Chapter XV

Variability Expression within the Context of UML: Issues and Comparisons

Patrick Tessier, CEA/List Saclay, France
Sébastien Gérard, CEA/List Saclay, France
François Terrier, CEA/List Saclay, France
Jean-Marc Geib, Université des Sciences et Technologies de Lille, France

Abstract

Time-to-market is one of the most severe constraints imposed on today’s software engineers. The increasing complexity of systems has also shortened the time available for designing them. Several solutions have therefore been proposed to decrease the time and cost of producing applications. This chapter presents the product line paradigm as an effective solution for managing both the variability of products and their evolutions. The product line approach calls for designing a generic and parameterized model that specifies a family of products. It is then possible to instantiate a member of that family by specializing the “parent” model or “framework.” In describing the latter, designers need to explicitly model variability and commonality points among applications. The following discussion explains in detail how UML models express
these different requirements. We then describe specific extensions of UML profiles and the way they are used in various product line methodologies.

**Introduction**

Thanks to increasing storage and processing capabilities, software-based applications now provide more and more functionalities. This is achieved at the cost of ever greater complexity and heavier developer workloads. Market conditions have fostered sharp competition that has accelerated the software production cycle, while maintaining the same high quality development criteria.

To reduce time and cost, companies must therefore be able to capitalize on their work. Their systems must be maintainable throughout their service life. They also must be updatable to integrate future evolutions (new functionalities, new hardware, improved optimization, etc.). This evolutivity is one of the most important features of industrial systems at a time when service life requirements are being stretched to a maximum. For example, automotive systems are adapted to each type of automobile and may need to evolve with changes in automobile design or the advent of new hardware platforms that are cheaper and more efficient.

One approach to managing the evolution of software-based systems consists of designing each new application as an application family (Bosch, 2001). Any subsequent evolution in that application is then viewed as a new member of the family.

The application family paradigm was first proposed by (McIlroy, 1968; Parnas, 1979). The main focus behind their idea was to foster reuse of the models for a whole set of applications. An application is then seen as a specific instance of an application family called an application domain. This is known as the “product line” approach.

Clements later defined the software product line as follows:

“A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way.”

*(Clements & Northrop, 2001)*

From this definition, one can deduce that a software product line relies on its capacity to use and compose common and specific features of an application domain called a framework. Each specific member of an application domain then results from one specialization of a framework.

The main requirement for modeling a product line is the ability to depict the commonalities and variabilities of an application domain in the context of a framework. Several formalisms such as FAST (Harsu, 2002), FODA (Kang, Cohen, Hess, Novak, & Peterson, 1990), and GenVoca (Batory, Chen, Robertson, & Wang, 2000) are still available to address this need.