Toward Adaptable Software Architecture for Dependability: ASAP

Thomas Robert and Thomas Pareaud
Université de Toulouse
LAAS-CNRS
7 avenue du colonel Roche,
31400 Toulouse
France
{trobert, tpareaud}@laas.fr

Inga Zutautaite-Seputiene
Vytautas Magnus University
Mathematics and Statistics Department
8 Viliekos str.,
LT-44404 Kaunas
Lithuania
ingaz@isag.lei.lt

Vladimir Stankovic
City University,
Centre for Software Reliability,
London EC1V 0HB,
United Kingdom
v.stankovic@city.ac.uk

Shen Lin
Lancaster University,
Computing Department
Lancaster LA1 4WA,
United Kindgom
s.lin6@lancaster.ac.uk

Abstract

This short paper describes the ASAP mini-project (Assessment-based Adaptable Software Architecture for Dependability) that was launched within the Network of Excellence ReSIST[1]. The main objective of the work is to address the adaptation of the fault tolerance software in a system according to on-line assessment-based triggers. On-line assessment and adaptive fault-tolerance have been identified in ReSIST as major research gaps for the future. The ambition of the mini-project is to gather recent work on fault tolerance software adaptation using open-component software based engineering techniques and advanced knowledge on on-line assessment using Bayesian inference. Some small case studies are planned to perform and early validation of the ideas and proposed approach.

Keywords
On-line assessment, adaptive fault tolerance, Bayesian inference, reflective computing.

1. Introduction

The evolution of systems at runtime is currently to be addressed in many application domains, those being critical or not. A solution to cope with this requirement for more flexible systems is to adapt the system design and service at run-time. The on-line adaptation may be required for many reasons like changes in the environment, components upgrades, system configuration, resource consumption, components transient failure but also new fault assumptions. These are some of the possible triggers for adaptation of the system. This kind of problem is concern safety critical embedded systems with limited resources and no means to repair, ubiquitous computing or mobile systems, and also large-scale distributed systems.

The solution must tackle two kinds of problems:

- The assessment of operational conditions and detection of a change to trigger the adaptation
- The adaptation of the target system, including in particular its mechanisms of achieving high dependability (e.g. fault tolerance).

The objective of this work is to propose an architecture, and methods to make adaptive resilient systems, in particular regarding fault tolerance. To reach this aim, we introduce several assessment techniques and a reflective component model as base modules to build up such an architecture. The reflective model is drawn from a complete open-component engineering framework.

This work is the continuation of the analysis of research challenges performed within the framework of the network of excellence ReSIST, in [3]. Among them, **Adaptation and self-organisation** concerns resilience of an evolving system that is highly affected by its ability to adapt to new requirements of the

[1] Network of Excellence ReSIST (*Resilience for Survivability in IST*, http://www.resist-noe.org)

environment. The problem is thus to dynamically reorganize the system, including its resilience mechanisms, according to new operational conditions.

In the context of fault tolerance adaptation, the two problems identified above have been spelled out more precisely as:

- An on-line assessment engine that deals explicitly with the assessment at runtime of the system resilience. Thus, the on-line (run-time) assessment is concerned with system operating in highly dynamic and evolving environment (e.g. network congestion, new/changing service customers, effects of exploits of faults / vulnerabilities). Its goal is assessment of the implications of the changing environment on system resilience ranging from simply signaling a sub-standard operation to triggering a system's adaptation.
- A design for adaptation applicable to fault tolerance –identified as an important research gap in [3]. It has been pointed out that the adaptation capabilities of a system depend on the early stages of its design, and the software development technology that was selected. In the case of fault tolerance, they identified the urging need for an environment that offers system architects the means to describe and design the adaptation of the fault tolerance [8].

This paper gives the scheme of an architecture to provide on-line adaptation of resilient systems. Such an architecture is built from different approaches and techniques to address the sub-tasks identified above.

2. Overall architecture

The system architecture we envisage will consist of several abstraction layers as illustrated in Figure 1 that gives a coarse view of the system. The lower layer corresponds to the system services provided to the user and defined in the functional specification documents. The fault tolerance layer corresponds to the set (pool) of possible mechanisms required to fulfill non-functional specifications (give examples?). The mechanisms may require different degrees of redundancy (including diverse redundancy) and offer different degrees of dependability assurance. The adaptation layer corresponds to the various scenarios enabling the fault tolerance strategies to be changed on-line. Last, the assessment layer corresponds to the set of mechanisms responsible for estimating the current situation and deciding which of the available fault-tolerant mechanisms is the most adequate for it. If the currently deployed mechanism is sub-optimal, then a change will be triggered. This change will lead to modifications of the active fault-tolerance mechanism and, if necessary, modifications of the base system. In practical terms, these layers correspond to components such as Base System (BSys), Fault Tolerance mechanisms (FTM), Adaptation Engine (AdE), Assessment Engine (AsE) that uses Failure Detectors (FDs), Software Sensors (SWS) and Software Actuators (SWA).

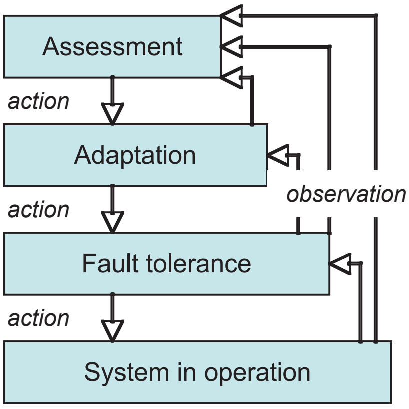

Figure 1. Abstraction layers for on-line adaptation

Figure 1 depicts in fact the proposed reflective architecture, each abstraction level corresponding to a metalevel. In reflective terms, observation and control facilities provided by open component technologies enable adaptation to be performed on each software level configuration on-line.

The **Base System** is the application software (and its supporting executive layers, operating system and middleware) without any fault-tolerant mechanisms, e.g. embedded software for automotive application, avionics, etc. Such subsystem consists of several components, with their respective APIs, which allow for interaction[2].

The **Fault Tolerance** software is defined as a set of interacting components that provides fault tolerance according to some requirement and resources availability. Their organization must be flexible enough to be manipulated at runtime. Novel software engineering techniques such as Fractal [1] or OpenCOM [2] can be used to reach this aim.

The **Adaptation** corresponds to a set of programs that are able to change the configuration of the fault tolerance software in order to modify on-line a given fault tolerance strategy. Because the open component technology enables individual components of a software layer to be manipulated at runtime (dynamic

[2] In case diverse components are used to achieve tolerance against software faults, the diverse components are still seen as part of the Base System, since they are assumed to be available (e.g. off-the-shelf) and developed by third parties in accordance with the 'design for adaptation' principles.

bindings), their objective is to update a given strategy while maintaining consistency.

The **Assessment** is a process of measuring the non-functional characteristics of the system, using a model of the system and the evidence available before and during operation. The Assessment in this particular framework is achieved by an Assessment engine, a piece of software, which allows for all necessary elements of the assessment to be represented: the system model, the pre-deployment knowledge about the non-functional property of interest, and the mechanism of updating the knowledge about the attribute of interest when new evidence becomes available. For the sake of concreteness in this report we limit ourselves to the following examples of assessment:

- Computation of dependability measures of interest associated with the particular deployment of BSys, AdE and FDs , e.g. the probability of system failure on demand, using Bayesian inference as described in a series of papers [4], [5]. Such a service ensures an up to date estimation of the dependability level of the whole system.

- Detection of the changes in the operational use of the system, i.e. of its operational environment, e.g. as a result of changes in the pattern the software is being used, which may invalidate the dependability measures computed in the past under different operational conditions.

- Detection by anticipation of real-time failure due to changes in the runtime conditions (i.e. scheduling errors) that lead to abnormal behaviours or erroneous states [6]; the approach consists of analyzing correct continuations using the automata time abstraction [7].

Finally, **SoftWare Sensors** will monitor the work of the deployed components and deliver input data to the specific Assessment engine deployed to keep the assessment up to date. The access to SWS will be via a defined API, in fact a meta-interface providing inputs to the adaptation and assessment engines.

Similarly, the adaptation will need means to act upon the system in operation through **SoftWare Actuators**.

3. Conclusion

Clearly, the on-line adaptation may be required for multiple reasons like changes in the environment, components upgrades, system re-configuration, resource usage, components transient failures as well as new fault assumptions. These are some of the possible triggers for adaptation. These situations are quite generic and apply to a wide range of systems, from resource-constrained safety critical embedded systems that cannot be repaired on-line, ubiquitous computing and mobile systems, to large-scale distributed infrastructure.

We argue that any solution to these challenges needs to address the following two problems:

1) How to properly assess operational conditions and trigger adaptation when required

2) Once adaptation has been decided, how to adapt a target system, and in particular how to adapt its dependability mechanisms.

In practice, the system in operation is composed of services implementing the functional specifications, those services being run on top of a middleware composed of an *assessment engine*, an *adaptation engine* and *fault tolerance mechanisms*.

The assessment engine obtains inputs from the system and its environment, and uses them to compute adaptation trigger. These triggers are passed on to the adaptation engine that is then responsible for dynamically modifying the software configuration (in particular the fault tolerance strategies) according to the needs that have been identified.

4. Acknowledgments

The authors want to give special thanks to Jean-Charles Fabre (*LAAS-CNRS, Toulouse*) and Peter Popov (*City University, London*) for their help in the elaboration of this mini project. They both played an important role in the organization of the project as well as in the contribution to the project content. We also thank François Taïani (*Lancaster University)* for his participation to this project.

5. References

[1] Bruneton, E., et al., *The Fractal Component Model and Its Support in Java*. Software Practice and Experience, 2006. 36 (11-12)(Experiences with Auto-adaptive and Reconfigurable Systems): p. 29.

[2] Coulson, G., et al. *Towards a Component-based Middleware Architecture for Flexible and Reconfigurable Grid Computing*. in *Workshop on Emerging Technologies for Next generation Grid (ETNGRID-2004), 13th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*. 2004. Italy.

[3] M. Banatre (Editor), *From Resilience-Building to Resilience-Scaling Technologies: Directions — Chapter 1, Evolvability of Resilient Systems*, Report D13, RESIST NoE, N°026764, Resilience for Survivability in IST, September 2007, 130 pages.

[4] Littlewood, B., P. Popov, and L. Strigini. *Assessment of the Reliability of Fault-Tolerant Software: a Bayesian Approach*. in *19th International Conference on Computer Safety, Reliability and Security, SAFECOMP'2000*. 2000. Rotterdam, the Netherlands: Springer.

[5] Popov, P. *Reliability Assessment of Legacy Safety-Critical Systems Upgraded with Off-the-Shelf Components*. in *SAFECOMP'2002*. 2002. Catania, Italy: Springer-Verlag.

[6] Robert, T., M. Roy, and J.C. Fabre, *Real-time run-time verifiers: theory and practice*, LAAS Research Report 07276, 2007, 10p.

[7] Alur, R. and D.L. Dill, *A theory of timed automata*. Theoretical Computer Science, 1994. 126: p. 183--235.

[8] T. Pareaud, J-C. Fabre, M-O. Killijian, Towards adaptive fault tolerance in Embedded Systems, International Congress on Embedded Real-Time Software, Toulouse, France, Jan. 2008, 9p.