Object-oriented frameworks and product line architectures have become popular in the software industry during the 1990s. A vast number of frameworks has been developed in the industry for various domains, including graphical user interfaces (e.g. Java's Swing, Microsoft's MFC), graph-based editors (HotDraw, Stingray's Objective Views), business applications (IBM's San Francisco), electronic commerce (Sun/IBM), network servers (Java's Jeeves), just to mention few. When combined with components, frameworks provide the most promising current technology supporting large-scale reuse.

A framework is a collection of several fully or partially implemented components with predefined cooperation patterns between them. A framework implements the software architecture for a family of applications, to be specialized by application-specific code. Hence, some of these components are designed to be replaceable (variation points). Applications built on top of a framework not only reuse its source code but architecture design, which we consider as the most important characteristic of frameworks. Besides the fact that reuse of architecture design amounts to a standardization of the application structure, frameworks offer an additional important advantage. Framework-based software development implies a significant reduction in the size of the source code that has to be written by the programmer who adapts a framework.

The bad news is that framework development requires an extra effort. The costs are higher compared to the development of a specific application. Today proper techniques for describing frameworks, especially the points where application developers need to adapt the framework, are not elaborated. Furthermore, tools and methods assisting in both framework development and adaptation are almost non-existent or in their infancy.

The recent standardization efforts of the Unified Modeling Language (UML) provide a unique chance to harness UML as notational basis for framework development projects. However, UML does not provide adequate constructs to model frameworks. There is no indication in UML design diagrams what are the variation points and how they need to be instantiated.

This tutorial shows how to explicitly model framework variation points in UML using static and dynamic description diagrams to capture allowed structure and semantics of the variation points. A new member of the UML language family is proposed, including a few extensions to assist framework development and adaptation. The new extensions have been defined mainly by applying the UML extensibility mechanisms: stereotypes, tagged values, and constraints. A case study in the e-commerce domain will be presented to illustrate the approach in a real-world framework development scenario.