Testing self-adaptive systems - an overview

Tilo Werdin, Dominik Olwig

May 26, 2017

Contents

1 Abstract

During the past years, the requirements for software-systems have changed dramatically. Software now often needs to be self-adaptive. This also leads to a different process of development. One important aspect of developing software is testing. Due to the unflexibility of traditional tests, there is a need for new testing-methods. There are some approches, that target this problem. What is missing is an overview about them.
Therefor we want to categorize, shortly describe and evaluate the different ideas, that exist in current literature.
In order to find as many different approches as possible, we apply the snowballing-method during our research and we structure them using different categorization techniques.
The outcome of our research will be a taxonomy and an evaluation of state of the art testing techniques for self-adaptive systems.

Keywords: self-adaptive systems; system testing

2 Introduction

For engineering software there are several approaches. One that got a lot more important over the past years is self-adaptive software for self-adaptive systems (SAS). SAS is able to adjust to different situations in different places all on it's own. The great benefit and the cause of why the field of SAS is researched now is that the software engineer does not program all exact situations that the system might be in. The software engineer just models some situations and abstracts from that. The system itself then should find a way to behave in a good way.

This is done by control loops. A control loop in a SAS contains four main parts: monitoring, analysis, planning and execution (MAPE). As the name loop tells, the control loop operates frequently and influences the behavior of the system. While monitoring and execution interact directly with physical parts of the system, the analysis and planning parts of the loop calculate the adjustments the system should made.

The greatest thinkable application of SAS might be a human like robot that behaves completely autonomous. It would be designed by some system architect and programmed by some software engineer, but they will never imagine every single situation the robot might be in. The solution is, developing a system that adjusts on it's own "learns" over the time. A smaller example could be an application that uses variable web services. In this application it shell not be important that a special service is reachable. The system chooses at runtime one of the services available and that could be very different ones each time the system runs.

Looking at the examples above, it is clear that these SAS need to be tested, in order to assure quality. This includes validation and verification of the systems. The worst case would be that there is a system produced that uses self-adaptation. It is delivered to the customer but was never tested and validated before. The customer starts the system and it not working as expected and the system failure leads to emergencies. Therefore it is very important to test self-adaptive systems.

While SAS are constructed in an other way than "normal" systems and the process of developing SAS is quite different there is the need of new ways to test systems and their behavior. Traditional Unit-Tests will reach their limits as the testing engineer needs to think of every single test case on his own and as mentioned before it is nearly impossible to think of every single situation the SAS might be in.

There are already different approaches on testing SAS that we want to discuss and compare to each other in this paper. On the one hand there is self-testing with different techniques and on the other hand there is model-based testing.

missing: method (snowballing), motivation (overview) maybe to much detail in the first to paragraphs

3 Method and result

3.1 Self-Testing

basic idea... - monitor system-components and if constraint violated -¿ do recovery-action - constraint can be a result or a QoS-constraint (e.g. response time) - keeping the monitored data can be a good source for later offline-tests

3.1.1 Testmanager

One idea for a self-test architecture is a testmanager. In general this is a software-component, that runs simultanously to the monitored system.
There are two forms of using a testmanager:

1. safe adaption with validation
Every time the system perceives a contextual change it notifys(?) an internal adaptation-manager. It decides wheather an adaption is needed. If the system wants to adapt, the adaptation-manager will initiate an adaption and at the same time notify the testmanager. After the adaption completed, all actions targeting the system were blocked. In the meanwhile the testmanager is executing a set of tests that depend on requirements that the adaption-manager sent. When all tests were finished the result is sent back to the system. The systems adaptation-manager will keep the changes, if the tests were successful or it will recover the old system-state, if a test failed.

2. replication with validation The main idea of this architecture is similar to the idea of safe adaption with validation. When the system needs to do an adaptation it notifies a test-manager. But in contrast

3.1.2 Corridor Enforcing Infrastructure

3.2 model-based testing

4 Evaluation

4.1 Criteria

what are our evaluation critera?
- control during runtime
- ensure quality before deploy
- performance overhead (time, memory)
- testing-cost (how complex is building the test)
- adaptability (how easy can the test-suite be adapted)
- maybe something like: Generalization (how general is the approach)

4.2 Comparison

5 test

"hello world"
[2] [1]

References

[1] R. de Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu, B. Schmerl, G. Tamura, N. M. Villegas, T. Vogel, D. Weyns, L. Baresi, B. Becker, N. Bencomo, Y. Brun, B. Cukic, R. Desmarais, S. Dustdar, G. Engels, K. Geihs, K. M. Göschka, A. Gorla, V. Grassi, P. Inverardi, G. Karsai, J. Kramer, A. Lopes, J. Magee, S. Malek, S. Mankovskii, R. Mirandola, J. Mylopoulos, O. Nierstrasz, M. Pezzè, C. Prehofer, W. Schäfer, R. Schlichting, D. B. Smith, J. P. Sousa, L. Tahvildari, K. Wong, and J. Wuttke. *Software Engineering for Self-Adaptive Systems: A Second Research Roadmap*, pages 1–32. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[2] T. M. King, D. Babich, J. Alava, P. J. Clarke, and R. Stevens. Towards self-testing in autonomic computing systems. In *Eighth International Symposium on Autonomous Decentralized Systems (ISADS'07)*, pages 51–58, March 2007.