# Chapter 3.10
# Evolution in Model-Driven Software Product-Line Architectures

**Gan Deng**
*Vanderbilt University, USA*

**Jeff Gray**
*University of Alabama at Birmingham, USA*

**Douglas C. Schmidt**
*Vanderbilt University, USA*

**Yuehua Lin**
*University of Alabama at Birmingham, USA*

**Aniruddha Gokhale**
*Vanderbilt University, USA*

**Gunther Lenz**
*Microsoft, USA*

## ABSTRACT

This chapter describes our approach to model-driven engineering (MDE)-based product line architectures (PLAs) and presents a solution to address the domain evolution problem. We use a case study of a representative software-intensive system from the distributed real-time embedded (DRE) systems domain to describe key challenges when facing domain evolution and how we can evolve PLAs systematically and minimize human intervention. The approach uses a mature metamodeling tool to define a modeling language in the representative DRE domain, and applies a model transformation tool to specify model-to-model transformation rules that precisely define

metamodel and domain model changes. Our approach automates many tedious, time consuming, and error-prone tasks of model-to-model transformation, thus significantly reducing the complexity of PLA evolution.

## INTRODUCTION

Software *product-line architectures* (PLAs) are a promising technology for industrializing software-intensive systems by focusing on the automated assembly and customization of domain-specific components, rather than (re)programming systems manually (Clements & Northrop, 2001). A PLA is a family of software-intensive product variants developed for a specific domain that share a set of common features. Conventional PLAs consist of *component frameworks* (Szyperski, 2002) as core assets, whose design captures recurring structures, connectors, and control flow in an application domain, along with the points of variation explicitly allowed among these entities.

PLAs are typically designed using *scope/commonality/variability* (SCV) analysis (Coplien, Hoffman, & Weiss, 1998), which captures key characteristics of software product-lines, including: (1) *scope*, which defines the domains and context of the PLA, (2) *commonalities*, which name the attributes that recur across all members of the product family, and (3) *variabilities*, which contain the attributes unique to the different members of the product family.

### Motivating the Need for Model-Driven Software Product-Line Architectures

Despite improvements in third-generation programming languages (such as C++, Java and C#) and runtime platforms (such as CORBA, J2EE and Web Services middleware), the levels of abstraction at which PLAs are developed today remains low-level relative to the concepts and concerns within the application domains themselves, such as manually tracking the library dependency or

*Figure 1. Using DSMLs and domain-specific component frameworks to enhance abstraction and narrow the gap between problem and solution space of software-intensive systems*