Testing Autonomic Computing Systems: Principles, Challenges, Future Directions

Manas Kumar Yogi[1] K.Chandrasekhar[2] Sai Indraja Munagapati[3]
[1,2]Assistant Professor [3]B.Tech. Student
[1,2,3]Department of Computer Science & Engineering
[1,2,3]Pragati Engineering College Surampalem, Andhra Pradesh, India

Abstract— Autonomic systems have ability to self-configure, self-manage, self-optimize .Additionally they can self-heal in case of failures. This paper envisions the key principles while testing autonomic systems and also discusses the difficulties encountered during this testing process. Researchers are actively working towards facing these challenges and in near future a comprehensive framework will be established to test the functionality of autonomic computing systems.

Key words: Autonomic, Self-Healing, Testing, Touch point, Knowledge Source

I. Introduction

Autonomic computing systems are future computing system armed with properties of self-management which makes problem solving easier than expected. But along comes design challenges. To make testing of autonomic computing system easier we have self-testing methods but they don't implement test cases to verify at unit level the functionality of task managers and autonomic managers which form the core of the architecture .Autonomic computing systems have the ability to manage the resources efficiently by sharing the server loads. Till now we do not have any cost effective testing framework for testing autonomic computing systems nor we do have established mechanisms to test the functionality of the system using suitable test metrics. Although researchers are working in this direction, as the customer requirements are changing with time the dynamic behaviour of the autonomic computing system is also ever changing thus making all the relevant static implementations of test design for the cause obsolete. The basic approach followed for testing such a system is a black box testing. Tests are designed in such a way to uncover potential errors in the system. Tests can be rerun after bug fixing work and evaluated for functional performance. Tests results are well documented and test logs are stored in a structured manner.

II. Principles

There are many aspects to be tested for autonomic computing systems. First one is testing the components of a autonomic system. The following components are to be tested:

1) Task manager: User interface component to perform management functions. To test the functionality of the task manager we use a black box testing approach where we develop sufficient number of test cases to validate the input from the user interface component. The test cases are designed in such a manner that they test each feature of the interface with respect to responsiveness of the system.
2) Autonomic Manager: Autonomic managers monitor resource details, analyze those details, plan adjustments, and execute the planned adjustments. To test the autonomic manager we verify if all resources under the system are correctly displayed along with their capability. Further we construct few mock plans and verify if they are adjustable under the operational limits.
3) Knowledge source: Source for business data and IT policies. This knowledge source can be tested on similar lines of database testing. We can implement tests to cover performance, stress,load on the knowledge source.
4) Touchpoint: It includes servers, databases, storage devices. To verify the functionality of touch point, we design and execute test cases for concerned servers as well as databases. Tests to be conducted are load, stress, performance. Testing storage devices is done as per hardware testing techniques like installation testing etc.
5) Enterprise service bus: It consists of communication components following web standards. Configuration testing is done to verify if the service bus is able to transmit required data from one component to other within the autonomic computing system.

Moving further, we need to test the self healing, self configuring, self protecting, self optimizing abilities of the autonomic system.

Testing the self configuring, self optimizing properties of a autonomic system uses a black box testing approach. The test cases take input a pre defined configuration and verify if the output is accordingly correct or not. The test cases are designed to verify the weighted average of the deviations of the response time ,probability of rejection metrics. The test cases are then executed and test results are documented .

For designing test cases to verify the self healing property we have to know the granularity of the components fault tolerance level. For example if performance degrades beyond certain level then automatically self healing mechanism must be activated. To test this principle we must introduce inputs which degrade the system and at one point a severe fault occurs. If the tests are able to find the fault then the specific self healing property is tested successfully. For example we have to design test cases which fail the system. If test cases are failed after execution then actual outcome, expected outcome are same and hence our test cases are deemed to be verify the self healing characteristic of the system correctly. For verifying the self protecting property we must perform security testing. We have to understand the various security measures incorporated in each autonomic components. As web standards are followed automatically security like encryption of user credentials will exist implicitly. Our test cases should be designed in such a manner that they bring out the weaknesses if any present in the system.

III. KEY CHALLENGES

Autonomic computing systems are highly adaptive in nature so we need to develop test cases which cover the adaptive behavior .But this is not as easy as we think. We need predictive test cases to determine the outcome of a operation in such dynamic systems. Construction of predictive test cases takes considerable amount of time, effort as designing such test cases are difficult. Test case automation tools existing now are incapable of efficient execution in a multi environment system. Existing self testing frameworks address this challenge but only partially. We assume the system is in safe state before performing all types of tests. The self-testing framework consists of test managers, test knowledge sources and auxiliary test services that collaborate to provide validation services for the autonomic system. But the major problem here is that there is no formal specification for generating test sequences. Determining the preconditions, post conditions, invariants for a autonomic computing system is exhaustive. Another set of challenges is maintaining test resources separately is costly for a autonomic system. So minimal testing resources are pooled to test the functionality of the system. Also instantiating similar copies of software for testing may introduce negative impact on the system. Black box testing technique needs training to give efficient performance and the training time is considerable which poses a difficulty.

IV. FUTURE DIRECTIONS

Researchers are working to create robust frameworks to test autonomic systems and also a test driven autonomic computing system approach. Models using petri nets, markov chains to validate the risk involved with testing a autonomic system are been looked into with substantial interest. Further work is related with mutation testing techniques in this area.

V. CONCLUSIONS

Testing autonomic computing systems is not easy as it seems due to multi variant factors .A robust technique is still need to test the functionality of a autonomic computing system. The self healing, self managing, self optimizing properties of a autonomic system are still to be tested to a satisfactorily level. We have included in this paper the key principles, challenges and future directions.

REFERENCES

[1] Philip Koopman. Elements of the self-healing system problem space. In Proceedings of ICSE Workshop on Architecting Dependable Systems (WADS), 31–36, May 2003.
[2] G. Eisenhauer and K. Schwan. An object-based infrastructure for program monitoring and steering. In Proceedings of the 2nd IGMETRICS Symposium on Parallel and Distributed Tools (SPDT98), 10–20, August 1998.
[3] S. M. Sadjadi and P. K.McKinley. Transparent self-optimization in existing CORBA applications. In Proceedings of the 1st IEEE International Conference on Autonomic Computing, 88–95, May 2004.
[4] Gail Kaiser et al. Kinesthetics eXtreme:An external infrastructure for monitoring distributed legacy systems. In Proceedings of The Autonomic Computing Workshop 5th Workshop on Active Middleware Services (AMS), 22–30, June 2003.
[5] Shang-Wen Cheng et al. Rainbow: Architecture-based self-adaptation with reusable infrastructure. In IEEE Computer, 46–54, October 2004.
[6] Bradley Schmerl and David Garlan. Exploiting architectural design knowledge to support self-repairing systems. In Proceedings of the 14th International Conference of Software Engineering and Knowledge Engineering, 241–248, July 2002.