Introduction to Software Product Lines: Engineering, Services, and Management Minitrack

Timo Käkölä
University of Jyväskylä, Finland
timokk@jyu.fi

Andrea Leitner
Graz University of Technology, Austria
andrea.leitner@tugraz.at

Software has become the key asset for competitive products and services in all industries. Thus, competitiveness in software development, maintenance, and related services has become a concern for organizations. Competitiveness can be increased through (1) internal strategies such as the strategic creation and reuse of software assets and (2) external strategies such as outsourcing software development, maintenance, and/or services from third party service providers and acquiring off-the-shelf components from providers and open source communities. A viable third strategy is to develop both strategies in parallel. This minitrack focuses on the first and third strategy.

Software product line engineering is an industrially validated methodology for developing software-intensive systems and services faster, at lower costs, and with better quality and higher end-user satisfaction. It differs from single system development, as:

1. It needs two development processes to work optimally: domain engineering and application engineering. Domain engineering defines and realizes the common and variable features of the product line by establishing a common software platform. Application engineering derives applications by exploiting the commonality and binding variability built into the platform.

2. It needs to explicitly define and manage variability. During domain engineering, variability is introduced in all assets such as requirements, architectural models, components, and test cases. It is used during application engineering to mass-customize applications to the needs of customers.

Software product line research has mostly focused on the modeling and management of variability in the context of embedded systems. Most insights have been obtained from large government and private organizations. The first three out of the five papers accepted this year reflect the mainstream product line research. The others deal with software product management and the provisioning of software products and services.

Kolassa, Rendel, and Rumpe study variability modeling in the context of Matlab/Simulink. Their paper evaluates three variability modeling concepts to find out which one is the best for modeling variability in the automotive domain. Through a controlled experiment with developers at Volkswagen, the study finds the so-called delta modeling approach to be both the preferred and the most efficient concept.

Research on Software Architecture Knowledge Management (SAKM) has largely focused on capturing and storing design decisions and their rationale, for example, to prevent the unintended violation of previous decisions. Approaches have been developed to combine SAKM with variability management in order to support the capturing of knowledge for a set of related products. Groher and Weinreich leverage a systematic literature review to identify the characteristics of the existing approaches and the gaps related to the application of the approaches in practice. The paper also contributes to documenting software product line architectural knowledge.

In their paper Diaz, Perez, Garbajosa, and Fernandez-Sanchez present the concept of Product-Line Architectural Knowledge (PLAK), encompassing the knowledge of product line architecture and its variability. This knowledge is managed with the help of the so-called PLAK Model, defining a set of design decisions. The usefulness of the PLAK Model in preventing the unintended violation of previous design decisions is evaluated through a case study in the context of a product line for power grids.

Kakar’s paper investigates the dilemma faced by software product managers who need to allocate resources to ensure the products appropriately meet users’ functional (Utilitarian) requirements and Hedonic requirements for pleasant experiences. The Utilitarian value and the Hedonic value are found to positively impact, respectively, user loyalty, thereby potentially enhancing the capacity to retain existing user base, and word-of-mouth of users, thereby increasing the ability to attract new users.

The extant literature offers practices service providers can use to execute international sourcing of software-intensive systems and services effectively. According to the paper authored by Käkölä and Lu, the literature falls short of providing a detailed enough set of classes of information systems that supports providers in executing the practices. This paper presents a set of best practices and supporting classes of information systems for service providers.