Introduction to the Minitrack on Software Architecture

stsar00.pdf

Rick Kazman
(kazman@sei.cmu.edu)
Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, 15213

Software architecture has become an increasingly important topic in the research and practice of software engineering over the past decade. It is concerned with the high-level structures of software systems (typically represented as “components” and “connectors”), and the properties of these structures. So why devote a minitrack to this topic? There are a number of reasons. Software architecture is the keystone in the successful development of complex software-intensive systems. It is the first and best opportunity to ensure that the system meets its quality goals; it is the basis for organizing the development effort; and it is the manifestation of the earliest design decisions, those which are most fundamental and hence hardest to change. A software architecture can be the basis for a software product line, delivering on large-scale reuse, in contrast to the largely failed approaches to component-based reuse that have been seen over the past few years. The ramifications of a product's architecture last its entire lifetime. The right software architecture is a prerequisite for smooth development; the wrong one can precipitate disaster.

In this minitrack, we will see five papers that address a representative selection of the topics that are of importance to architects today. The first paper, “Making Behaviour a Concrete Architectural Concept”, deals with overlaying a behavioural notation—Use Case Maps—onto architectural concepts, and demonstrates the advantages that accrue to this superimposition: a better understanding of the system’s performance and robustness. In the second paper, “Rose/Architect: a tool to visualize architecture”, we see an attempt to add architectural notions to a design tool (Rational’s Rose) and design notation (UML). Rose/Architect gives a user the ability to organize modeling elements, such as classes, objects, and packages into planes. The tool can visualize any plane or set of planes, and can even reconstruct missing relationships between elements using user-definable heuristics. The third paper, “The Ragnarok Architectural Software Configuration Management Model”, describes a software configuration management model and prototype tool that allows a developer to manage a complex software project—its versions and configurations—as an architecture, rather than simply as an undifferentiated set of files. It allows a developer to see the difference between versions of an architecture, and to trace how an architecture evolves. The fourth paper, “Reactive Software Architectures”, describes an agent-base approach to making architectures dynamically reconfigurable, rather than the existing case where architectures are typically only reconfigured at design time. The authors also describe a prototype tool that allows an architect to support reactive architectures. Finally, the fifth paper, entitled “Object-oriented Architecture Measures”, presents a way to measure an object-oriented architecture’s goodness, with respect to the abstractness and hence modifiability of the design.