Part 4
Frameworks for Rationale-Based Software Engineering

The case in support of rationale is a compelling one—the ability to capture and encode the decision-makers' intent as part of a knowledge management strategy aimed at using this knowledge to assist with future decisions so that we can learn from the past, rather than repeating it (or repeat it only when past decisions were successful). The importance of rationale and its potential value has resulted in a significant amount of research over the past 30 years yet there still remain many obstacles towards its acceptance and use in practice. Still, advances in technology have resulted in new opportunities for integrating rationale into practice and the increasing awareness of the relationship between process and product quality suggest that the reluctance to invest up-front effort for later benefit may be lessening.

The challenge is to move rationale outside the laboratory and into practice. Studies have shown that it takes 15–20 years to mature a technology (Redwine and Riddle 1985). In order to successfully transition a technology into practice it is important to understand what both the *obstacles* and *benefits* of that technology are.

In order to build a Rationale Management System (RMS) to support RBSE, we need to identify where and how rationale can be used in software development (benefits) and capture these uses, along with concepts needed to compare rationale approaches and relate them to software engineering, in a Conceptual Framework (Chapter 16). We also need to develop an Architectural Framework (Chapter 17) that identifies issues (obstacles) that must be addressed by an RMS architecture in order to successfully support software engineering. Past work in rationale has indicated that it shows great promise in providing significant benefits to software development and we need to look ahead (Chapter 18) to determine how those benefits can best be disseminated into software development approaches, processes, and tools.