Foreword

As software systems become more and more ubiquitous, the issues of dependability become more and more critical. And, as solutions to these issues must be planned in from the beginning of a system - grafting them on after the system has been build is very difficult - it is appropriate that these issues be addressed at the architectural level.

However, how they are to be addressed at this level is a critical question. Are the solutions to these dependability issues to be considered explicitly in the architecture, or are they to be considered implicitly by being buried in the design and implementation instead of the architecture? If they are to be considered explicitly, are they integral to the entire architecture or are they componentized within that architecture?

An example analogy for whether to make it explicit or implicit can be found in the issue of distribution. Do you want to make distribution an architecturally explicit issue or do you want the architecture to be independent of placement and topological issues. For example, in the quest for a generic architecture to cover a product line that ranged from centralized to a variety of distributed systems, it was decided to make the architecture distribution free (IWSAPF3, LNCS 1951, Springer-Verlag 2000). The solution incorporated an ORB (Object Request Broker) like architectural entity in the system and buried the details of placement and topology into that component, thereby removing distribution as an explicit architectural issue. Similarly we might treat dependability issues in the same way burying them in components that worry about how to provide them rather than making them explicitly part of the architecture.

If we decide to make them explicit, then there is still the issue of whether they are integral across the entire architecture or whether they are componentized within that architecture. If integral, then one way to ensure that all the components in an architecture conform appropriately would be to define a dependability-property-specific architectural style (Perry/Wolf, SEN 17:4, Oct 1992) that all components in the architecture must conform to. An example of this use of an architectural style for fault handling was defined and used in the above-mentioned product line architecture.

Far more interesting architecturally is an attempt to find solutions to dependability problems that are compositional, or additive, and that can be viewed as independent components (Brandozzi/Perry, WADS2002). The logical distinction between components and connectors is very suggestive in this direction. Connectors can be much richer than just means of communications (their initial and basic use). Indeed, the entire Coordination conference series is built around the premise of separating coordination from computation - in effect using connectors among computations to provide their coordination. We can approach dependability problems in the same way or by pushing the envelope of connectors even further. Not only can connectors be coordinative, the can be mediative as well (Keynote, Coordination97). Indeed, mediation is often precisely what we want to do in the context of making our systems more dependable.

In this book we have a variety of approaches in considering the problems of dependability architecturally. Some push in these new and interesting directions, others continue in more traditional modes. It is my belief that we will make our most significant progress exploiting these new possibilities in looking for compositional means for achieving dependability.

Dewayne E Perry
Motorola Regents Chair in Software Engineering
The University of Texas at Austin