Chapter VII

Developing Requirements Using Use Case Modeling and the Volere Template: Establishing a Baseline for Evolution

Paul Crowther, Sheffield Hallam University, UK

Abstract

A major contributor to the development of a quality final product is a complete, consistent, and detailed requirement specification (Pressman, 2000). No matter how good the specification and its translation into an initial system, it will evolve once it is released to users as the requirements and the environment change and the users develop. The aim of this chapter is to provide a method of establishing the baseline in terms of the requirements of a system from which evolution metrics can be effectively applied. UML (Rumbaugh, Jacobson, & Booch, 1999) provides a series of models that can be used to develop a specification which will provide the basis of the baseline system. This can then be used as a datum from which measurements can be made. One of the starting points for modeling is use case analysis. Other models can then be developed based on these initial models. One of the difficulties with this approach is...
that once the initial models have been agreed upon, they are not maintained as the later more detailed models evolve. The methods described in this chapter show how this can be achieved and measured.

**Introduction**

This chapter discusses the establishment of a baseline from which to measure the evolution of a software system. The work described is based on the development of a system designed to deliver a collaborative learning environment on personal Web-enabled mobile computing devices called MOBIlearn (2000). Further, the components which make up the system are being developed by a series of teams which are distributed throughout Europe.

A system in constant use evolves. This is because of changes in:

- Requirements
- Environment
- User development

However, the evolution needs to be both controlled and measured. The foundation of the initial instance of the system is the requirements on which it is based. Feedback from users and refinements as the environment is more completely understood will lead individual components to evolve and hence the overall system itself to evolve.

UML has the advantage that it can be used in conjunction with a variety of development methodologies while providing a readily understandable set of diagrams. These are based on a series of interconnected models that range from use cases used to develop requirements through collaboration diagrams used to determine how the use cases will be implemented on to logical models which will form the basis of the final software.

In this chapter, the primary emphasis will be on use cases and their role in establishing the base requirements of the system. These will be discussed in terms of their relationship to the Volere template which adds control and referencing. Finally, this will be tied into the UML component model and the use of XML in the resulting service-oriented architecture.