Automated Error Analysis for the Agilization of Feature Modeling [1]

P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, M. Toro

Dpto. de Lenguajes y Sistemas Informáticos, University of Seville
Av. Reina Mercedes s/n, 41012 Seville (Spain)

Abstract

Software product lines (SPL) and agile methods share the common goal of rapidly developing high quality software. Although they follow different approaches to achieve it, some synergies can be found between them by *i*) applying agile techniques to SPL activities so SPL development becomes more agile; and *ii*) tailoring agile methodologies to support the development of SPL. Both options require an intensive use of feature models, which are usually strongly affected by changes on requirements. Changing large–scale feature models as a consequence of changes on requirements is a well–known error–prone activity. Since one of the objectives of agile methods is a rapid response to changes in requirements, it is essential an automated error analysis support in order to make SPL development more agile and to produce error–free feature models.

As a contribution to find the intended synergies, this article sets the basis to provide an automated support to feature model error analysis by means of a framework which is organized in three levels: a feature model level, where the problem of error treatment is described; a diagnosis level, where an abstract solution that relies on Reiter's theory of diagnosis is proposed; and an implementation level, where the abstract solution is implemented by using constraint satisfaction problems (CSP).

To show an application of our proposal, a real case study is presented where the Feature–Driven Development (FDD) methodology is adapted to develop an SPL. Current proposals on error analysis are also studied and a comparison among them and our proposal is provided. Lastly, the support of new kinds of errors and different implementation levels for the proposed framework are proposed as the focus of our future work.

Keywords: feature models, agile methods, error analysis, theory of diagnosis, constraint programming

1 Introduction and Motivation

The so–called *agile methods* have arisen to face up to the problems that traditional, *heavyweight* software development methodologies have not satisfactorily solved yet. Agile methods pursue some main goals which are described in the *Agile Manifesto* (Fowler and Highsmith, 2001), where a number of key changes to traditional software development are proposed. For example: focusing the efforts during development in the interaction with customers through working software; collaborating with customers during development instead of negotiating contracts at the beginning of development; adapting software to changing requirements, etc. In other words, the aim of agile methods is producing high–quality software products in less time and cost than using traditional software development methodologies by reducing unnecessary tasks and increasing productivity.

On the other hand, the *software product line* (SPL) approach intend to develop a set or family of software products within a concrete application domain. In a SPL, software products are developed from a set of shared, common assets —the *core assets*— and a set of *product–specific assets*. The core–assets development process is known as *domain engineering* whereas the product–specific assets development process is known as *application engineering* (Pohl et al., 2005).

Although both approaches are very different from each other, they share the aim of reducing development time and cost while quality is not compromised, even increased. Our experience applying both approaches separately has made us think that is possible to find some synergies by sharing some practices and techniques so that SPL development becomes more *agile* and agile methods can adopt an SPL–like orientation.

Considering the *agilization* of SPL development, this article is focused on processes related to the so–called *feature models*, which are used to describe the products in an SPL and are intensively used in SPL development (see Section 2.1). For example, Czarnecki et al. (2005) and Sochos et al. (2004) propose inferring the core architecture from feature models; Batory et al. (2004) propose using *feature–oriented programming* (FOP) to implement an SPL decomposing the architecture into features and automatically deriving products from a selection of their features; Benavides et al. (2005) use feature models to support decision making during production.

Email address: `{ptrinidad, benavides, amador, aruiz, mtoro}@us.es` (P. Trinidad, D. Benavides, A. Durán, A. Ruiz-Cortés, M. Toro).

[1] This work has been partially supported by the European Commission (FEDER) and Spanish Government under CICYT project WEB–FACTORIES (TIN2006–00472).