Injecting Autonomic Computing into Legacy Systems: A Survey

Sangeeta P. Sangani and Dr.S.F. Rodd

Abstract--- *In the modern era of technology, many software systems are legacy software systems that have become too expensive to replace and too costly to re-engineer. Considering the need of improving the business values of such systems is the need to do so in the simplest, cost-effective, and least risky way available. There are a number of approaches used by software engineers to mitigate legacy systems by inserting new computing techniques. A relatively recent technique in software engineering concentrate on implementing self-managing attributes, like autonomic behavior in such legacy software and systems of applications in order to avoid the need for human monitoring and intervention. In this paper survey of different techniques and frameworks for building autonomous system have been presented. These techniques considerably reduces the cost of maintaining legacy systems.*

Keywords--- *Autonomic Computing, Legacy System, Self-Management.*

I. INTRODUCTION

IMPROVEMENT of existing legacy system has become a one of the major challenge now a days. Most of the IT vendors try to take advantage of new techniques that avoids the expensive human intervention in order to avoid human errors – including reducing costs and increasing profits. The major challenge is achieving it with adding minimal complexity to the existing system, and without compromising on performance or reliability. These legacy systems are old, and are themselves the “basic building blocks and mortar of many complex systems”, enhancing the functionality leads to new faults and inefficiencies [4]. Engineers always try to adopt best practices and to design better solutions to handle this challenge.

By using some advanced approaches software engineers are trying to face legacy software system challenges. By making use of reusability either for the code or for the architecture software engineers adopts service-oriented architecture design patterns, fault-tolerant, loose coupling between systems and the development of software that has self-managing autonomic attributes is achieved. Using “technology to manage technology” [5], new technique is designed to behave like admin to do monitoring and reconfiguring of a system in real-time and adapt to changing

conditions like change in the data volume, or connectivity, etc.

“Autonomic computing” is a term first proposed by IBM in 2003. Autonomic computing refers to the situation where distributed computing resources are capable of self-manage to unpredictable changes by hiding internal complexity to operators and users. And it is capable of self-managing – that is, self-monitoring, self-configuring, and self-healing. A self-monitoring capability is to “observe and confirm that system is running properly within expected scenario”. A self-configuring capability is “alter the system configuration during run-time automatically to take benefits of or to alter when the system is not performing well within system expected parameters”. A self-healing system “is to identify any anomaly actions and perform some configuring action by itself to correct that anomaly” [2][6].

The Autonomic Computing Initiative (ACI) is inspired by the autonomic behavior of human nervous system. It is capable of controlling body functions like respiration, heart rate, and blood pressure without any major intervention.

To make a system as self-managing system, the operator defines new policies/ rules to guide the system to manage itself instead of controlling the system directly. IBM has defined some major four properties referred as self-star (also called self-*, self-x, or auto-*) properties.

II. SELF -* PROPERTIES OF AUTONOMIC COMPUTING

A. Self-Configuration

Managing large scale systems with respect to Installing, configuring, and integrating is challenging and hectic job even for experts. Many of the corporate data centers are haphazard with servers, routers, databases, and other technologies on verities of platforms from many different vendors. To merge and install an e-commerce application like SAP, it takes months by experts. By using an autonomic system, it will configure automatically to satisfy some objectives at business level, like specifying what is expected, but not how it is to be accomplished. As a new cell in the body or a new person in a population will get incorporate by itself seamlessly, similarly whenever a new component is introduced it will incorporate itself with the other components of the system will adapt to its presence.

B. Self-Optimization

There are hundreds of tunable parameters with some complex middleware, such as Web Sphere, or database systems. Very few experts knowhow to set it properly to get optimal performance. Such self tunable systems are mostly integrated with complex systems. Consequently, performance-

Sangeeta P. Sangani, Assistant Professor., Dept of CSE, KLS Gogte institute of technology, Vishveswaraya University of technology, Belagavi Karnataka. E-mail:spsangani@git.edu

Dr.S.F. Rodd, Professor, Dept of CSE, KLS Gogte institute of technology, Vishveswaraya University of technology, Belagavi Karnataka. E-mail:Sfrodgit@git.edu

DOI:10.9756/BIJSESC.8284

tuning of such one large subsystem can have unanticipated effects on the whole system. The system can be made cost cost-effective by making use of autonomic as it always try better ways to improve its operation. It is as similar as “muscles become stronger through exercise, and the brain modifies its circuitry during learning”.

By monitoring systems` parameters and experimenting with and tuning their own parameters autonomic system will learn to make proper decision about keeping functions or outsourcing them. These Autonomic systems will proactively update themselves by finding, verifying, and applying the latest updates.

C. Self-Healing

Many IT companies including IBM have large department specially dedicated to detect, trace the root cause of failures in complex systems. Some major / minor problems may need some weeks to diagnose and fix it. It even happens that the problem may disappear mysteriously without any particular corrective action. Autonomic computing systems will be used to “automatically detect, diagnose, and repair localized problems resulting from bugs or failures through a regression tester”. Recent autonomic systems with knowledge-base uses historical data (log files) to do analysis of the situation and diagnosis the error. The system tries to compare with existing situation, and then apply the diagnosis procedure automatically without any human intervention.

D. Self-Protection

In most of the security tools software like firewalls & Intrusion Detection tool, a protecting a system is decided by an admin from malicious attacks and inadvertent cascading failures. Autonomic systems will address this problem in two senses. Autonomic system will defend the system as a whole against large-scale, correlated problems arising from malicious attacks or correlated failures that are left unaddressed by self-healing measures. They will tries to address problem based on early detection of errors and doing corrective action immediately.

III. SURVEY

A. Injecting Autonomic into Legacy Systems

In [15] Schanne et al. have briefed out, how to inject autonomic functionality into existing object oriented code. Standard proxy/wrapper class concept is used to incorporate autonomic functionality, like self-updating, self- configuration and self-optimization. The “static reflection” approach is used to get the class structure of a legacy code and then updating the methods with self handling capability. But the drawback observed in this technique user should provide the pre-processor meta-information about the existing code, like, pre and post-conditions and invariants of methods. This method fails to do the actual need, because its impossible to get meta information of many legacy systems. By looking at the byte code user can't get this required meta-information, they must have source code for that, which in real life, may not be available for a legacy system.

Abbas et.al [16] hadan idea of designing an infrastructure

to support existing systems with autonomic computing incorporated by dynamic linked libraries. As it links libraries during run time, it restricts itself to GNU C library. It inserts dynamic linker hooks into existing code wherever is required during runtime. These added blocks of code are used for diagnostic purposes to handle the situation on its own. Since their approach works only with C, it fails at platform independent environment.

Kinesthetic eXtrime [17] gives a completely new idea for injecting autonomic into legacy system. According to DASADA standards [18] probes are used to monitor the process of existing system and gauges to adapt an existing system. The legacy code is retrofitted with probes which collects information from all process at runtime and gives it to a centralized array of gauges. By these gauges overall system status is identified and these probes are replys back for any feedback. According to the vision of autonomic system in [19], “each sub-components of the system should be self managed by it-self and hence a whole the system would become self managed also”. In a similar way, this technique collects status information from the different components and tries to upgrade the whole system. There is chance of high communication traffic because all the probes and gauge need to communicate. Self-healing capabilities is incorporated at whole system level but not at single component level.

J. J. Mulcahy [3] has described code reuse concept of software engineering by using a concept of service oriented architecture. Self-* of autonomic computing system are also used to extend the legacy enterprises system of a multi-channel vendor of musical instrument. By these approaches, they were able to increase the orders placed by customers, and hence extends the business value of that system.

B. Autonomic Systems

As mentioned in [9], “there is no full-fledged autonomic system neither in the business domain nor in the research domain”. Most of the autonomic systems till today are with a limited amount of required functionality [8,10] of an autonomic system. The majority of the systems are lacking in explaining how to write programs in such systems or how to utilize such systems in a simpler fashion. They either starts with new metaphors or provide a completely new approach to autonomic computing which will increases the complexity. The main idea is to introduce a system which is simple to operate, by having transparent autonomic framework. Some of the systems are mentioned below.

The Unity system [11] designed a platformwhere interaction of autonomic elements with each other and with their environment is done properly. It can configure by itself by some utility function. But, this utility function assumes that one can quantify the utility of different choices. As this Unity system did not explains about complexity of using it by application programmers.

Autonomia [12] uses mobile agent technology to design autonomic applications. It provides a platform where application developers can specify and implement distributed application with autonomic properties. Autonomia can only address self-configuration and self-healing properties of

autonomic systems. The users of Autonomia are restricted to follow a well defined library and a pre defined Application Service Template to create their programs. So user needs to study the underlying system and understand all its interface specifications to program on Autonomia.

Auto Mate [13] is an environment for executing Grid based autonomic applications. Auto Mate uses these Grids services, and it calculates a composition plan of different components based on dynamically defined objectives and constraints that describe how a given high- level task can be achieved. It has a powerful set of tools for programming framework, autonomic composition, and for the coordination of middleware etc. However, as with IBM's toolkit, Auto Mate does not address the complexity of integrating autonomic functionality into applications. It has many new features and paradigms, learning these would be more complex.

C. Autonomic programming Environments

Most of research projects try to provide support for heterogeneous platforms but fails to achieve complete autonomic capabilities. Some projects comes with transparency of code and easy to use autonomic framework along with the programming environments.

IBM's Autonomic Toolkit [18] has set of tools and has different APIs for different phases like monitoring, analysis, planning, and executing autonomic applications. This toolkit contains several class libraries, plug-ins and tools for the Eclipse development environment [19]. Internally these tools have many inter-dependencies between functionality for an autonomic environment, But this will not effect complexity of integrating autonomic functionality into applications. But these tools are not helpful for the programmers to design their autonomic applications. It has limitations of communication primitives required for more general autonomic environment.

Developers of autonomic systems as in [20], believes that this toolkit is difficult to incorporate into other distributed environments and required some extra effort by the application programmer to use its functionality. Authors opinion that, this approach is completely diverse. Application programmers relieved from the burden of such complex and ever changing metaphors. It is proposed that, keeping programming an autonomic system transparent to the user and easy to use is necessary, otherwise the goal of autonomic computing is sacrificed.

Orso et. al. [20] gives a technique to update a running java application dynamically. By making use of proxy classes they can rewrite their code and allow substituting, adding and deleting of classes at run time. This technique modifies the application code by rewriting a class and by renaming statically. It will take care of hot swapping of class at run time whenever a new version of a class is available. As this technique provides a novel way of reconfiguring an existing distributed system, but hot-swapping is done through user only which violates self property of autonomic computing. By making use of self-configuration of autonomic elements it can be improved.

IV. CONCLUSION

In This paper brief survey of different techniques of making legacy system as autonomic systems, autonomic environments and autonomic systems has been presented. As there is essential need for improving the performance of legacy systems, this paper gives an idea about different techniques for injecting autonomic computing into legacy systems. And It has been observed that there is a scope for use of modern computing techniques to further improve the performance & efficiency of legacy systems. A new approach like machine learning technique can be used to impose self-configuring, self-healing, self optimizing and self-protection properties into legacy systems to improve the performance. Hence it becomes cost effective solution to maintain legacy systems, avoids complete replacement of old systems.

V. REFERENCES

- [1] M.J. Khan, S. Shamil and M.M. Awais, "Decision making in autonomic managers using fuzzy inference system", In 2009 Fifth International Conference on Autonomic and Autonomous Systems, Pp. 214-219, 2009.
- [2] C. Schneider, A. Barker and S. Dobson, "A survey of self-healing systems frameworks", Software: Practice and Experience, Vol. 45, No. 10, Pp. 1375-1398, 2015.
- [3] J.J. Mulcahy and S. Huang, "Autonomic Software Systems: Developing for self-managing Legacy systems", IEEE International Conference on Software Maintenance and Evolution. Pp. 549-552, 2014.
- [4] D. Garlan, "A 10-year Perspective and Software Engineering Self-Adaptive Systems", Keynote at the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, 2013.
- [5] Y. Brun, G. Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M and M. Shaw, "Engineering Self-Adaptive Systems through Feedback Loops", In Software Engineering for Self-Adaptive Systems. Springer Berlin Heidelberg, Pp. 48-70, 2009.
- [6] S. Neti, H.A. Müller, "Quality criteria and an analysis framework for self-healing systems", In International Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2007.
- [7] T. Haupt, "Towards mediation-based self-healing of data- driven business processes", In Software Engineering for Adaptive and Self-Managing Systems (SEAMS), 2012.
- [8] J.O. Kephart and D.M. Chess, "The vision of autonomic computing", Computer, Vol. 36, No. 1, Pp. 41-50, 2003.
- [9] M. Salehie and L. Tahvildari, "Autonomic Computing: Emerging Trends and Open Problems", SIGSOFT Software Engineering Notes, Vol. 30, No. 4. Pp. 1-7, 2005.
- [10] S.R. White, J.E. Hanson, I. Whalley, D.M. Chess, J.O. Kephart, "An Architectural Approach to Autonomic Computing", First International Conference on Autonomic Computing (ICAC'04), Pp. 2-9, 2004.
- [11] D.M. Chess, A. Segal, I. Whalley and S.R. White, "Unity: Experiences with a Prototype Autonomic Computing System", First International Conference on Autonomic Computing, Pp. 140-147, 2004.
- [12] X. Dong, S. Hariri, L. Xue; H. Chen; M. Zhang, S. Pavuluri and S. Rao, "AUTONOMIA: An Autonomic Computing Environment", In Communications Conference on Proceedings of the 2003 IEEE International Performance, Computing, Pp. 61-68, 2003.
- [13] M. Parashar, H. Liu, Z. Li, V. Matossian, C. Schmidt, G. Zhang and S. Hariri, "Auto Mate: Enabling Autonomic Grid Applications", Cluster Computing: The Journal of Networks, Software Tools, and Applications, Special Issue on Autonomic Computing, Kluwer Academic Publishers, Vol. 9, No. 1, 2006.
- [14] A.B. Brown, J. Hellerstein, M. Hogstrom, T. Lau, S. Lightstone, P. Shum and M.P. Yost, "Benchmarking Autonomic Capabilities: Promises and Pitfalls", First International Conference on Autonomic Computing , Pp. 266-267, 2004.
- [15] M. Schanne, T. Gelhausen and W.F. Tichy, "Adding autonomic functionality to object-oriented applications", 14th International Workshop on Database and Expert Systems Applications, Pp. 725-730, 2003.

- [16] N. Abbas, M. Palankar, S. Tambe and J.E. Cook, "Infrastructure for Making Legacy Systems Self-Managed", 2004 Workshop on Self-Managing Systems, 2004.
- [17] G. Kaiser, J. Parekh, P. Gross and G. Valetto, "Kinesthetics eXtreme: An External Infrastructure for Monitoring Distributed Legacy Systems", Fifth Annual International Workshop on Active Middleware Services, Pp. 22-27, 2003.
- [18] B. Balzer, "Probe Run-Time Infrastructure", .
- [19] P. Horn, "Autonomic Computing: IBM's Perspective on the State of Information Technology," IBM Corporation, 2001.
- [20] A. Orso, A. Rao and M. Harrold, "A Technique for Dynamic Updating of Java Software", 18th IEEE International on Software Maintenance, Pp. 649-658, 2002.

Sangeeta Sangani, currently working as Assistant Professor in the department of CSE, KLS, Gogte Institute of Technology, Belagavi, Karnataka, India. Area of interest are Autonomic Computing, Distributed Computing, Machine Learning

Sunil F Rodd, currently working as Head & Professor in the department of CSE, KLS, Gogte Institute of Technology, Belagavi, Karnataka, India. Area of interest are Autonomic Computing, Performance Tuning of Database Management Systems, Distributed Computing, Machine Learning.