed system and its components; section 3 presents the subsystem in charge of information collection relying on multi-agent concepts; section 4 is dedicated to the presentation of the machine learning and data-mining systems employed and the description of the decision-making support system that operates as an integrating mechanism and provides an intelligent interface for the network manager; finally, section 5 presents the conclusions and perspectives of this work.

2. Proposed Solution

The problem of network monitoring and management may be described as follows: a network operator or manager must be able to monitor and manage, from a given workstation, a myriad of resources and equipment that may be seen as belonging to an outside environment.

The objective of this effort is to ensure the performance of several users' tasks in a safe and efficient way. The environment is subject to failures and unforeseen events and suffers constant change, since the network configuration is seldom maintained for a long time. The operation must be continuous, and the effects of a complete halt or reconfiguration always result in problems. Time constraints are present either to ensure assistance to critical tasks or to maintain the service at an adequate level.

Figure 1 depicts the structure of a conventional network management system: the data concerning network devices and performance is captured by a monitoring system that provides it to the network operator; the entire data interpretation and decision-making task rests solely with the system operator.

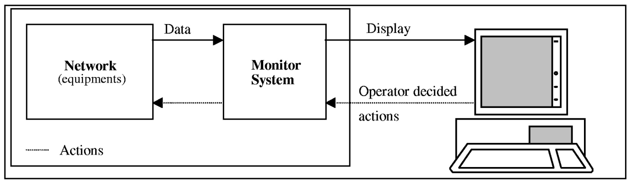


Figure 1. Conventional Network Management System Structure.

All these characteristics suggest that the adequate model for the automation of the problem is a control system in which the best control vector for the current situation, relying on data collected in the environment, is to be provided by the system. However, the diversity of elements involved and the complexity of the problem prevent the construction of proper algorithms to carry out that task. An alternative seems to be taking into account the experience of network operators and managers in hands-on problem solving.

This is precisely the strategy employed by the GIR system, using the information available in an intelligent manner, along three basic lines:

- distribution of agents that search the network for information in an intelligent manner, providing them in a more abstract and adequate way for decision-making;
- the use of machine learning and data-mining techniques that, starting from the log files which register previous problems and their solution, allow the use of experience obtained from previous situations;
- the use of heuristics to the conduction rules supplied by experts, through a decision-support system.

Figure 2 outlines the structure proposed for an intelligent system of network management.

The data on equipment and other devices is obtained in a selective way and at the adequate abstraction level, by means of a set of agents operating in a distributed way. A set of data mining tools allows the analysis of previous problem situations and the steps taken to correct them. A decision-making support system allows the operator to diagnose the situation on the basis of knowledge provided by network management experts, or by the identification of similar situations previously solved. The final decision concerning action still belongs to the system operator, but it can be made now with the support and suggestion offered by the automated system.

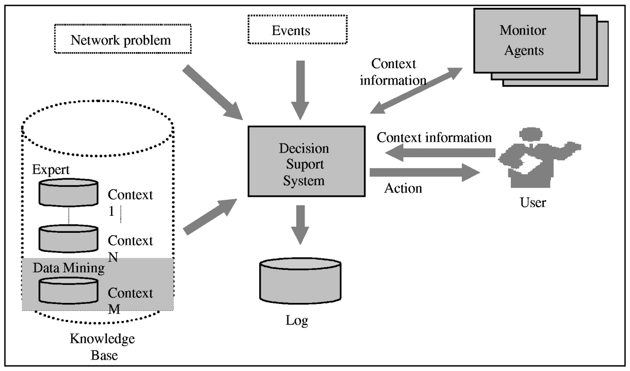


Figure 2. Intelligent Management Network System Structure

3. Information collection and handling agents

The first task needed in a management process is information collection. This process may include not

Type	Skill
Information	To provide information on a given resource (e.g. server, router).
Task	To filter and synthesize information provided by the *Information* agents, according to the different abstraction levels
Interface	To interpret and submit to the manager the present network situation, on the basis of knowledge provided by the *Task* agents, and facilitate the communication between manager and system.

Table 1. Agent types and skills.

only the pure and simple search for data, but also the treatment suited to the intended use.

In the GIR system data is collected in the computer network by a group of agents. This is a natural alternative, since the network environment is inherently distributed and many interactions may be needed in the process. On the other hand, the utilization of agents and their communication reduces the volume of useless information in the network, thereby reducing the overhead imposed by the management system.

The tasks are carried out by a set of agents organized into three different categories: Information, Task and Interface, the respective scope of which is shown in Table 1.

Agents act according to the following stages: (a) direct acquisition of information on the resources managed by a network; and (b) information treatment, performed by a filtration and synthesis process to obtain information with a higher level of abstraction to meet the requirement.

In computing terms, agents are processes distributed in the network, and they exchange information by sending messages. It is important to note that there are several agents of the same type, allowing parallel information acquisition and handling.

Some *Information* agents are SNMP managers that search information from SNMP agents (managed objects) [23] present in the system. Others were built to provide information from the workstations existing in each network segment, that is, they work on the basis of stimuli and answers [5].

Interface and *Task* agents in turn operate in multithread, allowing them to interact with several *Information* agents in parallel. Those agents have objectives and a pro-active behavior — being in this sense intelligent [28] [29].

Other tasks performed by the association of agents are: (1) the creation of logs containing relevant data such as route, response time, and time of log generation; such data are later used in the knowledge acquisition process; (2) the collection of MIBs from manageable agents already existing in the network environment, to provide them to the Decision Support service.

3.1. Example of operation

To illustrate the operation, a performance monitor process in a network segment is provided as an example (see Figure 3). There are several network segments, and each one of them has a Task agent. Initially, Interface agents query Task agents about the situation in their segment. One of these agents is the Ping Task agent. This agent builds an ICMP type message and send it to each route leading to Information agents that are, in general, servers scattered throughout the network. Thus, after the message returns, a description of the situation of each route between Task and Information agents is obtained, allowing the determination of the performance of the segment for each route. Interface agents are charged with providing such information to the operator.

When a too long response time is detected in a given segment or route, the Task agents takes over the role of SNMP managers, defined as the software elements installed in network equipment able to provide information on their interaction with the network [12]. Those agents collect from the equipment located on the route under study information that must be taken into account in problem solving. Such information is passed on to the Interface agents and later to the Decision Support module.

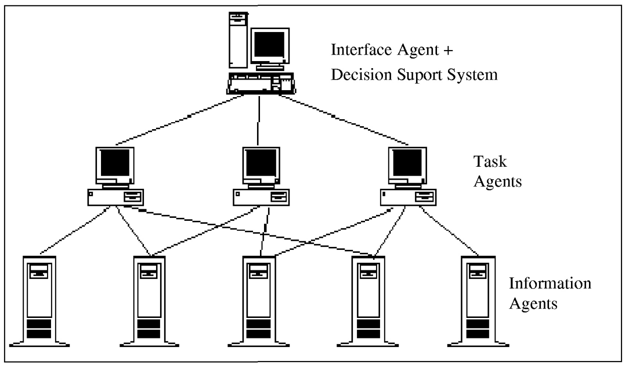


Figure 3. Positioning of Interface, Task and Information agents in a Computer Network.

3.2. Implementation

The system is implemented in C++ on a Windows NT platform, and employs RPC to carry messages between different agents [17]. To recover and check information about routes and overloads on network segments, ICMP packages (ping), socket raw and checksum were employed [18]. Finally, graphic interfaces were employed to present the manager with the network situation in the form of gauges, as shown in Figure 4.

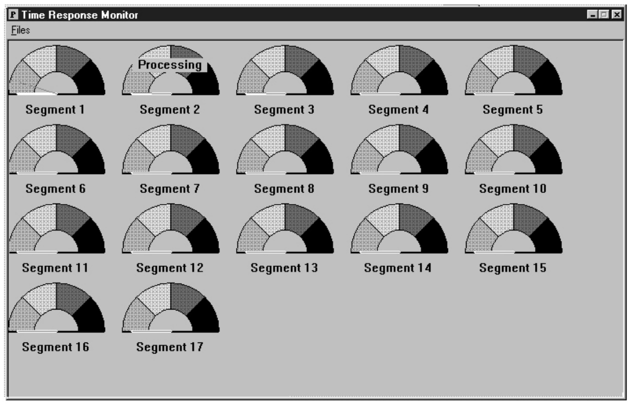


Figure 4. Representation of segment performance by means of gauges.

3.3. Agent Manager

A special type of Interface agent was built to allow agent management. It is the Manager, allowing graphic display of agent location in the network and automatic change of location of machines and segments in the network. Therefore such system agents are considered mobile so that they can act in different sites throughout their existence. With the use of agents they may be enabled to monitor specific network points with greater ease.

Agents can also be configured remotely from the Manager, by defining their scope according to a pre-established model. The interaction among agents is being performed by the information exchange language KQML [13].

4. The Decision Support System

This module consists in a Knowledge-Based System to support the decision-making process. In the context of computer network management, DSS tries to help the manager or operator with the task, providing suggestions about what to do in the face of a given event in the network, and suggesting action to enhance his/her performance.

One of the main advantages of using DSS is the reduction of the cognitive overload on the operator, since in many situations the number of variables to be considered in the search for a solution exceeds the memorization and handling capability of the human mind.

As seen in Figure 2, DSS integrates directly with the other system components, resulting in an intelligent user interface. Its entries are events or problems in the network originating from: (a) a conventional network monitoring system (HP OpenView e Optivity); (b) the "intelligent agents" scattered in the network performing some kind of monitoring; or (c) the network operator himself, in case he/she intends to carry out some sort of simulation in the environment. The events are then used to trigger the inference process that aims to define an action to be suggested to the network operator.

4.1. The DSS Structure

In general lines this module follows the classic structure of a rules-based production system [27], made up of a language to represent knowledge and one (or several) mechanisms to perform the inference. However, a few particular characteristics stand out: (1) the organization of the knowledge about the problem domain in contexts — groups of rules applicable to a particular situation, fulfilling the need to minimize the effort spent during a likely maintenance of the knowledge basis; and (2) the acquisition of "raw" knowledge in the form of rules originating from data mining techniques.

Basically, it is possible to split the DSS structure in three levels:

Data level

At the data level is found the momentary knowledge representing the current status of the system during the resolution of a problem. This is constantly fed by the agents charged with searching relevant information in the network, as well as by context information (or meta-information) that is of the utmost importance in solving a problem and suggesting an action.

Knowledge level

This level is represented by a knowledge base composed of production rules in the IF-THEN format. Each rule represents an atomic article of knowledge and belongs to a certain context. Thus, each context is composed of a set of related rules concerning the scope of a particular problem, the selection of which is driven by events originating in the network or by the inference process itself.

This division facilitates the process of knowledge acquisition and the maintenance of the system's knowledge base, and contributes to enhance system performance, since only the knowledge parcel more adequate to the situation is employed.

The natural style of production rules facilitates the inclusion of explanations about the reasoning employed, such as "how" and "why" questions about a given decision.

Control level

The control level contains the control strategy to handle knowledge and may be referred to in general as inference engine.

There are two possible approaches: (a) forward reasoning, starting from the evidence available (current status) in search of a solution; (b) backward reasoning, using a set of pre-defined assumptions as starting point and trying to determine which one of those assumptions is supported by the evidence. Certainly the definition of the best strategy depends on the problem in question [27]. The GIR system implements both above mentioned approaches, and each situation context may use a different strategy merely by indicating the desired type of engine (forward or backward).

4.2. Example of operation

Figure 5 shows an example of an information bloc that makes up a context concerning a communication failure problem. In this figure the following is observed as to the particular context structure: (a) the definition of the inference direction employed, in this case forward (DIRECTION: FORWARD); (b) the presence of context-specific variables; and (c) the set of rules that make up the context. As to the syntax of the defined language, the presence of QUERY and CHANGE commands stands out, enabling to request information on the network environment from system agents and the change of context.

A main context is used to start the inference process. Starting from the variables concerning the problem (events and alarms), a set of rules defines the next context to be used. This expert context is then triggered, and it is possible to move to another context if so indicated by the inference process that continues until a solution to the problem is found.

```
CLASS Communication_Fault_Context
{
    DIRECTION : FORWARD
    VAR
    {
        Fault CommunicationBetweenAandB SystemStatusA SystemStatusB
        IPAddressBintoA IPAddressBintoB IPAddressAintoB AdressIPAintoA
        RouteBettweenAandB LatencyBettweenAandB
    }

    RULES
    {
        RULE r1: IF QUERY SystemStatusA = 0 THEN Fault := 1

        RULE r2: IF QUERY SystemStatusB = 0 THEN Fault := 2

        RULE r3: IF SystemStatusA = 1 AND SystemStatusB = 1 AND
            QUERY IPAddressBintoA != QUERY IPAddressBintoB THEN Fault := 3

        RULE r4: IF SystemStatusA = 1 AND SystemStatusB = 1 AND
            QUERY IPAddressAintoB != QUERY IPAddressAintoA THEN Fault := 4

        RULE r5: IF SystemStatusA = 1 AND SystemStatusB = 1 AND
            IPAddressAintoB = IPAddressAintoA AND
            IPAddressBintoA = IPAddressBintoB AND
            QUERY RouteBetweenAandB = 0 THEN Fault := 5

        RULE r6: IF SystemStatusA = 1 AND SystemStatusB = 1 AND
            IPAddressAintoB = IPAddressAintoA AND
            IPAddressBintoA = IPAddressBintoB AND
            RouteBetweenAandB = 1 AND QUERY LatencyBetweenAandB > 1000 THEN Fault := 6

        RULE r7: IF SystemStatusA = 1 AND SystemStatusB = 1 AND
            IPAddressAintoB = IPAddressAintoA AND
            IPAddressBintoA = IPAddressBintoB AND
            RouteBetweenAandB = 1 AND LatencyBetweenAandB  <= 1000 THEN CHANGE Performance_Context
    }
}
```

Figure 5. Context with Knowledge about communication fault.

The DSS also possesses a user-friendly graphic interface designed to facilitate its interaction with the network operator. At the same time, the system possesses a simple protocol allowing communication with network monitoring agents. During the interaction with the agent, the network manager or operator may discard the action suggested and provide another considered more adequate. However, the system always records the problem and the action chosen in a proper log file of the triple type (problem, suggested action, action taken). This is important to allow feedback in the decision-making process, taking into account previous cases as a guiding mechanism in new, similar situations.

4.3. Building the knowledge-base

In order to fulfill the DSS knowledge-base, the acquisition of knowledge follows two approaches: (1) classical, by interviewing network experts, managers and operators [27]; and (2) automatic knowledge acquisition, based on data-mining techniques [8] and Machine Learning [14].

The classical approach emphasized the use of interview-based techniques, allowing the mastering of the terminology employed in computer network management, and the definition of heuristics indicated by network operators and managers on the basis of their experience.

However, the main factor distinguishing DSS is the use of automatically acquired knowledge. In different application areas, data mining techniques have been employed to extract knowledge from major information repositories, such as the examples reported by [2].

The use of data mining techniques is possible in the context of computer network management, since the log files generated by management assistance systems like HP OpenView and Optivity make up a bulk of data that may contain relevant information for managers in the form of logged events. Most of those events concern problems occurred in the network.

Thus, a knowledge acquisition system is applied to the available log files in DSS, in two main stages: information pre-treatment and data-mining proper.

Information Pre-Treatment

The initial stage consists in selecting, cleaning, enriching and formatting data, using as ancillary tool an appropriate interface to navigate the log file. Figure 6 depicts the main interface window.

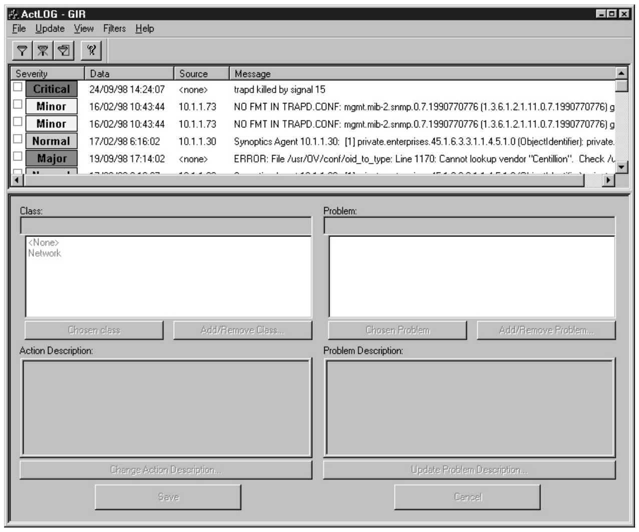
As already mentioned, important information on network events is recorded in the log files. However, those events are typically not related to the steps taken by the expert to solve the problem at that time. One of the characteristics of that interface is precisely that it adds ancillary information such as the problem class and the respective action on the part of the expert at the time the problem arose/was solved.

Figure 7 (a) contains a schematic depiction of information pre-treatment, showing the log file reading and the interaction with the expert to indicate the action taken after the recorded events. Thus the association (event-class-action) is obtained in an explicit way. An important by-product of this stage is the creation of an ontology of the area, needed to standardize the terms used in the statement of problem classes and actions during the pre-treatment process.

The final result of this stage is a training base adequately formatted and filtered to employ automatic knowledge acquisition algorithms to be used in the next stage of the process.

Data-mining

Like in a mining process [8] the training base obtained in the pre-treatment is searched — or mined —



Severity	Data	Source	Message
Critical	24/09/98 14:24:07	<none>	trapd killed by signal 15
Minor	16/02/98 10:43:44	10.1.1.73	NO FMT IN TRAPD.CONF: mgmt.mib-2.snmp.0.7.1990770776 (1.3.6.1.2.1.11.0.7.1990770776) g
Minor	16/02/98 10:43:44	10.1.1.73	NO FMT IN TRAPD.CONF: mgmt.mib-2.snmp.0.7.1990770776 (1.3.6.1.2.1.11.0.7.1990770776) g
Normal	17/02/98 6:16:02	10.1.1.30	Synoptics Agent 10.1.1.30: [1] private.enterprises.45.1.6.3.3.1.1.4.5.1.0 (ObjectIdentifier): private.
Major	19/09/98 17:14:02	<none>	ERROR: File /usr/OV/conf/oid_to_type: Line 1170: Cannot lookup vendor "Centillion". Check /u

Figure 6. Interface for navigating the log file.

for a precious thing: the knowledge about computer network management that will allow solving the problem.

The literature presents several data-mining processes. Algorithms for automatic knowledge acquisition such as CN2 [6], C45 and C50 [22] are especially employed to obtain classifiers (event-action) based on production rules and decision trees [20]. Such algorithms induce production rules from example files [19] [21].

As can be seen in Figure 7 (b) acquisition algorithms generate a knowledge basis (set of rules) from the training base generated during information pre-treatment. The final product of this module is pure knowledge about network management.

5. Conclusions and Perspectives

This paper describes a proposal for a system employing modern Artificial Intelligence techniques to support computer network management. Its main goal is to help the conduction (situation assessment) of the system operator as to the best alternative of action for each problem detected. As in most systems of this type, the goal is to decrease the cognitive overload falling on the operator, allowing an automatic analysis of the several alternatives presented. The distributed, selective, and intelligent information collection carried out by an association of expert agents may also be pointed out.

The knowledge employed by the system derives from two main sources: (a) heuristic, obtained from experts in network conduction that make up a Knowledge Based System; and (b) information obtained via

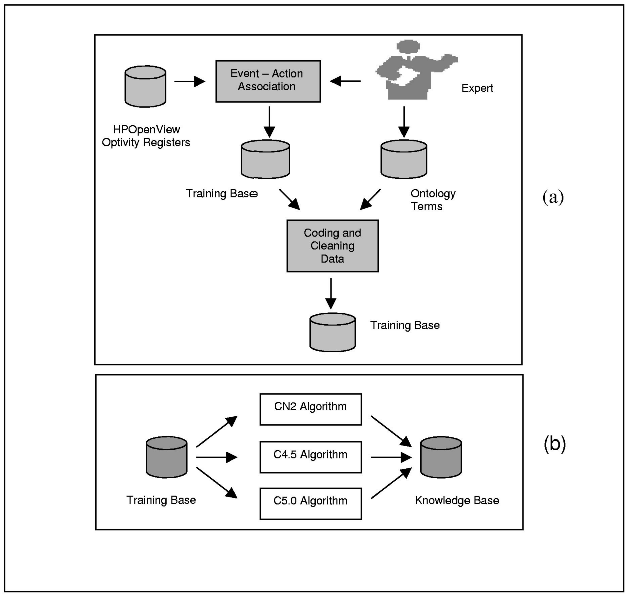


Figure 7. (a) Information pre-processing and (b) Data Mining.

automatic knowledge acquisition in a data-mining process carried out in files containing system logs; the aim is to detect similar situations and identify the actions taken by the operator in previous cases.

Systems with this focus take on an active role in modern computer network management. The intelligent participation ensures better use of the information offered by monitoring systems, as well as the standardization of actions among several managers, and facilitates training new operators.

The GIR system is undergoing the final implementation steps; most of the software modules described above have already been developed. The future stages of the program are the following:

- increase in the number of parameters manageable by information agents, granting network operators more flexibility;
- interaction with SNMP agents, enriching management possibilities by means of utilization of the information available in this protocol widely used in distributed environments;
- tests performed in the machine learning environment checking the actions suggested in typical failure situations;
- continuous acquisition of heuristic knowledge deriving from experts in the area, aiming to create a major Data Base on the domain; and
- performance of system operation tests in a real working environment, represented by a large corporate computer network.

References

[1] Abe, J.M.; Ávila, B.C.; Prado, J.P.A., *Multi-Agents and Inconsistency*, ICCIMA'98 International Conference on Computacional Intelligence and Multimedia Applications, Monash University, World Scientific Publishing Co. Pte., Ltd., Australia, pp. 137-142, February, 1998.

[2] Adriaans, P.; Zantinge, D. *Data Mining*, Addison-Wesley, England, 1996.

[3] Ávila, B.C., *Data Mining*, Livro da VI Escola Regional de Informática da Região Sul, SBC — Sociedade Brasileira de Computação, pp. 87-106, Brazil, Maio, 1998. (in portuguese)

[4] Barthès, J-P.A.; Scalabrin, E.E., *Cognitive Agents and Exchange Protocols*, MASTA Workshop, Coimbra, Portugal, October, 1997

[5] Brooks, R.A., *A Robust Layered Control System for a Mobile Robot*, IEEE Journal of Robotics and Automation, 2(1), pp. 14-33, March, 1986.

[6] Clark, P.; Niblett, T., *The CN2 Induction Algorithm*, The Turing Institute, Glasgow, UK, 1988.

[7] D'Ambrosio, B.; Fehling, M.; Forest, S.; Raulefs P.; Wilber M., *Real-Time Process Management for Materials Composition in Chemical Manufacturing*, IEEE Expert, pp.80-93, Summer. 1987.

[8] Fayyad, U.M.; Shapiro, G.P.; Smyth, P.; Uthurusamy, R.,*Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 1995.

[9] F., Yongjian, *Discovery of Multiple-Level Rules from Large Databases*, PhD Thesis, School of Computing Science, Simon Fraser University, 1996.

[10] Genesereth, M.R.; Ketchpel, S.P., *Software Agents*, Communications of the ACM, Vol. 37(7), pp. 48-53, July, 1994.

[11] Jacob, F.; Suslenschi, P., *Situation Assessment for Process Control*, IEEE Expert, pp. 49-59, April, 1990.

[12] OpenView, *HP OpenView: Using Network Node Manager*, Hewlett Packard Part No. J1169-90002, October, 1995.

[13] McGuire, J.; Pelavin, R.; Shapiro, S.; Finin, T.; Weber, J.; Wiederhold, G.; Genesereth, M.; Fritizson, R.; Mckay, D., Draft Specification of the KQML Agent-Communication Language, 1993.

[14] Michalski, R.S., *A theory and methodology of indutive learning*, In Michalski *et al.* (editor) Machine Learning: An Artificial Intelligence Approach, Vol. 1, pages 83-134, Morgan Kaufmann, 1983.

[15] Optivity, *Optivity LAN 7.0 for UNIX*, Bay Networks, Part no. 893-568-G, April, 1996.

[16] Paraiso, E.; Kaestner, C.; Ramos, M., *MASC: A Multi-Agent System for Control and Supervision of Industrial Plants*, Engineering of Intelligent Systems, Tenerife, Spain, February, 1998.

[17] Paraiso, E.; Vigo, J.; Ditzel, C., *Implementação e teste desempenho da linguagem KQML*, Relatório Técnico LASIN-002/97, Pontifícia Universidade Católica do Paraná, Mestrado em Informática Aplicada, Curitiba, Brazil, 1997. (in portuguese)

[18] Paraiso, E.; Vigo, J.; Camargo, G., *PING: Um Monitor de Tempo de Resposta*, Relatório Técnico LASIN-003/97, Pontifícia Universidade Católica do Paraná, Mestrado em Informática Aplicada, Curitiba, Brazil, 1997. (in portuguese)

[19] Quinlan, J.R., *Induction, Knowledge and Expert Systems*, Proceedings of the Australian Joint Conference on Artificial Intelligence, Sydney, Australia, 1987.

[20] Quinlan, J.R., *Generating production rules from decision trees*, In J. McDermott (editor), IJCAI-87, pp. 304-307, 1987.

[21] Quinlan J.R.; Compton, P.J.; Horn, K.A.; Lazarus, L., *Inductive Knowledge Acquisition: a case study*, In Applications of Expert Systems, Addison-Wesley, Wokingham, UK, pp. 157-173, 1987.

[22] Quinlan J.R., *C4.5 Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo, California, 1993.

[23] Rech Filho, A., *Estudos para Implantação de uma Gerência de Rede Corporativa Utilizando Arquitetura de Protocolos Abertos*, Tese de Mestrado, CEFET-PR, Maio, 1996. (in portuguese)

[24] Russel, S.J.; Norvig, P., *Artificial Intelligence: a modern approach*, Prentice-Hall, New Jersey, 1995.

[25] Santos, A.; Carvilhe, C.; Capeline, K.; Britto Jr., A., *Um estudo comparativo dos algoritmos CN2 e C4.5*, Relatório Técnico LASIN-001/97, Pontifícia Universidade Católica do Paraná, Mestrado em Informática Aplicada, Curitiba, Brazil, 1997. (in portuguese)

[26] Scalabrin, E.E., *Conception et Réalisation d'environnement de développement de systèmes d'agents cognitifs*, Ph.D. Thesis, Université de Technologie de Compiègne, France, 169 p., 1996. (in french)

[27] Stefik, M., *Introduction to Knowledge Systems*, Morgan Kaufmann, 1995.

[28] Sycara, K.; Decker, K.; Williamson, M.; Pannu, A., *Distributed Intelligent Agents*, IEEE Expert, July, 1996.

[29] Wooldridge, M.J.; Jennings, N.R., *Agent Theories, Architectures, and Languages: A Survey*, Workshop on Agent Theories, Architectures and Languages, ECAI'94, Amsterdam, 1994.