Autonomic Computing: Emerging Trends and Open Problems

Mazeiar Salehie
mazeiar@uwaterloo.ca

Ladan Tahvildari
ltahvild@uwaterloo.ca

Dept. of Elect. and Comp. Eng.
University of Waterloo
Waterloo, Ontario
Canada N2L 3G1

ABSTRACT

The increasing heterogeneity, dynamism and interconnectivity in software applications, services and networks led to complex, unmanageable and insecure systems. Coping with such a complexity necessitates to investigate a new paradigm namely *Autonomic Computing*. Although academic and industry efforts are beginning to proliferate in this research area, there are still a lots of open issues that remain to be solved. This paper proposes a categorization of complexity in I/T systems and presents an overview of autonomic computing research area. The paper also discusses a summary of the major autonomic computing systems that have been already developed both in academia and industry, and finally outlines the underlying research issues and challenges from a practical as well as a theoretical point of view.

Categories and Subject Descriptors

A.1 [**General Literature**]: Introductory and Survey; D.2.9 [**Software Engineering**]: Management

General Terms

Design, Management

Keywords

Autonomic Computing, Software Engineering, Software Management

1. INTRODUCTION

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. On the other hand, as systems became much more software-intensive, we can hear more about *software crisis*. The root cause of the crisis has always been *complexity*; complexity of business domains to be analyzed, the target system to be designed, implemented, maintained and managed, and the complexity of using systems in environments. As Robert Morris, director of IBM's Almaden Research Laboratory, said "There is no less than a crisis today in three areas: cost, availability and user experience".

Meanwhile, Information Technology (I/T) is called upon to deliver business services at higher speed and minimum cost. These services must be integrated into the existing infrastructure which lead to increase the complexity. Today, I/T organizations face severe challenges in managing complexity due to cost, time and relying on human experts. The growing complexity of the I/T infrastructure threatens to undermine the benefits information technology aims to provide [12]. According to a study published in March 2002 by researchers from University of California at Berkeley [29], the labor 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.

All these issues have necessitated the investigation of a new paradigm, *Autonomic Computing*, to design, develop, deploy and manage systems by inspiring from strategies used by biological systems to cope with complexity, heterogeneity, and uncertainty. An autonomic system has four major characteristics - self-configure, self-heal, self-optimize, and self-protect - which are often referred to as self-CHOP characteristics. According to [30], there are two visions in Autonomic Computing: "Vision in Administration" in which computing systems that can manage themselves given high-level objectives from administrators, and "Vision in Software Engineering", due to software intensiveness of I/T-based systems, in which autonomic functionality is standardized and given system engineers concentrate on a new or improved functionality. This survey paper identifies emerging trends in autonomic computing research, and enumerates a list of open questions.

Organization of this paper is as follows. Section 2 reviews the concepts and categories of complexity in I/T. Section 3 presents how the characteristics of autonomic computing help to cope with such increasing complexities. Section 4 presents the trends in the architecture and evaluation criteria of autonomous systems. Section 5 addresses two paradigms, agent-oriented and component-based, which can help us to develop autonomous systems. Section 6 discusses a summary of the autonomic computing systems that have been already developed both in academia and industry. Section 7 outlines a number of open questions from a fundamen-

DEAS 2005, May 21, 2005, St. Louis, Missouri, USA.

tal as well as a practical perspective, and finally Section 8 concludes.

2. A CATEGORIZATION OF COMPLEXITY IN I/T

After the economic stagnation in I/T industry at the end of 1990's, recent reports of International Data Corp. (IDC) indicate that total spending in I/T gradually have been increased especially in service and software sectors [19]. Such increases can address i) how much organizations and industry have become I/T-based, and ii) how much complex and expensive is to develop I/T systems especially software products. A study shows $25\% - 50\%$ of I/T resources are spent on problem determination. Across all industries, more than 40% of all investments in information technology are used just trying to get technologies to work together. In other words, half of the investment goes for issues that do not directly drive business value [14].

Figure 1 illustrates a synergy between three components namely: i) an I/T-based business system including layers of *business*, *I/T-based* and *software systems*, ii) a software quality model (based on ISO/IEC 9126-1 [15]) including internal, external and in use quality factors, and iii) a complexity model including attributes such as cost, time and size. A business system consists of I/T-based systems (e.g. information systems), people, machines, and other artifacts [5]. Generally an I/T-based system consists of software systems, network and hardware resources.We categorize such a complexity model into three following categories:

- *Business Domain Complexity* which embraces the complexity of business processes, organizations and resources in terms of cost, time, size, security and fault-proneness. The most important aspect of this type of complexity for I/T-based systems is how to translate business policies to I/T policies. For instance, in a banking system how security and assurance policies could be translated and realized by the I/T-based system.

- *System Development Complexity* which deals with designing, implementing, testing, evolving, and refactoring the system. Some aspects of complexity in this category reflect the business domain complexity such as computational complexity caused by massive business processes. In a nutshell, methodologies, tools, and resources play a key role in how easy we can develop and maintain a system. For instance, development of a system with object-oriented methodologies may have less complexity.

- *System Management Complexity* which encompasses installing, (re-)configuring, problem management (monitoring, detecting, and recovering), resource management, security management, and many other issues related to efficiency and service providing. For instance, administration of a large-scale data center deals more with this type of complexity.

We believe that the complexity model of I/T based systems are highly related to their quality model, because of relationship between complexity attributes (e.g. structure) and quality factors (e.g. functionality). In the following sections we elaborate further on these relationships.

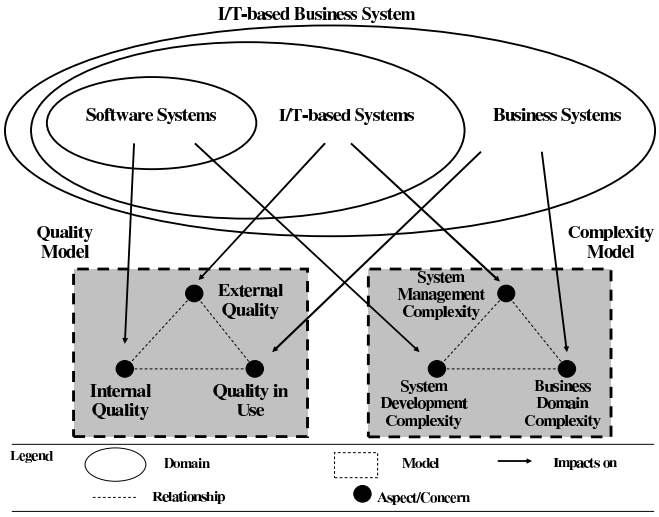

Figure 1: A Synergy Between Three Concerns in I/T.

3. FUNDAMENTAL CONCEPTS

To cope with management complexity in I/T with a reasonable cost, we need to have self-managing systems. *Autonomic Computing* which was first introduced by IBM supports such self-managing systems [13]. Autonomic Computing facilitates to design, develop, and deploy self-managing systems by inspiring from strategies used by biological systems. An autonomic system has four major characteristics – self-configure, self-heal, self-optimize, and self-protect – which are often referred to as self-CHOP characteristics, and four minor characteristics namely: self-awareness, open, context-aware, and anticipatory [4].

Table 1 shows which characteristic addresses one or more categories of complexity depicted in Figure 1. A ($\surd$) in each entry means that the characteristic in the corresponding row supports *coping with* the complexity in the corresponding column. While the following sections elaborate each of major and minor characteristics further, Figure 2 depicts a potential taxonomy to show potential relationships between quality factors [15] and autonomic characteristics.

3.1 Major Characteristics

This section reviews the core concept of the four major characteristics for an autonomic system and their relationships with quality factors as depicted in Figure 2.

- *Self-Configuring* is the capability of adapting automatically and dynamically to environmental changes. This characteristic has two aspects as follows: i) installing, (re-)configuring, and integrating large, complex network-intensive systems [18, 21], and ii) adaptability in architecture or component level to re-configure the system for achieving the desired quality factors such as improving performance in response to environmental changes [7]. The latter aspect is the reason why it addresses development complexity in Table 1. Self-configuring characteristic may have an impact on the quality factors such as: *maintainability*, *functionality* , *portability* , and *portability*.

- *Self-Healing* is the capability of discovering, diagnosing, and reacting to disruptions [18]. Such a system must be able to recover by detecting a failed component, taking it off-line to be fixed, and replacing the fixed component into the system without any apparent disruption. Such systems have to predict problems and take actions to prevent a failure. The main objective of self-healing is to maximize availability, survivability, maintainability and reliability of the system [11]. There are some solutions available for this aspect, such as Recovery-Oriented Computing (ROC) [29] which proposes micro-rebooting of the faulty module or modules to control fault propagation to the whole system. It requires that each module can be stopped and restarted independently, and the operating system can recognize faulty modules and automatically restart them.

- *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 [35]. 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 [11]. 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 [18].

- *Self-Protecting* is the capability of reliably establishing trust, and anticipating, detecting and recovering from the effects of attacks with two aspects [18]: i) defending the system against correlated problems arising from malicious attacks or cascading failures that remain uncorrected by self-healing measures, and ii) anticipating problems based on early reports from sensors and taking steps to avoid or mitigate them. Two existing solutions that exhibit such behavior are the *immune system* inspired by biological systems, and *secure distributed storage (SDS)* [8]. This characteristic also addresses business domain complexities such as fraud detection and prevention (as shown in Table 1).

3.2 Minor Characteristics

Beside of aforementioned characteristics, four additional subcharacteristics can be enumerated for an autonomic system, namely : i) *self-awareness* which means that an autonomic system is aware of its state and its behaviors for self-managing and also for collaborating with other systems, ii) *open* which means that an autonomic system must operate in an heterogeneous environment (to be interoperable) and is portable across multiple platforms, iii) *context-awareness* which means that an automatic system should be aware of its execution environment and is able to react to environmental changes such as new business policies, iv) *anticipatory* which means that an autonomic system will anticipate the optimized resources needed while keeping its complexity hidden. This characteristic adds proactiveness to self-managing paradigm [28]. As shown in Figure 2, such minor characteristics may have their impacts on quality factors such as *maintainability*, *functionality*, and *portability*.

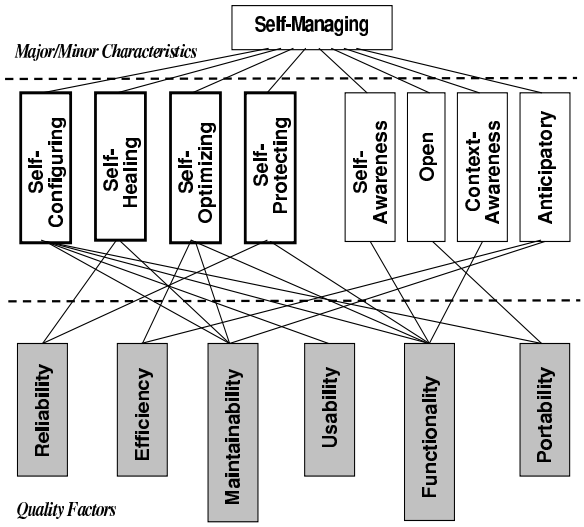

Figure 2: Relationships between Autonomic Characteristics and Quality Factors.

4. EMERGING TRENDS

This section deals with the emerging trends in creating autonomous systems at the architecture level and evaluation criteria to determine the level of autonomicity of such systems.

4.1 Autonomous System Architecture

As known in an autonomic component, the basic concept is a control loop [18]. This acts as a manager to monitor, analyze, and apply proper actions on a set of predefined system policies [22]. McCann et al. [20] identified two categories of the approaches to autonomic computing systems: *tightly coupled* and *decoupled* autonomic systems. The former is often built using intelligent agents with their own goals, and the latter is the one in which the infrastructure handles the autonomic behavior of the system. The two approaches have common concepts and it is sometimes difficult to place a research project in one particular category. In both approaches, there is a need to have two kinds of elements [36] : i) to implement functionalities of the target system, and ii) to introduce some patterns for system self-management. These sets of elements describes two levels of system architecture namely : *intra-element* and *inter-element* relations.

The higher level of the architecture deals with inter-element relationships in a coordination/cooperation fashion. In this level, global aspects of autonomicity are taken into account, such as global configuration or incremental repair in response to local failures [7]. Infrastructure elements may be needed for this level to provide naming service (registry), monitoring (sentinel), combination (aggregator), interaction (broker) and negotiation (negotiator). Section 5 elaborates

Characteristic	Business Domain Complexity	System Development Complexity	System Management Complexity
Self-Configuring	-	√	√
Self-Healing	-	-	√
Self-Optimizing	-	-	√
Self-Protecting	√	-	√
Self-Awareness	-	√	√
Open	-	√	√
Context-Aware	√	-	√
Anticipatory	-	-	√

Table 1: Weaving Between I/T Domain Complexity and Autonomicity Characteristics.

two well-known paradigms for creating autonomous systems in intra- and inter-element levels.

4.2 Evaluation Criteria

Autonomic computing has become inevitable and therefore become more prevalent, hence its evaluation is becoming increasingly more important. There are two aspects that need proper attention : i) evaluation of an autonomic system, and ii) assessment the autonomicity of an I/T environment.

McCann et al. [20] list a set of metrics by which the autonomic systems can be evaluated and compared, namely : Quality of Service (QOS), cost, granularity/flexibility, robustness, degree of autonomy, adaptivity, reaction time, sensitivity, and stabilization. IBM has created an autonomic assessment software tool to measure the level of autonomic function against each of the following six operational areas within any I/T environment : security management, user and resource provisioning, performance and capacity management, solution deployment, availability, and problem management. This tool analyzes an environment to determine its level of autonomic maturity. Such levels of maturity can be categorized as: *Basic*, *Managed*, *Predictive*, *Adaptive* and finally *Autonomic* levels [22].

5. AUTONOMIC COMPUTING TECHNIQUES AND FORMALISMS

As described in [13, 18], research areas like agent-based computing, grid computing, software engineering methodologies, and control theory can help us to achieve the objectives of autonomic computing. Due to space limitation, we discuss two of them which are addressed more in the literature.

5.1 Agent-Oriented Paradigm

Autonomy, proactivity, and goal-directed approach to problem solving and planning are major characteristics of agents [17]. Agents are capable of initiating action independent of any other entity. Software agents have their own thread of control, by localizing not only their code and states but also their invocation. Such agents can also have individual rules and goals, making them appear like "active objects with initiatives" [26].

While traditional approaches to computer systems management are often centralized and hierarchical, today's large-scale computing systems are highly distributed with complex connectivity and interactions, led to centralized management schemes be infeasible. Viewing autonomic elements as agents and autonomic systems as multi-agent systems makes it clear that agent-oriented architectural concepts seem to be substantially important [18]. Agent-oriented architecture can help self-managing approach of autonomic systems, for instance the system could encapsulate the local optimization of resource usage from the global allocation of resources. In [33], authors present an agent-based architecture, called *Unity* in which every component is an autonomic element and includes : computing resources (e.g., databases, storage systems, and servers), higher-level elements with some management authority (e.g. workload managers and provisioner), and elements that assist other elements in doing their tasks (e.g. policy repositories, sentinels, brokers, registries).

5.2 Component-Based Development

Component-based development is another research area that addresses encapsulation and self-sufficiency at the level of frameworks and middlewares. In addition to the interfaces exported by traditional components, autonomic components provide enhanced profiles or contracts that encapsulate their functional, operational, and control aspects. These aspects enhance the interfaces to export information and policies about their behavior, resource requirements, performance, interactivity and adaptability to dynamic systems. Several noteworthy research themes and industrial solutions address these aspects as follows :

- *Composition Formalisms* can help to automate generation of global configuration by considering certain constraints and/or optimization criteria (like path-based and graph-based methods) [7].
- *Smart Components* can help to adapt to environmental changes for providing building blocks in self management systems (like smart servers) [16].
- *Hot swapping* can help to develop self-management system either by *code inter-positioning* or *code replacement*. Inter-positioning involves inserting a new component between two existing ones. This allows us to enable monitoring in a more fashionable manner when problems occur by minimizing performance at minimum cost [2].

6. A LANDSCAPE OF AUTONOMIC SYSTEMS

This section reviews the research works have been done so far for autonomic computing (not merely under this name) in both academia and industry sectors. Our objective is to figure out how and in what extent available solutions have addressed goals presented in the autonomic computing vision [18]. Results of this section may help researchers to outline deficiencies of current research works and confronting challenges in the future.

Project / Characteristic	SMART	Oceano	Optimal Grid	Auto Admin	N1	Adaptive Enterprise	Ocean Store	AntHill	ROC	Autonomia	eBiquity	Software Rejuv.
Self-Configuring	√	√	√	-	√	√	√	√	-	√	√	-
Self-Healing	-	-	-	-	-	-	√	-	√	√	-	√
Self-Optimizing	√	√	√	√	√	√	√	√	-	√	√	-
Self-Protecting	-	-	-	-	-	-	√	-	√	√	-	√
Self-Awareness	√	√	-	-	-	-	-	-	√	-	√	√
Context-Aware	√	-	-	-	-	-	-	-	-	-	√	-
Open	-	-	√	-	√	√	√	√	√	√	√	-
Anticipatory	√	√	√	-	-	√	√	-	-	-	-	-

Table 2: A Comparison Between Academic and Industrial Projects in Autonomic Computing

6.1 Industry-Oriented Projects

The essence of simplifying management and operations of I/T-based systems led to many vendors who had no doubt enter the market of autonomic computing. Large enterprise-wide vendors such as IBM, Sun Microsystems, HP, Microsoft, Intel, Cisco, and several other vendors have developed variety of systems and solutions for this area of research. For this section, due to the space limitation, we focus on only four major vendors whose products are leading the market of autonomic systems, namely: IBM, Sun, HP and Microsoft with their major projects as follows:

- *SMART, IBM :* reduces complexity and improves quality of service through the advancement of self-managing capabilities within a database environment [31].
- *Oceano, IBM :* designs and develops a pilot prototype of a scalable, manageable infrastructure for a large scale computing utility powerplant [24].
- *Optimal Grid, IBM :* aims to simplify the creation and management of large-scale, connected, parallel grid applications by optimizing performance and includes autonomic grid functionality as a prototype middleware [27].
- *AutoAdmin, Microsoft*: makes database systems self-tuning and self-administering by enabling them to track the usage of their systems and to gracefully to adapt to application requirements [3].
- *N1, Sun :* manages data centers by including resource virtualization, service provisioning, and policy automation techniques [23].
- *The Adaptive Enterprise, HP :* helps customers to build a system in three levels namely business (e.g., customary relation management), service (e.g., security), and resource (e.g., storage) [22].

As shown in Table 2, the least focused autonomic characteristics in industrial projects are self-healing, self-protecting and context-awareness, and the most focused ones are self-configuring and self-optimizing.

6.2 Academic-Oriented Projects

As autonomic computing is a multi-disciplinary research area, a remarkable number of projects in computer science, software engineering, artificial intelligence are related to it. Because of space constraint, we are only able to address a part of these projects in this survey paper (mostly supported by IBM). This section provides a comparison among autonomic computing academic projects based on eight characteristics as discussed Section 3 to compare them in terms of accordance with the designated vision. While Table 2 depicts such a comparison, each project is elaborated further as follow :

- *OceanStore, UC Berkeley :* designs a global persistent data store for scaling to billions of users [25].
- *AntHill, University of Bologna :* supports the design, implementation and evaluation of P2P applications based on multi-agent and evolutionary programming borrowed from complex adaptive systems (CAS) [1].
- *Recovery-Oriented Computing, UC Berkeley/Stanford :* investigates novel techniques for building highly-dependable Internet services, recovery from failures rather than failure-avoidance [29].
- *Autonomia, University of Arizona :* provides the application developers with all the tools required to specify the appropriate control and management schemes to maintain any quality of service requirements [9].
- *eBiquity, University of Baltimore County :* explores the interactions between mobile, pervasive computing, multi-agent systems and artificial intelligence techniques [10].
- *Software Rejuvenation, Duke University :* develops fault management techniques aimed at cleaning up the system internal state to prevent the occurrence of more severe crash failures in the future [34].

As shown in Table 2, the least focused autonomic characteristics in academic projects are context-awareness and anticipatory, and the most focused ones are open and all major characteristics.

7. OPEN PROBLEMS

Although academic and industrial projects realized remarkable parts of the autonomic computing vision, there are still open problems to deal with in this promising research area. This section points out some future directions in the form of open questions that we believe have not been addressed yet or partially covered.

- *Which formalism and techniques are best suited for which purpose?* A wide range of formalisms and techniques can be used to support designing, developing, testing, and maintaining autonomic systems. Each one has its own merits and weaknesses. Hence, we need to identify which ones are most appropriate for which activity, and how the different formalisms and techniques can be combined.
- *How can we inject autonomicity to non-autonomous or semi-autonomous systems?* The problem lies in systems where source code is not available or the coupling between components are high, so it is virtually impossible to include monitoring and actuating functions. Is it possible to develop autonomic functionality separating from application/service functionality [30] or use Aspect-Oriented programming technology [6]? We believe that from software engineering point of view, autonomic characteristics are highly related to quality and non-functional requirements (NFR) [32]. To achieve this, we have to classify autonomic characteristics in terms of their measurable effect on internal quality metrics, and relate these metrics to the external quality factors to which they are correlated. We also need case studies, empirical studies, and controlled experiments to provide anecdotal or statistical evidence about this correlation.
- *How can we analyze and manage the dependencies between autonomous components to address business policies?* One of the points in autonomic computing vision [18] is to design interfaces for translating business policies into I/T policies. This feature is related to *usability* and it is crucial to determine which components are mutually independent. High-level communication languages and knowledge engineering methods may help to address this question but to the best of our knowledge much works still need to be done.
- *How can we build more open/extensible autonomic tools?* Lack of open standards is another challenge for thriving autonomic components and systems in I/T industry. Interpretability and applying global policies for security and configuration are all depend on existence of such standards.
- *How can we measure major and minor characteristics to evaluate autonomic systems?* A novel evaluation framework needs to be developed for bridging the gaps between theses characteristics and the quality factors. Quality assurance in autonomic systems is an area which is still in its infancy.

8. CONCLUSION

The research in autonomic computing continues to be very active. Although commercial tools are beginning to be developed and released, there are still a lots of open issues that remain to be solved. This paper briefly reviewed fundamentals of autonomic computing, discussed three different types of complexity that address characteristics of autonomous systems mentioned in literature. We also reviewed major research areas and techniques which can support autonomic computing with its objectives.

Our study of existing projects in academia and industry indicates that the current projects could be divided into : i) systems which address autonomic characteristics such as Smart, AutoAdmin, AntHill, and Software Rejuvenation, and ii) systems which help to build systems that address these characteristics such as eBiquity and Autonomia. In this paper, we raised a number of open questions that can be used as research agenda for future research within the area of autonomic computing.

9. REFERENCES

[1] Anthill, university of bologna. URL = http://www.cs.unibo.it/projects/anthill/.

[2] J. Appavoo and et al. Enabling autonomic behavior in systems software with hot swapping. *IBM Syst. J.*, 42(1):60–76, 2003.

[3] Autoadmin, microsoft corporation. URL = http//research.microsoft.com/dmx/autoadmin/.

[4] Autonomic computing, the 8 elements. URL = http://www.research.ibm.com/autonomic/overview.

[5] M. Azuma. Applying iso/iec 9126-1 quality model to quality requirements engineering on critical software. In *Proceedings of the* 3^{rd} *IEEE Int. Workshop on Requirements for High Assurance Systems (RHAS)*, 2004.

[6] H. Chan and T. C. Chieu. An approach to monitor application states for self-managing (autonomic) systems. In *Proceedings of the* 18^{th} *Conference on Object-oriented programming, systems, languages, and applications (OOPSLA)*, pages 312–313. ACM Press, 2003.

[7] S. W. Cheng and et al. An architecture for coordinating multiple self-management systems. In *Proceedings of the* 4^{th} *Working IEEE/IFIP Conference on Software Architecture (WICSA)*, page 243. IEEE Computer Society, 2004.

[8] D. M. Chess, C. Palmer, and S. R. White. Security in an autonomic computing environment. *IBM System Journal*, 42(1):107–118, 2003.

[9] X. Dong and et al. Autonomia: an autonomic computing environment. In *Proceedings of IEEE Int. Conference on Performance, Computing, and Communications (IPCC)*, pages 61 – 68, April 2003.

[10] ebiquity, university of baltimore county. URL = http://ebiquity.umbc.edu.

[11] A. G. Ganek and T. A. Corbi. The dawning of the autonomic computing era. *IBM Systems Journal, Special Issue on Autonomic Computing*, 42:5–18, 2003.

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

[13] P. Horn. Autonomic computing: Ibm's perspective on the state of information technology, 2001. http://www-1.ibm.com/industries/government/doc/ content/bin/auto.pdf.

[14] Ibm and cisco unveil innovative approach. URL = http://www-03.ibm.com/autonomic/press_cisco.shtml.

[15] ISO/IEC 9126-1 Standard: Software engineering -Product quality - Part 1: Quality model, Int. Standard Organization, 2001.

[16] J. Jann, L. M. Browning, and R. S. Burugula. Dynamic reconfiguration: Basic building blocks for autonomic computing on ibm pseries servers. *IBM Sys. J.*, 42(1):29–37, 2003.

[17] N. R. Jennings. On agent-based software engineering. *Artifitial Intelligence*, 117(2):277–296, 2000.

[18] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):41–50, 2003.

[19] A. Kluth. Make it simple, 2004. Economist magazine, Survey of Information Technology.

[20] J. A. McCann and M. C. Huebscher. Evaluation issues in autonomic computing. In *Proceedings of Grid and Cooperative Computing Workshops (GCC)*, pages 597–608, 2004.

[21] B. Melcher and B. Mitchell. Towards an autonomic framework: Self-configuring network services and developing autonomic applications. *Intel Technical Journal*, 08:279–290, Nov. 2004.

[22] R. Murch. *Autonomic Computing*. Prentice Hall, 2004.

[23] N1, sun microsystems. URL = http://www.sun.com/software/n1gridsystem/.

[24] Oceano, ibm. URL = http://www.research.ibm.com/oceanoproject/.

[25] Oceanstore, uc berkeley. URL = http://oceanstore.cs.berkeley.edu/.

[26] J. Odell. Objects and agents compared. *Journal of Object Technology*, 1(1):41–53, May 2002.

[27] Optimal grid, ibm. URL = http://www.alphaworks.ibm.com/tech/optimalgrid.

[28] M. Parashar and S. Hariri. Autonomic computing: An overview. *Hot Topics, Lecture Notes in Computer Science*, to appear, www.caip.rutgers.edu/TASSL/Papers/automate-upp-overview-05.pdf.

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

[30] M. Schanne, T. Gelhausen, and W. F. Tichy. Adding autonomic functionality to object-oriented applications. In *Proceedings of* 14^{th} *Int. Workshop on Database and Expert Sys. App. (DEXA)*, pages 725–730, 2003.

[31] Smart, ibm. URL = http://www.almaden.ibm.com/software/dm/SMART/.

[32] L. Tahvildari, K. Kontogiannis, and J. Mylopoulos. Quality-driven software re-engineering. *Journal of Systems and Software, Special Issue on : Software Architecture - Engineering Quality Attributes*, 66(3):225–239, June 2003.

[33] G. Tesauro and et al. A multi-agent systems approach to autonomic computing. In *Proceedings of the* 3^{rd} *Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 464–471. IEEE Computer Society, 2004.

[34] K. S. Trivedi and et al. Modeling and analysis of software aging and rejuvenation. In *Proceedings of the* 33^{rd} *Annual Simulation Symposium(SS)*, page 270. IEEE Computer Society, 2000.

[35] W. Walsh and et al. Utility functions in autonomic systems. In *Proceedings of IEEE conference on Autonomic Computing (ICAC)*, pages 70–77, 2004.

[36] S. White and et al. An architectural approach to autonomic computing. In *Proceedings Int. Conference on Autonomic Computing*, pages 2–9, NewYork, USA, 2004.