# A SURVEY OF AUTONOMIC COMPUTING SYSTEMS

Mohammad Reza Nami [1], Mohsen Sharifi [2]
[1]*Faculty of Computer Science, Shahid Beheshti University, The Iran* , [2]*Faculty of Computer Enginering, Iran University of Science and Technology, The Iran*
*Phone: +98-21-22403133,Fax: +98-21-22413139,*
nami@iau-saveh.ac.ir

, *mshar@iust.ac.ir*

**Abstract** The evolution of networks and the Internet, which have presented high scalable and available services have made environments more complex. The increasing complexity, cost, and heterogeny in distributed computing systems have motivated researchers to investigate a new idea to cope with the management of complexity in IT industry. For this, Autonomic Computing Systems (ACSs) have been introduced. In this paper, we present a complete survey of ACSs. It consists of characteristics, their effects on quality factors, architecture of ACS building blockes, and challenges.

**Keywords:** Agent, Multi-agent System, Autonomic Computing Systems, Self-managing Systems.

## 1. Introduction

In centralized applications, data and programs were kept at one site and this was a bottleneck in performance and availability of remote information in desktop computers. Therefore, the concept of distributed systems was emerged. During the 1990s, distributed databases and client-server packages were used for information exchange between remote desktop computers. In these years, Distributed Computing Systems (DCSs) consist of different computers which were connected to each other and located at geographically remote sites. This was the starting point for emerging concepts such as Peer, Peer-to-Peer (P2P) computing [3], Agent [1], and Grid [2]. The evolution of networks and the Internet, which have presented high scalable and available services have made environments more complex. This complexity has increased the cost and errors of managing IT infrastructures. The skilled persons who manage these systems are expensive and can't manage them in configuration, healing, optimization, protection, and maintenance. Moreover, IT managers look for ways

*Please use the following format when citing this chapter:*

Nami, M.R., Sharifi, M., 2006, in IFIP International Federation for Information Processing, Volume 228, Intelligent Information Processing III, eds. Z. Shi, Shimohara K., Feng D., (Boston: Springer), pp. 101–110.

to improve the Return On Investment (ROI) by reducing Total Cost of Ownership (TCO), improving Quality of Services (QoSs), and reducing the cost for managing of IT complexity. A study shows that 25 to 50 percent of IT resources are spent on problem determination and almost half of the total budget is spent to prevent and recover system from crashes [4]. All these issues have motivated researchers to investigate a new idea to cope with the management of complexity in IT industry and self-management systems have been introduced. On March 8, 2001, Paul Horn presented importance of these systems with introducing ACSs to the National Academy of Engineering at Harvard University. IBM Vice President of Autonomic Computing Alan Ganek [5] has written a message and explained the importance of autonomic computing and the aim of introducing ACSs as "The goal of our autonomic computing initiative is to help customers build more automated IT infrastructures to reduce costs, improve up-time, and make the most efficient use of increasingly scarce support skills." Some benefits of autonomic computing include reduction of costs and errors, improvement of services, and reduction of complexity.

This paper is organized as follows. Related works are surveyed in section 2. In section 3, we present an overview of ACSs including definitions, benefits, and their characteristics. Section 4 describes Autonomic Elements (AEs) architecture as building blocks in ACSs. In section 5, some challenges such as robustness,learning, and relationships among AEs are discussed. Finally, we present conclusions and further researches.

## 2. Related Works

On March 8, 2001, Paul Horn presented a link between pervasiveness and self-regulation in body 's autonomic nervous system and introduced ACSs to the National Academy of Engineering at Harvard University. With choosing the term *autonomic*, researchers attempted to make autonomic capabilities in computer systems with the aim of decreasing the cost of developing and managing them. S. White et al in [6], and R. Sterritt and D. Bustard in [7] have described some general architectures for ACSs and their necessary elements called autonomic elements. J. A. McCann and M. C. Huebscher in [?] have proposed some metrics to evaluate ACSs such as adaptability. Some performance factors such as security and availability have been discussed by others [9]. ACS properties have been discussed by many researchers. These properties include self-optimization [10], self-configuration [1], self-healing [11], and self-protection [7]. Grand challenges in engineering and scientific have been discussed in [12]. Different projects and products have been developed in both by the industry and the academic. M. Salehie and L. Tahvildari have outlined some of these products in [4]. From another view, we can categorize researches carried out in this field in two groups as the follows:

- **Group 1:** Researches which describe technologies related to autonomic computing.
- **Group 2:** Researches which attempt to develop autonomic computing as an unified project.

However, the lake of appropriate tools for managing the complexities in large scale distributed systems has encouraged researchers to designing and implementing ACSs features.

## 3. Autonomic Computing: Definitions and characteristics

This section present an overview of autonomic computing systems. The autonomic concept is inspired by the human body 's autonomic nervous system. The human body has good mechanisms for repairing physical damages. It is able to effectively monitor, control, and regulate the human body without external intervention. An autonomic system provides these facilities for a large-scale complex heterogeneous system. An ACS is a system that manages itself. According to Paul Horn 's definition, an ACS is a self-management system with eight elements. Self-configuration means that An ACS must dynamically configure and reconfigure itself under changing the conditions. Self-healing means that An ACS must detect failed components, eliminate it, or replace it with another component without disrupting the system. On the other hand, it must predict problems and prevent failures. Self-optimization is the capability of maximizing resource allocation and utilization for satisfying user requests. Resource utilization and work load management are two significant issues in self-optimization. An ACS must identify and detect attacks and cover all aspects of system security at different levels such as the platform, operating system, applications, etc. It must also predict problems based on sensor reports and attempt to avoid them. It is called as Self-protection. An ACS needs to know itself. It must be aware of its components, current status, and available resources. It must also know which resources can be borrowed or lended by it and which resources can be shared. It is Self-awareness or Self-knowledge property. An ACS must be also aware of the execution environment to react to environmental changes such as new policies. It is called as context-awareness or environment-awareness. Openness means that An ACS must operate in a heterogeneous environment and must be portable across multiple platforms. Finally, An ACS can anticipate its optimal required resources while hiding its complexity from the end user view and attempts to satisfy user requests. We consider self-configuration, self-healing, self-optimization, and self-protection as major characteristics and the rest as minor characteristics. We are going to present a survey of current definitions of ACSs which have been derived from Horn 's definition. The aim of this survey is to identify all the possible definitions about ACSs. The common researchers in this field have been considered for this survey. They are first author in their publications. Table 1 shows the list of each researcher 's autonomic computing definition. The list of definitions in table 1 shows that there are differences in interpretation and definition of ACSs. Of course, with closer examination of the papers, it is found that these definitions are derivred from the eight elements proposed by Horn in 2001. For example, D. M. Chess et al have used the term 'self-configuration' similar to Horn 's definition and have presented 'self-assembly' property in Unity as an autonomic computing product. As mentioned, the aim of Autonomic Computing (AC) is to improve the system abilities. Therefore, autonomic computing characteristics affect various measurements of quality. Table 2 specifies the relationships between autonomic computig properties and quality factors.

## 4. Toward Autonomic Element Architecture

The goal of an autonomic computing architecture is to reduce intervention and carry out administrative functions according to predefined policies. Moving from manual to autonomic systems is introduced in a step-by-step manner by Tivoli group in IBM. ACSs also can make decisions and manage themselves in three scopes: resource element scope, group of resource elements scope, and business scope. In resource element scope, individual components such as servers and databases manage themselves. In group of resource elements scope, a pool of grouped resources that work together perform self-management. For example, a pool of servers can adjust work load to achieve high performance. Finally, overall business context can be self-managing. It is clear that increasing the maturity levels of AC will affect on level of making decision. The path to AC consists of five levels: basic, managed, predictive, adaptive, and autonomic. They are explained in the following [17]:

- **Basic Level:** At this level, each system element is managed by IT professionals. Configuring, optimizing, healing, and protecting IT components are performed manually.

- **Managed Level:** At this level, system management technologies can be used to collect information from different systems. It helps administrator to collect and analyze information. Most analysis is done by IT professionals, but it is starting point of automation of IT tasks.

- **Predictive Level:** At this level, individual components monitor themselves, analyze changes, and offer advices. Therefore, Dependency on persons is reduced and decision making is improved.

- **Adaptive Level:** At this level, IT components can individually and group monitor, analyze operations, and offer advices with minimal human intervention.

*Table 1.* List of Different Definitions for Autonomic Computing System.

| *First Author* | *Definition of Autonomic Computing* |
|---|---|
| Kephart [12] | Major characteristics and Self-managing |
| Chess [13] | Major characteristics |
| IBM Tivoli Group [14] | Major and Minor characteristics |
| Sterritt [15] | Major and Minor characteristics except anticipatory |
| Tianfield [16] | Self-mechanism including major characteristics, Self-planning, Self-learning, Self-scheduling, Self-evolution |
| Parashar [2] | Major characteristics, Self-adapting |
| Murch [17] | Major and Minor characteristics |
| Tesauro [1] | goal-driven self-assembly, self-healing, |
|  | real-time self-optimizing |
| De Wolf [10] | Major characteristics |
| White [6] | Major characteristics, Self-managing |
| Ganek [18] | Major characteristics, Self-managing |

*Table 2.* Relationships Between Autonomic Computig Properties and Quality factors.

| *Autonomic Properties* | *Quality Factors* |
|---|---|
| Self-configuration | Maintainability, Usability, Functionality, Portability |
| Self-healing | Reliability, Maintainability |
| Self-optimization | Efficiency, Maintainability, Functionality |
| Self-protection | Reliability, Functionality |
| Self-awareness | Functionality |
| Openess | Portability |
| Context-awareness | Functionality |
| Anticipatory | Efficiency, Maintainability |

- **Autonomic Level:** At this level, system operations are managed by business policies established by the administrator. In fact, business policy drives overall IT management, while at adaptive level, there is an interaction between human and system.

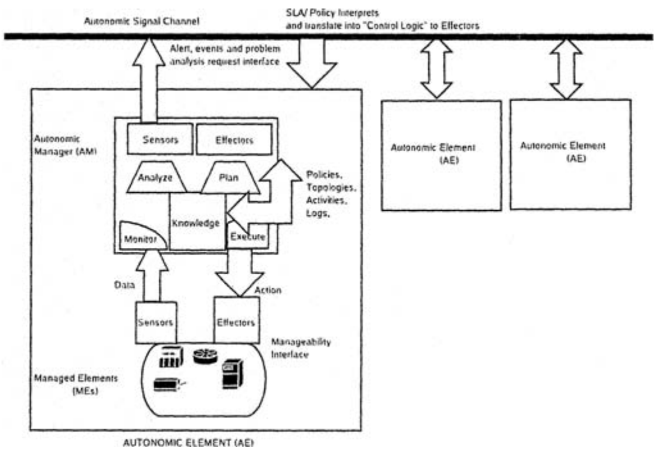

*Figure 1.* Autonomic Element architecture

Autonomic Elements (AEs) are the basic building blocks of autonomic systems and their interactions produce self-managing behavior. We can consider AEs as software agents and ACSs as multi-agent systems. Each AE has two parts: Managed Element (ME) and Autonomic Manager (AM). In fact, ACSs are established from Managed Elements (MEs) whose behaviors are controlled by Autonomic Managers (AMs). AMs execute according to the administrator policies and implement self-managing. An ME is a component from system. It can be hardware, application software, or an entire system. Sensors retrieve information about the current state of the ME, then compare it with expectations that are held in knowledge base by the AE. The required action is executed by effectors. Therefore, sensors and effectors are linked together and create a control loop.

Autonomic Managers (AMs) are the second part of an AE. An AM uses a manageability interface to monitor and control the ME. It has four parts: monitor, analyze, plan, and execute. The monitor part provides mechanisms to collect information from a ME, monitor it, and manage it. Monitored data is analyzed. It helps the AM to predict future states. Plan uses policy information and what is analyzed to achieve goals. Policies can be a set of administrator ideas and are stored as knowledge to guide AM. Plan assigns tasks and resources based on the policies, adds, modifies, and deletes the policies [ 6]. AMs can change resource allocation to optimize performance according to the policies. Finally, the execute part controls the execution of a plan and dispatches recommended actions into the ME. These four parts provide control loop functionality. Communications between AMs provide self-managing and

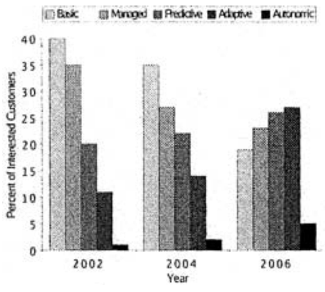

*Figure 2.* Estimate of people trends toward autonomic products

context-awareness. External behavior of AEs is related to relationships among them. Figure 1 shows detailed architecture of AEs in an AC environment. AMs can be linked together via an autonomic signal channel.

Tivoli group has also presented an estimation of people trends to autonomic operations from 2002 to 2006. Figure 2 shows results of this estimate.

## 5. Autonomic Computing Challenges

Since autonomic computing is a new concept in large-scale heterogeneous systems, there are different challenges and issues. Some of them have been explained in the following:

### Issues in Relationships among AEs

Relationships among AEs have a key role in implementing self-management. This Relationships have a life cycle consists of specification, location, negotiation, provision, operation, and termination stages. Each stage has its own challenges [12]. Expressing set of output services that an AE can perform and set of input services that it requires in a standard form and establishing syntax and semantics of standard services for AEs can be a challenge in specification. As An AE must dynamically locate input services that it needs and other elements that need its output services must dynamically locate this element with looking it up, AE reliability can be a research area in location stage. AEs also need protocols and strategies to establish rules of negotiation and to manage the flow of messages among the negotiators. One of challenges is the designer to develop and analyze negotiation algorithms and protocols, then determine which negotiation algorithm can be effective. Autonomated provision can be also a research area for next stage. After agreement, The AMs of both AEs control the operation. If the agreement is violated, different solutions can be introduced. This can be a research area. Finally, after the both AEs agree to terminate the negotiated agreement, the procedure should be clarified.

### Learning and Optimization Theory

A question raises this challenge: how can we transfer the management system knowledge from the human experts to ACSs? The master idea is that by observing that how several human experts solve a problem on different systems and by using traces of their activities, a robust learning procedure can be created. This procedure can automatically perform the same task on a new system. Of course, facilitating the knowledge acquisition from the human experts and producing systems that include this knowledge can be a challenge. One of reasons of the success of ACSs is their ability to manage themselves and react to changes. In short, in sophisticated autonomic systems, individual components that interact with each other, must adapt in a dynamic environment and learn to solve problems based on their past experiences. Optimization can be also a challenge, because in such systems, adaptation changes behavior of agents to reach optimization. The optimization is examined at AE level.

### Robustness

There are many meanings for robustness. Robustness has been served in various sciences and systems such as ecology, engineering, and social systems. We can interpret it as stability, reliability, survivability, and fault-tolerance, although it does not mean all of these. Robustness is the ability of a system to maintain its functions in an active state and persistence, when changes occur in internal structure of the system or external environment. The persons often mistake it with stability. Although both stability and robustness focus on persistance, but robustness is broader than stability. It is possible that components of a system are not themselves robust, but interconnections among them make robustness at the system level. A robust system can perform multiple functionalities for resistancing without change in the structure. With design of instructions that permit system to preserve its identity even when it is disrupted, the robustness in systems can be increased. Robustness is one of grand scientific challenges which can be also examined in programming.

## 6. Conclusions and Future works

In a Distributed Computing System, users and multiple computers are interconnected in an open, transparent, and geographical large-scalable system. Therefore, development and management of these systems are master problems for IT professionals. IBM proposed Autonomic Computing Systems as a solution. ACSs manage themelves. Four major characteristics of such systems include self-configuration, self-optimization, self-protection, and self-healing.

To achieve them, ACSs have four minor characteristics as self-awareness, context-awareness, openness, and anticipatory. Autonomic Elements (AEs) provide self-managing behavior in ACSs. They are the building blocks of ACSs and their interactions produce self-managing behavior. The various parts of AEs have been automated with evolution of AC levels. Engineering and scientific challenges have discussed in this field such as robustness, learning, and relationships among AEs.

In this paper, a survey of autonomic computing systems and their importance are presented. As future researches, the following topics can be proposed in autonomic distributed computing domain:

1 Performance evaluation of applying the autonomic behavior in a DCS model.

2 Designing an autonomic manager in multi-layer P2P form, so that autonomic behavior and management information as a knowledge base are stored in separated layers.

3 Studying languages which develop autonomic management behavior in a distributed computing environment.

4 Implementing a self-healing system in a virtual organization while one of partners failed.

## Acknowledgement

Mohammad Reza Nami is a faculty member and researcher in the Islamic Azad University and Shahid Beheshti University from Iran. He has been ranked as number 1 in computer science examination in Iran in October 2002. For this, he has a scholarship from the Ministry of Science, Research, and Technology (MSRT) of Iran. He can be reached at nami@iau-saveh.ac.ir and nami1352@yahoo.com

## References

1. Tesauro, G., and et al.:*A Multi-agent systems approach to autonomic computing*. IBM Press, (2004) 464-471.

2. Parashar, M., and et al.:*AutoMate: Enabling Autonomic Grid Applications*. Cluster Computing: The Journal of Networks, Software Tools, and Applications, Special Issue on Autonomic Computing, Kluwer Academic Publishers, Vol. 9, (2006).

3. Milojicic, D. S., and et al.:*Peer-to-Peer Computing*. In: Proceedings of the Second International Conference on Peer-to-Peer Computing, (2002) 1-51.

4. Salehie, M., Tahvildari, L.:*Autonomic Computing: emerging trends and open problems*. ACM SIGSOFT Software Engineering Notes, Vol. 30, (2005) 1-7.

5. Ganek, A.:*IBM Initiatives in autonomic computing and policy*.
http://www-03.ibm.com/autonomic/letter.shtml.

6. White, S., and et al.:*An architectural approach to autonomic computing*. In: Proceedings "International Conference on Autonomic Computing" (ICAC'04), NewYork, USA, (2004) 2-9.
7. Sterritt, R., and Bustard, D.:*Towards an autonomic computing environment*. In: 14th International Workshop on "Database and Expert Systems Applications" (DEXA '03), (2003) 694-698.
8. McCann, J. A., Huebscher, M. C.:*Evaluation issues in autonomic computing*. In: Proceedings of "Grid and Cooperative Computing" workshop (GCC-2004), Vol. 15, (2004) 597-608.
9. Chess, D. M., Palmer, C., and White, S. R.:*Security in an autonomic computing environment*. IBM System Journal, Vol. 42, (2003) 107-118.
10. De Wolf, T. and Holvoet, T.:*Evaluation and comparison of decentralised autonomic computing systems*. Department of Computer Science, K.U.Leuven, Report CW 437, Leuven, Belgium, (2006).
11. Hariri, S. and Parashar, M.:*Autonomic Computing: An overview*, Springer-Verlag Berlin Heidelberg, Vol. 3566, (2005) 247-259.
12. Kephart, J. O. and Chess, D. M.:*The vision of autonomic computing*. IEEE Computer, Vol. 36, (2003) 41-50.
13. Chess, D. M., Segal, A., Whalley, I., and White, S. R.:*Unity: experiences with a prototype autonomic computing system*. In: 1st "International Conference on Autonomic Computing" (ICAC 2004), New York, NY, USA, (2004) 140-147.
14. IBM Corporation Software Group:*The Tivoli software implementation of autonomic computing guidelines*. http://www-03.ibm.com/autonomic/pdfs/br-autonomic-guide.pdf.
15. Sterritt, R., Parashar, M., Tianfield, H., and Unland, R.:*A concise introduction to autonomic computing*. Advanced Engineering Informatics, Vo. 19, (2005) 181-187.
16. Tianfield, H.:*Multi-agent autonomic architecture and its application in e-medicine*. IEEE/WIC International Conference on "Intelligent Agent Technology" (IAT 2003), (2003) 601-604.
17. Murch, R.:*Autonomic Computing*. Prentice-Hall, (2004).
18. Ganek, A. G. and Corbi, T. A.:*The dawning of the autonomic computing era*. IBM System Journal, Vol. 42, (2003) 5-18.