Analysis of Autonomic Computing Concepts in Computational Grid Based on the ACLM Model

GORAN MARTINOVIC, DAMIR FILKO, MIRAN KARIC
Faculty of Electrical Engineering
Josip Juraj Strossmayer University of Osijek
Kneza Trpimira 2B, 31000 Osijek
CROATIA

Abstract: - This paper deals with basic concepts of autonomic computing. Each concept is associated with an autonomy indicator. Quality indicators for autonomic environments are also given. They describe system capability for self-healing, -optimization, -configuration and -protection. The proposed Autonomic Computing Layer Model (ACLM) is described in detail. It enables a framework for implementation of autonomic computing concepts at the levels of security and authentication, policy and goals, management, translation, security as well as communication and resource level. Autonomy of a computational grid as a heterogeneous and dynamic environment is discussed. Also, usability of the proposed ACLM model for the implementation of autonomic computing concepts in the grid is analyzed.

Key-words: - autonomic computing, autonomy indicators, computational grid, layered model.

1 Introduction
A constant increase of computer environment complexity requires a higher level of automation in resource management. According to [3], it is the main reason for the evidently fast development of autonomic computing. To be autonomic, a system needs to «know itself». It must configure and reconfigure itself under varying and unpredictable conditions (self-configuring). An autonomic system always looks for ways to optimize its working (self-optimizing). An autonomic system must be able to recover from routine and extraordinary events that might cause some parts to malfunction (self-healing). Such system must ensure self-protection by resource management in often virtual environment. The environment of non-dedicated resources requires a very high level of system autonomy.

According to [10], an autonomic computing environment has a certain ability for self-management and a dynamic change of its inner structure in compliance with owner’s business policy and goals, as well as users’ needs and in accordance with environment’s performance. An autonomic computing environment adapts to user’s demands and surroundings in correlation to the information gained by observing the aforementioned requests and performances.

Basic features of an autonomic computing environment are given in Chapter 2. To achieve desired levels of autonomy we should define some demands with indicators of autonomy and quality of service. That is given in Chapter 3. Chapter 4 gives short overview of standards in autonomic computing which contributed to principles of autonomic computing. In Chapter 5 we elaborate a layer model for autonomic computing (ACLM) and suggest ways of its application. A proposal for the implementation of ACLM in the grid system is given in Chapter 6.

2 Autonomic Computing
Inspiration for autonomic computing was found in the human autonomous nervous system, which controls our heart rate, respiration and generally performs all kinds of adjustments for various internal and external conditions without any conscious thought. Such similar self-managing and adaptive properties we require from autonomic computing. As in [10], IBM divided self-managing mechanisms in four distinct possibilities: self-healing, -optimization, -configuration and -protection.

These four possibilities are definitely very much interconnected. For example, an autonomic system has an ability of automatic configuration as a response to varying and unpredictable conditions of surroundings it is situated in. That in certain cases of unknown intrusion includes self-protection [17]. Self-configuration demands from the system the ability to learn, which in turn contributes to system
optimization as a whole. The ability to learn also enables system recovery and healing in case of unexpected events which can lead to failure of certain parts. Additionally, self-managing mechanisms can have an endless list of specific abilities, such as: self-governing, -correction, -organization, -scheduling, -planning, -administration, -optimization, -tuning, etc. In the paper we will focus on the basic four. In order to implement properly the concept of autonomic computing in a certain system, according to [9], one would need an evolutionary approach and progressive advancement of current systems whereby a significant degree of automation can be achieved without a need for replacing the whole system environment.

Fig. 1 presents levels of system autonomy. A majority of current computer systems can be placed on a level 1. Every subsystem is monitored and controlled individually by highly qualified staff who originally set it up into working condition and who will also replace it. At the level 2 management tools and technologies are used which can collect information from multiple subsystems and then deliver it to a limited number of consoles. This reduces time administrators of complex systems would need to gather and synthesize information themselves. On the predictive level 3, new technologies are introduced which enable us to link more elements of the system together where they can perceive patterns themselves and predict optimal configurations. All this enables them to counsel administrators and recommend further steps to be taken for the purpose of solving current situations. By developing new technologies and gaining more trust in system predictions and recommendations, we come to level 4, where the system acts independently and automatically to perform correct actions based on information available to it and with sense of what is happening in the system. SLA’s (Service Level Agreements) govern the entire system management. On a fully autonomic level, system activities are managed by business polices and goals and users access the system only to supervise operation or change policy and goals, as in [9], [17] and [14].

3 System Autonomy Indicators

3.1 Self-healing

According to [10], self-healing enables detection of faulty system behavior and launching of corrective action before any error actually occurs. It should do so in such a manner that it does not introduce any disturbance in the system, and it has a goal to increase system resilience. For the system to be self-healing it must have the ability to isolate a faulty component, shut it down and find adequate replacement or repair the component itself. In so doing, the system will often have to anticipate possible problems and act just to prevent failure. That is expected, because the main goal of this environment is to minimize outages and to maximize reliability and availability.

Large systems are usually built from many different components made by numerous manufacturers. This could lead to a problem with transmitting and understanding massages between various components, which is a fundamental prerequisite for failure identification. One of the basic assumptions for self-healing is problem diagnostics.

![Fig. 1 Levels of system autonomy](image-url)
possibilities of internal communication and message recognition, i.e. symptom recognition. The capacity to act is determined by system ability to independently work towards failure elimination. That activity can be manifested by repairing a faulty component, changing configuration or finding adequate replacement. Or, the system can have more modest abilities as to making propositions on malfunctions that need to be resolved later on by the staff. These capabilities depend on the level of system autonomy displayed in Fig. 1, whereas the manner of operation is determined by business policy of the institution. Its policy determines which actions should be taken in case of individual failures. Indeed, the capacity to act in case of finding adequate replacement depends on the failure severity and the availability of system resources. For example, if we have a cluster of 10,000 equal computer units keeping additional 500 units in cold reserve is logical if we want to enable redistribution of load in case of malfunction or increased demands for resources. Furthermore, if we have a breakdown of the cooling system and one third or 3,334 computer units go down, then we can replace only 500 of them. It is then questionable whether that figure is sufficient for the system to continue working on demanding applications. From the aforementioned facts and from [9] and [10] there follows expression (1), which defines the quality of a self-healing environment.

\[ Q_{SH} = p_a p_b (a + b) \]  

where \( p_a \) is the level of autonomy (1-5), \( p_b \) is a redundancy factor (0.1-1), \( a \) equals reliability of failure detection and \( b \) is a reaction speed factor.

### 3.2 Self-optimization

Self-optimization enables effective scheduling, allocation and utilization of resources in order to fulfill user demands and allow optimal quality of service. A self-optimization environment requires thereby hardware and software components which will increase resource performance and satisfy user needs without human. We must also enable dynamic redistribution of load toward systems which posses required resources. Similarly, all systems for storage, database, network and other resources, require constant calibration for the purpose of increasing efficiency and adaptability in unpredictable and growing environments [9].

Quality of a self-optimizing environment depends almost exclusively on response time, i.e. on the time needed for the required element of self-optimization to change preferences of some resource or to redistribute resources in such manner that will enable fulfillment of newly created needs. Furthermore, it can place resources in the state of readiness and only leave as many resources as needed for optimal performance. Although we have to bear in mind system unpredictability or at least its demands for system resources, the situation is not that chaotic and it can often boil down to rough time periods which can be predicted and approximated. Therefore, the quality of self-optimization, or the response time usually depends on algorithms used for prediction of workload, as in [11], or demands for a certain type of resource, respectively. Consequently, expression (2) equalizes quality of a self-optimizing environment with response times, i.e. their related frequencies and their weight/influence factors for certain types of resources.

\[ Q_{SO} = \sum_i \frac{w_i}{t_i}, \quad \sum_i w_i = 1 \]  

where \( t_i \) is the response time for the \( i \)-th resource and \( w_i \) is a weight/influence factor for the \( i \)-th resource.

### 3.3 Self-configuration

According to [10], self-configuration represents an ability of dynamic adoption to changes with minimum of human intervention. For the system to support such functionality it must be designed from the start with possibilities such as Plug&Play, configuration setup wizards and wireless network managers. In accordance with [9], self-configuration does not only apply to the possibility that each individual subsystem adjusts itself during operation, but also to the whole system in the enterprise to configure itself in a way that a user or an environment demands of it. Self-configuration is clearly connected with a self-optimization environment. That is because optimization requires configuration by demand, i.e. self-configuration. Because of that, we can associate the quality of a self-configuration environment with the one of self-optimization. Consequently, the quality of self-configuration element could also depend on response time, i.e. the time needed to pass from the request to change or install configuration till the actual configuring takes place and the normal operation of that component or the system begins. Indeed, that response time depends on the complexity of the component or the system that needs to be configured. In other words, it is not the same if we want to configure the router, a grid node or the entire system. Furthermore, it is essential to
know if the system can configure individual components of some subsystem, complete subsystems or the entire system as a whole. Expression (3) confirms that the quality of self-configuration depends on all previously mentioned parameters.

$$Q_{sk} = p_a \cdot \sum_{i} \frac{k_i}{t_i}$$ \hspace{1cm} (3)

where $p_a$ is the level of system autonomy (1-5), $k_i$ is the factor of complexity of an individual configuring element and $t_i$ is the response time needed to configure individual elements (components, subsystems, systems).

3.4 Self-protection

According to [9] and [10], self-protection enables prediction and detection of hostile and intrusive behavior, along with reducing the effects of unintentional human errors and malicious activities. Therefore, the system with this ability of self-protection can define and manage protection from internal and external attacks. The system must thereby have a possibility to create backups and restore resources which are just as safe as the original ones. Furthermore, the system would also have to be built around some basic security technologies such as LDAP (Lightweight Directory Access Protocol), Kerberos [13], hardware encryption and SSL (Secure Socket Layer). While implementing all this we must not forget to keep the simplicity of understanding and using user identities in different contexts for the purpose of reducing the stress of administrators.

Due to its very nature, quality of self-protection depends on detection of unauthorized activities, i.e. a probability of detecting them. This detection only comes into consideration after harmful activities or intrusions have already happened. Hence the self-protective environment has a role to limit damage and restore the system to its last acceptable status. Because of the aforementioned demands, self-protection is very much entwined with a self-healing environment. It is rather difficult to assess the quality of self-protection in computer environments because of its complexity, entwined nature and numerous ways of harmful activities.[1] Since detection differs for almost every type of a dangerous activity, from viruses and trojans to hostile intrusions, the quality of self-protection can easily be associated with the time for detecting every type of a harmful activity, and if we also assign a certain weight factor which determines seriousness of a harmful activity, we come to expression (4).

$$Q_{sp} = \sum_{i} \frac{sf_i}{MTTD_i}$$ \hspace{1cm} (4)

where $sf_i$ is a seriousness factor for a certain type of a harmful activity and $MTTD_i$ is Mean Time To Detection for a certain type of a harmful activity.

4 Standards and Standardization Bodies

The nature of the concept of autonomic computing prevents a company from delivering the whole and complete system solution. As it has been said before, self-managing systems are made from various resources, managing mechanisms, knowledge sources and other components from different manufacturers and suppliers. Therefore, there is a necessity for those kinds of systems to be founded on open industry standards, as described in [2] and [10]. Some of the standards and standardization bodies according to [10] are: DTMF, WS-CIM, IETF, SNMP, OASIS, WS-RF, ARM, etc.

5 Autonomic Computing Layer Model - ACLM

Although there already exist several models of autonomic computing [12], a new model shown in Fig. 2 is proposed in this paper. It is called Autonomic Computing Layer Model (ACLM).
As it can be seen in Fig. 2, ACLM consists of six layers, each of them being in charge for one part of the functionality of the whole system. The main layer is Management layer which holds the entire logic and “reason” of the autonomic system. There follows a description of individual layers.

5.1 Security and Authentication Layer
This layer consists of mechanisms which ensure safe access to the system for legitimate users and system administrators. Thus, this layer ascertains authenticity for a user who accesses the system, secures safe access as well as detects and prevents illegitimate access. It can support security protocols such as LDAP, SSL and Kerberos [13], as well as firewalls.

5.2 Policy and Goals Layer
The second layer is actually the only layer users and system administrators need and have access to in order to modify it. It determines general goals and policies, i.e. system purpose and tasks, as well as the manner of its behavior in certain situations. It is one of the main layers since it “defines” the system.

5.3 Management Layer
As it can be seen in Fig. 3, this layer includes all elements needed to fulfill all four components of self-management. The previous layer has a great influence on this layer since it defines the manner of action and management this layer executes.

- **Knowledge base** holds all necessary data needed for normal operation of an autonomic system. It includes data from patterns of behavior for viruses, trojans and similar malware, to router configurations, software etc.
- **Monitor engine** consists of mechanisms used for observing, i.e. supervising and filtering of the input system data. It is connected with the knowledge base which tells it what components need supervision.
- **Analysis engine** includes mechanisms for analyzing data coming from the monitor. It is also connected with the knowledge base which informs it about ranges of input data, patterns of behavior and sequences of events for the purpose of symptom identification.
- **Planning engine** devises a plan about dealing with a current situation, demand or assignment based upon the data provided to it by the analysis engine and depending on the current situation in the system. Information on the manner of approach and planning is received from the knowledge base and the upper layer.
- **Decision and Execute engine** performs a final check about the system status and brings a decision whether that something needs to be executed, and after that it commits execution. Operations which can be executed include for instance a change of router configuration, resource redistribution etc.

5.4 Translation Layer
Autonomic systems are usually a heterogeneous environment, as shown in [9], [10], [14] and [17]. They are composed of different hardware and software. Consequently, every component communicates in its own specific way and the job of the translation layer is to translate all manner and form of communication to some common language. This translation is two-way and every instruction from the management layer will be translated to a “language” of a specific component.

5.5 Security and Communication Layer
It consists of security (e.g. SSL, Kerberos) and communication (e.g. TCP/IP) protocols and it is used to connect and setup networks of system components.

5.6 Resource Layer
It physically consists of all system resources including *inter alia* processors, memory, hard drives and network.

6 Autonomic computing in grid
6.1 Grid computer systems
Grid environment [6] is a distributed and heterogeneous computer environment in which huge computing power of networked machines is used for execution of one or more tasks in accordance with performances. Such environment can also be called and often it really is a “virtual organization” [6]. Similar concepts of distributed computing existed as
early as the middle of the 20\textsuperscript{th} century [15]. Explosion of grid computing is recorded at the end of 90-ies when the Globus alliance developed the first standard. According to [4], [6] and [7], it is the “Globus Toolkit”, which defines basis possibilities and interfacing of components used for communication, information access, discovery and mapping of resources, authentication and other goals. Globus Toolkit is a set of services and libraries that enable us to work on the grid. It is based on an open code. Rapid development of technology and awareness of open architectures has resulted in the birth of an advanced standard by the Global Grid Forum (now Open Grid Forum) [16] called the Open Grid Services Architecture (OGSA), which is conceptually based on the Globus Toolkit, and is acknowledged by the industry [8]. OGSA is a service-based distributed architecture which enables mutual operation and communication of various resource types. As in [5], OGSA might be considered a sophisticated form of a web service architecture adapted to work in a grid environment.

### 6.2 Autonomy of Grid Computer Systems

According to [5], currently quite a few components have been implemented into grid computer systems making them autonomic to a certain degree. It especially refers to parts related to discovery and management of resources in distributed systems. However, for a higher degree of autonomy in a grid environment, all aforementioned and some other mechanisms should necessarily be accomplished. They would enable basic autonomic computing concepts. First of all, it is necessary to adjust complexity and the quantity of jobs being executed to grid system performance, increase the rate of response to efficiency of changes of system settings depending on conditions in the system and the environment in question. Since the grid environment is heterogeneous, further advanced strategies should be developed for the purpose of joining various components and mechanisms of various possibilities. It is also necessary to efficiently use available hardware (disks, memory, processors and network). All changes referring to components demanded by the environment must be carried out dynamically and timely. During its exploitation period, grid environment is rather unstable, i.e. it is subject to changes of its own structure. The main reason for that is changeability of both user demands and other systems [5]. These are only some of the reasons why accomplishing a more complete grid environment autonomy can be considered a challenging task.

### 6.3 Application of the ACLM to Grid System

Proposed ACLM autonomy model can increase computer system autonomy, hence grid environment autonomy as well. Most acceptable actions in the grid environment encompassed by the ACLM model are subjects of research that follows, and can be reduced to:

- Adjustment of the existing grid architectures and tools contained in the OGSA and the Globus for operation within the ACLM model primarily for the purpose of implementing layers 1, 5 and 6.
- Policy and Goals layer can be implemented by developing a new language of its own syntax and semantics or by using XML. That will enable users to write complex manuals in a logical and simple way defining thereby its autonomy.
- Translation layer can use ontology in future development, i.e. it might be based on databases and complex algorithms for recognition and translation of messages, records and states.
- The middle and most important layer, Management layer, requires most interventions implying a complex system of sensors, databases, as well as software support that would carry out policy, logic and heuristics necessary for system monitoring and management.

### 7 Conclusion

This paper gives an overview of basic features of autonomic computing and describes its basic goals and functional demands. Additionally, quality indicators are defined. They are rather simple, but contain values obtained by complex procedures and analysis. A real power of these indicators will be shown by analysis in experimental applications in future research. Furthermore, a six-layer autonomic computing model is described. It is applicable in a series of computer environments, thus also in a grid environment. Therefore, a series of interventions as to implementation of the autonomic computing principle per model layers are proposed for the grid environment. The described actions include the application of the existing OGSA and Globus tools, as well as some new ones that have to be developed. Future research would imply additional elaboration of the ACLM model and developing of a software tool for the purpose of model implementation and its experimental use in the grid environment.
References:


