

Product Line Architectures for Embedded Real-Time Systems

Anders Wall, Kristian Sandström, Jukka Mäki-Turja, Joakim Fröberg¹, and Christer Norström

Mälardalen Real-Time Research Centre, Department of Computer Engineering
Mälardalen University, Västerås, Sweden

¹Volvo Construction Equipment AB, Sweden
{awl, ksm, jma, jfg, cen}@mdh.se

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. To enable the use of a PLA and derivation of product architectures from the PLA we need a design language. Such a language is often called *architecture description language* (ADL) [2]. The ADL should have a precise syntax and semantics to enable architectural analysis, including analysis of, for instance, performance, maintainability, flexibility, and temporal properties. By making early analysis of the temporal behavior, for each product, based on the PLA and the product specific features, we can extract the following information: Traditional real-time measurements such as system utilization, response times for each feature, distribution of response times for a specific feature, and jitter information for features. However, this information is not only used for schedulability analysis, it is also used for analyzing the flexibility of the PLA with respect to implementation constraints. For example, what will happen if a non-implemented component will utilize more resources than estimated in the architectural analysis?

Furthermore, robustness with respect to internal errors and erroneous assumptions about the environment can be analyzed. Typical analysis is based on “what if”-questions such as: ‘*What will happen if a specific component slightly overruns its time budget?*’ or ‘*What will happen if events from the environment are generated in a higher frequency than assumed?*’

Moreover, the ADL must facilitate constructions for modeling of flexible components. The flexibility mechanisms specified on components constitutes the variation points that are used when product architectures are derived from the PLA, i.e. when the PLA is tailored for a particular product

The contributions of this work with respect to embedded real-time 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. Finally, we propose constructions that must be supported by an ADL in order to be suitable for describing product line architectures for embedded real-time systems.

2 Product line based development

In this section we discuss the development process in which a PLA for embedded real-time products is constructed. The process is iterative and includes architectural analysis of properties that are of vital importance for a PLA, e.g. flexibility. Moreover, the derivation of products from a PLA is dealt with. The design process proposed in this paper is shown in Figure 1 where the process is divided into *requirements capturing*, *PLA development*, and *product development*. The proposed design process will be further described in this section.

Developing a PLA is done either in an *evolutionary* or *revolutionary* way [2]. The evolutionary approach to PLA design is conducted by a generalization of existing products, whereas in the revolutionary approach, the common architecture is developed, rather than extracted from existing implementations. Independent of whether the evolutionary or the revolutionary approach is taken, the first

step when developing a PLA is to capture the requirements for every product in the product line. One way to do this is to organize and group required functionality into *features*. We consider features to be functional entities seen from a stakeholder’s perspective.

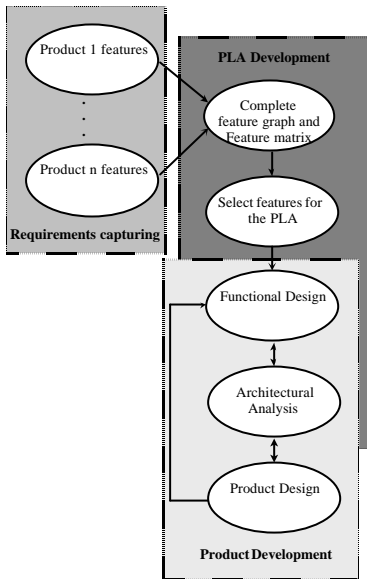


Figure 1 The process of developing a product line based on a PLA.

For instance, the function for adding items to an address book in an ordinary cellular phone can be a feature. A feature typically groups a set of requirements and therefore it also simplifies the requirements handling. At this point, we must consider all features in the product line. The products and their features make up a matrix, the *feature matrix* for the product line. The feature matrix shown in Table 1 has m features distributed over n products. A position in the matrix marked with X indicates that the feature in that column is present in the product of that line.

	Feature 1	Feature 2	Feature m
Product 1	X	X	
Product n		X	X

Table 1 A feature matrix

Features are rarely independent from each other. A feature can for instance depend on other features in order to deliver the desired functionality. Another example of a relation between features is the mutually exclusive relation, implying that only one of the related features can appear in the final product. If mutually exclusive features must co-exist in the product, effort has to be made to resolve the conflict. Features in real-time systems will also exhibit dependencies related to the temporal domain. For instance, consider the lock-free break feature and the anti-slid feature in automotive vehicles. Both features need information about

the wheels’ velocity, thereby having a shared temporal dependency related to the freshness of the sensor data. Such relations on features are specified in a *feature graph*.

The feature matrix and the feature graph constitute the basis for deciding what features to include in the PLA. Typically, features that are common among a majority of the products in the product line are included. Consequently, the PLA may provide features that are superfluous for some specific products. Identifying the commonality among the features for real-time systems is more complicated since we also have to take the temporal domain into consideration.

When the scope for the PLA has been decided, the features should be mapped to components that, together with their interrelations, constitute the actual architecture. This part of the development process is referred to as *functional design*. In the functional design of a PLA, the designer must take into consideration that features may have different implementations for different products. Depending on how implementations differ, the correct mechanisms for obtaining the desired flexibility must be selected.

For real-time systems we also have to consider temporal behavior. A typical example of how the temporal requirements influence the functional design is the following. Consider features that are functionally equivalent between a set of products. If these products will run on different infrastructures, i.e. operating system and hardware platform, the functionality may be partitioned among components in different ways to fulfill the timing requirements. In the high-end product we can partition a feature in such a way that it will be easy to maintain while in a low-end product we have to make an architecture that is focused on performance to be able to fulfill the timing requirements due to the limited resources in the low-end product.

After the functional design the PLA must be analyzed in order to verify that the architecture is flexible enough to facilitate all products in the product line. If not sufficiently flexible the architecture must be transformed. Thus, iterations between analysis and functional design are required. Since our focus is on real-time systems we would like to gain confidence in that the architecture is sufficiently flexible to be used in all products in the product line without violating the temporal constraints.

3 References

[1] N. E. Fenton, S. L. Pfleeger, Software Metrics: A rigorous & practical approach, International Thomson Computer Press, ISBN 0-534-95600-9, 1996

[2] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice, Addison-Wesley, ISBN 0-201-19930-0, 1997