Model-Driven Engineering of Fault Tolerance in Enterprise Distributed Real-time and Embedded Systems

Sumant Tambe*, Aniruddha Gokhale, Jaiganesh Balasubramanian, Krishnakumar Balasubramanian, Douglas C. Schmidt
Vanderbilt University, Nashville, TN, USA
Contact : *sutambe@dre.vanderbilt.edu

Large scale enterprise distributed real-time and embedded (DRE) systems, such as those found in aerospace, defense, telecommunications and healthcare domains, have stringent, simultaneous QoS requirements in terms of performance, availability and reliability. In spite of rigorous software design, test and certification processes, however, it is difficult to rule out all latent defects in the system. In particular, it is difficult, if not impossible, to deal with unplanned failures such as hardware, software or network link failures. Redundancy by means of replication of critical system components in fault-tolerant systems is a popular and successful solution to this problem. The spectrum of fault-tolerant systems ranges from totally application controlled fault-tolerance (FT) at one end to completely application-agnostic or infrastructure controlled fault tolerance to the other.

Building fault-tolerant enterprise DRE systems is hard due to the effort required to address these concerns in addition to actual system development work. For example, application-controlled fault-tolerance incurs additional development work for application developers. Conversely, infrastructure-controlled fault-tolerance alone can be insufficient to cater to the dependability needs of different kinds of applications. Providing *ad hoc* modifications or extensions to infrastructure-provided solutions engineered in this way could potentially impact its reusability or affect project schedules when provided by applications alone. Moreover, the complexity of provisioning fault-tolerance QoS requirements can distract application developers from their primary job of developing business-logic.

Model-Driven Engineering (MDE) is a promising approach to address these challenges because it raises the abstraction of system design to a level higher than is possible with third-generation programming languages alone. Modeling different system components as fault-tolerant and auto-generating supporting run-time components helps to separate the dependability concerns from other system development concerns. Moreover, information from models can be used to auto-generate near-optimal deployment plans, which helps decrease the probability of simultaneous failures in deployed enterprise DRE systems.

Our fault-tolerance modeling environment extends the Platform Independent Component Modeling Language (PICML), which is a domain specific modeling language that deals with the different design and deployment phases of component based systems including specification, assembly and packaging, with new FT elements. This presentation will focus on the following three contributions of applying MDE to design fault tolerant enterprise DRE systems: (a) Demonstrating how modeling FT elements at different granularities can help in separation of concerns, and (b) how generative technologies can be used to rapidly and reliably enhance the system with FT capabilities, and (c) how system deployment concerns can be alleviated applying placement decision algorithms on the models.

1. **Modeling fault-tolerance aspects of enterprise DRE systems.** The newly introduced elements allow control over the granularity of protected system components such as software components, component assemblies and application workflows. It allows FT elements to be

modeled orthogonally to the system components and therefore achieves separation of FT concerns from the main system composition and functionality development concerns. The following modeling elements are supported:

a. **Fail-over units (FOUs)**, which enable control over the granularity of protected system components, such as software components, component assemblies, and operational strings. These modeling abstractions can capture the fail-over granularities of system entities, the degree of replication for FOUs and requirements for liveness monitoring of FOUs.

b. **Replication groups (RGs)**, which allow capturing the replication requirements of software components within a FOU. These models will specify replication strategies, such as active, passive or other variants, and state synchronization policies for components.

c. **Shared Risk Groups (SRGs)**, which define groupings of application components that share the risk of simultaneous failure by virtue of failure of resources they share, such as processes, nodes, racks or even data centers. Risk factors will be determined by assigning metrics, such as co-failure probabilities for the components in a risk group or the geographic distance between the placement of replicas that may determine availability.

2. **Generative capabilities for provisioning FT capabilities.** The model interpreters and generative tools use the dependability requirements captured by the modeling tools for synthesizing metadata to describe replica deployment plans and FT provisioning mechanisms in component middleware systems, such as Lightweight CCM as follows:

   a. The placement model interpreter provides a strategizable framework to use different constraint-based algorithms to come up with a component and replica placement plan to minimize the co-failure probability of the system as a whole.

   b. The deployment plan model interpreter translates the models into supporting run-time components to realize ready-to-deploy, robust, fault-tolerant component-based enterprise DRE systems. This includes placing the status monitoring and fault recovery components without the need for the application developer having to model these explicitly.