SED Lecture 10
Using Rationale to Document Design

Objectives
To be able to use issues, proposals, criteria, arguments, and resolutions to document the rationale for major software analysis and design decisions.

Key Points
• Rationale is the fundamental reason or logical basis for a decision
• We need to record formally the rationale for major decisions in design and analysis to make visible the reasons for decisions so that when changes are proposed, the effect of change on all parts of the system design can be evaluated
• Rationale is documented using issues, proposals, criteria, arguments, and resolutions

Core reading
Bruegge & Dutuoit, 12.2 an overview of rationale 12.4 Rationale activities 7.4.6 reviewing system design
Bruegge & Dutoit, Rationale exercise (see attached)

UWA CITS2220, Semester 1, 2010
Rationale Exercise

The following excerpt, from Bruegge and Dutoit, presents the rationale for using a relational database for a distributed data processing application.

Rewrite this prose argument as a structured rationale argument using the headings: issue, proposal, alternatives, arguments, criteria and resolutions.

One fundamental issue in database design was database engine realization. The initial non-functional requirements on the database subsystem insisted on the use of an object-oriented database for the underlying engine. Other possible options included using a relational database, a file system, or a combination of the other options. An object-oriented database has the advantages of being able to handle complex data relationships and is fully buzzword compliant. On the other hand, OO databases may be too sluggish for large volumes of data or high-frequency accesses. Furthermore, existing products do not integrate well with CORBA, because that protocol does not support specific programming language features such as Java associations. Using a relational database offers a more robust engine with higher performance characteristics and a large pool of experience and tools to draw on. Furthermore, the relational data model integrates nicely with CORBA. On the downside, this model does not easily support complex data relationships. The third option was proposed to handle specific types of data that are written once and read infrequently. This type of data (including sensor readings and control outputs) has few relationships with little complexity and must be archived for extended periods of time. Files offer an easy archival solution and can handle large amounts of data. Conversely, any code would need to be written from scratch, including serialization of access. We decided to use only a relational database, based on the requirement to use CORBA and in the light of the relative simplicity of the relationships between the system's persistent data.