Setting the context for variability integration in software product line

Abstract

The need for a faster, better and cheaper production of software has motivated the intention to use again and again the repetitive structure from the previous software development project in a new but similar context. However, routinely practical and realistic problem which occurs in software development force software developers towards producing fast and ad hoc solutions to solve the problem. This scenario hinders the payoffs of productivity and quality that can be reaped from software reuse. Thus the problem of software reuse has been highlighted in (Prieto-Diaz, 1993) as lack of widespread, formalize and systematic reuse. In (Frakes & Isoda, 1994) the success factors of systematic software reuse lies on its repeatable process, domain specific focus and the reuse itself primarily concentrates on higher level lifecycle artefacts such as requirement, design and subsystems.

One of the notable approaches for systematic software reuse is Software Product Line (SPL) (Paul Clements & Northrop, 2002; Pohl, Böckle, & van der Linden, 2005). SPL fulfils the criteria of systematic reuse where it focuses on domain specific approaches and enables the reuse of higher and lower level artefacts in software development. In SPL reuse happens with the use of core assets (Paul Clements & Northrop, 2002; McGregor, Northrop, Jarrad, & Pohl, 2002). With core assets, overlaps among members of the family can be leverage by merging common parts (known as commonality) and at the same time managing its variabilities. Thus, systematic reuse and automation in producing products from SPL depends on how well its core asset is designed. The more generic the core assets the more products can be generated. This generality requires postponing the design decisions with variation points to represent variabilities. Due to the numerous feature interactions and variation points to represent the variability in different level of abstractions in software development the amount of variability that has to be supported in software artefacts is growing considerably and also variability among individual products in the software product line also increases. As in (Berg, Bishop, & Muthig, 2005) managing variations at different levels of abstraction and across all generic development artefacts is an overwhelming task. Thus a central issue in SPL is the systematic management of variability where it has been the key difference with conventional software and also has been a major challenge in SPL development (Jan Bosch, 2004; Krueger, 2002). This chapter focuses on the

variability between requirements and architectural levels of SPL development phase. The explicit integration between the different phases is hoped to have a significant benefit in the variability representation between different abstraction levels and also leads towards a more efficient product derivation in SPL.