Some Challenges in Adaptive Fault Tolerant Computing

François Taïani, Jean-Charles Fabre

Some Challenges in Adaptive Fault Tolerant Computing*

François Taiani
Computing Dept.
Lancaster University
Infolab21
Lancaster, UK
f.taiani,@lancaster.ac.uk

Jean-Charles Fabre
LAAS-CNRS ;
Université de Toulouse
INP-INSA-UPS-ISAE
Toulouse, France
jean-charles.fabre@laas.fr

Contact author: François Taiani, Computing Dept., Lancaster University, Infolab2, Lancaster, UK

Keywords: adaptive systems, reflection, component based software engineering

1. Problem statement

As mission-critical computer-based systems grow in size, they must provide increasing levels of flexibility to address evolving requirements and cater for rapidly changing operational conditions. In this context, *dynamic adaptation* appears as a powerful enabler to allow these systems to change while maintaining their services, a key requirement in large long-running applications. Because they are mission-critical, these systems must also adapt to evolving threats, and be able to react to changes in service priorities, which naturally leads to the need for *adaptive fault tolerance*, a notion formulated a decade ago [1]. Unfortunately, and in spite of a number of pioneering works [2, 3, 4], adaptive fault tolerance remains a challenging and poorly understood area, with a number of technological roadblocks preventing wide industrial adoption.

One of the key challenges of adaptive fault tolerance arises from the additional coupling that adaptation introduces between the functional and non-functional parts of a system. Over the past decade, a number of approaches have been proposed to provide a clean *separation of concerns* between a system's functional implementation (base level) and its fault-tolerance mechanisms. Unfortunately, dynamic adaptation, both at the functional and non-functional level, introduces new interdependencies that cannot be handled by these approaches. A change in the system's functional architecture might for instance modify underlying assumptions about diversity, and hence require the fault-tolerance to evolve. Or a change of fault-tolerance mechanisms in a real-time system might consume more resources, and hence require a change at the functional level to insure the overall system still meets its deadline. These interdependencies influence the rest of the adaptation cycle, and must be captured by the online operational monitoring to trigger reconfigurations.

This coupling also impacts the traditional definitions of fundamental notions of fault tolerance, which might either no longer apply, or only partially capture reality. Especially, dynamic adaptation introduces new hazards that are not covered by conventional fault models: Mismatches between subsystems, bad interpretation of meta-descriptions, obsolescence of software modules may have at least as much impact as commonly currently considered faults.

Functional and non-functional adaptations are therefore linked, and this coupling must be understood and controlled to provide a principled and robust development approach for fully adaptive systems. This requires the development of appropriate programming abstractions, mechanisms, and architectural guidelines to support adaptive fault-tolerance across a wide range of areas in a controlled and repeatable manner.

2. Approach

We contend that a solution to the above challenges should focus on three key aspects: (i) *separation of concerns*, (ii) *programmability*, and (iii) *scope control*.

Separation of concerns We argue for a three-tiered separation, between (i) the *functional level*, (ii) *fault-tolerance*, and (iii) *adaptation* itself. Only so can adaptation become a first-class entity and be reasoned about in a well-defined manner. By recursion, this separation can be extended by considering the *adaptation* of the *adaptation software* itself. For instance, such a platform could support monitoring and triggering mechanisms that are deployable on the fly, or could allow inference mechanisms to adapt their accuracy and resource footprint according to evolving constraints.

Programmability To foster wider adoption, adaptive fault-tolerance should be supported by a set of

* This work has been partially supported by RESIST, *Resilience in IST*, the Network of Excellence n°026764, through an internal project called ASAP (*ASAP: AsseSsment-based AdaPtable Software Architecture for Dependability*), and by the FP7 IST Project DiVA (Dynamic Variability in complex, adaptive systems) n°215412.

well-structured and clearly organized high-level abstractions. We argue here for a declarative approach, that would allow developers and fault-tolerance experts to express the dependencies and requirements we have mentioned in an appropriate domain-specific language. This language should cover fault-tolerance assumptions and needs (fault-model, failure unit, level of confinements), as well as channels of interdependency between the functional and non-functional levels (timeliness constrains, shared resources, operational constrains). Besides simplicity and expressiveness, a key challenge of this approach resides in the mapping of this language to an underlying fault-tolerance middleware that supports the three levels of concerns discussed above.

Scope control proposed by Kiczales and Lamping [5], this refers to the ability to operate small changes with a small effort. We argue here for fine-grained adaptation units, to encourage reuse, and support resource-constrained environments (e.g. embedded processors in mass-products such as cars). A fine-grained approach will also result in a smoother adaptation process, as only the parts of a fault-tolerant mechanism that need to be adapted will be impacted. This is crucial for continuous long-running systems that can neither afford downtimes nor provide the resources for monolithic switchovers between two configurations.

3. Technologies, challenges and outlook

Dynamic software engineering, especially *component-based software development* (CBSE) and *reflective architectures* are ideal candidates the support the above three-fold approach. Reflection in particular is a core technological enabler, thanks to its ability to perform on-the-fly operational modifications. Meeting the challenges of adaptive fault-tolerance require however that we go beyond the current state of the art in this area by considering simultaneously how dynamic adaptation can be provided both at functional and fault tolerance levels, without endangering the subsequent integrity and consistency of the resulting system.

CBSE allows a decomposition into small components that can easily be assessed operationally [6]. This appears as a necessary first step to then progressively take into account distributed adaptation, using the notion of *component federation* to help organise coordination and support reasoning about distributed adaptive fault tolerance mechanisms.

To use these technologies, a number of key challenges will have to be met:

- *Design for adaptation*: fault tolerance mechanisms should be decomposed into fine-grained *Lego-like* components that can be assembled on-line to realize a given fault tolerance strategy;
- *Mastering distributed state*: run-time state will need to be captured and mapped to different software configuration in a consistent manner;
- *Synchronisation of modifications*: the component architecture should be modeled and monitored on-line to perform component updates without endangering consistency and dependability;
- *On-line assessment* of system configuration parameters and of the dependability of the whole software system, including both application and fault tolerance mechanisms, to trigger adaptation;
- *Development of a resilient adaptation process* able to guarantee dependability properties during the modification of the fault tolerance software.

On a longer term, the kind of adaptive fault-tolerance we advocate opens up the prospect of *proactive dependable systems,* i.e. systems that can tailor their strategies according to predicted changes in their environments and internal conditions (see for instance [7]). Although adaptive fault tolerance will obviously be a key building block of these systems, proactivity raises a number of much larger challenges in terms of modeling, evaluation, and reasoning, which will all need to be reconsidered in this new light.

4. References

[1] J. Goldberg, R. J. Stroud, “Adaptive Fault-Tolerant Systems and Reflective Architectures”, LNCS no. 1357, Proc. of the *Workshop on Object-Oriented Technology* (ECOOP'97), Jyväskylä, Finland, pp. 80-88, June 1997.

[2] Y. J. Ren, D. E. Bakken, T. Courtney, M. Cukier, D. A. Karr, P. Rubel, C. Sabnis, W.H. Sanders, R. E. Schantz, and M. Seri, "AQuA: An Adaptive Architecture that Provides Dependable Distributed Objects," *IEEE Trans. on Computer*, vol. 52, pp. 31-50, 2003.

[3] T. A. Dumitras, D. Srivastava, and P. Narasimhan, "Architecting and Implementing Versatile Dependability," in *Architecting Dependable Systems III*, WADS, pp. 212, 2005.

[4] T. Pareaud, J.-C. Fabre, and M.-O. Killijian. "Componentization of Fault Tolerance Software for Fine-Grain Adaptation," in Proc of the *14th Pacific Rim Inter. Symp. on Dependable Computing* (PRDC'08), Taipei, Taiwan, pp. 248-255, 2008.

[5] G. Kiczales and J. Lamping. “Operating Systems: Why Object-Oriented?”, Proc. of the *Third International IEEE Workshop on Object-Orientation in Operating Systems,* pp. 25-30, 1993.

[6] G. Coulson, G. Blair, P. Grace, F. Taïani, A. Joolia, K. Lee, J. Ueyama, T. Sivaharan. "A generic component model for building systems software" *ACM Trans. on Computer Systems* (TOCS) 26(1), pp. 1-42. 2008.

[7] A. Casimiro, P. Lollini, M. Dixit, A. Bondavalli and P. Verissimo. “A framework for dependable QoS adaptation in probabilistic environments”, Proc. of the 23rd ACM Symp. on Applied Computing, (SAC'08), Fortaleza, Ceara, Brazil, pp. 2192-2196, 2008.