Information Sciences and Computer Engineering, Vol. 1, No. 1, (2010) 53–59

PHARUS

International Journal of
Information Sciences and Computer Engineering

journal homepage: http://www.ijisce.org

Autonomic Computing Strategy for Server Virtualization

Marius-Constantin Popescu*

Faculty of Mathematics and Informatics, University of Craiova, Romania

Abstract– **This paper presents a technique specially adapted for the implementation of some techniques for self-supply and self-optimization for autonomic computing with application in server virtualization. The autonomic computing processes are considered non-linear, variable systems with constant linear parameter variations. The approach is based on the fact that a computer system behaves in time in a non-linear way, its parameters having a random variation and the components of the autonomic environment can fail occasionally.**

Keyword: Redundant structures, automatic control, server virtualization.

1. Introduction

Definition of the usual time fault-tolerant electronic systems, is that the correct implementation of the system continues to function entry / transfer / exit, in the presence of a certain set of faults that may occur during operation without corrective action from the outside [1, 2, 3]. Fault-tolerant systems, according to this definition, assumes on the theory that the system design faults were removed before entry into service. The actions of tracking system defects can be considered as included in the diagnostic algorithm of failure, distinguishing between the following approaches [4, 5]:

a) The initial test is carried out before putting into service of equipment and allows identification of hard defects introduced during manufacturing processes, design errors or software errors.
b) Testing on line takes place simultaneously with the normal operation of the system and can be implemented with both hardware and software means This operation involves the use of error detecting codes, doubling the elements, comparing the output variables, the use of surveillance systems such as maintenance microprocessors that run monitoring system software. The main advantage of online testing is to detect and diagnose faults before the occurrence of significant damage to the system.
c) Testing the redundant modules must determine whether the backup modules are able to take over the functions of the functional modules. For this purpose it is used either online testing methods (self-testing systems) or off-line testing methods (diagnostic software for preventive maintenance) depending on the type of protective redundancy used. Redundancy is a convenient method for increasing the reliability of the electronic systems if the elements malfunctions that compose the system are independent events. But practice has shown that sizing redundant systems, the simplifying assumption of independence of failure, leading to inaccurate assessments. Therefore for a realistic design of complex electronic systems with redundant structure is necessary to take into account the effect of dependent failures of common node.

Techniques to achieve structural redundancy can be classified as [6, 7]:

a) Static redundancy (masking defects techniques) involving the coding of the logical functions of the system with redundant codes and techniques of errors recognition and instant masking effect of a defect in the system, by activating the automatic backup elements. This multiplication of reserves in the system structure can be achieved from component level to the whole system level (global and individual structures). The main advantages offered by the application of static type redundancy can be summarized as follows: instant corrective action, prior diagnosis of defects is not necessary, switching from non-redundant systems to the same system with redundant structure is relatively simple. It should be noted however, that digital systems redundant and static structure raises numerous problems of synchronization between system modules and introduces many fan-in and fan-out limitations.
b) Dynamic redundancy (switching) is the devoted term of techniques that involves using multiple back-up modules with the same functions, of which only some are operational, others are in stand-by (to be connected only in the moment of identifying a failure). This technique is used to achieve self-preparation of the systems by automatically switching the reserves, or performing reconfiguration when the system must be reorganized in a different layout than the original. Using dynamic redundant structure has the following advantages: there is needed to be electrically powered only one back-up module; the switch that performs the activation stock in the system provides a insulation of the defects; the number of reservations system can be op-

*Corresponding author:
Email address: popescu_ctin2006@yahoo.com, Ph: +40 745438287

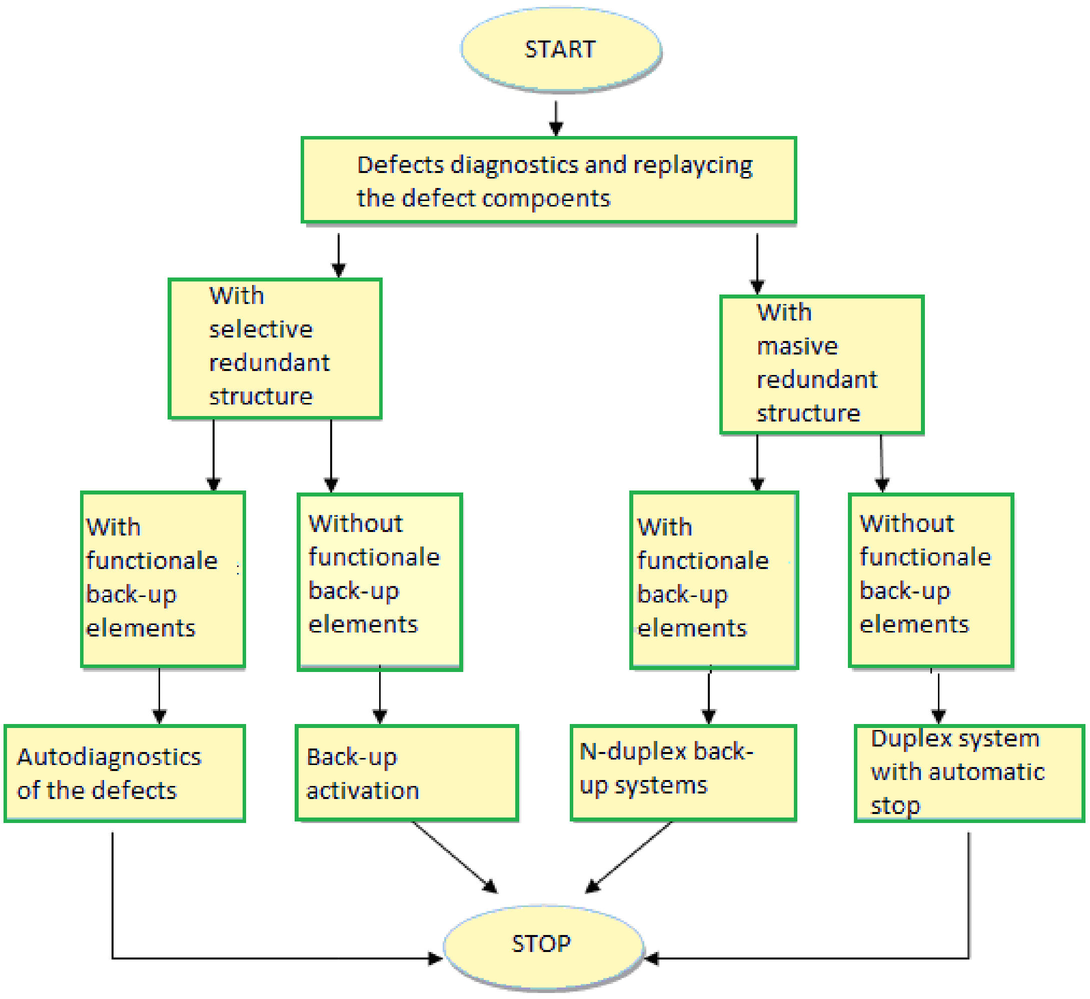

START
Defects diagnostics and replaycing the defect compoents
With selective redundant structure
With masive redundant structure
With functionale back-up elements
Without functionale back-up elements
With functionale back-up elements
Without functionale back-up elements
Autodiagnostics of the defects
Back-up activation
N-duplex back-up systems
Duplex system with automatic stop
STOP

Fig. 1. The diagram of an algorithm for detection and diagnosis of faults.

timized for a given task, without changes in the underlying structure; does not introduce any construction limits as the static redundancy structure, regarding the equipping the digital systems.

2. Reconfigurable Computing Architecture for Autonomous Systems

In the present, the data structure of an enterprise is very complex so that the operation, functioning and the maintenance require considerable material and intellectual efforts. The idea of building an intelligent information management structure has long been developed in academic and industrial environment. A higher degree of automation in monitoring and controlling all the elements of the information structure is given by the principles of autonomous computers [8, 9]: self-configuration, self-protection, self-optimization and self-preparation.

In Fig. 2 is presented a structure of constructive elements of an autonomous system and interaction between them. This architecture is similar to the automatic control systems. Although the approach is general and can be applied to many situations, it can analyze the main times of measurement from the multitask applications. They may be elements of trading in banking, insurance or data systems.

Fig. 3 shows the interconnection structure of a self management system that has a single resource (called an element of management). In this architecture, the resource can be a single computer system, a software application hosted on a computer system, or a collection of more logic related systems (distribution computer system).

The sensors provide measurement data acquisition of the resources and ensure the implementation of the means to modify the times of behaviour of the resources.

Current research on developing optimal architecture of an autonomous computer system for a specific application is based on the development of requirements on which constituent elements are built.

It proposes a set of seven basic elements for building an autonomic computing system, each with a clear function, well defined and distinct [10, 11]:

Sensor Component - Is responsible for notification and data acquisition from the entity or entities managed and that describe quantitatively the state of the managed systems. Such a component comprises several sensors, if they are to the supervision of various entities.

Filter Component - Filter data from sensors by eliminating noise or making changes requested by the controller who processes them. Filter component implements a series of filters that allow processing through a series of consecutive filters.

Coordinator Component - Organize the process of predic-

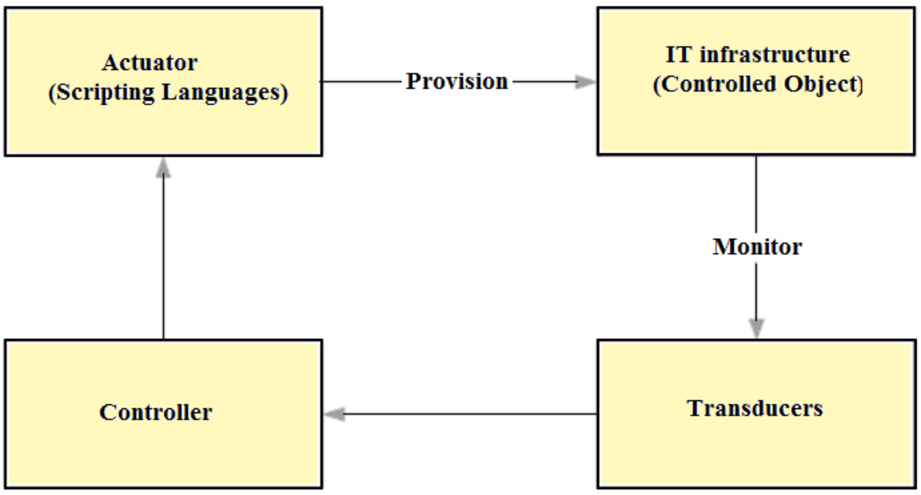


Fig. 2. The structure of an autonomous computing environment.

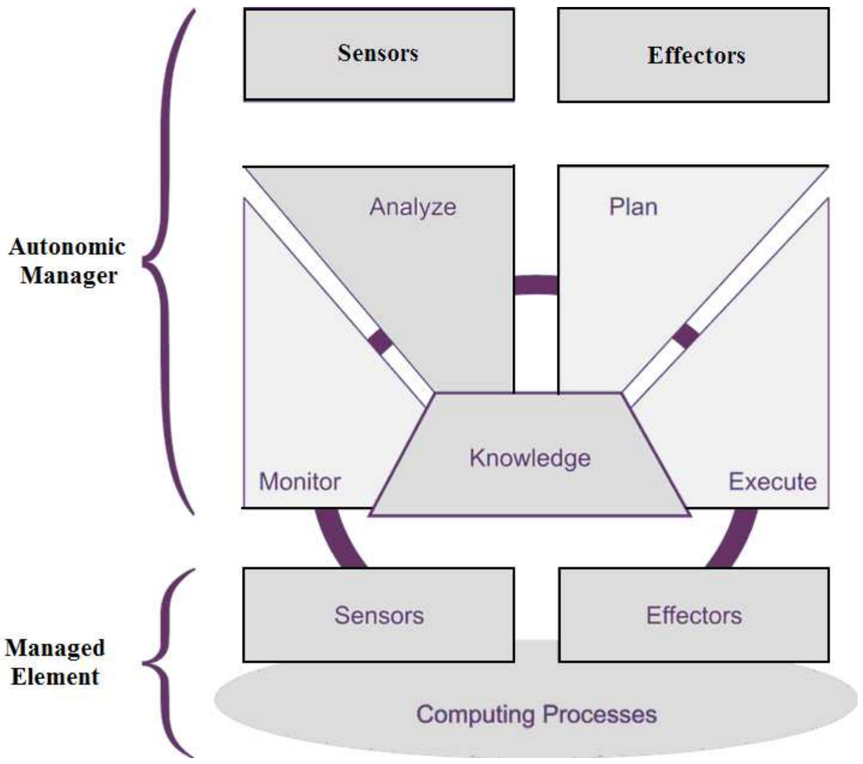


Fig. 3. A generic architecture for autonomous computing systems.

tion. It receives data from the filters, uses the modelling block, the estimator and the decision block, for predictive deciding and sends this decision to the actuators. Once known the decision sensors are added or removed.
Modelling Block - Notes the model entity or entities generated. It is used by the estimator and by the decision block. It can update the model given a prior estimation or of changes to the modelled infrastructure.
Estimator – makes estimations on future modelled system state on the current status, or if necessary, it is using informations on previous states.
Decision Block - Make predictive decisions based on estimated data.
Actuator Block – activates the state of the actuators based on signals provided by the decision block. It comprises several actuators.

Control theory uses a simple reference architecture and generally applicable. The main objective of this architecture is to provide components to handle a target system to reach the wanted forecasted target. Figure 4 shows the reference architecture. The handling component of the system target is called a ”controller”.

The target system is the computerized resource, the controller is an independent leader and the controlled target is part of the knowledge component, regarding the autonomous computer architecture.

The other components of the reference architecture are:

A reference input r(k), which is the desired value of the output

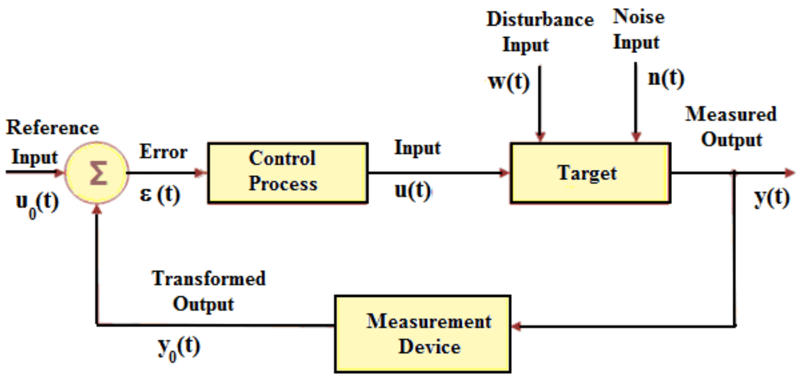


Fig. 4. Block diagram of a control system.

size (measured) or a transformation of it. Examples include the desired response time, the CPU utilization, and the descent time in a trust frame. A control error e(k), the difference between the reference input and transformed output. The control error is used as input for the controller component. A control input u(k), which is determined by the controller component and used to influence the behaviour of target system. The control input will sometimes influence target system through actuator. Examples include the number of servers in a scattered placement of conduct and grant parameters for the software. A measured output y(k), which is a measurable characteristic of the target system. Examples include CPU utilization and response time of the application. The entry noise n(k), which can be any changes affecting the measured output produced by the target system. Output disturbance d(k), which is any change that affects how the input control influence the measured output.

The examples include a backup process or a virus-scan process running during the measurement. The target system, which is the computerized system to be controlled. Measurement Unit, which converts the measured output such that it can be compared with the input reference. An example of transformation is filtering the measured output to reduce stored noise and low pass filtering to exclude high-frequency data. The Controller, which is the component that determines the setting of input control necessary to achieve the desired value as set by the input reference. The controller calculates the values for the input control based on current values, or in the past, of the control errors [12].

3. Automatic Control in the Virtual Environment

The technique applied in this case is based on real-time measurement of the values of the load generated by the central unit for the running processes on each server, by calculating the ratio between the two. Based on the known coefficient from the stage of allocation of the tasks, the processing central unit coefficients are adjusted as the servers serve the individual customer requests. Since the reallocation of the coefficients of the central processing unit is a slow process and should respond to long-term changes of the load, the response mechanism must eliminate all the rapid changes of the measured parameters, such as the central processing unit load. As a result, the output of the transducer is conditioned, calibrated and placed in a mediation filter. In this experiment autonomous computer system architecture considers that the computer system whose output is the central processing unit load, is nonlinear, stochastic and with slow parameters. The model of the system under consideration is described by a nonlinear discrete time system having both parametric and non-parametric uncertainties. For all servers running on the same hardware support, the adaptive robust controller calculates the percentage of allocation of central processing unit. Considering that on each support of each computer hardware there are running j different virtual servers then the system will run autonomously computing j different robust adaptive controllers. However, it is not a realistic solution because it involves much consumption of the computing power for the load control. Therefore the existing computer resources will be allocated to active processes. It will be considered separately for each hardware will be calculated at intervals depending on the slow dynamics of the server process a single adaptive robust law governing.

4. Implementation

The architecture presented above for self-configuration of the central processing unit's load using a robust adaptive controller was implemented and tested on a computer network operating in real time. The test configuration is shown in Fig. 5. The virtual server was put into service by installing a program on a computer with VMM Intel Core 2 Duo.

The actuator of the whole system, represented by IBM Tivoli Provisioning Manager (TPM) was installed on another IBM computer with a processor operating at 2.6 GHz. Virtual computers were obtained by virtualization with Debian Etch Linux. Autonomic computing environment components were implemented as Web services, all running on a computer with a 3.2 GHz Intel processor. To configure the entire system it was designed a graphical user interface. For implementation there were used Java Server Faces libraries. The robust adaptive controller was included in the checklist interface, as exemplified in Fig. 6.

The controller is selectable it from a list of synthetic control algorithms. Until a few controllers are currently available: Convergence Notification Controller, Robust Adaptive Controller and PM Controller, which is a classical proportional controller. For the simple architecture of virtual servers that does not require the installation of hardware and complex software, you can use a simple control law, which gives good results for the control of the coefficients of the central processing unit. Sophisticated

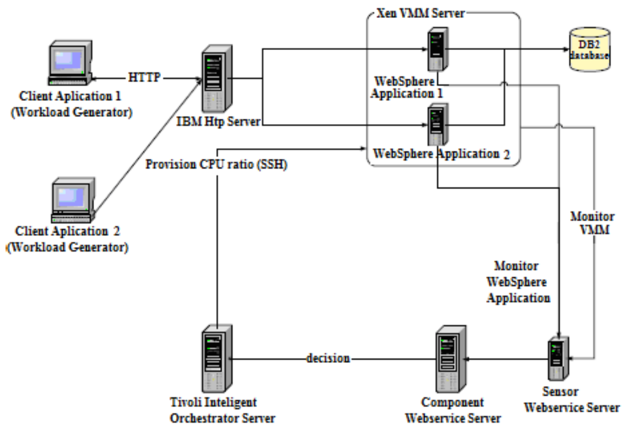

Xen VMM Server
DB2 Database
HTTP
Client Aplication 1 (Workload Generator)
IBM Htp Server
WebSphere Application 1
WebSphere Application 2
Provision CPU ratio (SSH)
Client Aplication 2 (Workload Generator)
Monitor VMM
Monitor WebSphere Application
decision
Tivoli Inteligent Orchestrator Server
Component Webservice Server
Sensor Webservice Server

Fig. 5. Test configuration used.

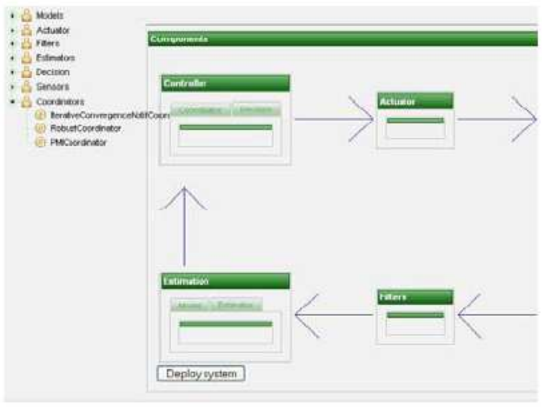


Fig. 6. Graphic interface with the user hand machines for setting the type of controller in the autonomous computing system.

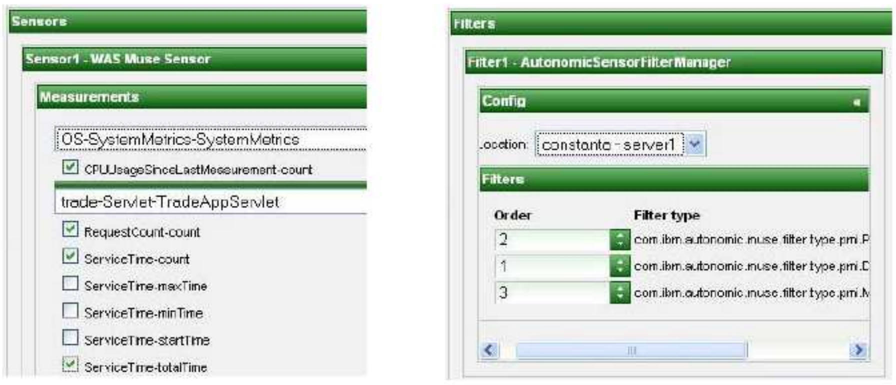
algorithms can be used for complex architectures of servers, operating in critical systems. Design tools developed as shown in Fig. 7a, allow configuration of several control models. To configure the system, the system engineer selects with the mouse components while the components of the system can be connected and turned on during operation. The user must connect the components through arrows. Fig. 7a Configuration window shows the original state of the sensors. The load of the central processing unit, response time and number of calls are set to provide the coordinator corresponding measured values. The filter is also configured with a configuration window that is shown in Fig. 7b.

Estimator is configured using the windows in Fig. 8. The configuration of the estimator includes choosing a model, as shown in Fig. 8a, where is shown how the filtered variables are transformed into model variables of the filtered variables. The parameters to be set are the number of iterations required to achieve a best possible estimate for the variables modelled, the convergence measure and sampling period as explained in this part. The control law is interpreted and a decision is made to change the central processing unit's load according to certain thresholds that are used to alleviate the rapid changes of the central processing unit's load.

The controller sends the appropriate commands to the actuator represented by IBM Tivoli Provisioning Manager. Based on values obtained by comparing the control achieved by the robust adaptive controller and mitigating factors represented in the form of thresholds, the system provides the actuator how to act. The engineer system engineer must indicate to the system which actuator is used and on which physical computer is it so that it can launch all commands related to configure the entire computing environment autonomously and to specify what parameter must be used for the initialization of the actuator. Set up links between adaptive robust controller and IBM Tivoli Provisioning Manager is done through a window of configuration, as shown in Figure 9.

5. Conclusion

Based on the study it can be concluded that the implementation of redundant structures in order to ensure tolerance to malfunctions is the most modern method of increase in reliability of complex digital electronic equipment. Computers tend to incorporate autonomous structures in automatic data management systems so as to adapt to the changes in configuration, protection and reconfiguration of available resources during operation. One of the current research directions consider autonomous computers as an adaptive control system that solves the limitations imposed by optimal use of available resources as a result of external requests. The proposed application for testing ”The automatic control in the virtual environment” is a simulation of the performance obtained using the robust adaptive control algorithm and method. In this experiment, it was proposed an original architecture for autonomic computing system whose active task is a central processing unit with slow time-varying parameters. To get more features in the simulation, it was assumed that every


Sensors
Sensor1 - WAS Muse Sensor
Measurements
Filters
Filter1 - AutonomicSensorFilterManager
Config
Filters

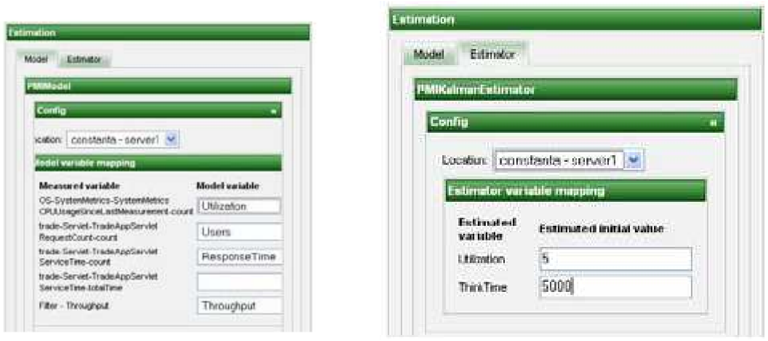
OS-SystemMetrics-SystemMetrics
☑ CPUUsageSinceLastMeasurement-count
trade-Servlet-TradeAppServlet
☑ RequestCount-count
☑ ServiceTime-count
☐ ServiceTime-maxTime
☐ ServiceTime-minTime
☐ ServiceTime-startTime
☑ ServiceTime-totalTime

Location: constanta - server1

Order	Filter type
2	com.ibm.autonomic.muse.filter.type.pmi.P
1	com.ibm.autonomic.muse.filter.type.pmi.D
3	com.ibm.autonomic.muse.filter.type.pmi.N

a) b)

Fig. 7. Window configuration: a) the sensor; b) filters.



Location: constanta - server1

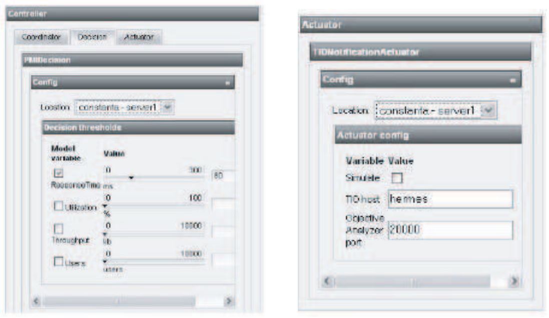
Measured variable	Model variable
OS-SystemMetrics-SystemMetrics CPUUsageSinceLastMeasurement-count	Utilization
trade-Servlet-TradeAppServlet RequestCount-count	Users
trade-Servlet-TradeAppServlet ServiceTime-count	ResponseTime
trade-Servlet-TradeAppServlet ServiceTime-totalTime	
Filter - Throughput	Throughput

Location: constanta - server1

Estimated variable	Estimated initial value
Utilization	5
ThinkTime	5000

a) b)

Fig. 8. Window configuration: a) allows the user to choose a right model; b) corresponding to the processing time and estimator use.



Location: constanta - server1

Model variable	Value	
☑ ResponseTime	0 – 300 ms	80
☐ Utilization	0 – 100 %	
☐ Throughput	0 – 10000 t/b	
☐ Users	0 – 10000 users	

Location: constanta - server1

Variable	Value
Simulate	☐
TIO host	hermes
Objective Analyzer port	20000

a) b)

Fig. 9. Window configuration: a) the parameters for the decision block decision; b) actuators servers.

hardware support for each computer connected to cluster there are running different virtual servers whose operating is managed by an adaptive, robust, autonomous computer system.

References

[1] H. Cârstea, D. Margeloiu, and C. Mitaru, "Redundancy and testability in digital filters," *Bul. Stiint. Univ. Politehnica Timisoara, Seria Electronica si Telecomunicatii, Tomul*, vol. 53, no. 67, pp. 204–206, 2008. Fascicola 1.

[2] M. M. aescu, "Contribuii la creêrea disponibilitãii sistemelor digitale complexe utilizând reconfigurarea autonomã dinamicã," (Timisoara), 2009.

[3] M. C. Popescu, O. Olaru, and V. Balas, "Identification of the de-synchronization, synchronization and forced oscillation phenomenon of a nonlinear system," *WSEAS Transactions on Systems and Control*, vol. 4, pp. 177–187, April 2009.

[4] H. Cârstea, O. Mitaru, and R. Negrea, "Method for computing a reliability function of digital systems with redundant structure through coding," *ISSE 2008, 31st International Spring seminar on Electronics Technology*, pp. 48–53, 2008.

[5] H. Lagar-Cavilla, J. A. Whitney, A. M. Scannell, P. Patchin, S. M. Rumble, L. Eyal, M. Brudno, and M. Satyanarayanan, "Snowflock: rapid virtual machine cloning for cloud computing," *EuroSys*, pp. 1–12, 2009.

[6] M. C. Popescu, O. Olaru, and N. Mastorakis, "Processing data for colored noise using a dynamic state estimator," *WSEAS Transactions on Communications*, vol. 8, pp. 321–330, March 2009.

[7] B. Solomon, D. Ionescu, M. Litoiu, and M. Mihaescu, "Toward a real-time reference architecture for autonomic systems," *29th International Conference on Software Engineering and Workshops*, pp. 1224–1231, 2007.

[8] M. Litoiu, M. Mihaescu, B. Solomon, and D. Ionescu, "Scalable adaptive web services," in *Proceeding of SDSOA*, pp. 1107–1112, Leipzig, May 2008.

[9] A. A. Soror, U. F. Minhas, A. Aboulnaga, K. Salem, and S. K. P. Kokosielis, "Automatic virtual machine configuration for database workloads," *SIGMOD Conference*, pp. 953–966, 2008.

[10] N. M. M. C. Popescu, O. Olaru, "Equilibrium dynamic systems intelligence," *International Journal of Mathematical Models and Methods in Applied Sciences*, vol. 3, no. 2, pp. 133–142, 2009.

[11] B. Solomon, D. Ionescu, M. Litoiu, and M. Mihaescu, "A real-time pattern based approach to autonomic computing," *SEAMS ACM Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pp. 78–85, May 2007.

[12] M. Steinder, I. Whalley, D. Carrera, I. Gaweda, and D. M. Chess, "Server virtualization in autonomic management of heterogeneous workloads," *Integrated Network Management*, pp. 139–148, 2007.

Marius-Constantin Popescu was born in Gorj – Tismana, on the 19-th of may 1965. In 1990, has gratuated the University of Craiova, Romania, Faculty of Automation and Computer and in 1995 he graduated from the Faculty of Mathematics and Informatics, University of Craiova. Author and co-author of 175 scientific papers, 10 textbook and 11 books. In 2003 has received scientific degree Ph.D, he is currently Associate Professor of University of Craiova, Romania. Professional skills: measurements in industrial processes, fuzzy system modeling, artificial intelligence, optimization and automation of industrial processes, web intelligence, intelligent communication network control.