Quality Assurance for Autonomous Systems – A Review of Model-Based Methods

Ina Schieferdecker
Technical University Berlin/Fraunhofer FOKUS
ina@cs.tu-berlin.de

Abstract

A major field in research and development are open, distributed, communication-intense systems, which are currently characterized by increasing heterogeneity in structure and technologies and by growing complexity and behavior dynamics. Autonomous systems are being investigated in order to return to manageable and stable systems whose operation and maintenance is affordable. By resolving the obstacles of current systems via enlarged system autonomy, the quality assurance of such dynamic, reconfigurable, adjustable systems becomes problematic. This keynote reviews requirements, methods and approaches for model-based quality assurance of autonomous systems.

1. Overview

Autonomous systems are an emerging approach to the design of open, distributed, communication-intense systems that maintain continuity in operation in a continuously changing environment by context and situation dependent adaptations. Autonomous systems provide build-in features like scalability, robustness, security, quality-of-service, etc.

Autonomous systems are being developed for the provisioning of highly flexible and self-adaptable services which cannot be achieved by todays engineering principles. Major software and system vendors such as IBM (Autonomic Computing [1]), HP (Adaptive Infrastructure [2]), Sun (N1 [3]), CISCO (Adaptive Network Care [4]), Microsoft (Dynamic Systems Initiative [5]), and Hitachi (Harmonious Computing [6]) are concluding that the only viable long-term solution is to create systems that manage themselves.

Such systems change continuously their configuration and behaviour. In addition, the system environment is under permanent change: autonomic systems react differently depending on the location, the system invocation time, the usage preferences of system service usages and alike. Therefore, quality assurance of autonomous systems is particularly complicated. Pre-deployment tests are not sufficient – instead, a continuous monitoring, auditing and supervision of autonomous systems have to be used.

2. Autonomous System Paradigms

Autonomic system engineering is inspired by various engineering and science fields dealing with complex systems and structures. Basic paradigms encompass:

- selfware for implementing self-* properties
- sensing, context- and self-awareness as enablers for autonomicity
- functional composition for system dynamics and evolvability
- autonomic governance for system management

Self-* properties relate to:

- *Self-configuration*. A system is able to bootstrap and to be fully reactive to the needs and changes of the environment. The contextual analysis is fully distributed.
- *Self-optimisation.* A (sub)system and its service capabilities are optimised in an automated and distributed manner using localized information only.
- *Self-learning.* A (sub)system is able to learn from current or past experiences in order to properly react or to act proactively.

Eighth International Symposium on Autonomous Decentralized Systems (ISADS'07)
0-7695-2804-X/07 $20.00 © 2007 IEEE

- *Self-protection*. A (sub)system knows/learns about thread, intrusion and breakdown structures and initiates automatically maintenance tasks for system security.
- *Self-healing*. A (sub)system is able to detect errors or situations that may later manifest as errors, and to automatically initiate corrective means.
- *Self-awareness.* A (sub)system is able to reflect (sub)system and context properties – meta-information on the (sub)system – via an explicit, possibly built-in model.

Selfware is a (set of) components or middleware/OS elements providing autonomous system functionalities and controls needed to realize these self- properties. Hereby, sensing is basic to the self-* properties and includes monitoring, probing, measuring, and collecting system and context information. For example, an external or built-in supervisory component may control the system via a feedback loop or a decision mechanism based on context and/or system information or on the (perhaps precomputed) model of the system and its environment, which is parameterized and updated by the sensed information.

Autonomic systems leverage the strict binding of system delivering functions and services and replace it by on-demand system composition. This is characterised by the presence of a constant (invariant) task, which can be performed by variable structures and algorithms that finally produce a constant (invariant) result. This on-demand composition is not only characterised by the dynamic topology of the system structure, but also by dynamic (sub)system configurations together with potential mobility of the (sub)systems.

Finally, autonomic systems suggest a definition of behaviour in relation to policies: policy is a rule defining choices in the behaviour of a system. In this view policy and behaviour are two separate concerns that can be engineered separately and independently. A policy is basically a rule '[ON *event*] IF *condition* THEN *action'*. It is easy to define context aware rules, i.e. ones where conditions will represent contexts that are desired to be taken into account. Thus without altering the system functionality (i.e. without reboots or even suspending (sub)system operations) new policies can change (sub)system behaviours triggered by changes within the (sub)system contexts.

3. Quality Assurance

System quality encompasses functional and non-functional aspects: the correct system/service execution combined with responsiveness, robustness, availability, scalability, performance, etc. considerations. In addition, the internal quality (the static quality of the system's constituting artefacts), the external quality (the dynamic quality of the system's constituting artefacts in interaction with the system environment) and the quality in use (the quality perceived by an end user who uses a system in a specific context) can be differentiated.

Traditionally, system quality is engineered constructively throughout the system design and development process and assured analytically by review, verification, validation and testing methods. Already for static systems, analytical quality assurance methods fail in providing 100% correctness statements for real systems (this is not only true for sample-based reviews, validation and testing methods, but also true for verification methods, which do not scale for real system and hence do selected, but not exhaustive verifications only).

This problem exponentiates for dynamic systems. Hence, although the application of pre-deployment analytical methods are a prerequisite for system quality, the quality assurance for autonomous systems has to be extended to the system operation and maintenance phase.

Various approaches have been developed already:

- Online validation/tests [7]: active tests are performed to analyse the system behaviour in its current context.
- Passive tests [11]: sampled system traces are offline analysed for system anomalies and system errors.
- Built-in tests [8]: (sub)system components are equipped with tests, that are able to evaluate actively the correctness of the component environment with respect to the component-environment interaction. They also allow the environment to evaluate the component correctness.
- Auditing [9]: system policies are passively monitored to detect policy conflicts (activated/enabled policies contradict each others) or policy violations (activated/enabled policies are not obeyed)

- Supervision [10]: the system is continuously sensed in order to infer if faulty or pathological system behaviour is exhibited. Smart monitors provide problem-oriented, filtered and pre-evaluated system information, the supervisor deduces problems and reactions (potentially, by use of additional active tests), and actuators perform finally problem-specific corrective measures.

These approaches for the quality assurance of autonomic systems will be reviewed and open issues identified. A special attention is given to model-based techniques for active, passive or built-in tests of autonomous systems. Established techniques including TTCN-3 (Testing and Test Control Notation [12]) and U2TP (UML 2.0 Testing Profile [13]) are analysed with respect to the application to autonomous systems.

4. References

[1] IBM Autonomic Computing: http://www.research.ibm.com/autonomic/research/

[2] HP Adaptive Infrastructure: http://h71028.www7.hp.com/enterprise/cache/342611-0-0-0-121.html

[3] Sun N1: http://www.sun.com/software/n1

[4] CISCO Adaptive Network Care: http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/anc/index.htm

[5] Microsoft Dynamic Systems Initiative: http://www.microsoft.com/windowsserversystem/dsi

[6] Hitachi Harmonious Computing: http://www.hitachi.co.jp/Prod/comp/soft1/global/vision/harmonious.html

[7] P. H. Deussen, G. Din, I. Schieferdecker: A TTCN-3 Based Online Test and Validation Platform for Internet Services, Sixth International Symposium on Autonomous Decentralized Systems - Advanced Distributed Transportation Systems, ISADS 2003, Pisa, Italy, IEEE Press, April 2003.

[8] H.-G. Gross, I. Schieferdecker, G. Din: Specification and Implementation of Built-in Contract Tests, in Testing COTS Components and COTS-based Systems, Springer 2004.

[9] R. Chaparadza, M. Peter, I. Schieferdecker: FANSA - a Framework for Automated Network Security Auditing. In Proceedings of the International Conference on Late Advances in Networks - ICLAN2006, Paris, France, Dec. 2006.

[10] L. Baresi, M. Baumgarten, M. Mulvenna, C. Nugent, K. Curran, P.H. Deussen: Towards Pervasive Supervision for Autonomic Systems. In Proceedings of the IEEE Workshop on Distributed intelligent Systems: Collective intelligence and Its Applications (Dis'06) - IEEE Computer Society, Washington, DC.

[11] B.T. Ladani, B. Alcalde and A. Cavalli, Passive Testing - A Constrained Invariant Checking Approach, In Proceedings of TestCom 2005, May-June, Montréal, Canada.

[12] J. Grabowski, D. Hogrefe, G. Rethy, I. Schieferdecker, A. Wiles, C. Willcock: An Introduction into the Testing and Test Control Notation (TTCN-3). - Computer Networks Journal, Vol.42, Issue 3, 2003.

[13] P. Baker, Z. R. Dai, J. Grabowski, Ø. Haugen, S. Lucio, E. Samuelsson, I. Schieferdecker, and C. Williams: The UML 2.0 Testing Profile, Conquest 2004, ASQF Press, September 2004, Nuremberg, Germany.

Acknowledgement

This overview paper has been developed with the help of ACCO (Autonomic Communication Coordination Group) at FOKUS, Berlin lead by Michail Smirnov, Tanja Zseby, Peter Deussen and Stefan Arbanowski.