Adding value to the Unified Process

Richard Mitchell
InferData Corporation, USA

Both the Unified Modeling Language and the Unified Process are quite loosely defined. Guidelines that constrain the language and the process in very practical ways can make the process much more effective. This tutorial proposes a coherent body of such guidelines.

The tutorial is intended for developers and managers who are considering adopting Rational's Unified Process for use with the Unified Modeling Language. The tutorial will assume an elementary knowledge of object-oriented concepts and their use in OO modeling.

The Unified Process (UP) from Rational Corporation uses the Unified Modeling Language (UML). The UP can be thought of as a layer above the UML.

The UML and the UP are both quite loosely defined, so that different projects working in different ways might all claim to be using the UP and UML.

This tutorial presents key elements from a layer we think of as coming between the modeling language and the process. This layer clarifies and constrains the UP and its use of the UML, and hence provides support for modeling and support for the process.

We have found in practice that such clarifications and constraints add value to the UP by building on its strengths and overcoming some of its weaknesses. The strengths of the UP include its emphasis on iterative working. Weaknesses include failing to tighten the underlying language, UML, and failing to distinguish clearly between development activities and development phases.

Issues the tutorial will concentrate on

- The UP is “use-case driven.” Generalize this to “behavior driven.” For example, events can be more helpful than use cases when modeling a domain with no business or system boundaries.
- The UP is “architecture-centric.” Distinguish the many kinds of architecture needed (technically-oriented, subject-based, logical, physical, etc.).
- Raise the status of domain, or business, models. A model of the key types in a domain can provide a single vocabulary for other models.
- Separate the content of a model from its time of construction. For example, making a design decision early in a project does not turn it into an analysis issue, as the UP implicitly suggests.
- Distinguish adding detail in order to increase precision from adding detail in order to support refinement. For example, avoid premature design of behavior by describing use cases with informal preconditions and postconditions rather than with sequences of actions.
- Individual models are projections of a single, underlying model, and can sometimes be combined to good effect. For example, you can get the benefits of a sequence diagram and a collaboration diagram by overlaying the diagrams.
- Define the UML carefully enough to support cross-checking between diagrams. For example, check that states are distinguishable on type models. (The tutorial will offer a list of more than 20 cross checks.)
- Work with example-level models as well as general-level models. For example, use snapshots to uncover subtleties of type models.
• Define maturity tests on models. For example, ascertain whether certain cross-checks have been performed, and determine whether key example-level models are supported by the general-level models.

This list does not define a new process. Rather, it reduces the number of interpretations of the Unified Process. The list is not comprehensive. For example, we could add support for project-management through time-boxing of development activities, or support for high levels of precision through the use of OCL. Or we could focus more on modeling support, through such topics as design patterns. Our particular selection focuses on process issues, and reflects what we have found in practice adds considerable value to many kinds of projects with only a small investment in learning.

By design, the elements in the selection work together to form a coherent whole. For example, it is easier to avoid premature design of behavior when you have an architectural model that shows where such design choices belong.

Richard Mitchell works as a senior consultant for InferData Corporation, a training and consultancy company specializing in object technology. He has published many contributions to the object-oriented field, particularly in the areas of design by contract and development methods. Richard has presented papers and tutorials at conferences in America, Asia, Australia, and Europe. His consulting work mostly involves supporting modeling efforts in Fortune 500 companies.

Richard chairs the program committee for TOOLS Europe, and serves as a member of the program committees for the OT and UML series of conferences.