Analysis and Design of Metrics for Autonomic Computing

Deepak Kumar Tyagi[*1], Anuraag Awasthi[*2], Ritesh Rastogi[*3]

**Dept. of Computer Applications, Noida Institute of Engg. & Tech. (NIET), Greater Noida, UP, India,*

[1] geit.deepak@gmail.com,

[2] anuraag_awasthi@hotmail.com,

[3] rit_ras@hotmail.com

***Abstract*--Today the computer software/systems have become extremely complex. The increased need to distribute data, applications and system resources across geographical boundaries has led to complexity of the software applications. This situation results in overall increase in maintenance efforts and cost. There is a need to have systems which are self-aware, self-protecting, self-healing, self-optimizing, and self- configuring. This would help in cheaper and faster maintenance. Such computing systems and infrastructure would take care of managing themselves. In an autonomic environment the IT infrastructure and its components are self-managing leading to reduced cost of owning and operating computer systems. The present paper does a study of existing quality characteristics and corresponding metrics, and based on analysis proposes a new set of characteristics as well as metrics to design a robust autonomic computing system.**

***Keywords*–Autonomic computing, self-aware, self-protecting, self-healing, self-optimizing, and self-configuring.**

I INTRODUCTION

The current method of dealing with the increased level of the software application complexity is proportionally increasing the number of persons involved in maintenance. However this manual method is unsustainable at best as the exponential growth of the software industry makes it prohibitively time consuming, expensive and error prone to match the number of maintainers with the complexity of the software systems. The possible solution could be to incorporate self-management in the applications without human interactions. This means letting computing systems and infrastructure take care of managing themselves. In an autonomic environment the IT infrastructure and its components are self-managing. Systems with self-managing components reduce the cost of owning and operating computer systems. Autonomic computing capabilities simply make applications that can take care of themselves. Applications would automatically diagnose their own problems and fix them, reasoning out how to protect them from future bugs. The applications would become self-aware, self-protecting, self-healing, self-optimizing, and self- configuring.

Autonomic systems make informed, constrained decisions without human intervention. They enable dynamic self-management, self-protection and self-optimization of software systems, in turn allowing software systems to react to the changing requirements of their environment. Within the last 20 years the number of computer devices have increased at an unparalleled rate; as has the complexity of both the software and communications infrastructure that supports these devices [1]. An autonomic system provides these facilities for a large-scale complex heterogeneous system. It is a system that manages itself. This definition shows that Autonomic System is emerging as an approach to the design, development, and management of large-scale distributed computing systems.

Salehie et.al [5] proposes a category of complexity in IT Systems. According to them, there are two visions of Autonomic Computing. The first is "Vision in Administration" and second is "Vision in Software Engineering". Nami et.al [9] presented thorough survey of autonomic computing systems, presenting their characteristics, effects on quality factors, their building block architecture and challenges. King et.al [11] presented a reusable object-oriented design for developing self-testable autonomic software. Arnautovic et.al [8] proposed a model software management interactions and tasks in the form of a discourse between the administrator and the software system.

II AUTONOMIC COMPUTING SYSTEMS (ACS)

An Autonomic Computing System (ACS) provides above facilities for a large-scale complex heterogeneous system. An ACS is a system that manages itself. This definition shows that ACS is emerging as an approach to the design, development, and management of large-scale distributed computing systems. According to Paul Horn's definition [4], an ACS is a self-managing system with eight elements. Self-configuration means that an ACS must dynamically configure and reconfigure itself under changing conditions. Self-healing means that an ACS must detect failed components and eliminate 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 [5]. 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 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 lent by it and which resources can be shared. It is self-awareness property. An ACS must be also aware of the execution environment to react to environmental changes such as new policies. It is called 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 enduser view and attempts to satisfy user requests. We consider self-configuration, self-healing, selfoptimization, and self-protection as major characteristics and the rest as minor characteristics.[2] defines decentralized autonomic computing systems as systems that are constructed as a group of locally interacting autonomous entities that cooperate in order to adaptively maintain the desired system-wide self-* properties without any external or internal control. For these systems, implications and interpretation of decentralization on self-* properties can be grand challenges [6][7].

- *Self-Management:* The Autonomic Computing system must free system administrators from the details of system operation and maintenance.
- *Self-configuration:* An autonomic computing system configures itself according to high-level goals, i.e. by specifying what is desired, not necessarily how to accomplish it. This can mean being able to install itself based on the needs of a given platform and the user.
- *Self-optimization:* An autonomic computing system optimizes its use of resources. It may decide to initiate a change to the system proactively (as opposed to reactive behavior) in an attempt to improve performance.
- *Self-healing:* An autonomic computing system detects and diagnoses problems. What kinds of problems are detected can be interpreted broadly: they can be as low level as a bit-error in a memory chip (hardware failure) or as high-level as an erroneous entry in a directory service (software problem). Fault- tolerance is an important aspect of self-healing. Typically, an autonomic system is said to be reactive to failures or early signs of a possible failure.
- *Self-protection:* An autonomic system protects itself from malicious attacks but also from endusers who inadvertently make software changes, e.g. by deleting an important file. The system autonomously tunes itself to achieve security, privacy and data protection. Thus, security is an important aspect of self-protection, not just in software, but also in hardware.
- *Openness:* The Autonomic Computing system must function in a heterogeneous world and implement open standards.
- *Context-awareness:* The Autonomic Computing system must find and generate rules for how best to interact with neighboring systems.
- *Anticipatory:* The Autonomic Computing system must have a projection of the user needs and actions into the future.

TABLE I
AUTONOMIC COMPUTING VS CURRENT 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 follow 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-protections	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.

Table 1 compares the four states of autonomic computing with how we manage today and what it will be like with full autonomic systems.

- *Self-Awareness:* The Autonomic Computing system must be aware of its internal state. 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 landed by it and which resources can be shared.
- *Self-Regulation:* The capability of adapting automatically and dynamically to environmental changes. This characteristic has two aspects as follows: Installing, (re)-configuring, and integrating large Adaptability in architecture or component level to re-configure the system.
- *Environment Awareness:* The Autonomic Computing system must find and generate rules for how best to interact with neighboring systems.
- *Self-Adaptive:* An autonomic computing system must know its environment and the context surrounding its activity, and act accordingly. It will find and generate rules for how best to interact with neighboring systems. It will tap available resources, even negotiate the use by other systems of its underutilized elements, changing both itself and its environment in the process, in a word, adapting.
- *Self-Defining:* The Autonomic Computing system must free system administrators from the details of system operation and maintenance.
- *Self-Expression:* Concerns the possibility of radically modifying at run-time the structure of components and ensembles. For components, this could imply internalizing capabilities previously unavailable to them (e.g., a new function or new sensors) as well as changing their internal architecture (e.g., switching from being reactive entities to goal-oriented ones).

TABLE III
RELATIONSHIP BETWEEN NEW AUTONOMIC CHARACTERISTICS AND QUALITY FACTORS

Characteristics	Metrics
Self-Awareness	Functionality
Self-Regulation	Reliability, Efficiency and Maintainability
Environment Awareness	Usability, Efficiency
Self-Adaptive	Reliability, Portability and Functionality
Self-Defining	Functionality and Portability
Self-Expression	Efficiency and Reliability

IV CONCLUSION

An Autonomic Computing System to be truly self-managing, it is required that it possesses features like self-awareness, self-regulation, environment awareness, self-adaptive, self-defining, and self-expression, apart from the existing ones like self-optimizing, and self-protecting etc. For these additional characteristics, new metrics have been proposed as well some have been reclassified to make them more realistic and logical. These metrics such as maintainability, understandability, reusability, usability, adaptability and modularity can be used to determine external characteristics.

The proposed dynamic metrics will work as indicators in predicting Autonomic Software Systems' external quality characteristics.

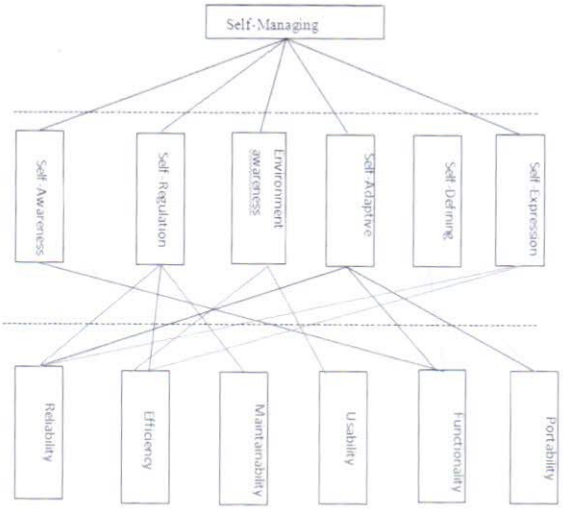

Fig. 2 Relationships between New-Autonomic Characteristics and Quality Factors.

REFERENCES

[1] Naganathan E. R. and Eugene X. P. 2009. Architecting Autonomic Computing Systems through Probabilistic Software Stability Model (PSSM). *In Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human* (ICIS '09). ACM, New York, NY, USA, pp: 643-648.

[2] Abbas N., Andersson J., and Lowe W. 2010. Autonomic Software Product Lines (ASPL).*In Proceedings of the Fourth European Conference on Software Architecture: Companion* Volume (ECSA '10), Carlos E. Cuesta (Ed.). ACM, New York, NY, USA, pp: 324-331.

[3] Sharma, A., Kumar, R., Grover, P. S. 2009, "Reusability Assessment for Software Components – a Neural Network Based Approach", Accepted for publication *in International IEEE Conference* (IACT 09) to be held at Thapar University, Patiala from 26-28 March.

[4] Bradley Simmons and Hanan Lutfiyya. 2005. Policies, Grids and Autonomic Computing. *In Proceedings of the Workshop on Design and Evolution of Autonomic Application Software* (DEAS '05). ACM, New York, NY, USA, pp:1-5.

[5] Salehie M. and Tahvildari L., 2005. *Autonomic computing: emerging trends and open problems.* SIGSOFT Software Engineering. Notes 30, 4 (May 2005),pp:1-7.

[6] Lin P., MacArthur A., and Leaney J.. 2005. Defining Autonomic Computing: A Software Engineering Perspective. *In Proceedings of the Australian Conference on Software Engineering (ASWEC*

'05). IEEE Computer Society, Washington, DC, USA, pp: 88-97.

[7] Beckmann B. E., Grabowski L. M., McKinley P. K., and Ofria C. 2008. Autonomic Software Development Methodology Based on Darwinian Evolution. *In Proceedings of the International Conference on Autonomic Computing (ICAC '08).* IEEE Computer Society, Washington, DC, USA, pp: 203-204.

[8] Arnautovic E., Kaindl H., Falb J., and Popp R.. 2008. High-Level Modeling of Software-Management Interactions and Tasks for Autonomic Computing. *In Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems (ICAS '08).* IEEE Computer Society, Washington, DC, USA, pp: 212-218.

[9] Nami M.R. and Bertels K., 2007. A Survey of Autonomic Computing Systems. *In Proceedings of the Third International Conference on Autonomic and Autonomous Systems (ICAS '07).* IEEE Computer Society, Washington, DC, USA, 26-20.Quitadamo R. and Zambonelli F., Autonomic Communication Services: A New Challenge for Software Agents, Autonomous Agents and Multi-Agent Systems Volume 17, Number 3, pp : 457-475.

[10] Amoui M., Salehie M., Mirarab S, and Tahvildari L. 2008. Adaptive Action Selection in Autonomic Software Using Reinforcement Learning. *In Proceedings of the Fourth International Conference on Autonomic and Autonomous Systems (ICAS '08).* IEEE Computer Society, Washington, DC, USA, pp: 175-181.

[11] King T. M., Ramirez A., Clarke P. J., and Quinones-Morales B., A Reusable Object-Oriented Design to Support Self-Testable Autonomic Software. *In Proceedings of the 2008 ACM symposium on Applied computing (SAC '08).* ACM, New York, NY, USA, pp: 1664- 1669.

AUTHOR BIOGRAPHY

Mr. Deepak Kumar Tyagi is working as Senior Lecturer in department of MCA. He has also guided many students of MCA in various projects. His qualification is M.Tech in Software Engineering from MTU Noida. His area of interest is C, C++, JAVA, Data structures using C.

Dr. Anuraag Awasthi, Director (MCA) is a multifaceted personality with rich corporate, government and academic experience. He has served industry in senior leadership positions in global organizations like IBM, HCL, Bharti Airtel Group, Nihon Unisys etc. in India and many countries like Japan, France and Thailand. He has also been Dean, Faculty of Engineering & Technology, JVW University, Jaipur, and Director, IIMT, Dehradun, before joining NIET. During his vast corporate career, he has been Head of various functions like SW Development, Quality, HR, IT and Customer Support. He has hands-on exposure of leading areas like Information Security, SW Quality, Employee Engagement, HR Services and Systems, Learning & OD. During his long career, he has won several performance excellence awards. He has also been involved with social work as Citizen Warden appointed by Lieutenant Governor of Delhi.

Dr. Awasthi obtained Master of Computer Applications (MCA) from Birla Institute of Technology, Mesra (Ranchi) in January 1988, M.Sc. (TQM) from Kuvempu University (Gold Medalist), PGDHRM from IMT Ghaziabad, LLB from Delhi University and Ph.D. (CS) from Kumaun University in 2005.

Mr. Ritesh Rastogi is working as an Associate Professor in NIET in Department of MCA. He has guided many of the M.Tech students and many projects of MCA. His qualification is M.Tech in Computer Science. His area of interest is Software Engineering, Software Quality Assurance, Software Testing, Managing People, Training & Development, C Programming.