

Software Product Lines in Automotive Systems Engineering

Steffen Thiel, Muhammad Ali Babar and Goetz Botterweck
Lero – The Irish Software Engineering Research Centre, University of Limerick

Liam O'Brien
NICTA – National ICT Australia

Copyright © 2008 SAE International

ABSTRACT

Product line approaches are well-known in many manufacturing industries, such as consumer electronics, medical systems and automotive [1]. In recent years, approaches with a similar background have rapidly emerged within Software Engineering, so called Software Product Line (SPL) approaches [2], [3].

As automotive manufacturers and suppliers design and implement complex applications, such as driver assistance [4], they strive for mechanisms that allow them to implement such functionality on integrated platforms. This offers the opportunity to build a variety of similar systems with a minimum of technical diversity and thus allows for strategic reuse of components. This has resulted in a growing interest in SPL approaches both in the software engineering and the automotive systems domain.

This paper discusses the increasing importance that SPL approaches could play within the context of Automotive Systems Engineering. To accomplish this, we first provide an overview of the major challenges faced by Automotive Systems Engineering [5]. We then present a selection of SPL approaches, which could provide solutions for the described challenges. To complement this we make the case for empirical evaluation as a basis for well-founded decisions and selection of techniques.

Finally, we present an in-depth discussion of how the approaches and techniques outlined can be used to address the identified challenges. The paper concludes with an overview of open research questions and expected benefits for the development of automotive systems.

INTRODUCTION

Product line approaches are well-known in many manufacturing industries, such as consumer electronics, medical systems and automotive [1]. In recent years, approaches based on similar paradigms have rapidly

emerged within Software Engineering, so called Software Product Line (SPL) approaches [2], [3].

As automotive systems get more and more complex and software-intensive manufacturers and suppliers face numerous challenges, such as the handling of complexity and variability, the need for integrated platforms and architectures and the inherent conflict between limited time and resources and high-quality products. In this paper we explore and discuss the contributions SPL approaches could make to provide solutions for these challenges.

The remainder of the paper is structured as follows: In the next section we give an overview of challenges faced by automotive engineering. Next, we first explain (on a general level) which role SPL Engineering could play within automotive systems engineering. We then discuss particular approaches which could offer solutions to the challenges discussed in the preceding section. We finish the paper by outlining related work and drawing conclusions.

SPL IN AUTOMOTIVE ENGINEERING

For more than 30 years reuse has been the long-standing notion to solve the cost, quality, and time-to-market issues associated with the development of software-based systems [6]. A major addition to existing reuse approaches since the 1990s are software product lines (e.g., [3], [7], [8], [9], [10], [11]). A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way [7]. The basic reuse philosophy of software product lines is intra-organisational reuse through the explicitly planned exploitation of similarities between related products. This philosophy has been adopted by a wide variety of organisations and has proved to substantially decrease costs and time-to-market and increase the quality of their software products (e.g., [7], [8], [9], [10], [11]).

Software product lines are particularly important to the automotive domain as car suppliers tend to integrate and combine more and more software functionality such as driver assistance, car dynamics, and airbag control systems on powerful platforms on different makes and models to meet different requirements across multiple markets. Many suppliers and manufacturers already use a product line approach so as to be able to build different variants of their products for use within a variety of automotive systems (e.g., [7], [12], [13], [14]).

In the software product line context, software products are developed in a two-stage process, i.e., a domain engineering and a concurrent application engineering process, as depicted in Figure 1 (e.g., [3], [8], [15]). The *domain engineering* process is responsible for establishing the reusable platform [3]. The platform consists of artefacts such as the requirements specification, architecture documentation, design specification, implementation, and test cases. Domain engineering defines the commonality and variability between product line members. It involves, among other activities, the implementation of an adequate product line architecture and a set of reusable software components such that the commonalities can be exploited economically while retaining the ability to vary the products.

The *application engineering* process is responsible for deriving product line applications or products from the platform established in domain engineering. It exploits the variability of the product line and ensures the correct binding of the variability to the specific needs of the customer. In many industrial organisations, these artefacts consist of both hardware and software components (e.g., [16]). Domain and application engineering is also known as core asset and product development (e.g., [7]).

Domain and application engineering are typically supported and coordinated by the technical and organisational management. The technical management supports co-operation between domain and application engineering. It includes tasks such as project planning, configuration management and support by software tools. The organisational management must create a basis that supports the technical activities within the existing organisational structure. Besides, it must ensure that the business goals of the product line are specified and communicated. In addition, the organizational management is responsible for the commitment of sufficient resources for the development of the reusable artefacts such that the advancement and effective use of those artefacts is ensured.

The idea behind a product line approach to product engineering is that the investments required to develop the reusable artefacts during domain engineering are outweighed by the benefits in deriving the individual products during application engineering (e.g., [7], [11]). A fundamental reason for researching and investing in approaches and technologies for product lines is to obtain the maximum benefit out of this upfront investment – in

other words, to minimise the application engineering costs of automotive systems.

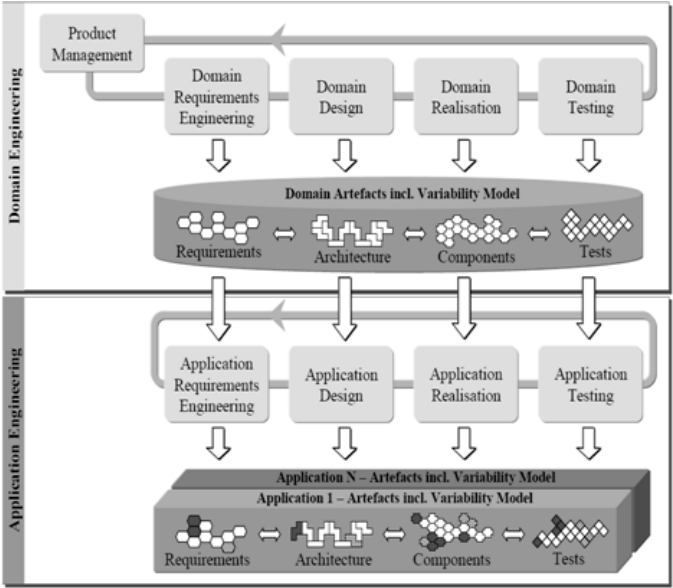


Figure 1 – Software Product Line Engineering [3]

Clements et al. [7] provide a product line practices framework that describes essential practice areas as well as the relationship of these practice areas within system and software engineering, technical management, and organizational management. The framework identifies 29 practice areas that need to be considered when moving from single system development to product line development (see Figure 2).

Technical Management	Systems and Software Engineering	Organizational Management
Scoping	Understanding Relevant Domains	Market Analysis
Technical Planning	Requirements Engineering	Technology Forecasting
Data Collection, Metrics, and Tracking	Architecture Definition	Building a Business Case
Technical Risk Management	Architecture Evaluation	Funding
Make/Buy/Miner/Commission Analysis	Component Development (Make)	Structuring the Organization
Configuration Management	COTS utilization (Buy)	Operations
Tool Support	Mining Existing Assets (Mine)	Launching and Institutionalizing
Process Definition	Developing Acquisition Strategy (Commission)	Customer Interface Mgmt
	Software System Integration	Organizational Planning
	Testing	Organizational Risk Mgmt
		Training

Figure 2 – Practice Areas in Product Line Development (adapted from [7])

For example, the practice area “Scoping” refers to the specification of products and features that should be in scope for the product line under consideration. Scoping requires inputs from market analysis, business case and technology forecasting. The result provides a clear guideline of “what to build” during the development. It also provides support for the development of the required core assets. Other practice areas refer to similarly important topics of product line development and address and avoid common pitfalls.

CHALLENGES IN AUTOMOTIVE ENGINEERING

We will now present some challenges that automotive engineering faces today. These have been collected from a review of the relevant literature [4, 5] as well as from workshops with industry partners.

REDUCTION OF COMPLEXITY

Many automotive suppliers and manufacturers such as Cummins [7], Bosch [12] [17], DaimlerChrysler [14], and Volkswagen [18] use a product line approach so as to be able to build different variants of their products for use within a variety of automotive systems. The size of the product lines is usually large since only in this case significant economies of scale can be achieved. Therefore, automotive software platforms are typically developed in such a way that they can be customized and used in hundreds of products (e.g., [17]). These platforms can easily incorporate thousands of variation points and configuration parameters.

Managing this amount of variability is extremely complex and requires sophisticated modelling techniques. In particular, there is a strong need for appropriate approaches that support the different stakeholders in carrying out their development tasks in complex software product line efforts with a large number of variants [19].

IMPROVED ARCHITECTURAL DESIGN PRACTICES

Software architectures for automotive systems need to be comprehensive enough to capture and describe the multi-functionality and all related issues. A comprehensive architecture should provide a sophisticated structural view that covers all the aspects that are relevant to different kinds of stakeholders such as drivers, passengers, maintainers, and production staff. A comprehensive car architecture should be described at different levels such as functionality level, design level, cluster level, hardware level, and deployment level [5].

Architecture and interface specification is another big challenge in software engineering for automotive systems. There is general lack of suitable and reliable methods to accurately and sufficiently provide interface specifications. This is particularly important in the automotive industry because of its distributed mode of software development.

From software to systems engineering

Traditionally the design, manufacturing and assembly of cars relied on a modular approach, which required that each component was as independent as possible from others. However, in recent years we see more and more complex functions (such as occupant protection systems or driver assistance systems) which (1) require complex control processes (which lead to software-intensive implementations) and (2) require the integration with various subsystems in the car (which results in clusters of

hardware and software components that are highly integrated and interdependent).

This change in implementation structures has to be reflected in the engineering processes used to create these structures. Hence, automotive engineering has to adapt methods and techniques from software engineering. But that's not enough. What really is required are approaches that incorporate both hardware and software components and their interplay. Hence, we have to move on from software engineering to system engineering.

IMPROVED EVIDENCE IN MODELING AND EVALUATING QUALITY ATTRIBUTES

A quality attribute is a non-functional requirement of a software intensive system, e.g., reliability, modifiability, performance, usability and so forth. According to IEEE Standard 1061 [20], software quality is the degree to which the software possesses a desired combination of attributes. Quality is a major issue for automotive systems engineering. Like any other domain, software quality is fundamental to any automotive system's success. However, quality as a concept is very challenging to define, describe and understand. This aspect of any system has very strong subjective interpretation. That is why it is vital to systematically elicit and precisely define quality aspects of a software intensive automotive system. There are a number of classifications of quality attributes. McCall listed a number of classifications of quality attributes developed by software engineering researchers including himself [21]. A later classification of software quality is provided in [22].

An examination of quality definitions, meanings and views in [23] describes quality as hard to define and measure but easy to recognize. However, quality experts including some with a software background have proposed models, e.g., [24], [25], [26], not to measure quality itself but to measure surrogate attributes which, when combined, can provide a notion of the quality of the product. While, there are several quality attributes required by different automotive systems, there are two main quality attributes that must be satisfied at a level required by the law of a country, where an automobile is going to be sold. These two quality attributes are reliability and safety.

Legislation at national levels, European Union and global levels along with market place demand is directing the automotive industry to comply with more stringent safety levels, increasing reliability levels, higher fuel efficiency and low emission levels [27]. The AUTOSAR standard is a move to establish an open standard for automotive embedded electronic architecture. AUTOSAR tries to achieve modularity, scalability transferability and reusability of functions. However there is no formal verification of reliability and safety concerns or formal (mathematically based) testing in AUTOSAR.

DESIGN AND EVALUATING ARCHITECTURES FOR FAMILY OF SYSTEMS

Software architecture provides the key framework for the earliest design decisions taken to achieve functional and quality requirements [28]. Architecture for a family of systems also helps identify the commonality among different systems and explicitly document variability. As we have mentioned architecting automotive systems is a complex and challenging design activity, and architecting a product family is even more challenging. It involves making decisions about a number of inter-dependent design choices that relate to a range of design concerns. Each decision requires selecting among a number of design options; each of which impacts differently on various quality attributes. Additionally, there are usually a number of stakeholders participating in the decision-making process with different, often conflicting, quality goals, technical and project constraints, such as existing platforms, cost and schedule [29].

Apart from the challenge of devising optimal architectural solutions, specifying the architecture and interfaces of components of an automotive system is a difficult task, which poses several kinds of challenges. Usually OEMs (Original Equipments Manufacturers) have to provide an overall architecture of the automotive systems in its cars and distribute these to potential suppliers of systems and components who do the implementation. The AUTOSAR standard is a move to establish an open standard for automotive embedded electronic architecture. AUTOSAR tries to achieve modularity, scalability transferability and reusability of functions. However, even if the architecture and components are specified using AUTOSAR, there is still no checking of conformance or conformance validation. Hence, there is a need for specific methods and tools to validate that those implementations actually conform to the specifications and that the combination of the various implementations conforms to the OEMs' specifications.

ARCHITECTURE KNOWLEDGE MANAGEMENT

We have mentioned that designing and evaluating software architectures of a family of systems in automotive domain involves complex and knowledge intensive tasks. The complexity lies in the fact that tradeoffs need to be made to satisfy current and future requirements of a potentially large set of stakeholders, who may have competing vested interests in architectural decisions. The knowledge required to make suitable architectural choices is broad, complex, and evolving, and can be beyond the capabilities of any single architect [30]. Due to the recognition of the importance and far reaching influence of the architectural decisions, several approaches have been developed to support architecting processes. Examples are the Generic Model for architecture design [31], Attribute-driven design [32], Architecture Tradeoff Analysis Method (ATAM) [33], 4+1 views [34], Rationale Unified Process (RUP) [35] and architecture-based development [36]) While these approaches help to manage

complexity by using systematic approaches to reason about various design decisions, they provide very little guidance or support to capture and maintain the details on which design decisions are based, along with explanations of the use of certain types of design constructs (such as patterns, styles, or tactics). Such information represents architecture knowledge, which can be valuable throughout the software development lifecycle [37].

Lack of a systematic approach to capture and use architecture knowledge may preclude organizations from growing their architecture capability and reusing architectural assets. Moreover, the knowledge concerning the domain analysis, architectural patterns used, design alternatives evaluated and design decisions made is implicitly embedded in the architecture and/or becomes tacit knowledge of the architect [38].

INCREASED PROCESS EFFICIENCY

Reduction of costs-per-product, shorter time to market

One of the greatest challenges for engineering approaches are the conflicting goals of creating products with a sufficient quality level and reducing the costs-per-product. On the one hand, components have to fulfil the required quality criteria, especially for safety-relevant functions. On the other hand, in automotive engineering we have to deal with target costs which are fixed in advance and hence dictate the resources which are available for component design and implementation. This conflict has to be dealt with by approaches and process models that allow for efficient creation of components with the required quality level.

Seamless model-driven development

The intelligent use of models promises to be one of the major foundations for efficient processes in automotive systems engineering. Models can be used to describe the different forms of knowledge which are captured and transformed during the engineering process, such as requirements, high-level or detailed design, implementation or test cases. This is the foundation for tooling and automation, which in turn promotes efficient engineering processes [5].

However, the current usage of models in automotive systems engineering is insufficient and is far from realising its full potentials. For instance, models are used in isolated areas, without an integrated flow of information. Often models are used only in a semiformal way as a form of communication on the whiteboard or as illustrations in a textual specification.

This lack of clearly and precisely defined semantics undermines the use of real *model-driven* approaches, where models are expressive enough to be used in powerful interactive tools and the automatic derivation of further artefacts including the implementation.