Applying Product Line to the Embedded Systems

Haeng-Kon Kim

Department of Computer Information & Communication Engineering,
Catholic University of Daegu, Korea
hangkon@cu.ac.kr

Abstract. For software intensive systems, a reuse-driven product line approach will potentially reduce time-to-market, and improve product quality while reducing uncertainty on cost and sc11edule estimates. Product lines raise reuse to the level of design frameworks, not simply code or component reuse. They capture commonality and adaptability, through domain and variability analyzes, to be able to create new products easily by instantiating prefabricated components, adapting their design parameters, and leveraging from established testing suites. In this paper, we examine software technology and infrastructure (process) supporting product lines more directly to embedded systems. We also present evaluation criteria for the development of a product line and give an overview of the current state of practices in the embedded software area. A product line architecture that brings about a balance between sub-domains and their most important properties is an investment that must be looked after. However, the sub-domains need flexibility to use, change and manage their own technologies, and evolve separately, but in a controlled way.

1 Introduction

Today the trend in computer-based products, such as cars and mobile phones, is shorter and shorter lifecycles. As a consequence, time spent on development of new products or new versions of a product must be reduced. One solution to this emerging problem is to *reuse* code and architectural solutions within a product family. Besides shortening development time, properly handled reuse will also improve the reliability since code is executed for longer time and in different contexts [1]. However, reuse is not trivial when applying it to real-time systems since both functional behavior as well as the temporal behavior must be considered.

We propose a design process suitable for developing product lines for real-time systems. The process starts in a requirement capturing phase where the requirements from all products in the line are collected. Communalities in functional- and temporal requirements among the products will be considered when the actual PLA is designed. The PLA is then analyzed. The objective of analyzing the PLA is to gain confidence in that the PLA is flexible enough to be a base on which all products can be realized without violating any temporal constraints. The contributions of this work with respect to embedded systems are an outline of a development process, focusing on the special considerations that must be taken into account when designing a PLA for embedded real-time systems . We also present an industrial case study of the use of a PLA for construction equipment.

M. Gavrilova et al. (Eds.): ICCSA 2006, LNCS 3982, pp. 163 – 171, 2006.

2 Related Works

2.1 Principles of Software Product Lines

The software market - just like other markets - has a great demand for variety in products. The entirety of product-variants of one software is also referred to as software system family or software product line. Manufacturing was the first discipline that provided an answer how to efficiently build varying products. Instead of building single system family members, interchangeable parts were assembled to products . Actually the same principle can be transferred to software products, which is depicted in Figure 1. Reusable parts are identified & selected from a pool of reusable assets - the asset base - and integrated into single products.

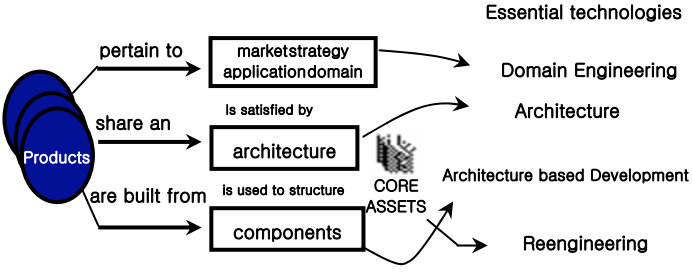

Fig. 1. Components in Product Line

2.2 Commonalities and Variabilities

Reuse implies that the asset base contains artifacts which can be reused in more than only one single product. This means that the asset base contains common parts which do not change between product line members and variable parts that feature different functionalities from member to member. Variability can be realized on run-time or development time. Product line engineering is concerned with development time variabilities as in figure 2.

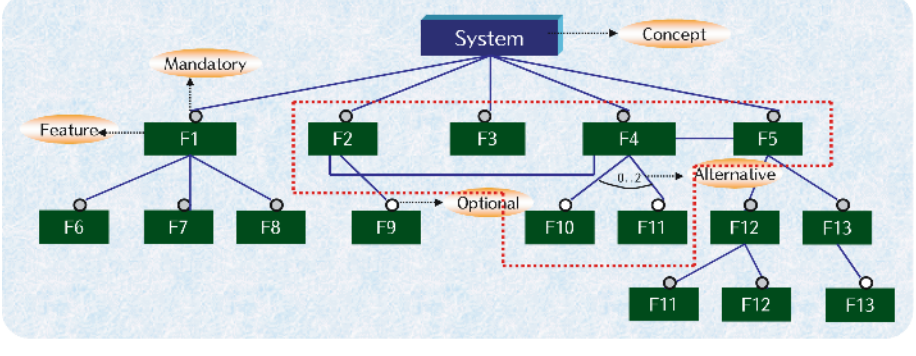

Fig. 2. Commonalities and Variabilities