

The Role of Software Engineering in Future Automotive Systems Development

Siobhán Clarke¹, Brian Fitzgerald², Paddy Nixon³, Klaus Pohl², Kevin Ryan²,
David Sinclair⁴ and Steffen Thiel²

Lero – The Irish Software Engineering Research Centre

¹Trinity College Dublin

²University of Limerick

³University College Dublin

⁴Dublin City University

Copyright © 2008 SAE International

ABSTRACT

The amount and complexity of software in automotive systems is constantly increasing. Today's luxury cars include numerous electronic control units. A large part of the functionality of these units is driven by software.

In the future even more software-intensive automotive systems are expected as automotive manufacturers and suppliers tend to integrate and combine applications on more powerful platforms. The increasing amount and complexity of software in these platforms has led to the situation where software engineering has become an essential discipline within automotive systems development.

This paper identifies essential areas of software engineering that will have a significant impact on future automotive systems and systems development. Particularly, it discusses how the software engineering discipline can be developed in the context of overall automotive systems development while considering essential requirements and constraints that are or will become prevalent in this domain.

1 INTRODUCTION

Over the past decade, the amount and complexity of software in automotive systems has increased substantially. Today's luxury cars include more than 70 electronic control units (ECUs) that operate as partly networked systems to improve passenger comfort, safety, economy, and security. Many of these systems are driven by software [1].

In the future more software-intensive automotive systems are expected as automotive manufacturers and suppliers tend to integrate and combine applications such as driver assistance, car dynamics, and airbag control systems on more powerful platforms (e.g., [2]). The integration and combination of automotive

applications is particularly crucial for the exploitation of strategic reuse that can drive the reduction of time-to-market and of the costs of automotive applications.

The increasing amount and complexity of software in automotive platforms has led to the situation where software engineering has become an essential discipline within automotive systems development [3]. Manufacturers and suppliers already make great efforts in adopting, tailoring, and improving software engineering processes and practices while working in globally distributed teams to build high-quality systems (e.g., [4]).

This paper identifies essential areas of software engineering that will have a significant impact on future automotive systems and systems development. Section 2 motivates domain-oriented software engineering as a general approach to provide effective research results in a given application domain, such as automotive. Sections 3-6 discuss important areas of software engineering that we believe will have an effect on automotive systems development. The areas include software product line engineering (Section 3), global software development (Section 4), service-oriented architectures (Section 5), and mathematics applied to software engineering (Section 6). We describe particular challenges with respect to the automotive domain and outline approaches within the area to address these challenges. Finally, Section 7 concludes the paper with a summary.

2 DOMAIN-ORIENTED SOFTWARE ENGINEERING

Software Engineering is a broad, dynamic and multi-faceted field, where research can vary from highly theoretical, abstract models, through the development of numerous branches of technology, into empirical studies of industrial practice.



Documents

Document Sharing Hub

The development of quality software requires, in addition to software engineering capabilities, knowledge about the application domain in which software is going to operate. Research in Software Engineering has progressed from seeking universal solutions, whether through concepts, notations, methods or tools, to the development of domain-specific solutions that take advantage of domain specificities. Domain-specialised workshops, conferences and publications have increased and many general conferences and journals devote sessions or issues to specific domains (e.g., [5]).

Domain-oriented research has the potential to

- narrow the problem focus,
- reflect domain standards, constraints, models and priorities,
- facilitate rapid validation, and
- accelerate industrial uptake.

We believe that domain-oriented Software Engineering can provide more effective research results as it includes the peculiarities of the given domain. Once domain-specific solutions have been developed they are available for adaptation and exploitation in other domains. Research outcomes that are proven useful in a number of domains may be generalised to provide domain independent solutions.

Within Lero, the Irish Software Engineering Research Centre, we have chosen the automotive domain as our initial focus. Within that we have identified different research areas that we believe will have a significant impact on the development of future automotive systems. The research areas include:

- Software Product Lines
- Global Software Development
- Service-Oriented Architectures
- Mathematics Applied to Software Engineering

In the following sections we will introduce these areas, describe particular challenges regarding automotive system development, and propose approaches and research directions to address these challenges.

3 SOFTWARE PRODUCT LINES

3.1 INTRODUCTION

Product line approaches are well-known in many manufacturing industries, such as consumer electronics, medical systems and automotive [6]. In recent years, approaches with similar roots have rapidly emerged as important paradigms within Software Engineering, so

called Software Product Line (SPL) development approaches [7, 8].

As automotive manufacturers and suppliers design and implement complex applications, such as driver assistance systems [9], they strive for mechanisms that allow them to implement major functionalities on integrated platforms. The move towards those larger platforms has provided an opportunity for strategic reuse of software components. One key aspect of this strategic reuse is the idea to build a variety of similar systems with a minimum of technical diversity. This has resulted in a growing interest in SPL approaches both in the software engineering and the automotive systems domains.

3.2 CHALLENGES

There are several challenges related to the adoption of product line approaches in automotive software-based systems (e.g., [3, 10]). The challenges include the following topics:

- Management of large numbers of variants
- Development of product line architectures
- Integration of model-driven practices

3.2.1 Management of Large Numbers of Variants

Many automotive suppliers and manufacturers such as Cummins [7], Bosch [2, 4], and DaimlerChrysler [11] 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 a way such that they can be customized and used in hundreds of products simultaneously (e.g., [4]). These platforms could easily incorporate thousands of variation points and configuration parameters.

Managing this amount of variability is extremely complex and requires sophisticated modelling and representation techniques that can cope with large data sets. In particular, there is a strong need for appropriate approaches that support the different stakeholders in carrying out their development tasks in software product line efforts with a large number of product variants [12].

3.2.2 Development of Product Line Architectures

Software architecture provides the key framework for the earliest design decisions taken to achieve functional and quality requirements. An architecture for a family of systems needs to identify the commonality among different systems and must explicitly include a variability documentation. 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.

3.2.3 Integration of Model-driven Practices

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, the high-level or detailed design, the implementation or test cases. This is the foundation for tooling and automation, which in turn promotes efficient engineering processes [3].

However, the current use of models in automotive systems engineering does not realise its full potential. 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.

3.3 APPROACHES AND RESEARCH DIRECTIONS

Our research focuses on the following approaches to address the challenges mentioned:

- Visually informed variability management
- Architecture-Based Development
- Seamless model-driven development

3.3.1 Visually Informed Variability Management

As mentioned previously, industrial size automotive product lines can easily incorporate thousands of variation points and configuration parameters for product customization. A promising approach to address the complexity problem in automotive systems engineering is to improve the efficiency of variability management and product derivation.

Variability management is the process by which the variability of the product line development artefacts (e.g.,

architectural models, software components, and hardware components) is planned, documented, and managed throughout the development lifecycle. Variability management supports critical product line engineering tasks such as product derivation. Variation points identify locations in product line artefacts at which variation will occur [13].

A large part of the application engineering activities in an established and optimized product line approach consist of reusing the platform artefacts from domain engineering and binding the variability as required for the different applications/products. In this process, the variability is resolved and fixed to the particular customer product. Potentially pre-developed customer-specific application components are integrated into the product infrastructure. If domain and application engineering are coordinated in such a way then product production becomes more a configuration and composition than a development activity. In this way, a mass-customization of products can be achieved and the upfront investment in domain engineering can be more than justified.

Systematic variability management and product derivation can and should be supported by visualisation techniques and tools that support the understanding, management, and effective use of product line development artefacts, their built-in variability, and the dependencies among them. With suitable techniques such visualisations can also amplify the cognition about large and complex data sets created and used in industrial software product line engineering. Our research is focused on exploring the potential of visual representations such as trees and graphs combined with the effective use of human interaction techniques such as dynamic queries and direct manipulation when applied in a software product line context.

3.3.2 Architecture-Based Development

Software architecture embodies some of earliest design decisions, which are hard and expensive to change if found flawed during downstream development activities. The role of software architecture in a family of system becomes much more vital as architectures of individual products are derived from the core architecture. Hence, any flaw in the core architecture usually has ramifications for the achievement of required quality attributes for individual products in a family. A systematic and integrated approach is required to address architectural issues throughout the software development lifecycle.

Hofmeister et al. [14] have proposed a general model of software architecture design. This model has three activities: architectural analysis, architectural synthesis, and architectural evaluation. However, it does not consider the post-architecting activities, which are equally vital to ensure that the intent behind the architecture design remains correct during implementation and maintenance of the software architecture. One of the main characteristics of architecture-based development is the role of quality attributes and architecture styles and pat-