Computer Science • Vol. 12 • 2011

Włodzimierz Funika*,**, Filip Szura*,**, Jacek Kitowski*,**

AGENT-BASED MONITORING USING FUZZY LOGIC AND RULES

In this paper we present two solutions of monitoring automation for distributed systems. We develop this system to automate monitoring of distributes systems. Both solutions are aimed to monitor data storage and web services like web page servers. The first solution implemented in a system called Saude-Net, is an rule-based top level monitoring tool. In this system there are implemented rules which provide conditions which refer to one or more measured values. This system is able to choose the best action for an observed situation, e.g. a failure. It is possible to define more than one rule which relate to the same monitoring resource. The second concept presented in this paper refers to a fuzzy logic agent based approach to network monitoring. It is called SAMM compliant Agent. It is an extension to the Semantic-based Autonomous Monitoring and Management system (SAMM). On the one hand, it uses rules to define simple actions, based on a simple condition and an action description. On the other hand the main knowledge of this solution is defined by fuzzy logic. This system is able to manage and modify its knowledge to better fit to monitored resources. The knowledge in this concept is distributed among all the agents. The agents residing on a different hosts handle their parts of the knowledge and are capable to share/exchange them.

Keywords: *resource monitoring, faults, automation, fuzzy sets, rules, agents*

AGENTOWE PODEJŚCIE DO MONITORINGU WYKORZYSTUJĄCE LOGIKĘ ROZMYTĄ ORAZ REGUŁY

W pracy przedstawiono system agentowy, który pozwala na monitoring rozproszonych systemów. Każdy agent posiada własną wiedzę na temat możliwych reakcji na zaobserwowane sytuacje, np. awarie. W artykule przedstawiamy koncepcję automatyzacji monitoringu nazwaną SAMM compliant Agent. Jest ona rozszerzeniem systemu SAMM. Do opisu wiedzy użyte zostały zbiory rozmyte, a także reguły, przy czym reguły używane są do określenia prostych czynności, które może wykonać dany agent. W zaimplementowanej wersji reguły służą jako statyczna wiedza, natomiast logika rozmyta została wykorzystana jako główna idea reprezentacji wiedzy w przedstawionych agentach, z których każdy zarządza swoim segmentem wiedzy i może swoją wiedzę przekazywać innym agentom.

Słowa kluczowe: *monitoring, logika rozmyta, systemy regułowe, agenci, automatyzacja*

* AGH University of Science and Technology, Faculty of Electrical Engineering, Automatics, IT and Electronics, Department of Computer Science, al. A. Mickiewicza 30, 30-059 Krakow, {funika,szura,kito}@agh.edu.pl

** AGH University of Science and Technology, ACC Cyfronet AGH, ul. Nawojki 11, 30-950 Krakow, {funika,szura,kito}@agh.edu.pl

1. Introduction

Nowadays, monitoring automation is a current trend of emerging software. The market of distributed systems grows rapidly so there appears a need in automation which implies relieving the system administrator of having to fulfill some tedious management actions which can be realized on their behalf by a program. The scope of our research is the automation of monitoring for system management, based on artificial intelligence.

By saying "system management" we mean monitoring, reporting on failures (faulty operation) and responding (reacting) on them. Also we regard the equipment configuration and reconfiguration as a domain referring to network management. The term "system management" is closely related to the duties of the system administrator. This activity requires human involvement but some of its subtasks like system monitoring and reaction on errors (faults) can be automated which should imply cost reduction. By saying "system monitoring" we mean also control over monitored resources, the measurement of load of network services.

One of the approaches to automating reactions to failure is related to introducing the concept of agent it is aimed at enabling automatic actions whenever the agent decides that it should respond on the received notifications. The selection of actions is done by a knowledge engine which uses fuzzy sets [1] and rules [2]. This will be discussed more detailed later on in the paper. The agent is able to react on events which occur in monitored resources but it is also able to learn and exchange its knowledge with other agents, each one being specialized in a part of infrastructure or a part of knowledge domain.

To accomplish the above task we focused on two approaches based on fuzzy logic and rules. Both of them are used in our system. Another feature which will be important for our system is the use of agents [3, 4]. This approach is implemented in many existing monitoring systems, but in our research each agent will be able to learn, observe and respond on failures in the monitored system. Each agent is assumed to possess personal knowledge. The system under discussion, called SAMM-CA, has knowledge distributed among the agents located on separate hosts and is aimed to automate reactions on failures to automate reactions on failures. It is intended for administrators of large enterprise infrastructure like grids but also for small networks comprising a few servers. Therefore we try to develop a solution which will be able to work in both types of infrastructures. The idea is to make system management less tiring and less expensive. The presented solution is focused to monitor infrastructure resources, among others those related to data storage [5]. Our system is aimed for a large infrastructure where it will be able to install separate agents for parts of the whole infrastructure.

The paper is organized as follows: the related work section is followed by the concept of agent we exploit. In Section 4 we present more detailed data on the configuration modes of our agent. The tests performed on the agent are presented in Section 5 and followed by conclusions and ideas for the future work.

2. Related work

On the market these are available numerous monitoring systems, designed for different purposes. There are solutions developed for large enterprise infrastructures or for small Local Area Networks(LAN). Both groups have the goal to provide a centralized view of the monitored network. It is done because the administrator should see the whole monitored network to be able to perform quick repairs of a faulty resource. Some tools use agents which are installed on monitored hosts to acquire data, other ones use well defined interfaces or protocols like SNMP to acquire monitoring data.

Automation of monitoring is based on reducing the workload required to monitor simple problems occurring in the monitored resource. To achieve the automation many existing solutions use various methods such as fuzzy sets, neural networks, etc. These methods allow to create an expert system which will be able to resolve a given problem, it might be a monitoring automation or a generation or classification. Monitoring automation continues to attract more and more attention because the IT industry is still growing. Due to it there is still a need to create new automation for monitoring which will be more suitable and matched to specific tasks like the monitoring of distributed systems.

In this paper we focus only on a few monitoring systems which are popular on the market. The Nagios [6] monitoring tool often underlies other monitoring systems. It provides many plugins and packages which are delivered by many companies and also by this tool users. This system can monitor many different kinds of resources, also the system administrator can define a new one if needed. Nagios is one of the most popular network monitoring tools. Zabbix [7] - this is another monitoring tool which aims to help the system administrator to diagnose and monitor the network. This system uses agents to collect data from foreign hosts. Zabbix is not only an agent-based approach but it uses also network protocols like SNMP to collect data and monitor network equipment like router and switches.

Another monitoring system, the Semantic-based Autonomous Monitoring and Management (SAMM) tool uses ontologies to describe monitored resources and to manage them based on the measurable parameters.

The Saude-Net system [8]was implemented on top of Zabbix as a low-level monitoring system. It allows to automate monitoring by using of rules. These rules can be modified during monitoring with this tool, to better fit monitored failures.

At the end we want to mention Autopilot [9, 10] which is a system for automation of monitoring. It uses fuzzy logic as its knowledge engine. Autopilot is a monitoring infrastructure for dynamic performance tuning of heterogeneous computational grids based on closed loop control. This software is able to take actions which can optimize data centers and distributed applications.

All the mentioned monitoring systems feature different functionalities. Both Zabbix and Nagios are able to perform only simple actions like service restart when a failure is captured. Autopilot chooses an action based on its knowledge described by

fuzzy sets. The SAMM system features a central point of knowledge which is represented by ontologies and by agents which are able to collect data from monitored resources.

For our research we have chosen SAMM due to the benefits from using ontologies and the its ability to monitor distributed systems. Like our tool, it uses Java. SAMM provides a user interface which needs little modification for our purposes. Last but not least we try to focus in our research on an agent-based approach for monitoring automation. In its basic version SAMM possesses a kind of agents used for monitoring purposes.

3. Concept of SAMM compliant Agent (SAMM-CA)

Our main focus is on the monitoring automation. With this goal in mind we have decided to develop an agent provided with a knowledge engine which will be able to choose appropriate actions when it captures an error (failure) in one of the monitored resources. We want to create an agent which will monitor local resources only. A communication mechanism is intended to allow the system administrator to remotely manage agents. The management is supported via using the SAMM system [11] which delivers the appropriate GUI for the system administrator. Our agent is able to exchange its knowledge with other agents. This function allows for a better learning procedure, because the agent can be informed which function is used by other agents as a reaction on an error in the monitored resource. Each resource is tagged with a type which is used later in the learning procedure and in the knowledge exchange. The agent does not send the full description of an observed value but only the type of the resource which failed and the id of an action which was used. The idea is depicted in Figure 1. Other agents accept the obtained data or reject it. The behavior of the agent depends on its work mode. Later on it is discussed in more details.

Each rectangle describes only one host in the monitored infrastructure. On that host there are running a few services like a data storage service described in the agent as a resource. All these resources are monitored by the Monitoring module (step 1). This module is only responsible for data collection. Another module is the Knowledge module which obtains notifications from the Monitoring module (step 2). The Knowledge module decides which action is possible to run and which one should be used in the monitored system. If it decides which action should be used the Knowledge module passes the action id to the Execution module (step 3). The Execution module passes a list of available actions and their ids. When this module has received the action id it performs an action on the resource which was failed or on another one which should be started (step 4). The SAMM-CA passes the action id and resource tag to other agents in parallel with the action execution. It is done by the Communication module. This module obtains the data sent from the Knowledge Module (step 5). After that it transmits them to other agents in the monitored distributed system (step 6). On the right side of this figure we present the other agents. Each of them may add statistics on the executed action to its knowledge (step 7).

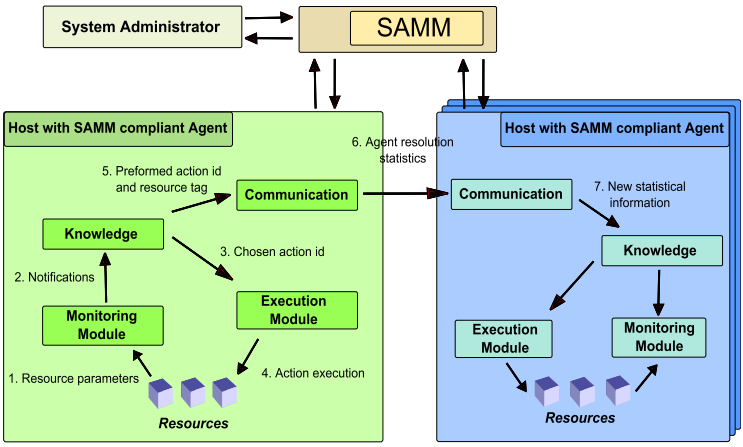

Fig. 1. The SAMM-CA communication between agents with short description of exchanged messages

The SAMM-CA consists of four major modules:

- Monitor module is only responsible for acquiring data from monitored resources
- Knowledge module is the decision component of the whole agent. It possesses all fuzzy sets and rules, both containing an action id which is used by the following module.
- Execution module is a component which is responsible for action execution. It stores all actions definitions and relevant ids. When it receives an action id it performs an appropriate action.
- Communication module enables communication between agents. It passes through the network a statistics for each agent. It allows to exchange knowledge between agents.

SAMM-CA is able to learn from the results of performed actions. After each action execution it checks the state of the monitored resource. If the monitored parameters for this resource are better than the most recent measured one, it changes the borders of fuzzy sets. If the measured parameter was in the interval of the maximum values it expands this interval. Due to it the shape of the set is closer to rectangle. But when the measured parameter is between the points which describe the left or right set border, the agent modifies the lower interval and makes this set closer to a trapezoidal shape. The rules which are used by these agents are not modified. They are used only as a static reaction on the observed failure.

4. Operation modes of the agent

We designed our agents to work in one of five modes. All of them may use a slightly different functionality. A configuration mechanism allows the administrator to change

the mode of the agent during its work. These changes can be performed via the SAMM system GUI. It is possible that when the system administrator changes the mode of the agent, it might work differently - its function can be combined from both modes (a recent and a new one) but it will last a few seconds only. After that the agent should work only in the new mode. We present all the implemented modes in Table 1. It contains a mode number and the corresponding functionality.

Table 1
Agent operation modes, each implying a different agent behaviour

Mode	Functionality (en-/dis-abled capabilities)		
	Possess History	Learn	Exchange information
1	No	No	No
2	Yes	No	No
3	Yes	Yes	No
4	Yes	Yes	Yes
5	Yes	No	Yes

The above table refers to the agent operation modes which are described more detailed:

- Mode 1 may be used when the system administrator do not want to enable the agent to modify its knowledge. In this mode our agent only reacts on the failures reported. It uses only its knowledge without any modifications. This mode is fully static. All the information that SAMM-CA possesses were supplied by the system administrator or by the previous mode.
- Mode 2 is similar to the previous one but in this mode the SAMM-CA collects data and stores it in a data base of historical data. The data contains information on the actions performed and resources failed. In this mode our agent is also unable to change its knowledge or exchange the collected information with other agents.
  Modes from 2 to 5 are based on the history of action usage and error reports.
- In mode 3 the agent is able to change its knowledge but it does not exchange (send and receive) statistical information with other agents. The knowledge changes are based on modifying the ranks of fuzzy sets. The ranks of these sets may be extended or reduced.
- Mode 4 is designed to allow our agent to be fully intelligent: in this mode it is capable of learning like in mode 3 but also it could receive statistical information from other agents in the monitored system. This information is also used in the learning phase.
- Mode 5 was developed only for the exchange of statistical data between agents. In this mode our agent is not allowed to change its knowledge. The agent can be used as an information source for other agents located in the same monitored system.

All the above modes are designed for different purposes but the most interesting one is mode 4 which uses all the functions of the agent.

5. Knowledge of the agent

As the knowledge engine the agent uses a combination of fuzzy sets and rules. The SAMM-CA uses fuzzy sets when actions are defined and performed dynamically, but it also can perform actions which should be used by default. These actions are defined by rules.

Rules can be managed or created only by the system administrator. They are used in our agent as the static knowledge. These rules define only simple actions like resource restart. This agent allows for only one rule for each monitored resource. If the system administrator defines more than one rule for the same resource then the agent may perform one of them but it is not known which one. These rules posses only a simple condition and may posses a short Java code or a unix script.

When running our agent monitors its related resources. When an error is detected the agent searches for rules to resolve it. If there are no solutions described by rules it uses fuzzy sets.

These sets relate to a defined action. Each action can possess at least one fuzzy set defined for one monitored parameter. If one action has two or more sets the intervals must not overlap. Otherwise they will be defined by the agent as one big fuzzy set with a maximum value between the sets.

In these sets each action is described by a set of values which may refer to many monitored parameters. For each parameter the agent possesses one dimension of fuzzy sets. Each set is described by nine values which are as follows:

- number of reported resource faults which were resolved by the action which is described by a current fuzzy set,
- four points which determine points of a fuzzy set - two for the value of zero (x_0 and x_4) and two for the value of one (x_1 and x_3)
- four parameters that describe a function, which are the bands of a current set. Two parameters for each function. For the left band these values are marked as `a` and `b`, for the right one these are marked as `c` and `d`.

The use of the above parameters is depicted in Figure 2.

After each modification the agent has to change the ranks of fuzzy sets. This can be done for all sets or only for the sets which are involved in the knowledge process. This is dependent on the complication of fuzzy sets and on their total number.

This modification was described earlier in this paper.

Each fuzzy set can be modified by our agent. The agent modifies the points of the upper and lower intervals. All these points reflect the functions of the borders of sets. The modification of these sets are dependent of the count of failures which were resolved by the action that is described by the appropriate fuzzy set. Each set which

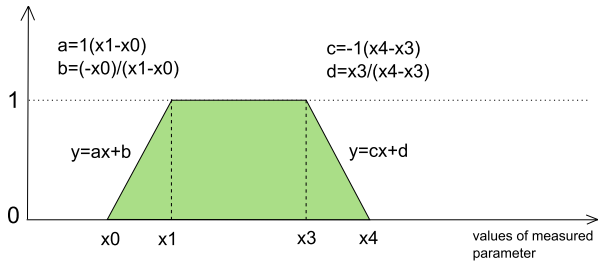

Fig. 2. SAMM-CA membership function in function of measured parameter values

describes a used action is modified whenever the agent is able to know the result of previous actions.

6. Tests

The system was tested with several scenarios.

These tests were performed on the infrastructure which contained three hosts with Unix systems. All these hosts have an installed distribution of Apache Server, MySQL and Tomcat. All three hosts were connected by a switch. On each one of them we have installed an instance of SAMM-CA. A single installation of SAMM system was placed on one of these hosts. This monitoring tool was responsible of monitoring the whole infrastructure.

The tests performed may be divided into two groups. The first group was aimed to test the functionality of the agent. During these tests we observed that all modes work fine but when the system administrator changes them it is possible that the SAMM-CA will work at the beginning slightly different than it was intended to according to a selected mode. For example when we switch the mode of the agent from mode 5 to mode 3 the agent behaved at the beginning like in mode 4. This situation held until a first error notification. Due to that we think that it is not serious problem so we let the agent to behave as described above.

Because this iteration occurs only when the agent changes its mode, which is done rarely, it is an acceptable situation and in some cases it is an expected one.

The second group of tests was performed to check the knowledge engine. In this part of tests we checked how many mistakes our system will make if it will be taught with a different number of actions. At the beginning we used a set of 100 error reports. When testing we changed the size of the learning data set. We started from a 5% learning set and proceeded up to a 80% one. The results are presented in Figure 3.

This diagram represents a test which was performed on our engine. The error was calculated as the ratio of incorrect actions to all the performed actions. Every single test is assigned an action id which describes an action which was in our opinion the best one for an observed situation.

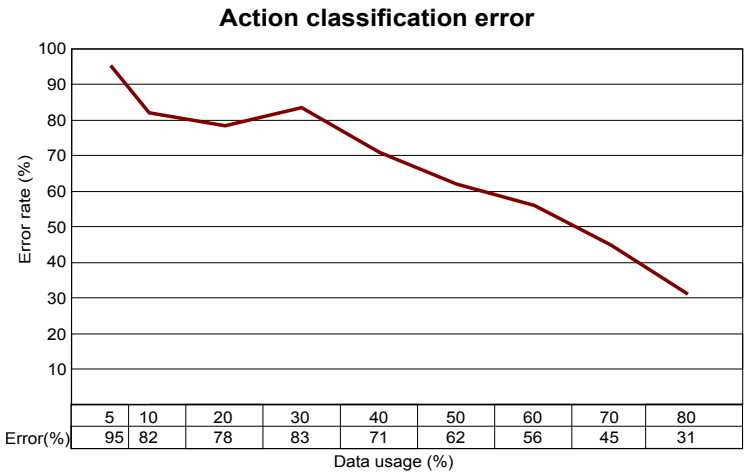

Fig. 3. Error rate in function of learning set size

The tests performed were dependent on a set test arrangement. We obtain a different behavior for the same test set but for its different arrangement.

Sometimes our agent was able only to learn a few sets. Due to it a situation which does not fall into fuzzy sets was wrongly classified, which resulted in a large error. During consecutive tests we observed that learning from more cases contributes to a better fit ratio of the agent.

Both groups of tests showed that SAMM-CA is able to react on the failures observed. It is also able to learn and improve its operation.

7. Conclusions and future work

In this paper we focused on the research aimed to create a solution which will enable to support the system administrators in their daily work. Our idea is to minimize the cost of system monitoring since it is constantly growing, due to the growing sizes of systems and companies. Our concept is concentrated on the monitoring and reacting on the failures observed in resources. The observed reactions of the system under discussion are good enough and it is able to learn and improve its knowledge. We believe that this solution may be used in both small and large enterprise infrastructures.

When testing our agent we observed that the starting knowledge and the sequence of reported errors influences the knowledge of the agent. Therefore it is very important that the system administrator should correctly and exactly set the starting knowledge of the agent. A properly set initial knowledge should result in a smaller percentage of actions which were incorrect for the situations observed.

In the future we are going to extend the functionality of the agent. We want to introduce the monitoring of non-Java applications by our agent as well as to manage them.

In the on-going research we are going to extend the knowledge handling capabilities of agents by: a function which will be able to optimize fuzzy sets. A function which is intended to allow agents to exchange knowledge and possible actions, and finally, the ability to exchange their fuzzy set ranges and defined rules. These extensions are intended to enable agents to learn from each other and to optimize their functioning and knowledge when exploiting information obtained from other agents.

Acknowledgements

The research presented in this paper has been partially supported by the European Union within the European Regional Development Fund program no. POIG.02.03.00-00-007/08-00 as part of the PL-Grid Project (`www.plgrid.pl`).

References

[1] Hellmann M.: *Fuzzy Logic Introduction.* `http://diuf.unifr.ch/ds/courses/dss2002/pdf/FuzzyLogic.pdf`, access: 16.06.2011.

[2] JBoss Community. *Drools Expert User Guide.* `http://downloads.jboss.com/drools/docs/5.0.1.26597.FINAL/drools-expert/html/index.html`, access: 16.06.2011.

[3] Cetnarowicz K., Kisiel-Dorohinicki M., Nawarecki E.: *The application of evolution process in multi-agent world to the prediction system.* [in:] American Association of Artificial Intelligence Journal, Menlo Park, USA, 1996, pp. 26–32.

[4] Cetnarowicz K.: *From algorithm to agent.* [in:] Gabrielle Allen et al., (Ed.), Proc. of ICCS, Baton Rouge, LA, USA, 2009. LNCS 5545, Springer-Verlag, pp. 825–834.

[5] Slota R. et al.: *Replica management for national data storage.* [in:] Wyrzykowski et al. (Eds.), editor, Proc. PPAM 2009, LNCS 6068, vol. II, Springer, 2009, pp. 184–193.

[6] Stone S.: *Monitoring systems comparison.* Technical report, The Forbin Group (TFG), San Francisco, USA, 2007.

[7] Kamarudzzaman K. A., Rusalan N.: *Comparison report on network monitoring systems (nagios and zabbix).* Technical report, Malaysian Administrative Modernisation and Management Planning Unit (MAMPU), 2010.

[8] Funika W., Szura F.: *Automation of decision making for monitoring systems.* [in:] K. Wiatr (Eds.) M. Bubak, M. Turała, editor, Proc. CGW'10, Krakow, October 11–13 2010. ACC-Cyfronet AGH, pp. 164–171.

[9] Ribler R. L., Simitci H., Reed D. A.: *The autopilot performance-directed adaptive control system. FGCS*, 18(1), September 2001, pp. 175–187.

[10] Ribler R. L., Vetter J. S., Reed D. A., Simitci H.: *Autopilot: Adaptive control of distributed applications, high performance distributed computing.* [in:] *Proc. the seventh International Symposium on High Performance Distributed Computing*, 1998, pp. 172–179.

[11] Funika W., Kupisz M., Koperek P.: *Towards autonomic semantic-based management of distributed applications.* Computer Science Annual of AGH-UST, 11, 2010, pp. 51–63.