Autonomic Computing: An Evidence of Better Management Technology

Amadin, F. I. & Obienu A. C
Department of Computer Science
University of Benin
P.M.B. 1154, Benin City. Nigeria.
frankamadin@uniben.edu, and obienuac@gmail.com

ABSTRACT

The increasing complexity, heterogeneity, dynamism and interconnectivity in software applications, services and networks has led to complex, unmanageable and insecure systems. This complexity has increased the cost and errors of managing information technology infrastructures, as well as, threatens to undermine the benefits information technology aims to provide. All these issues has necessitated the search for an alternate paradigm of system and application design, which can based on biological systems strategies, to deal with similar challenges of scale, complexity, heterogeneity, and uncertainty – a vision that has been referred to as Autonomic Computing. Autonomic Computing is a new innovation that is gaining awareness and acceptance in several fields due to its practical relevance in computing systems improvement. This paper presents an overview to autonomic computing, its architecture, examples, and promises to future applications.

Keywords: Architecture, Autonomic Element, Managed Elements, Multi-Agent Systems, Software

SMART-SMART-iSTEAMS Conference Proceedings Paper Citation Format
Amadin, F. I. & Obienu A. C (2018): Autonomic Computing: An Evidence of Better Management Technology. Proceedings of the SMART-iSTEAMS Multidisciplinary Conference, February, 2018, Ogwuashi-uku, Delta State, Nigeria. Pp 941-954

1. BACKGROUND TO THE STUDY

The advent and evolution of networks and internet, which has delivered ubiquitous services with extensive scalability and flexibility, continues to make computing environments more complex [1]. Moreover, advances in networking and computing technology and software tools have resulted in an explosive growth in networked applications and information services that cover all aspects of our life [2]. However, these sophisticated applications and services are extremely complex, heterogeneous and dynamic. This increasing complexity is overwhelming the capabilities of software developers and system administrators, who design, evaluate, integrate, and manage these systems [3]. Several researchers such as [4] are of the opinion that for a technology to be truly successful, its complexity has to disappear. Currently, several computing systems include complex infrastructures and operate in complex heterogeneous environments.

With the proliferation of handheld devices, the ever-expanding spectrum of users, and the emergence of the information economy with the advent of the web, computing vendors have difficulty providing an infrastructure to address all the needs of users, devices, and applications. Service-Oriented Architecture with Web services as their core technology try to address some of this issues, hereby raising numerous complexity issues [5].

Information Technology (IT) is called upon to deliver business services at higher speed and minimum cost. These services must be integrated into the existing infrastructures which lead to increase in complexity. Today, IT organizations face severe challenges in managing these complexities due to cost, time and relying mainly on human experts. This growing complexity of the IT infrastructure threatens to undermine the benefits IT aims to provide [6]. According to [7], the labour costs outstrip equipment by factors of 3 to 18, depending on the type of system, and one third to one half of the total budget is spent preventing or recovering from crashes. This necessitated IT managers to look for ways to improve the return on investment by reducing the total cost of ownership, improving quality of services and reducing the cost for managing of IT complexity– a vision that has been referred to as autonomic computing [8].

By attacking the software complexity problem through technology simplification and automation, autonomic computing promises to solve software evolution problems [5]. This research provides an overview of the autonomic environment and discusses some of the possibilities regarding how this technology might be able to adapt to changes in the evolving software crises and to take advantage of technology to better accomplish the needs of software developers. So, by embedding autonomic principles into existing system architecture, we can move one step further to achieving success.

2. CONTEMPORARY COMPUTING

Over the past era, the expeditious surge in computer technology has helped to produce more strained hardware and software applications. As a result, long term feasibility and sustainability are usually ignored which, results in "ball-of-mud" applications where peripherals have been added in a continued manner without paying any sort of attention to the resulting complex system integrity [9].The increase in the number of IT professionals is directly proportional to optimizing level of system complexity [10]. However, the rampant growth of information systems involves more than expected cost of computer system components, so there is a lot of resource wastage.

Moreover, the increasing heterogeneity, dynamism and interconnectivity in software applications, services and networks led to complex, unmanageable and insecure systems. This problem poses a great challenge for both science and industry because the increasing complexity of computing systems makes it more difficult for the IT staff to deploy, manage and maintain such systems. This dramatically increases the cost of management. Furthermore, if not properly and timely managed, the performance of the system may drop or the system may even fail.

Another drawback of increasing complexity is that it forces us to focus more on handling management issues instead of improving the system itself and moving forward towards new innovative applications. Autonomic computing focus on tackling the problem of growing software complexity. Autonomic systems are designed to take over routine, repetitive and manually intensive IT operations tasks that IT professionals choose to delegate [11].

3. AUTONOMIC COMPUTING

In 2001, Paul Horn from IBM coined the term Autonomic Computing (AC) to mark the start of a new paradigm of computing [10]. Autonomic Computing (AC) is a paradigm that aims at reducing administrative overhead by using autonomic managers to make applications self-managing. Autonomic computing was inspired from the autonomic nervous system that continuously regulates and protect our bodies subconsciously [12] 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 behaviour of the system. This will reduce the cost of management, improve performance, and enable the development of new innovative applications. Thus purpose of autonomic computing is not to replace humans entirely but rather to enable systems to adjust and adapt themselves automatically to reflect evolving policies defined by humans.

AC conceals the complexity of design and management in system equipment. Autonomic Elements (AEs) are the basic building blocks of autonomic systems and their interactions produce self-managing behavior. We can consider Autonomic Elements as software agents and Autonomic computing systems as multi-agent systems. Each Autonomic element has two parts: Managed Element and Autonomic Manager. In fact, Autonomic Computing Systems are established from Managed Elements whose behaviors are controlled by Autonomic Managers. Autonomic Managers execute according to the administrator policies and implement self-management. Managed Element is a component from system. It can be hardware, application software, or an entire system. Sensors retrieve information about the current state of the Managed Element and then compare it with expectations that are held in knowledge base by the Autonomic Elements. The required action is executed by effectors. Therefore, sensors and effectors are linked together and create a control loop. Autonomic Managers are the second part of an Autonomic Elements. An Autonomic Manager uses a manageability interface to monitor and control the Managed Element. It has four parts: monitor, analyze, plan, and execute, as shown in figure 1.

The Monitor part provides mechanisms to collect information from a Managed Element, monitor it, and manage it. Monitored data is analyzed. It helps the Autonomic Manager 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 Autonomic Manager. Plan assigns tasks and resources based on the policies, adds, modifies, and deletes the policies [13]. Autonomic Managers 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 Managed Element. These four parts provide control loop functionality. Communications between Autonomic Managers provide self-managing and context-awareness.

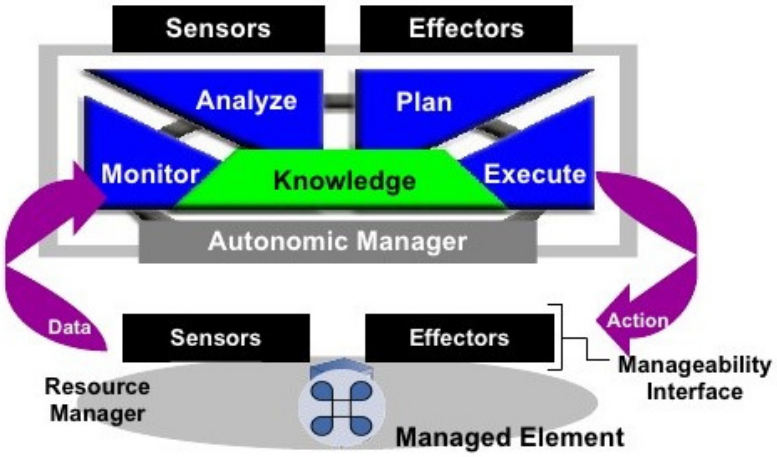

Figure 1: Autonomic Element [14].

All these different modules share same knowledge through resource details, change exploit policies based on environment plans. External behavior of Autonomic Elements is related to relationships among them. There exist self-managed architectures for sensors and handheld devices [15]. Figure 2 shows detailed architecture of Autonomic Elements in an Autonomic Computing environment. Autonomic Managers can be linked together via an autonomic signal channel.

4. ATTRIBUTES OF AUTONOMIC SYSTEMS

The properties that a system should have to constitute autonomicity are depicted in Figure 3. These properties of an autonomic (self-managing) system can be summarised into four objectives: self-configuring, self-healing, self-optimising and self-protecting. An autonomic system can self-configure at runtime to meet changing operating environments, self-tune to optimize its performance, self-heal when it encounters unexpected obstacles during its operation, and—of particular current interest—protect itself from malicious attacks. Research and development teams concentrate on developing theories, methods, tools, and technology for building self-healing, self-configuring, self-optimizing, and self-protecting systems.

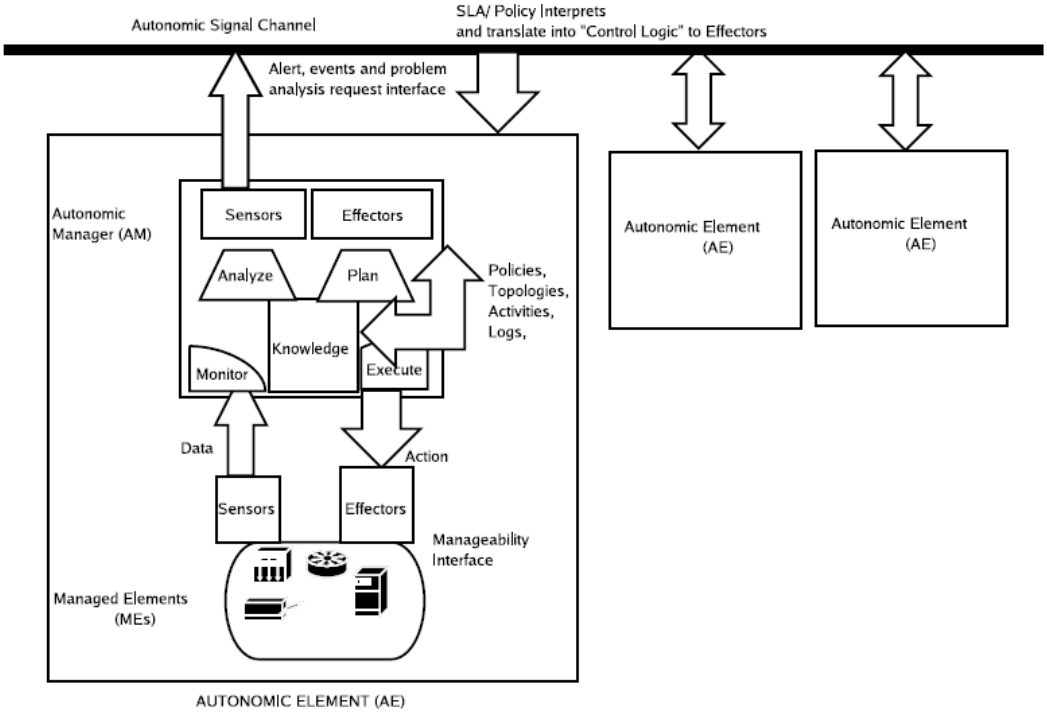

Figure 2: Autonomic Element architecture [16].

1. Self-configuring

Self-configuring systems provide increased responsiveness by adapting to a dynamically changing environment. A self-configuring system must be able to configure and reconfigure itself under varying and unpredictable conditions. Varying degrees of end-user involvement should be allowed, from user-based reconfiguration to automatic reconfiguration based on monitoring and feedback loops [17], [18]. For example, the user may be given the option of reconfiguring the system at runtime; alternatively, adaptive algorithms could learn the best configurations to achieve mandated performance or to service any other desired functional or nonfunctional requirement.

Variability can be accommodated at design time (e.g by implementing goal graphs) or at runtime (e.g, by adjusting parameters). Systems should be designed to provide configurability at a feature level with capabilities such as separation of concerns, levels of indirection, integration mechanisms (data and control), scripting layers, plug and play, and set-up wizards. Adaptive algorithms have to detect and respond to short-term and long-term trends.

2. Self-optimizing

Self-optimizing is the capability to efficiently maximize resource allocation and utilization for satisfying requirements of different users. Resource utilization and workload management are two important aspects necessitate for such a characteristic. One of the common models used in resource utilization is utility function [19]. Existing technologies in the workload management aspect, such as logical partitioning and dynamic server clustering, should be extensible to heterogeneous systems.

In this way, a single collection of computing resources will be provided which is manageable by a "logical" workload manager across the enterprise [20]. While in a short term, self-optimizing can address the complexity of managing system performance, in a long run its components will automatically and proactively seek ways to tune their operation, and make themselves more efficient in cost [17].

3. Self-healing

Self-healing systems provide resiliency by discovering and preventing disruptions as well as recovering from malfunctions. Such a system will be able to recover—without loss of data or noticeable delays in processing—from routine and extraordinary events that might cause some of its parts to malfunction. Self-recovery means that the system will select, possibly with user input, an alternative configuration to the one it is currently using and will switch to that configuration with minimal loss of information or delay. The main objective of self-healing is to maximize availability, survivability, maintainability and reliability of the system

[20].

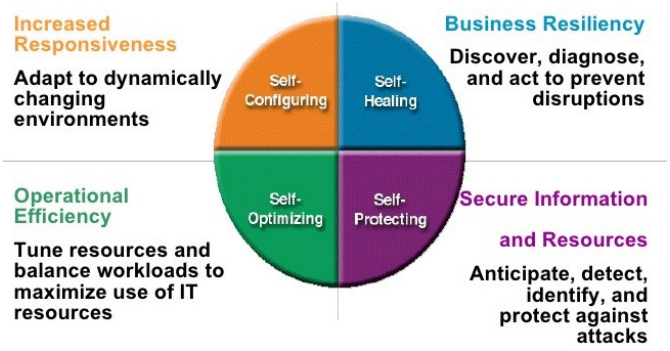

Figure 3: Autonomic Computing Attributes

4. Self-protecting

Self-protecting systems secure information and resources by anticipating, detecting, and protecting against attacks. Such a system will be capable of protecting itself by detecting and counteracting threats through the use of pattern recognition and other techniques [17]. This capability means that the design of the system will include an analysis of the vulnerabilities and the inclusion of protective mechanisms that might be employed when a threat is detected. The design must provide for capabilities to recognize and handle different kinds of threats in various contexts more easily, thereby reducing the burden on administrators.

Beside of aforementioned characteristics, two additional sub-characteristics can be enumerated for an autonomic system, namely:

1. Reflexivity: An autonomic system must have detailed knowledge of its components, current status, capabilities, limits, boundaries, interdependencies with other systems, and available resources. Moreover, the system must be aware of its possible configurations and how they affect particular nonfunctional requirements.
2. Adapting: At the core of the complexity problem addressed by the Autonomic Computing initiative is the problem of evaluating complex tradeoffs to make informed decisions. Most of the characteristics listed above are founded on the ability of an

autonomic system to monitor its performance and its environment and respond to changes by switching to a different behavior. At the core of this ability is a control loop. Sensors observe an activity of a controlled process, a controller component decides what has to be done, and then the controller component executes the required operations through a set of actuators. The adaptive mechanisms to be explored will be inspired by work on machine learning, multi-agent systems, and control theory.

5. TOWARD AUTONOMIC COMPUTING 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. Most existing systems cannot be redesigned and redeveloped from scratch to engineer autonomic capabilities into them. Rather, self-management capabilities have to be added gradually and incrementally—one component (such as, architecture, subsystem, or service) at a time. With the proliferation of autonomic components, users will impose increasingly more demands with respect to functional and nonfunctional requirements for autonomicity. Thus, the process of equipping software systems with autonomic technology will be evolutionary rather than revolutionary.

The path to Autonomic Computing consists of five levels: basic, managed, predictive, adaptive, and autonomic. They are explained in the following [21]:

1. Basic Level: At this level, each system element is managed by IT professionals. Configuring, optimizing, healing, and protecting IT components are performed manually.
2. Managed Level: At this level, system management technologies can be used to collect information from different systems. It helps administrators to collect and analyze information. Most analysis is done by IT professionals. This is the starting point of automation of IT tasks.
3. 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.
4. Adaptive Level: At this level, IT components can individually or group wise monitor, analyze operations, and offer advices with minimal human intervention.
5. 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 4 defined five levels of maturity to characterize the gradual injection of autonomicity into software systems

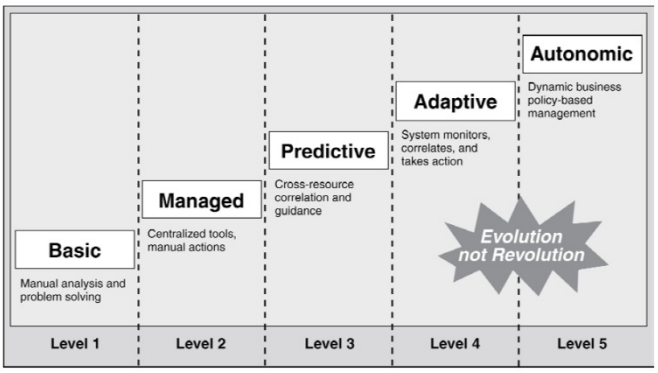

Figure 4: Increasing Autonomic Functionality

6. EXAMPLES OF AUTONOMIC SYSTEMS AND APPLICATIONS

There have been a number of research efforts in both academia and industry to develop autonomic systems and applications as identified by [22]. A few sample projects deployed in the time period starting from 2001 are:

OceanStore [23, 24], which is a global, consistent, highly-available persistent data storage system that supports self-healing, self-optimization, self-configuration, self-protection, policy based caching, routing substrate adaptation, autonomic replication, continuous monitoring, testing, and repair.

Storage Tank [25], is a multi-platform, universally accessible storage management system. It supports self-optimization, self-healing, policy based storage and data management, server redirection and log-based recovery.

Oceano [26], facilitates cost effective scalable management of computing resources for software farms. In terms of autonomic behaviour, it handles self-optimization, self-awareness, autonomic demand distribution, and constant component monitoring.

AutoAdmin [27], sets out to reduce Total Cost of Ownership (TCO) through self-tuning, self-administration by usage tracking, index tuning and recommendations based on workload.

Q-Fabric [28], provides system support for continuous online management through self-organization. It features continuous online quality management through 'customizability' of each application's Quality of Service (QoS).

Autonomia [29] presents one of the initial architectures implementing self-configuring and self-healing characteristics. It introduces an Autonomic Middleware service (AMS) comprising of a component and resource repository, and Fault and Security Handlers.
Focale [30] introduces a semantically rich architecture for orchestrating the behavior of heterogeneous and distributed computing elements with support of ontologies, policies and knowledge engineering.

Paws [31] presents an adaptive framework based on web-services. A BPEL (Business Process Execution Language) editor is provided, with which the business process and the constraints in terms of QoS can be defined.

Sassy [32] provides a framework for systems adapting to requirements by selecting services from service providers, that satisfy the utility function. SASSY introduces SAS (Service Activity Schemas), a visual requirements specification language, which describes the required services and activities from the domain ontology, and SSS (Service Sequence Scenarios), which defines the OS requirements. SASSY works in the framework of an SOA architecture to provide the self-architecting framework.

IPAutomata [33] is a part of IPCenter, IPSoft's commercial product for enterprise wide service delivery, an autonomic component managing multiple intelligent agents.

Applications of autonomic computing in engineering applications, such as autonomic (urban) traffic systems, autonomic industrial/residential building systems, autonomic industrial process systems, or autonomic manufacturing systems will increasingly come to the fore. As would be expected, early adaptors have come from within computing, for instance, efforts to add autonomic capabilities to instant messaging, spam detection, load balancing and middleware have been reported [34].

Database systems in particular, have been an early success within the AC initiative [35] due to the evolution of the DBMS towards more complex features and a resulting move towards self-tuning. SMART DB2 [36] provides for the reduction of human intervention and cost for DB2 through such self-management systems as self-optimization, self-configuration, autonomic index determination, disaster recovery, continuous monitoring of DB2's health and alerting the DBA.

7. PROMISES OF AUTONOMIC COMPUTING TO FUTURE APPLICATIONS

Over the past century there have been many profound technological, economic and social transformations. Currently, full development and diffusion of innovations such as software systems, telephones and automobiles have accompanied the emergence of mass production, mass consumption and mass government. There are many who, facing the next century, wonder if it will be possible and/or desirable to continue along the path of such prodigious change. Some worry about the capacity, both technological and social, to continue advancing and inventing new tools, new products and new ways of organizing everyday work and home life. Others worry that the ongoing transition costs may be too high, or that the risks to cherished traditions or the threats to environmental sustainability will, singly or together, be too great to bear. Preservation versus dynamism, incrementalism versus radicalism, these are the polar extremes that, unsurprisingly, haunt many end-of-the-century, future-of-the-millennium debates.

Against this background that IT is called upon to deliver services at greater speed and minimum cost. These services were integrated into the existing infrastructures which lead to increase in the complexity. This complexity has in turn increased the cost and errors of managing IT infrastructures. Also, the needed personals to manage these systems are expensive. Autonomic computing evolved as a discipline to create software systems and applications that is self-manage.

The goal is to overcome the complexities and inability to maintain current and emerging systems effectively. So, by embedding autonomic principles into existing system architecture, we can move one step further to achieving success.

Table 1 shows four aspects of self-management as they are now and would be with autonomic computing. It can be deduce that autonomic computing would be of great benefit in time to come since autonomic computing will aid computing systems to autonomously deal with unpredictable change, so as to fulfill the objectives they were constructed for, as well as conceals the complexity of design and management in system equipment.

TABLE 1: CONTEMPORARY VERSUS AUTONOMIC COMPUTING.

Concept	Current computing	Autonomic computing
Self-configuration	Corporate data centers have multiple vendors and platforms. Installing, configuring, and integrating systems is time consuming and error prone.	Automated configuration of components and systems follows high-level policies. Rest of system adjusts automatically and seamlessly.
Self-optimization	Systems have hundreds of manually set, nonlinear tuning parameters, and their number increases with each release.	Components and systems continually seek opportunities to improve their own performance and efficiency.
Self-healing	Problem determination in large, complex systems can take a team of programmers weeks.	System automatically detects, diagnoses, and repairs localized software and hardware problems.
Self-protection	Detection of and recovery from attacks and cascading failures is manual.	System automatically defends against malicious attacks or cascading failures. It uses early warning to anticipate and prevent system wide failures.

Moreover, Software has never been as important as today and its impact on life, work and society at large is growing at an impressive rate. We are in the flow of a software-induced transformation of nearly all aspects of our way of life and work. The dependence on software has become almost total. Malfunctions and unavailability may threaten vital areas of our society, life and work at any time.

With regards to its effects on people, organizations and society, Autonomic computing was conceived to lessen the spiraling demands for skilled IT resources, reduce complexity and to drive computing into a new era that may better exploit its potential, to support higher order thinking and decision making. Immediate benefits will include reduced dependence on human intervention to maintain complex systems accompanied by a substantial decrease in costs. Long-term benefits will allow individuals, organizations and businesses to collaborate on complex problem solving.

This is to say that Autonomic principles are key enablers for organizations seeking to take advantages of current technologies since they help mask complexity by simplifying infrastructure management. As such, Autonomic Computing has been identified by as a key area [22, 37, 38] and research is underway to utilize it in addition to autonomy [39].

8. MERITS OF AUTONOMIC COMPUTING

IT related benefits include:

1. Simplified user experience through a more responsive, real-time system.
2. Cost-savings – scale to use.
3. Increase stability of IT through automation.
4. Provides server consolidation to maximize system availability, minimize cost and human effort to manage large server farms.
5. Scale power, storage and costs that optimize usage across both hardware and software.
6. High security system. Less system or network errors due to self-healing.
7. Full use of idle processing power, including home PC's, through networked system.

9. CONCLUSION

In a distributed computing system, users and multiple computers are interconnected in an open, transparent, and geographical large-scale system. Therefore, development and management of these systems are master problems for IT professionals. IBM proposed Autonomic Computing Systems as a solution. Autonomic Computing Systems manage themselves. The prime purpose is to overcome the complexities and inability to maintain current and emerging systems effectively. Various benefits have been put forward for the adoption of autonomous computing principle including Self-healing, Self-optimizing, Self-protecting and Self-configuring. Autonomic capabilities are critical to businesses with large and complex IT environments, those using Web Services and/or Service Oriented Architecture (SOA) models, and those that leverage e-business or e-commerce. They are also key enablers for smaller businesses seeking to take advantage of current technologies, because they help mask complexity by simplifying infrastructure management.

The future for autonomic computing is bright. The big companies in computers are throwing lots of resources into this. The autonomic concept has been adopted by today's leading vendors and incorporated into their products. Aware that success is tied to interoperability, many today are participating in the standards development necessary to provide the foundation for self-managing technological ecosystems, and are integrating standards into their technology. So, by embedding autonomic principles into existing system architecture, we can move one step further to achieving success.

REFERENCES

[1] Salehie, M. and Tahvildari, L. (2005). "Autonomic Computing: Emerging Trends and open Problems" DEAS"05 St Louis, Missouri USA. ACM SIGSOFT Software Engineering Notes. 30 (4), 1- 7.

[2] Parashar, M. and Hariri, S. (2005). Autonomic computing: An overview. *Hot Topics, Lecture Notes in Computer Science*. Springer-Verlag Berlin Heidelberg 2005. Pp. 247-259. Available @ www.caip.rutgers.edu/TASSL/Papers/automate-upp-overview-05.pdf.

[3] Ganek, A. (2006). Overview of Autonomic Computing: Origins, Evolution, Direction, in Autonomic Computing – Concepts, Infrastructure, and Applications.

[4] Kluth, A. (2004). "Survey: Information Technology. Make It Simple." The Economist. http://www.economist.com/surveys /showsurvey.cfm?issue=20041030

[5] Müller, H. A.; Brien, L.; Klein, M. and Wood, B. (2006). Autonomic Computing. Carnegie Mellon University, Technical Note CMU/SEI-2006-TN-006, 2006. Downloadable from:http://www.sei.cmu.edu/reports/06tn006.pdf [last accessed 14.1.2016]

[6] Hariri, S. (2004). Autonomic computing: research challenges and opportunities. In Proceedings of IEEE conference on Pervasive Services (ICPS), page 7, 2004.

[7] Patterson, D. and et al. (200).Recovery oriented computing (roc): Motivation, definition, techniques, and case studies. UC Berkeley CS Tech. Rep. UCB/CSD-02-1175, March 2002.

[8] Hariri, S. and Parashar. M. (2005). Handbook of Bioinspired Algorithms and Applications, chapter The Foundations of Autonomic Computing. CRC Press LLC, 2005.

[9] Foote, B. and Yoder, J (2000). "Big Ball of Mud", in Pattern Languages of Program Design 4, ed. N. Harrison, B. Foote, H. Rohnert, Addison-Wesley, 2000.

[10] Horn, P. (2001). Senior Vice-President, IBM Research. Autonomic Computing: IBM's Perspective on the State of Information Technology, IBM Research, October 2001. http://www-1.ibm.com/industries/government/doc/content/bin/auto.pdf.

[11] Mittal, P., Singhal, A., and Bansal, A. (2014). A Study on Architecture of Autonomic Computing-Self Managed Systems. International Journal of Computer Applications 92: (6), 6 - 9.

[12] IBM, (2006) "An architectural blueprint for autonomic computing, 4th edition."http://www.01.ibm.com/software/tivoli/autonomic/pdfs/AC_Blueprint_White_Paper_4th.pdf, June 2006.

[13] White, S.; Hanson, J.; Whalley, I.; Chess, D., and Kephart, J. (2004). An architectural approach to autonomic computing. In Proceedings International Conference on Autonomic Computing (ICAC'04), NewYork, USA, pages 2–9, May 2004.

[14] Vassev, E and Hinchey, M. (2011). "Knowledge Representation and Awareness in Autonomic Service-Component Ensembles-State of the Art", in proc. Int. Symp. On O/C/S-Oriented RT Dist. Comp. Workshops, Newport Beach, CA, USA 2011

[15] Ayala, I. (2012). Self-StarMAS: A Multi-Agent System for the Self Management of AAL Applications, Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing. 2012.
[16] Nami, M. R. and Bertals, K. (2006). A Survey of Autonomic Computing Systems. Computer Engineering Laboratory, Delft University of Technology. Available @ www.*ce-publications.et.tudelft.nl/.../610_a_survey_of_autonomic_computing_systems.pdf*
[17] Kephart, J. O. and Chess, D. M. (2003). "The Vision of Autonomic Computing." IEEE Computer 36, 1: 41-50. N
[18] Melcher, B and Mitchell, B (2004). Towards an autonomic framework: Self-configuring network services and developing autonomic applications. Intel Technical Journal, 08:279{290, Nov. 2004.
[19] Walsh, W.; Tesauro, G.; Kephart, J and Das, R. (2004). Utility functions in autonomic systems. In Proceedings of IEEE conference on
[20] Ganek, A. G and Corbi, T. A.(2003). The dawning of the autonomic computing era. IBM Systems Journal, *Special Issue on Autonomic Computing*, 42:5{18, 2003.
[21] Murch. R. (2004). Autonomic Computing. In Prentice-Hall, pages 0–20:25–40, October 2004.
[22] Sterritt, R., Parashar, M.; Tianfield, H.; Unland, R. (2005). A concise introduction to autonomic computing. Advanced Engineering Informatics 19 (2005) 181–187
[23] Kubiatowicz, J. (2001). 'OceanStore: global-scale persistent storage' Stanford Seminar Series, Stanford University, http://oceanstore.cs.berkeley.edu/publications/talks/StanfordOceanStore.pdf; Spring 2001.
[24] Berkeley, U. C. (2002). Computer science division. Hildrum K. 'The OceanStore Project', project overview. http://oceanstore.cs.berkeley.edu/info/overview.html, July 8 2002. Project Page.
[25] Menon, J., Pease, D., Rees, R., Duyanovich, L., and Hillsberg B. (2003). IBM storage tank-A heterogeneous scalable SAN file system. IBM Syst J 2003; 42(2):250–67.
[26] IBM Research. The oceano project. http://www.research.ibm.com/oceanoproject/. IBM Corp.
[27] Narasayya V. (2002). AutoAdmin: towards self-tuning databases; November 13 2002. Guest Lecture at Stanford University.
[28] Poellabauer C. (2002). Q-fabric—system support for continuous online quality management; 2002 http://www.cc.gatech.edu/systems/projects/ELinux/qfabric.html
[29] Dong, X., Hariri, S. and Xue, L. (2003) "AUTONOMIA: An Autonomic Computing Environment," 2003.
[30] Strassner, J. C., Agoulmine, N. and Lehtihet, E (2006). "FOCALE –A Novel Autonomic Networking Architecture," 2006.
[31] Ardagna, D., Comuzzi, M. and Mussi, E. (2007). "PAWS: A Framework for Executing Adaptive Web-Service Processes," 2007.
[32] Menascé, A., Gomaa, S., Malek, H. and Sousa, P (2011). "SASSY: A Framework for Self-Architecting," IEEE Software, pp. 78-85, November 2011.
[33] IPSOFT (2012). February 2012. [Online]. Available: http://www.ipsoft.com/
[34] Kaiser G, Parekh J, Gross P, and Valetto G. (2003). Kinesthetics eXtreme: an external infrastructure for monitoring distributed legacy systems Proceedings of the autonomic computing workshop, fifth international workshop on active middleware services (AMS 2003), Seattle, WA; 2003. p. 22–30.

[35] Diao Y, Eskesen F, Froehlich S, Hellerstein J, Spainhower F, and Surendra M.(2003). Generic online optimization of multiple configuration parameters with application to a database server. Proceedings of the 14th IFIP/IEEE workshop on distributed systems: operations and management (DSOM), LNCS 2867. Berlin: Springer; 2003. p.3–15

[36] Lohman, M., and Lightstone, S. (2002). SMART: making DB2 (More) autonomic. In: VLDB 28th international conference on very large data bases, Kowloon Shangri-La Hotel, Hong Kong, China; August 20–23 2002.

[37] Clancy, D. J. (2002). NASA challenges in autonomic computing, Almaden Institute 2002, San Jose, CA: IBM Almaden Research Center; April 10, 2002.

[38] Sterritt, R. (2002). Towards autonomic computing: effective event management. Proceedings of 27th annual IEEE/NASA software engineering workshop (SEW), Maryland, USA, December 3–5.: IEEE Computer Society; 2002. p. 40–7.

[39] Truszkowski W, Rash J, Rouff C, Hinchey M. (2004). Asteroid exploration with autonomic systems. Proceedings of IEEE workshop on the engineering of autonomic systems (EASe) at the 11th annual IEEE international conference and workshop on the engineering of computer based systems (ECBS 2004), Brno, Czech Republic; 24–27 May 2004. p. 484–90.