Chapter 55
Adding More Agility to Software Product Line Methods:
A Feasibility Study on Its Customization Using Agile Practices

Kun Tian
University of Wisconsin, USA

ABSTRACT

Software Product Line Methods (SPLMs) have been continuously gaining attention, especially in practice, for on one hand, they address diverse market needs while controlling costs by planned systematic reuse in core assets development (domain engineering), and on another hand, they reduce products' time-to-market, achieving a certain level of agility in product development (application engineering). More cost-effective and agile as they are than traditional development methods for producing families of similar products, SPLMs still seem to be heavy weight in nature. In SPLMs, significant up-front commitments are involved in development of a flexible product platform, which will be modified into a range of products sharing common features. Agile Methods (AMs) share similar goals with SPLMs, e.g., on rapidly delivering high quality software that meets the changing needs of stakeholders. However, they appear to differ significantly practices. The purpose of this work is to compare Agile and Software Product line approaches from fundamental goals/principles, engineering, software quality assurance, sand project management perspectives, etc. The results of the study can be used to determine the feasibility of tailoring a software product line approach with Agile practices, resulting in a lighter-weight approach that provides mass customization, reduced time-to-market, and improved customer satisfaction.

1. INTRODUCTION

The need to rapidly produce quality software and respond to changes in a flexible and quick manner has driven the definition of new techniques, tools, and notations. These approaches share some common goals: increasing the productivity of the development teams, reducing products' time-to-market, reducing development costs and improving customer satisfaction. Two such methods are agile methods (AMs),

DOI: 10.4018/978-1-5225-3422-8.ch055

including Scrum (Schwaber & Beedle, 2001), Extreme Programming (Beck, 1999), DSDM (DSDM Consortium, 2006), and FDD (Palmer, 2003), and software product line methods (SPLMs), including PLUSS (Eriksson, Börstler, & Borg, 2005), COVAMOF (Sinnema, Deelstra, Nijhuis, & Bosch, 2004) and FOOM (Ajila, & Tierney, 2002)..

Software product line methods (SPLMs) are practices-based, or plan-driven, software development approaches in which a set of software-intensive systems that share a common, managed set of features are produced from a set of re- usable core assets in a prescribed way (Clements & Northrop, 2001)(Pohl, Böckle, & Van Der Linden, 2005). A core asset is a software artifact that is re-used in the production of customized products in a software product line (SPL). The assets include the requirements, architecture, components, modeling and analysis, plans, etc. A SPL product can be quickly assembled from core assets, and hence it achieves manufacturing efficiency. SPLMs support mass customization, which is "producing goods and services to meet individual customer's needs with near mass production efficiency" (Pine & Davis, 1999). Mass customization in SPLMs is transparent: customers can obtain a unique product by having their special requirements implemented; their common requirements are assessed before production begins (A Framework for Software Product Line Practice, 2006)..

The purpose of this paper is to present a comparison of SPLM and AM methods to make a preliminary study on the possibility to introduce more agility into SPLM using Agile practices. The comparison criteria span fundamental goals and principles, engineering activities, software quality assurance activities, and project management activities. This paper uses well established SPLMs and AMs in the comparison. The comparison is expected to provide a useful foundation to the community, which can be also extended to include additional results available in the literature and additional comparison criteria.

2. BACKGROUND

AMs are software processes that share the same values: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to changes over following a plan. The Agile Manifesto inspired 12 principles for Agile process (Martin, 2002). Among these principles, the highest priority is to satisfy the customer through early and continuous delivery of software. Agile methods use short iterations (sprints) that are typically two to four weeks long. Satisfying the customer also involves recognizing the need to change requirements, even late in development, to support the customer's competitive advantage. The customers are highly involved, as they receive frequent deliverables of working software and work together with the technical people daily throughout the project. Working software is the primary measure of progress on the project, as opposed to modeling artifacts, etc. The software is built by motivated individuals, who have an environment and the support they need to get the job done. The self-organizing teams strive for technical excellence (i.e., best requirements, best architecture, etc.) and simplicity (i.e., maximizing the amount of work not done, such as extensive documentation for planning, requirements, architecture, etc.). The project proceeds at a pace that is sustainable over the long run and includes regular reflections on how to become more effective at implementing necessary changes.

Figure 1 presents an overview of an Agile engineering process. AMs minimize requirement engineering and design modeling practices so that the team can begin working on code as soon as possible.