Advances in Computer Science and Information Technology (ACSIT)
p-ISSN: 2393-9907; e-ISSN: 2393-9915; Volume 3, Issue 5; July-September, 2016, pp. 448-451

http://www.krishisanskriti.org/Publication.html

Current State of Research on Self-healing Approach

Surbhi Khurana[1] and Aashish Grover[2]

[1]IGDTUW, Delhi
[2]IIIT, Delhi
E-mail: [1]surbhikhurana04@gmail.com, [2]grover.aashish498@gmail.com

Abstract—*Nowadays, computing system environments are intricate and multiphase. Due to their competence of having heterogeneous configuration they may escort to some erratic state. Extensive research on designing domains and decision making techniques enhances these systems with an autonomous behavior. The endeavor of this survey, is to focus on the self healing approach from Self-* properties of autonomic computing and to provide an overview of the current existing approaches. In this survey we classify the need of self-healing, self-healing systems fundamental principles, approaches and autonomic computing tools.*

1. INTRODUCTION

1.1. Autonomic systems

An autonomic system controls the functionality of computer applications and systems with minimal or almost zero dependency on user for inputs, in the same way, human body nervous system regulates [1]. The goal of autonomic computing is to create a system that run themselves, capable of high-level functioning while keeping the system's complexity invisible from the user. Hence, the fundamental for autonomic computing includes four properties also known as self * properties:

- *Self-configuring:* The capability of system to alter the functionality of its components itself, during runtime.
- *Self-healing:* The capability of system to discover, diagnose and recover from the breakdown to normal state.
- *Self-optimization:* The capability of system to maximize its efficiency by altering the resource utilization in order to meet the end-user need.
- *Self-protection:* The capability of system to sense, recognize and protect itself from malicious codes.

1.2. Reliability

Reliability can be defined as a measure of trust and belief that, the system will deliver the exact result which is expected from it, regardless of any internal or external conditions, without any deviation. Thus, the motivation behind introducing the autonomic behaviour totraditional computing environment is, to make the user, free from the task of discovering the system failure and system recovery from unwanted state. Hence, the objective is to increase the reliability of system. A self healed system provide this functionality by discovering system faults and hence, takes considerable steps to acquire a system from ambiguous to unambiguous state without human intervention [1].

1.3. Modularity

Traditionally computing systems were developed with an intention to fulfill a specific set of requirements, which make them simple and less complex. Now, the trend has been shifted to diversifying the computing capabilities of a system, by introducing modularity into system to amplify its functionality. Thus, because of these heterogeneous configurations, maintenance costs of the systems are escalating hastily [2].This results to the increase in system complexity [3].

2. SELF-HEALING IDEOLOGY

This segment focuses on the detailed ideology of self-healing in systems. We begin with the definition of self-healing systems and then, proceed with recognizing the different phases of a self-healing in system.

2.1 An outline of self healing

IBM [1] points self healing, as one of the foremost property, while defining an autonomic system as,

“a system which recovers from the ambiguous (or “faulty”) condition to the normative or standard condition, without any degradation in the functionality of other healthy modules”. [4]

Systems with self-healing technique might get confused with fault-tolerant or survivable systems, as the expectation from both the systems are alike. For the purpose of recovery, fault-tolerant systems encompass various stabilizing technique and replication strategies [5].

As a result, Ghosh et al. [6] divulge that, in some circumstances fault-tolerant systems works as chief to self-

healing systems. Whereas, survivable systems bulge wicked actions and secure the "essential services", a minimal set of functionality like system configurations [6, 7].

System resource availability and its efficiency are the foundation for enhancing a system with self-healing property. With respect to transactional phase, self-healing techniques takes the charge of low maintenance for the system. For incessant work, it includes buoyancy against obligatory adaptations and unpremeditated arbitrary behavior. Self-healing execution works, by finding failure causes, in order to derive a perfect solution and a sound strategy for recovery. Additionally, some hardware like sensors and actuators are also embedded for the accuracy, but the only constraint for success, is the timely detection of system misbehavior. This can be achieved by incessantly analyzing the sensed data as well as observing the output of necessary adaptation deeds.

2.2 Self healing elements

Autonomic capacity of a system depends upon the working of major elements of autonomic control loop. These are referred as MAPE loop [1, 8]. This loop consists of a manager that holds five distinct functions. These can be defined as follows:

- *Monitor:* It assembles the status information from the system through the sensors.
- *Analyse:* It analyses the gathered data and verifies whether the monitored information is pursuing the designated set of actions or not.

Plan: A system behaves differently in different scenarios. So a precise, accurate, and polished deployment of the actions is required. This phase keep track of policies that must be followed, to achieve the specified set of goals.

- *Execute:* It executes the parts of previously envisaged plans on the managed elements.
- *Knowledge:* It represents the knowledge base consumed and produced by all four previously mentioned tasks.

These five autonomic processes are now reduced into three main stages of a loop. Kephart and Chess [9] named these stages as detection, diagnosis and repair. According to Salehie and Tahvildari, [10] it can be considered as integration of self-diagnosing and self-repairing with discovery, diagnosing and reacting stages. Parashar and Hariri [11] believe in the existence of only "detect" and "recover" stage. According to Huebscher and McCann, [8] the three main stages in the loop can be categorized into three actions namely detect, diagnose, and fix. Taking into consideration all of the above researches, we can say that detection is the first phase of action.

2.3 Stages of Self healing

System robustness should never depend upon a single element. Even if there is some failure then it must opt for graceful degradation approach and the system as a whole should be able to recover from the encountered failures [12].

Ghosh et al. [6] suggests a model, featuring a fuzzy transition zone describing an unclear "Degraded State".
According to this concept, major super computers usually do not immediately quit operations when smaller portions fail, but continue to operate with a possible considerable loss on performance. This provides enough amount of time to cure the system and to bring the system back, without complete disruption.

The components of Ghosh et al. model can be described as follows:

- Normal state: This state included the normal working of a system
- Broken State: In this state the system becomes a dead system
- Degraded state (fuzzy zone): This state includes the fault detection phase

The next issue observed is the state explosion; it can be defined as a problem of large systems with many concurrent processes, where the number of processes may cause the number of possible states to grow exponentially. Clarke and Grumberg [13] proposed a solution to handle the above stated issue. They proposed a scheme where firstly all the states are discovered which are sharing some common properties with other states and after that a clustered set for the above states is made.

2.4 Self healing policies

Policies defines boundary for the actions to be taken, in order to cure the faulty system. Kephart and Walsh [14] in their research proposed three different types of policies:

- *Action Policies (Reaction):* This is a type of policy that defines all the set of actions to be taken on a certain fault; however it is similar to an IF-THEN statement.
- *Goal Policies (Routine):* Here, the goal or desired state is specified and after that the system takes necessary steps in order to take the machine from the current sate to a specified or desired state. Here, the routine level is defined to be the one, where largely routine evaluation and planning behaviours takes place.
- *Utility Function Policies (Reflection):* Utility function policies attach a significance value to each possible state at runtime, depending on the current state. Here, the results of the problem are dependent on the information obtained from its history, system capabilities, current system state, and current environment state.

3. IMPLEMENTED APPROACHES

3.1 Embedded systems

Embedded systems works in a very different environment with very limited hardware components or constrained environment. Glass et al. states that, redundancy models (with duplicate parts) must be dedicated to single resource for one specific task, and suggests a check pointing approach. Failure

detection can be done by exchanging a “keep alive” message between a task and its clone called as shadow-task. Thus, the shadow task is ready to take over, once the “keep alive” message disappears and reconfiguration of the network is initiated [15, 16].

3.2 Operating systems

In order to handle recovery, Herder et al. [17] in their paper suggests two special types of server named as *reincarnation and data server*. The reincarnation server is a master process and is responsible for noticing slave thread fails. The data server holds status of the slave threads and the master. For each element, recovery policies are stored and the common component replaces the fading application.

4. TYPES OF SYSTEMS

4.1 Intrusive and non-intrusive systems

The aim of self healing systems is not only to recover from ambiguous state but also to adept environmental changes to make system more reliable.

Intrusive systems are adapted to support the self healing extensions. There are various methods which are used for attaining system awareness, support adaptations. It might hinder with the original design and alter the timings of data exchange and logic decisions. Apart from these disadvantages, a buffed and firm guarded system can guarantee an optimal integration in the time domain. Detection and Recovery becomes more precise in such cases.

A non-intrusive alignment of self-healing techniques respects the guarded system as a complete unit. It never interferes with the functionality of other modules. In addition to this, it even works and deploys as a single module itself which has the capability of integrating with the original system. Adaptation and monitoring are optional in this case. Even though, Non-intrusive approach is the preferred way of integration, however it is less applied because its efficiency depends fully on the capabilities and characteristics of the supported system with whom it is attached.

4.2 Closed vs. Open system

Stabilization is one of the necessary requirement and need of any system. However, guaranteeing stability is an almost impossible task in systems with volatile behavior. Therefore, to neglect this behavior designers try to avoid the known failure sources. Closed-loop implementation is adopted by most of the self-healing approaches. Self-healing techniques are usually aligned to systems to enhance the long term use. Thus, some of the researched works introduce at least indirect influence by allowing dynamic handling of policies.

4.3 Recovery techniques

Various recovery techniques are proposed to take out the system from the state of failure. However, it may somehow be redundant in itself. In hardware recovery self healing implementations can be done by only spare parts duplicates or additional resources, software recovery can also be done by relocation of resources or services.

Various approaches that can be used are as follow:

- *Replacement:* This includes replacing the faulty parts. Software rerun includes freeing the allocated resourced from the faulty instance, and starting a fresh new application instance.
- *Balancing:* This includes introducing extra resources or killing some processes.
- *Isolation:* This includes cutting of a failing part of the system.
- *Persistence:* It assumes no further degradation, and if faulty modules wants to rejoin, then it must takes its own actions.
- *Redirection:* This includes navigating the data flow for new routes.
- *Relocation:* This includes moving an application to a different host i.e. re-directing.
- *Diversity:* This includes moving to a different approach to solve the tasks once.

5. DIFFERENT TOOLS

In this section we have discussed some of the tools which can be used for Windows as well as for UNIX operating system for the fault detection purposes.

Tool or Service	Use and Compatibility
Windows Event Viewer	Windows
Window self monitoring tool	Windows
Service.msc	Windows
ABLE toolkit[18]	UNIX
ABLE toolkit	To increase Web server performance[19]

6. CONCLUSION

Systems based upon autonomic computing would control the functioning of computer applications, without taking input from the user. The goal of autonomic computing is to create systems that can run themselves and are capable of high-level functioning while keeping the system's complexity invisible to the user. This initiative ultimately aims to develop computer systems capable of self-management, to overcome the rapidly growing complexity of computing systems management, and to reduce the barrier that complexity poses to further growth. Now, the system makes decisions on its own, using high-level policies. System will constantly check and optimize its status and automatically adapt itself to changing conditions. In this survey we discussed about various techniques and tools to support self-healing, a property of autonomic computing.

REFERENCES

[1] IBM, "An architectural blueprint for autonomic computing", *IBM, 4th edition*, June 2006.

[2] Jeongmin P, Jinsoo J, Shunshan P and Eunseok L,"Self-healing Mechanism for Reliable Computing", *International Journal of Multimedia and Ubiquitous Engineering*, vol. 3, issue. 1, 2008.

[3]Ganek AG and Corbi TA, "The dawning of the autonomic computing era", *IBM*, vol. 42, issue. 1, January 2003, pp. 5–18.

[4] Paul H "Autonomic computing: IBM's Perspective on the State of Information Technology", 2001.

[5] Pierce W, "Failure-tolerant computer design", *Academic Press*, New York, 1965

[6] Ghosh D, Sharman R, Raghav Rao H and Upadhyaya S, "Self-healing systems—survey and synthesis,Decision Support Systems", vol. 42, issue. 4, 2007, pp. 2164–2185.

[7]. Ellison R, Fisher D, Linger R, Lipson H, Longstaff T and Mead N (1999) "Survivability: protecting your critical systems", *IEEE*, vol. 3, issue. 6, November/December 1999, pp. 55–63.

[8]Huebscher MC and McCann JA, "A survey of autonomic computing—degrees, models, and applications", *ACM Computer Security*, vol. 40, issue. 3, 2008, pp. 1–28.

[9]Kephart JO and Chess DM, "The vision of autonomic computing", *IEEE Computer Soc Press*, vol. 36, issue. 1, 2003, pp. 41–50.

[10] Salehie M and Tahvildari L, "Self-adaptive software: landscape and research challenges", *ACM Trans Auton Adapt System*, vol. 4, issue. 2, 2009, pp. 1–42.

[11] Parashar M and Hariri S, "Autonomic computing: an overview. In: Unconventional programming paradigms", *Springer*, Berlin, 2005, pp. 247–259.

[12] White S, Hanson J, Whalley I, Chess D and Kephart J, "An architectural approach to autonomic computing", *In: Proceedings international conference on autonomic computing*, 2004, pp. 2–9.

[13] Clarke EM and Grumberg O, "Avoiding the state explosion problem in temporal logic model checking", *In: PODC Proceedings of the sixth annual ACM Symposium on Principles of distributed computing ACM*, New York, 1987, pp. 294–303

[14] Kephart J and Walsh W, "An artificial intelligence perspective on autonomic computing policies", *In: Proceedings fifth IEEE international workshop on policies for distributed systems and networks, POLICY*, 2004, pp. 3–12.

[15] Glass M, Lukasiewycz M, Streichert T, Haubelt C and Teich J, "Reliability-aware system synthesis. Design, Automation and Test", *In Europe Conference & Exhibition*, 2007, pp. 1–6.

[16] Glass M, Lukasiewycz M, Reimann F, Haubelt C and Teich J, "Symbolic Reliability Analysis of Self-healing Networked Embedded Systems", *In: SAFECOMP '08: Proceedings of the 27th international conference on computer safety, reliability, and security. Springer*, Berlin, 2008, pp. 139–152.

[17] Herder JN, Bos H, Gras B, Homburg P and Tanenbaum AS, "MINIX 3: a highly reliable, self repairing operating system", *SIGOPS OperSyst Rev*, vol. 40, issue. 3, 2006, pp. 80–89

[18] Bigus JP et.al., "ABLE: a toolkit for building multiagent autonomic systems," *IBM Systems J.*, vol. 41, issue. 3, 2002, pp. 350-371.

[19] Y.Diao, J. L. Hellerstein, S. Parekh and J. P. Bigus, "Managing Web server performance with AutoTune agents", *IBM Systems Journal*, vol. 42, 2003.