Towards a System Architecture for Resilient Computing

Miruna Stoicescu, Jean-Charles Fabre, Matthieu Roy

Towards a System Architecture for Resilient Computing

Miruna STOICESCU* Jean-Charles FABRE Matthieu ROY
CNRS ; LAAS ; 7 avenue du colonel Roche, F-31077 Toulouse Cedex 4, France
Université de Toulouse ; UPS, INSA, INP, ISAE ; UT1, UTM, LAAS ; F-31077 Toulouse Cedex 4, France

***Abstract*—Nowadays, systems are not only becoming increasingly complex and heterogeneous but are also opened towards their environment by means of context-awareness. Furthermore, in the vast majority of cases, their specifications periodically evolve, leading to new versions. As resilient systems are expected to continuously provide trustworthy services, they must cope with changes coming from the environment or from the specifications and reconfigure in order to adapt to them. We propose a framework for designing and developing such systems.**

***Keywords*—reconfigurable, adaptive, pervasive, fault-tolerant**

I. PROBLEM STATEMENT

The evolution of systems is ineluctable and leads to the notion of *resilient computing*[3]. Among the evolution scenarios that can occur during the operational life of a fault-tolerant application, we focus on those that may have an impact on the fault tolerance mechanisms:

- one or several changes in the application assumptions which influence the choice of the fault tolerance mechanism,
- changes in the requirements in terms of fault tolerance,
- changes in the system configuration (e.g. number of processors, memory resources, network bandwidth).

The essence of a *resilient application* lies in the fact that when faced with any of these changes or even with a combination of them, it must adapt itself while ensuring the observance of the safety criteria.

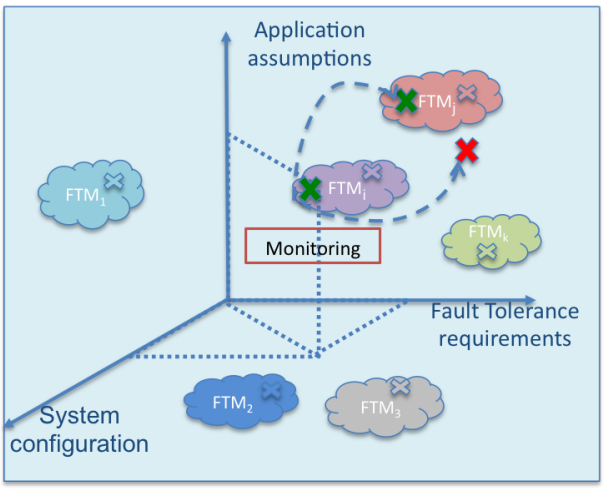

Fig. 1. Context-aware fault tolerance computing

This work has been supported by the french national agency (ANR) under contract ANR-BLAN-SIMI10-LS-100618-6-01.

∗ Corresponding author: miruna.stoicescu@laas.fr

We view the evolution of a system during its operational life as a trajectory in a space characterized by the parameters which can cause the aforementioned evolution. The three evolution scenarios can be aggregated into a frame of reference for this space, the three axes being: the application assumptions (e.g., whether it is deterministic or not), the fault tolerance requirements (e.g., the fault model) and the current system configuration (e.g., the resources it is using). We claim that a system's evolution can be represented in this vector space as shown in Figure 1. The combination of these three types of information indicates the most convenient fault tolerance mechanism to use, if any exists.

Given this frame of reference, we consider that each fault tolerance mechanism covers a certain region of space as shown in Figure 1 . Obviously, the union of all our fault tolerance mechanisms does not cover the entire space. A monitor keeps track of values in terms of the three axes and allows placing the state of the system for a given configuration in a certain region of this space. A change observed by the monitor triggers a transition towards a new region which might be covered by another fault tolerance mechanism or might be empty if there is currently no solution. In this case, we can still give some indications as to what must be adjusted in order to reach a region for which a fault tolerance mechanism exists.

Once our view of the problem stated, a certain number of issues arise, such as: how do we associate fault tolerance mechanisms with the combination of the three measures? how do we describe these fault tolerance mechanisms in a way which allows us to change them dynamically? how do we actually perform the transition from one mechanism to another? The following section presents our view on these matters.

II. OUR APPROACH

Separation of concerns is a generally accepted idea for introducing fault tolerance mechanisms in an application in a flexible way which allows subsequent modification and reuse. According to this principle, software architectures consist of two layers where the base provides the required functionalities and the top contains the fault tolerance mechanism(s). As we target the adaptation of fault tolerance mechanisms, we must manage the dynamics of the top layer, which can have two causes:

- The application layer remains unchanged but the fault tolerance mechanism must be modified either because the fault model changes or because an event in the environment, such as resource availability, makes it unsuitable.
- Changes in the top layer are indirectly triggered by modifications in the application layer which make the

fault tolerance mechanism unsuitable. In this case both layers execute a transition to a new state.

In order to achieve the adaptation of the fault tolerance mechanisms in both scenarios, we build on the representation from the previous section and refine it in three steps.

A. Description of the frame of reference

The three parameters labeling our axes are actually multidimensional vectors but for the sake of visibility we have chosen an elementary representation.

- The fault tolerance requirements are part of the non-functional specifications of an application. Our main focus is on the fault model, i.e., the types of faults which must be tolerated by the application. We base our fault model classification on known types [1], e.g., physical faults, design faults.
- The application assumptions regroup the characteristics of an application which have an impact on the choice of the fault tolerance mechanism but are not the same as the functional specifications of the application. These characteristics include: whether the application is stateless or stateful (i.e., if in case of failure we must rebuild its state for it to keep running), whether its state is accessible or not and whether the application is deterministic or not.
- The system configuration contains information such as the number of processors available, the available bandwidth, the memory resources.

B. Fault Tolerance Mechanisms

In order to place the mechanisms in the previously described frame of reference, we must first be able to classify them using the given criteria. In our approach, there are three steps in the selection process for finding the most adequate fault tolerance mechanism for an application: first, the fault model is identified, then the application assumptions and finally the current system configuration. The same process will be applied for classifying them.

C. System evolution

The state of a fault-tolerant application represents a point in the space given by our frame of reference. The application being fault-tolerant, this point must be in a region covered by a certain mechanism. A change (in terms of one of the three axes) is equivalent to a new state of the application, therefore a new point. As our purpose is to guarantee resilience when facing change, the application must always be "accompanied" by an adequate fault tolerance mechanism on its evolutionary trajectory. Therefore, we must either provide a fault tolerance mechanism encompassing a region which contains the new point or, should we fail to do so, "guide" the application towards a new region where a mechanism can be provided. We must design and build our fault tolerance mechanisms in view of such transitions and adaptations. The "distance" between two mechanisms as represented in our frame of reference should be equivalent to the difference between their implementations. If the new mechanism is close to the old one, we should be able to generate it through minor adjustments. The transition between distant mechanisms will most likely demand more complex modifications or even complete replacement.

The planned development process for achieving the smooth and safe transition between fault tolerance mechanisms consists of the following elements:

- a complete classification of the fault tolerance mechanisms we intend to use
- a method for describing and storing configurations corresponding to the fault tolerance mechanisms
- a set of tools allowing us to develop applications which are easily reconfigurable at runtime
- one or several algorithms for performing the transition between mechanisms

III. Conclusion and perspectives

Our work lies at the intersection of three research domains, namely Fault Tolerant computing, Autonomic Computing [2] and Ubiquitous Systems. The fault tolerance mechanisms we are considering are well-established solutions for certain use-cases/scenarios. This has led us to the concept of design patterns for fault tolerance, by drawing a parallel with the design patterns from software engineering. Each fault tolerance design pattern has an associated architecture. In order to operate a transition between fault tolerance mechanisms, we must be able to manipulate the configuration through its description file. A design principle that is commonly accepted in the area of reconfigurable systems is the use of component-based technologies (CBSE) for developing the management framework. We will use component-based middleware as a basic layer for our adaptive architecture. Finally, a decision needs to be made concerning the granularity of the reconfiguration. We can imagine two approaches for replacing a fault tolerance mechanism with another one:

- an atomic approach, where the old mechanism is completely replaced by the new one
- a differential approach, where we operate at a finer level by adding/removing the components corresponding to the difference between the two mechanisms

The transitions between mechanisms will most likely be described through state-machines. The algorithms behind the transitions could lead to design patterns for system evolution.

References

[1] A. Avizienis, J-C. Laprie, B. Randell, and C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. Dependable Secur. Comput.*, 1:11–33, January 2004.

[2] J-O. Kephart and D-M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[3] J-C. Laprie. From dependability to resilience. In *International Conference on Dependable Systems and Networks (DSN 2008), Anchorage, AK, USA, pp. G8-G9*, volume 8, 2008.

Miruna Stoicescu is a 1st year PhD student at LAAS–CNRS. Her research interests include reconfigurable systems and ubiquitous computing. She received the Research Master's degree in Computer Science and Telecommunications, Multimedia stream, from Institut National Polytechnique de Toulouse, France in 2010, the Certified Engineer degree from Ecole Nationale Supérieure d'Electronique, d'Electrotechnique, d'Informatique, de Télécommunications et d'Hydraulique de Toulouse, France in 2010 and the Certified Engineer degree from the Polytechnic University of Bucharest, Romania in 2009.