Chapter 13
Model-Driven Impact Analysis of Software Product Lines

Hyun Cho
University of Alabama, USA

Jeff Gray
University of Alabama, USA

Yuanfang Cai
Drexel University, USA

Sonny Wong
Drexel University, USA

Tao Xie
North Carolina State University, USA

ABSTRACT

Software assets, which are developed and maintained at various stages, have different abstraction levels. The structural mismatch of the abstraction levels makes it difficult for developers to understand the consequences of changes. Furthermore, assessing change impact is even more challenging in software product lines because core assets are interrelated to support domain and application engineering. Model-driven engineering helps software engineers in many ways by lifting the abstraction level of software development. The higher level of abstraction provided by models can serve as a backbone to analyze and design core assets and architectures for software product lines. This chapter introduces model-driven impact analysis that is based on the synergy of three separate techniques: (1) domain-specific modeling, (2) constraint-based analysis, and (3) software testing. The techniques are used to establish traceability relations between software artifacts, assess the tradeoff of design alternatives quantitatively, and conduct change impact analysis.

DOI: 10.4018/978-1-61692-874-2.ch013

INTRODUCTION

Changes are inevitable in software development and maintenance. Software adaptation and evolution represent changes that occur throughout the software lifecycle from conception to termination, such that change management influences both cost and quality (Lehman & Belady, 1985). Thus, impact analysis, which identifies the ripple effects of proposed software changes, is beneficial before developers make actual modification to a software asset. However, it is challenging for developers to analyze multiple candidate options for changes and make decisions that may have significant consequences (Arnold & Bohner, 1993; Bohner, 2002). Furthermore, it is difficult for developers to understand the consequences of changes across various software assets due to the structural mismatch of abstraction levels at different stages of the software lifecycle (De Lucia et al., 2008).

The challenges of software change are even more problematic for a software product line, which supports the derivation of a wide range of software products (members of a product family) through composing or modifying the core assets of its architecture. Developers can make changes to the problem domain and/or application domain of a software product line either to enhance the core architecture, impacting all the derived products, or to add more products as new members in the product family. Making changes to a software product requires the consideration of multiple constraints from different stakeholders and users of the product line family. It is possible that one stakeholder proposes a change to the requirements to maximize the value of his/her own product derived from the product family, but the change may positively or negatively influence other products or other properties of the product family.

Impact analysis (Arnold & Bohner, 1993; Bohner, 2002) accepts as input a root asset to which an initial proposed change is made, and then performs three main steps: (1) The analysis traces the relationships between the root asset and other assets to identify related assets; (2) The analysis examines each related asset to determine if it will be affected by the proposed change, and if so, what changes must be made by developers to accommodate the initial proposed change; (3) The analysis adds the effort to make additional changes on related assets to the total needed effort, producing an estimated scope and cost of the proposed change as the results of the analysis. Analyzing the impact of changes before performing actual modification to an asset has been recognized as an important task in the software development lifecycle (Arnold & Bohner, 1993; Bohner, 2002; Jönsson, 2007; Anquetil et al., 2008). The analysis results can serve as a preliminary input to planning project costs and predicting software system quality (Ajila, 1995). Despite its importance, the majority of support offered in current requirements and design tools provides only limited functionality. For example, given multiple alternatives to accommodate a change, quantitative and automated techniques are needed to assess the tradeoffs of each alternative, to balance the constraints from different perspectives, and to minimize the impact on existing products. Furthermore, it is difficult to assess change impact on heterogeneous software artifacts generated at different stages of the software development process. Manually creating traceability relations (the basis of impact analysis) is time-consuming, error-prone, and tedious. Although predicting change impact facilitates project planning and quality prediction, it is often omitted because of these preceding obstacles.

The Unified Modeling Language (UML) has been widely used for system analysis and design and a large number of UML diagrams have been developed to assist with lifecycle concerns. Some researchers have proposed impact analysis based on UML models to accomplish changes in the system while minimizing potential consequences, such as cost overrun and intermingled evolutions (Briand et el., 2006; Briand et el., 2002). However,