22th International Scientific Conference
Strategic Management and Decision Support Systems
in Strategic Management

May 19, 2017, Subotica, Republic of Serbia

Dejan Zdraveski
Faculty of Economics
Prilep, R. Macedonia

Margarita Janeska
Faculty of Economics
Prilep, R. Macedonia

Gjorgi Manceski
Faculty of Economics
Prilep, R. Macedonia

Kosta Sotiroski
Faculty of Economics
Prilep, R. Macedonia

PERFORMANCE EVALUATION OF AUTONOMIC COMPUTING SYSTEMS

Abstract: Autonomic computing systems is a new paradigm and they are characterized with the self-management properties. Autonomic computing systems are emerging as a significant new strategic comprehensive type of information systems, aimed at the design of complex distributed computing systems. Their work was inspired by the functioning of the human nervous system and is intended for designing and building self-management systems. These systems are continually evaluated in terms of their optimization and automatically adjust to changing conditions. Meeting these challenges, autonomic computing systems requires scientific and technological achievements in various fields, also new architectures that will support the effective integration of technologies used in these systems.
For the realization of self-activity in the autonomic computing systems is required to provide appropriate systems, technologies and services. In recent decades is characteristic extended parallel and distributed computing that support network environment. The technologies of autonomic computing systems can integrate with existing programming systems, in order to realize self-management applications.
The aim of this paper will be to define systems, technologies and services that are necessary for realization of the basic concepts and functions in autonomic computing systems and applications. It also would include the implementation of these systems, in terms of techniques of autonomic computing system that can be used with dynamically allocated servers to provide maximum functional usefulness. Also, will be presented system with self-managed properties that will be capable for reallocation of servers in case when there are changes in overload or when an error occurs on the server. The end of the paper will be presented performance evaluation and metrics of the autonomic computing systems. Namely, it is necessary to determine the indicators through wich will evaluate the performance and effectiveness of autonomic computing system.

Key words: autonomic computing system, technology, service centers, metrics, evaluation.

1. INTRODUCTION

The autonomic computing paradigm has been inspired from the autonomic nervous system that continuously regulates and protects our bodies subconsciously, leaving us free to focus on other work. Similarly, an autonomic system should be aware of its environment and continuously monitor itself and adapt accordingly with minimal human involvement. Human managers should only specify higher level policies that define the general behavior of the system. This will reduce the cost of management, improve performance, and enable the development of new innovative applications. Autonomic computing systems must (IBM, Autonomic Computing 2005):

- to have knowledge (for its components, status, capacity, the context of this activity and for other resources within the infrastructure),
- to be able to sense and analyze environmental conditions and
- to be able to plan for and affect change by altering its own state.

These systems are characterized by following properties: self-configuration, self-healing, self-optimization, and self-protection. Self-configuration is refers to a variety of responses of a system, such as changing network topologies or changing and setting up various software and hardware components. Self-healing is focused on maintaining or restoring a system's safety properties. Self-optimization focuses on the value of an objective function as the system property, and aims at maximizing (or minimizing) this objective function. A system is self-protecting when it autonomously anticipates, detects, identifies, and protects against malicious attacks or cascading failures to maintain overall system security and integrity.

The architecture of autonomic computing contains two main entities: autonomic manager and managed resources. It contains an autonomic control loop that domineer the flow of work done between sub constituents of autonomic elements. In a following figure is presented the control loop that is the core of the autonomic architecture:

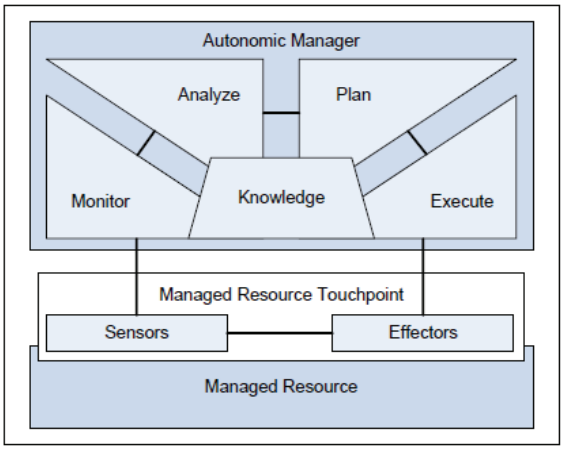

Figure 1. Autonomic computing control loop
Source: IBM International Technical Support Organization, 2005

In the autonomic control loop, the managed element represents any software or hardware resource. Sensors collect information about the managed element. Effectors carry out changes to the managed element. The knowledge used by a particular autonomic manager could be created by the monitor part, based on the information collected through sensors, or passed into the autonomic manager through its effectors. The most important component is autonomic manager. The role of autonomic manager is to monitor the events and actions using sensors. In the previous figure are presented four primary function of an autonomic manager:

- The monitor function – collect, aggregates, correlates and filters details from managed resources;
- The analyze function - provides the mechanisms to observe and analyze symptoms to determine if some change needs to be made;
- The plan function - creates or select a procedure to change managed resources; and
- The execute function - provides the mechanism to schedule and perform the necessary changes to the system.

The correlation between self-properties and autonomic architecture can be seen from the following figure:

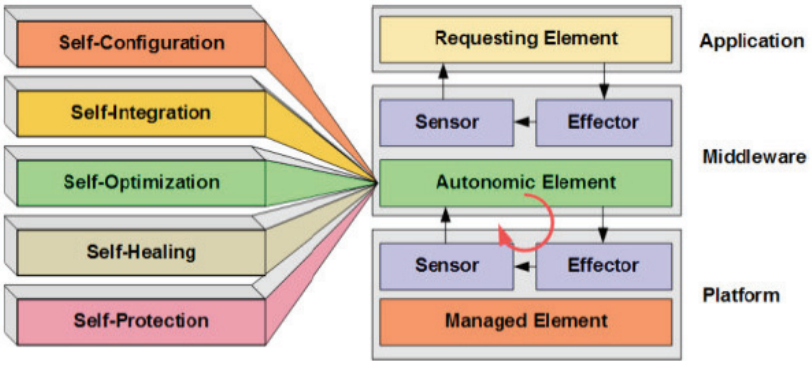

Figure 2. Correlation between self-properties and autonomic architecture
Source: Sakthi Nathiarasan A, 2014

2. SYSTEMS, TECHNOLOGY AND SERVICES FOR REALIZATION OF SELF-PROPERTIES IN AUTONOMIC INFORMATION SYSTEMS AND APPLICATIONS

In the last few decades is characteristic adapted/extended parallel and distributed computing to support grid environments. These can be classified as communication framework of distributed object systems, component-based systems and service-based systems. Current communication frameworks for distribute and parallel computing (i.e. message passing model and shared memory models) supplement existing programming systems to support interactions between distributed entities. These systems typically make very strong assumptions about the behavior of the entities and their interactions, especially about their static nature and reliable behaviors, which limit their applicability to highly dynamic and uncertain computing environments.
Current component-based programming systems, such as CORBA Component Model (CCM), Java Beans and Ccaffeine Common Component Architecture (CCA), also not directly support adaptation (Szyperski C, 2002).
Following table summarizing the main differences between object-oriented and component-based systems:

Table 1. A summary comparison of component-based and distributed-base systems

Comparison Factors	Object-oriented systems	Component-based systems
Flexibility	Less flexibility in term of hardware and software	More flexibility in term of hardware and software
Reliability	Dependes od developers	Thread safe and secure
Building strategy	Composition and inheritance are both used	Composition is major, and inheritance is unnecessary
Deployment	Monolithic software applications	Independent parts of software
Interoperability	Development is restricted with one or more technologies on one platform	Provides communication between different technologies on different platform

Comparison of component-oriented and service-oriented systems in terms of process, technology, quality and composition are summarized in follow table:

Table 2. Summary of similarities and difference of component-based and service-based oriented systems

	Component-based systems	Service-based systems
Process	- Building system from preexisting components; - Separate development process of components and system; - More activities involved in design time.	- Building system from preexisting services; - Separate development process of services and system; - More activities involved in run time.
Technology	- Constrained by component models; -Ranging from white box, gray box to black box. Static and dynamic binding between components; -Dynamic discoverability is not a major concern.	- Platform independency; -Black box; -Only dynamic building between services; Dynamic discoverability.
Quality	- Interoperability concern between heterogeneous components; -Achieve component substitutability through explicit specification; - Bettter predictability.	-Interoperability through universally accepted standards; -Achieve service substitutability through service description; - Predictability issue.
Composition	- Homogenous composition; -Design time and run time composition and design time composition allows for optimization; - Pipe and filter, event mechanism etc.; -Composition is made out of several component instances.	- Heterogeneous composition; -Services are composed at run time; - Pipe and filter, orchestration etc.; -Composite services are built by composing service description.

Autonomic computing technology can be integrated existing programming systems to realize self-management applications. This approach involves three key steps: (1) Specifying adaptation behaviors, (2) Enforcing adaptation behavior and (3) Conflict detection and resolution. (P. Boinot, 2000)
An autonomic service (shown in follow figure) consist of WS-resource, providing functionalities and stateful information, a coordination agent sending and receiving interaction messages for the associated WS-resource, and a

service manager that manages the runtime behaviors WS-resource and its interactions with other autonomic services. (Foster I. and all, 2004). The coordination agent acts as a programmable notification broker for the associated WS-resource.

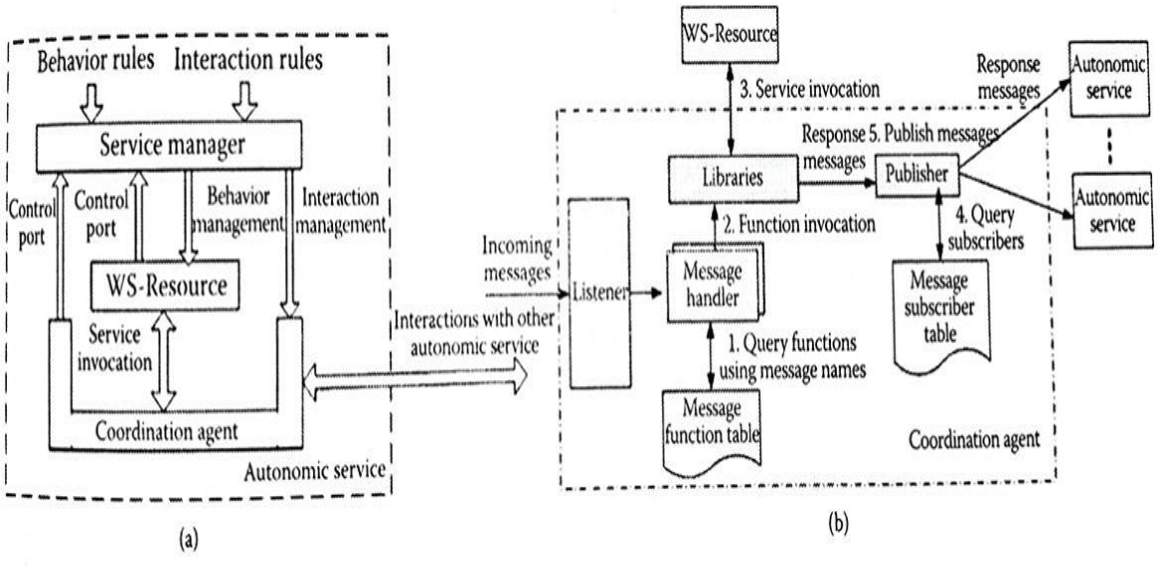

Figure 3. Autonomic service
Source: Foster I. and all, 2004

As shown in figure, a coordination agent consists of four modules that work in parallel: (1) listener module, that listens to the incoming messages from other autonomic services, (2) message handlers, that process the messages using functions defined in the message function table, (3) libraries, that provide functions for processing messages and involving the associated WS-resource and getting response messages, and (4) a publisher, that sends that response messages to the subscribes.

3. EVALUATION AND METRICS OF AUTONOMIC COMPUTING SYSTEMS

In autonomic computing, as in any other domain, it is important to be able to evaluate and compare different solutions. However, today there is no common and generally usable approach. Anyway, it is necessary metrics that allows evaluating and comparing the performance of solutions on multiple characteristics related to autonomic computing.
In autonomic computing there is a quantitative and qualitative evaluation, as in any other domain. Qualitative evaluations, is comparing different systems characteristics or properties on a non-numeric and discrete scale. This is often a subjective approach but can have more meaning than a single number. Quantitative evaluations, is mostly an objective approach but a single number is not expressive enough to evaluate every characteristic or property of the system. Therefore, it is best to combine the qualitative evaluation by quantitative. In this paper we suggest follow set of metrics:
Costs. Autonomic computing costs, and their measurement include many aspects. Agent-based systems typically compare the amount of communication, actions performed, and cost of actions required to reach the goal, while for many commercial systems the aim is to improve the cost of running an infrastructure, which includes primarily people costs in terms of systems administrators and maintenance. This means that the reduction in cost for such systems cannot be measured immediately but over time and as the system becomes more and more self-managing. (Nussey I., Telford R., 2003). Cost comparison is further complicated by the fact that adding automaticity means adding intelligence, monitors and adaptation mechanisms – and these cost. (McCann J .A., Jawaheer G., 2003). The actual architecture can also impact in the measurement of the cost of a self-adaptive system. Namely, in this context, costs could be in terms of extra hardware and communications to that hardware node. Also, autonomic computing system typically consists of networks of sensors working together to create intelligent homes, monitor the environment etc (McCann J .A. 2003). That means that the cost of autonomic computing involves resource consumption. There are a number of cost-related metrics such as business cost, i.e. savings on administrator personnel; or overhead cost, i.e. cost of extra autonomic activity inside the system (cpu usage, storage). The cost index, denoted as Index Cost, can be calculated as the average of internal (e.g. CPU usage, storage, etc.) and external cost (e.g. overhead of knowledge monitoring, distributed intelligent adaptation, etc.). In follow figure each axis depicts the extra percentage needed (savings on administrator personnel, CPU usage, storage) compared to the same system without the autonomic capabilities:

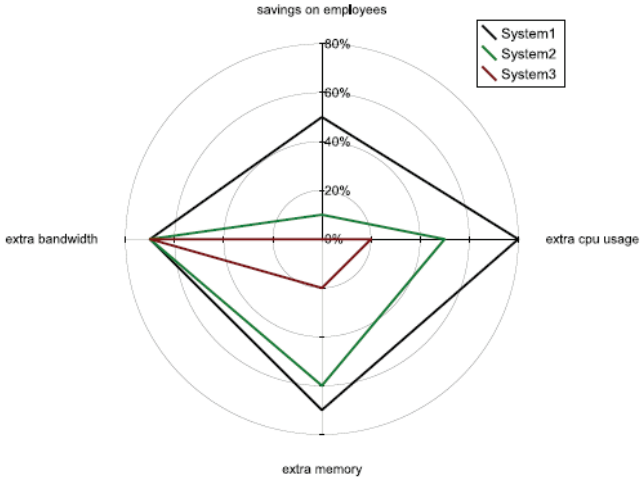

Figure 4. Cost of autonomic computing
Source: Wolf T., Holvoet T., 2006

Granularity/Flexibility. The granularity of automaticity is an important issue for comparing autonomic systems. Granularity is the degree of distribution of the intelligence. The distribution impacts the management responses. Also, granularity is in correlation with decentralization of the autonomic control. Granularity is important where binding and loading a component took a few seconds. Fine-grained components with specific adaptation rules (i.e. decentralised control) will be highly flexible. Many current commercial autonomic endeavours are at the thicker high service level. This is one of the eight characteristics of autonomic systems. Typically many autonomic systems are designed to avoid failure at some level. Many are designed to cope with hardware failure such as a node in a cluster system or a component that is no longer responding. Some avoid failure by retrieving a missing component. Either way the predictability of failure is an aspect in comparing such systems. Some systems will be designed for their ability to cope with predicted failure e.g. using a mean time before failure metric of hardware and others to cope with unpredicted environments. To measure this, the nature of the failure and how predictable that failure is, needs to be varied and the systems' ability to cope measured. Ability to cope could be in terms of a quality of service metric that pertains to the application domain.

Degree of Autonomy. There are two approaches for evaluation autonomic system for this metrics. First approach is behaving on *coverage of self-* requirements (*self-configuring, self-healing, , self-protecting, and self-optimizing). For each self-* property there are a number of related requirements. Example, on the follow figure is show that system1 achieves all self-protecting requirements and only 25 percent of the self-configuring requirements, while system 3 achieves 75 percent of the self-configuring requirements and self-optimizing requirements and only 50 percent of the self-protecting requirements. The second approach is behaving on degree of autonomy of autonomic manager function: monitoring, analysis, planning and implementation for each of autonomic requirement. Namely, for each autonomic requirement it is estimated degree of autonomy of separate functions. For example, one system can perform more autonomic requirements of planning and analysis functions, while another system perform more autonomic requirements of monitoring function.

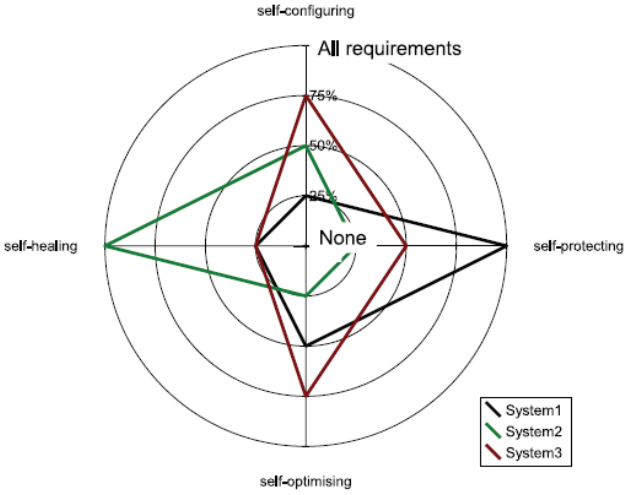

Figure 5. Coverage of self-requirements
Source: Wolf T., Holvoet T., 2006

Adaptability. Adaptability can be seen in the ability of the system reconfiguration when to replace a components, and the system working during a reconfiguration. Some systems are designed to continue execution whilst reconfiguring, while others cannot. Furthermore the location of such components again impacts the performance of the adaptivity process. So, adaptability is taken as metrics of autonomic computing systems.
Time to adapt, reaction time and stabilization time. Time to adapt is time a system takes to adapt to a change in the environment. That is, the time taken between the identification that a change is required until the change has been effected safely and the system moves to a continue state. Reaction time can be seen to partly envelop the adaptation time. This is the time between when an environmental element has changed and the system recognizes that change, decides on what reconfiguration is necessary to react to the environmental change and get the system ready to adapt. The reaction time affects the sensitivity of the autonomic system to its environment. The stabilization time is time between when the system starts adaptation and the moment where the system is back in a ready state or in other words a stable state.
Sensitivity. This is a measurement of how well the self-adaptive system fits with the environment. At one extreme a highly sensitive system will notice a change as it happens and adapt to improve itself based on that change. High sensitivity contributes to faster response to changes.
Learning index. It is ability to learn from past experiences to enhance future operations. Learning allows the network to cope with unpredicted situations. As a consequence, it becomes more difficult to verify behavior and operation of the system since there is no clear optimal state towards which the system converges. In this context we can to define two extremes of the learning capability as follows: closed-adaptive and open-adaptive. A system is called closed-adaptive when it is not enabled with learning ability. Open-adaptive architectures learn from past experiences to continuously evolve and enhance their applied adaptation strategies. This could be achieved by applying some artificial intelligence techniques. Learning index will be ratio of the number of learnable management objects, versus the total number of network management parameters. This index can be relative number of well-performing learning-based decisions. (G. N. Yannakakis, J. Levine, J. Hallam, and M. Papageorgiou, 2003) The learning index is a number between 0.0 and 1.0 which indicates the overall learning ability of the architecture. The minimum value of 0.0 corresponds to a closed-adaptive system where. A complete open-adaptive system gives the value 1.0 for the learning index.
In this section we emphasize the major evaluation metrics (qualitative and quantitative) of autonomic information systems. Of course, it is not a final list of metrics, given the fact that this issue is multidimensional and there are many aspects to choosing the right metrics, from the aspect of: networking, communication, centralization, architecture, intelligence, self-management characteristics, the autonomic manager functions, workload and so on. Namely, single metrics measured in isolation may not be enough to evaluate the system. Evaluation have to be derived from combinations of such metrics. An example would be where an autonomic system is required to balance the conflicting goals of maximizing performance while minimizing energy consumption, and if the CPU is saturated (exhibits a high utilization), is need to start a new virtual machine and then carry out load balancing between the two of them.

CONCLUSION

The purpose of autonomic computing is not to replace humans but rather to enable systems to adjust and adapt themselves automatically to reflect evolving policies defined by humans. Autonomic computing systems, also known as self-systems, who can regulate and maintain themselves without human intervention. Also, the target of autonomic computing is improving the management of IT process where business processes can be rapidly adapted to realize on demand goals. The main four characteristics (self-properties) of autonomic computing are: self-configuring, self-

healing, self-optimizing, and self-protecting. An autonomic system consists of a set of autonomic elements that contain and manage resources and deliver services to humans or other autonomic elements. An autonomic element consists of one autonomic manager and one or more managed elements. At the core of an autonomic element is a control loop that integrates the manager with the managed element. Autonomic computing systems are quite complex and are composed of many different types of hardware and software components, which are organized in architecture.
The emergence of pervasive wide-area distributed computing environments, such as pervasive information systems and computational grids, has enabled new generations of applications that are based on seamless access, aggregation, and interaction. However, the inherent complexity, heterogeneity, and dynamism of these systems require a change in how the applications are developed and managed. In this paper are described the systems, technologies and services to support the development of autonomic self-managing applications.
In autonomic computing, as in any other area, it is important to be able to evaluate and compare different solutions thoroughly. We presented some qualitative and quantitative metrics, such as: costs, granularity/flexibility, degree of autonomy, adaptability, time to adapt, reaction time and stabilization time, sensitivity, and learning index. This metrics are essential for quantifying and guiding progress in autonomic computing Of course, it is not a final list of metrics, given the fact that this issue is multidimensional and there are many aspects to choosing the right metrics, from the aspect of: networking, communication, centralization, architecture, intelligence, self-management characteristics, the autonomic manager functions, workload and so on.

REFERENCES

Allan B. A., and all (2002), The CCA core specification in a distributed memory SPMD framework, Concurrency Computation

Boinot P., and all. (2000), Declarative approach for designing and developing adaptive components, In 15 th IEEE International Conference on Autonomic Software Engineering, 111-119

Bosh J. (1999), Superimposition: A component adaptation technique, Information and Software Technology, 257-373

De Wolfn T. and T. Holvoet (2006), Evaluation and comparison of decentralized autonomic computing systems, in In Department of Computer Science, K.U.Leuven. Leuven, Belgium: Report CW 437, March 2006

Foster I. and all (2004)., Modeling Stateful Resource with Web Services, www-128.ibm.com/developerworks/library/ws-modelingresources.pdf

IBM (2005), Autonomic Computing: IBM's Perspctive on the State of Information Technology

IBM (2005), Problem Determination Using Self-Managing Autonomic Technology, International Technical Support Organization

Kephart J. O. and Chess D. M. (2003), The vision of autonomic computing, Computer, vol. 36, no. 1, 2003

McCann J .A., ANS (Autonomic Networked System): A Position Paper, 1st UK-UbiNet Workshop,25-26th September

McCann J .A., Jawaheer G. (2003), Experiences in Building the Patia Autonomic Webserver, 1st International Workshop "Autonomic Computing Systems", DEXA

Nathiarasan S. A. and all (2014)., Autonomic Computing: Overview, International Journal of Advancement in Engineering Technology, Management&Applied Science, Vol.1, Issue 3, August

Nussey I and Telford R (2003), Presentation part of the IBM Academic Autonomic Computing Day, London, 29th October

Parker S. and Johnson C. (1998) An integrated problem solving environment, The SCIRun computational steering environment, In proceeding of 1st Hawaii International Conference on System Sciences, 147-156

Szyperski C. (2002), Component Software Beyond Object-Oriented Programming. Component Softvare Series, Addison-Wesley, Boston, 2nd.ed., 2002

Yannakakis G. N., Levine J., Hallam J., and Papageorgiou M., (2003), Performance, robustness and effort cost comparison of machine learning mechanisms in flatland, in Proc. 11th Mediterranean Conference on Control and Automation MED03. IEEE