Editorial for the SCP SPLC 2010 Special Issue

Jan Bosch[1] and Jaejoon Lee[2]
Guest Editors

[1] Chalmers University of Technology, SE-412 96 Gothenburg, Sweden
jan@JanBosch.com

[2] School of Computing and Communications, Lancaster University, Lancaster, UK
j.lee3@lancaster.ac.uk

Since its rise to general awareness and popularity starting close to two decades ago, the concept of software product lines has taken the center stage in the software reuse community. After more than four decades of research into effective and efficient reuse of software inside the four walls of the organization, and countless initiatives, software product lines presented an approach that has proven to provide real productivity improvements in the development cost of software intensive products. This has allowed companies to increase their product portfolio with an order of magnitude, to allow for much higher degrees of configurability by customers, facilitated common look-and-feel across a wide product population and enabled companies to be more innovative by decreasing the cost of new product experiments. It achieved this by broadening the scope of study from technology to include process, business strategy and organizational aspects. Successful product lines address all aspects relevant to the organization and then adopt and institutionalize the approach in the company.

This special issue features eight papers from the 14th Software Product Line Conference (SPLC 2010). SPLC provides an institution and the premier meeting place for the software product line community. In particular, SPLC 2010 was held in Jeju, South Korea, and the accepted papers covered various areas of software product line engineering including product line contexts, variability management, formal approaches, product validation, and feature modeling. We invited eight top quality papers and they were significantly improved and extended for this special issue.

The paper of Karataş et al. introduces a mapping from extended feature models to constraint logic programming over finite domains. The mapping is used to translate into constraint logic programs; basic, cardinality-based and extended feature models, which can include complex cross-tree relationships involving attributes. This translation enables the use of off-the-shelf constraint solvers for the automated analysis of extended feature models involving such complex relationships. The authors also present the performance results of some well-known analysis operations on an example translated model.

The paper by Hartmann et al. describes the limitations of the current practice of combining heterogeneous components in a product line and describes the challenges that arise from software supply chains. This paper is motivated by the fact that software product lines are increasingly built using components from specialized suppliers. Hartmann et al. introduce a model driven approach for automating the integration between components that can generate a partially or fully configured variant, including glue between mismatched components. In particular, the authors analyses the consequences of using the approach in an industrial context, using a case study derived from an existing supply chain.

Ubayashi et al. propose a product line engineering method, which focuses on constructing embedded systems that take into account the contexts such as the external physical environments. The main goal is to avoid unexpected and unfavourable behaviours that might emerge in a system if a developer does not recognize any possible conflicting combinations between the system and contexts. In this paper, the authors provide the notion of a context-dependent product line, which is composed of the system and context lines and show a development process that includes the creation of both product line assets as well as context assets.

Eklund et al. present an in-depth view of how architects work with maintaining product line architectures at two internationally well-known automotive companies: the truck and bus manufacturer Scania and the car manufacture Volvo. The case study shows several interesting results: the process of managing architectural changes as well as the information the architects maintain and update is surprisingly similar between the two companies, despite that one has a strong line organization and the other a strong project organization. What does differ is that the architects studied see themselves interacting much more with other stakeholders than architects in general. The authors conclude that the results indicate how the company's different core values influence the architects when defining and maintaining the architectures over time.

Ganesan et al. present an analysis of the unit testing approach developed and used by the Core Flight Software System (CFS) product line team at the NASA Goddard Space Flight Center (GSFC). The goal of the analysis is to understand, review, and recommend strategies for improving the CFS' existing unit testing infrastructure as well as to capture lessons learned and best practices that can be used by other software product line (SPL) teams for their unit testing. The authors found that the unit testing approach incorporated many practical and useful solutions such as allowing for unit testing without requiring hardware and special OS features in-the-loop by defining stub implementations of dependent modules. Ganesan et al. conclude that these solutions are worth considering when deciding how to design the testing architecture for a SPL.

Marinho et al. introduce an approach for the development of mobile and context-aware software using the Software Product Line (SPL) paradigm. Mobile devices are multipurpose and multi-sensor equipment supporting applications able to adapt their behavior according to changes in the user's context (device, location, time, etc.). Although several solutions have been proposed to facilitate their development, reuse is not systematically used throughout the software development life-cycle. The authors present a Nested SPL for the domain of mobile and context-aware application, and discuss lessons learned in the SPL development with a context-aware visit guide product line.

Cetina et al. introduce Dynamic Software Product Lines (DSPL), which encompass systems that are capable of modifying their own behavior with respect to changes in their operating environment by using run-time reconfigurations, and prototype a Smart Hotel DSPL to evaluate the reliability-based risk of the DSPL. The authors discuss some guidelines learned in the case study and suggest that DSPL engineers should provide users with more control over the reconfigurations.

Heuer et al. propose a formal syntax and semantics for defining variability in Petri nets and use these extended Petri nets as a foundation to formally define variability in UML activity diagrams. UML activity diagrams serve as a basis for several testing techniques in product line engineering. The authors illustrate the contribution of such formalization to assurance activities in product line engineering by describing its usage in three application examples.

We sincerely thank all the reviewers; Creation of this special issue was not possible without their contributions.