The Architect

**Integrating Architecture Methods: The Case of the Rational Unified Process**

Rick Kazman and Robert L. Nord

In a previous column ("Rethinking the Software Life Cycle"), we took a look at the traditional software-development life cycle in the context of the architecture-centric methods that we have developed at the Carnegie Mellon® Software Engineering Institute (SEI) over the past 10 years. These methods include the Architecture Tradeoff Analysis Method® (ATAM®) [Clements 02], the SEI Quality Attribute Workshop (QAW) [Barbacci 03], the SEI Attribute-Driven Design (ADD) Method [Bass 03], the SEI Cost Benefit Analysis Method (CBAM) [Bass 03], and SEI Active Reviews for Intermediate Design (ARID) [Clements 02]. This column shows how these architecture-centric methods fit into the framework of the Rational Unified Process (RUP).

The SEI’s architecture-centric methods were developed at the same time that the RUP was being developed. The RUP is an object-oriented development framework. It provides guidelines, templates, and examples for all aspects and stages of a software-intensive system’s life cycle, although it treats software architecture obliquely.

The SEI’s architecture-centric methods have long demonstrated that they can illuminate important characteristics of architectures and the quality-attribute requirements that shape them. Until now, such considerations have been relegated to a separate “supplementary requirements” document in the RUP. Also, business drivers, long a key part of SEI methods, have just recently found a place in the RUP.

The SEI architecture-centric methods can provide explicit and detailed guidance on eliciting the architectural requirements, on designing the architecture, and on analyzing the resulting design.

- The architecture-centric methods place an emphasis on quality attributes rather than functionality.
- The architecture-centric methods help fill gaps in the RUP design process by providing specific advice on
  - the elicitation and documentation of quality-attribute requirements
  - which design operation will achieve a desired quality-attribute response
  - how to understand and predict the consequences of the design decisions in terms of risks, trade-offs, and ultimately return on investment
- The architecture-centric methods all use common concepts: quality attributes, architectural tactics, and a “views and beyond” approach to documentation that leads to increasingly efficient and synergistic use [Clements 03].

Table 1 shows where specific SEI architecture-centric methods can help to produce artifacts required in different RUP phases, or how the methods can enhance the activities of the RUP. More details are available in a forthcoming technical report [Kazman 04].
Through the process of the QAW, vague requirements would be refined into several quality-attribute scenarios. The ADD Method defines a software architecture by basing the design process on the quality-attribute requirements of the system. The ADD approach follows a recursive decomposition process where, at each stage in the decomposition, design decisions are made to satisfy a chosen set of high-priority quality scenarios.

It is clear that design decisions interact. For this reason, we need an organized method for understanding the interaction of the many decisions that are made in creating a complex system architecture. The ATAM provides software architects with a framework for understanding the technical tradeoffs and risks that they face when making architectural design decisions. In addition, the CBAM helps software architects consider the return on investment of any architectural decision and provides guidance on the economic tradeoffs involved. Finally, the ARID evaluates whether the design can be used by the software engineers who must work with it.

The benefit of including the SEI methods is to address quality attributes in an explicit, methodical way. Quality-attribute requirements drive the software architecture, and architecture-centric activities (with an explicit focus on quality attributes) drive the software system life cycle. Properly managed, the architecture-centric methods can be a low-cost addition to the RUP that will increase the quality of the systems and products developed.
References


About the Authors

Rick Kazman is a senior member of the technical staff at the SEI, where he is a technical lead in the Architecture Tradeoff Analysis Initiative. He is also an adjunct professor at the Universities of Waterloo and Toronto. His primary research interests within software engineering are software architecture, design tools, and software visualization. He is the author of more than 50 papers and co-author of several books, including a book recently published by Addison-Wesley titled *Software Architecture in Practice*. Kazman received a BA and MMath from the University of Waterloo, an MA from York University, and a PhD from Carnegie Mellon University.

Robert L. Nord is a senior member of the technical staff in the Product Line Systems Program at the Software Engineering Institute (SEI) where he works to develop and communicate effective methods and practices for software architecture. Prior to joining the SEI, he was a member of the software architecture program at Siemens, where he balanced research in software architecture with work in designing and evaluating large-scale systems. He earned a Ph.D. in Computer Science from Carnegie Mellon University. Dr. Nord lectures on architecture-centric approaches. He is co-author of *Applied Software Architecture* and *Documenting Software Architectures: Views and Beyond*.
The views expressed in this article are the author’s only and do not represent directly or imply any official position or view of the Software Engineering Institute or Carnegie Mellon University. This article is intended to stimulate further discussion about this topic.

The Software Engineering Institute (SEI) is a federally funded research and development center sponsored by the U.S. Department of Defense and operated by Carnegie Mellon University.