From Product Architectures to a Managed Automotive Software Product Line Architecture

Benjamin Cool, Christoph Knieke, Andreas Rausch, Mirco Schindler, Arthur Strasser, Martin Vogel
TU Clausthal, Department of Computer Science, Software Systems Engineering
Julius-Albert-Straße 4
D-38678 Clausthal-Zellerfeld, Germany
{benjamin.cool|christoph.knieke|andreas.rausch|mirco.schindler|arthur.strasser|m.vogel}@tu-clausthal.de

Oliver Brox, Stefanie Jauns-Seyfried
Volkswagen AG, Powertrain Electronics
P.O. 16870
D-38436 Wolfsburg, Germany
{oliver.brox|stefanie.jauns-seyfried}@volkswagen.de

ABSTRACT
To keep the software development for vehicles cost efficient, software components are reused for different variants as well as for succeeding generations. Furthermore, cost reductions are achieved by software sharing between the Original Equipment Manufacturer (OEM) and the suppliers. However, as a consequence of the blackboxed view caused by software sharing, no common detailed software product line architecture specification for the Electronic Control Unit (ECU) software exists, as it would be required for analyzing the quality of the product line architecture, planning changes on the product line architecture, checking the compliance between the product architecture and the product line architecture, and therefore, avoiding architecture erosion. Thus, after several product generations, software erosion is growing steadily, resulting in an increasing effort of reusing software components, and planning of further development. Here, we propose an approach for repairing an eroded software consisting of a set of product architectures by applying strategies for recovery and discovery of the product line architecture. Moreover, we give a methodology for a long-term manageable, plannable, and reusable software product line architecture for automotive software systems.

Keywords
Architecture Evolution; Software Product Lines; Software Erosion; Architecture Quality Measures; Automotive

1. INTRODUCTION
Nowadays, many variants of a vehicle exist. This variability is mainly achieved by reusing software of the ECUs for a huge number of vehicles. Furthermore, for each new vehicle generation, the software of preceding generations of the vehicle is reused or adopted.

In the automotive software development the reusability is enabled by applying a shared software development process between the OEM and one or several suppliers for automotive ECUs. The OEM develops or reuses parts of the software and delivers the object code to the supplier for integration. The software components have to provide a certain degree of modularity so they can be integrated hardware independent. We call this software components modules in the following. In order to realize the communication between OEM and supplier modules, some interfaces are shared. All other interfaces and modules are blackboxed for know how protection purposes. Therefore, as a basis for software sharing, merely a coarse grain view on the top level groups of modules exists, e.g., in the case of the engine control unit, combustion engine functions, power train functions, and vehicle functions.

An explicit top level architecture (Product Line Architecture - PLA) is important to coordinate the shared development between the OEM and the suppliers. Each product that is developed has an architecture (Product Architecture - PA) whose structure should be mapped onto the top level architecture. This top level architecture describes the structure of all realizable PAs. However, because of software sharing an overall assignment of top level groups to modules, and their interface, is missing (depicted by boxes with question marks in Figure 1). The knowledge of the overall, product independent structure is not explicitly documented, and therefore exists only implicitly in the minds of the partic-
ipants. Further development of existing products and the development of new products lead to an eroded PA [17] as an initially demanded structure is not available [11], see PA in Figure 1. At some point structural difference between the PA and the PLA reaches a degree of degeneration that it will be increasingly difficult to integrate new features into the system with the required quality, cost and time [17].

We propose an approach for repairing an eroded software product line. As demonstrated in [17], by applying architecture regeneration, a strong cohesion of the software components can be achieved improving maintainability and extensibility of the software. Furthermore, we give a methodology for a long-term manageable, plannable, and reusable software product line architecture for automotive software systems.

To describe the PLA and PA in our methodology, we use a domain specific architecture description language (so-called EMAB) which we have introduced in a previous work.

The paper is structured as follows: The related work is discussed in Section 2. In Section 3, we propose our methodology for recovering an eroded ECU software PL from a set of PAs for long-term further development. Section 4 concludes.

2. RELATED WORK

To the best of our knowledge no continuous overall development cycle for automotive software product line architectures exists. Several aspects of our process are already covered in literature:

In [2] de Silva and Balasubramaniam provide a survey of technologies and techniques either to prevent architecture erosion or to detect and restore architectures that have been eroded. However, each approach discussed in [2] refers to architecture erosion for a single PA, whereas architecture erosion in software product lines are out of the scope of the paper.

The initial phase architecture recovery and discovery within our approach is closely related to the domain of software product line extraction (cf. [5], [10]). In numerous publications, Bosch et. al. address the field of product line architecture [3], software architecture erosion [19], and reuse of software artefacts [1].

Automotive manufacturers have to cope with the erosion of their ECU software. Holdschick [9] addresses the challenges in the evolution of model-based software product lines in the automotive domain. [12] propose a systematic approach to architecture regeneration. In [17], a methodical and structured approach of architecture restoration in the specific case of the brake servo unit (BSU) is applied. Thiel and Hein [18] propose product lines as an approach to automotive system development. Flores et. al. [4] explain the application of 2GPL (Second Generation Product Line Engineering) - an advanced set of explicitly defined product line engineering solutions - at General Motors.

Several approaches deal with the topic reuse of software components in the development of automotive products [6], [16]. In [6] a framework is proposed, which focuses on modularization and management of a function repository. Another practical experience describes the introduction of a product line for a gasoline system from scratch [16]. However, in both approaches a long term minimization of erosion is not considered.

3. METHODOLOGY

Our methodology is divided into three phases:

- **Recovery & Discovery**: As a major challenge, we have to deal with product line development where a set of modules constitutes a basis for deriving a huge number of products. Therefore it is necessary to know about the derivable PAs from a given PLA. Two PLAs are distinguished: Current derivable PLAs are captured by the Actual PLA (APLA). All planned PAs for future development are captured by the Target PLA (TPLA). In this phase we recover an APLA and discover a TPLA candidate.

- **Plan & Evolution**: Further development, e.g. of new functions, may require changes on the APLA resulting in a new TPLA. However, changes to the APLA have to be done and planned carefully as each change of a module may effect a huge number of products.

- **Realize & Minimize**: During product development, we minimize PA erosion by architecture compliance monitoring, i.e., checking architecture conformance using architecture rules derived from APLA.

In the Recovery & Discovery phase a subset of all existing PAs are selected first (see step a) in Figure 2). From this set of PAs a TPLA candidate and a number of APLA candidates are derived independently. The TPLA candidate is discovered by domain experts (see step e) in Figure 2). To recover PLAs candidates a great number of recovery approaches and techniques are existing in literature. We choose a data mining and a functional analysis approach (see steps b) and c) in Figure 2). For the data mining based technique the so called Spectral Analysis of Software Architecture (SPAA) [13], [14] approach is used. It is a generic approach to cluster software elements by their dependencies. The functional analysis was performed by domain experts.
Due to the fact that in the automotive domain software modules are developed in the context of product instead of the product line architectures, a mapping between a PLA and reused and further developed modules is not available. Therefore the APLA is created which describes the hierarchy of module groups and dependencies as the actual used product independent architecture. Moreover target quality requirements that need to be reached as soon as changes are applied to the APLA are kept in the initial TPLA. The resulting architectures (APLA, TPLA) after the first phase are illustrated on the top left of Figure 3.

After the initial creation of the APLA and TPLA candidate in the Recovery & Discovery phase, which is only performed once as an entry point for the management cycle, both are used as input for the following phase Plan & Evolution. The phase Plan & Evolution concerns the two top quadrants of Figure 3. "Act" describes the application of changes to the APLA that results in the TPLA after the successful quality compliance check. The APLA and the desired TPLA candidate from the phase Recovery & Discovery serve as input for this phase. Phase Recovery & Discovery is applied once only. Hence, in further development cycles, new desired TPLA candidates have to be planned in phase Plan & Evolution: New functions may require changes on the APLA resulting in a new desired TPLA. Objective of the phase Evolution & Plan is to identify a new TPLA with a chain of change operations (CO) to transform the APLA into a TPLA. An attempt is made to approximate the desired TPLA.

We propose a cycle of four consecutive steps executed iteratively to identify the set of COs to transform the APLA into the TPLA. As depicted in Figure 4 in the first step (Development) one change operation is applied to the APLA resulting in a "potential" TPLA candidate $S'_0$. Next, the TPLA candidate is measured by metrics (Measurement) and the result is compared with the result of the measurement of the APLA (Evaluation). If the TPLA candidate is better than the APLA, test cases are executed (Inspection). In case those are inspected positive the CO is discarded and another CO is applied to the APLA. If the test cases are inspected positive and the TPLA candidate fits the expectation of the evolution plan, the TPLA candidate is set as the new TPLA. Otherwise, in the next step another CO is applied to the TPLA candidate. The output is a TPLA with a chain of COs.

In the automotive domain software sharing is done in the context of PAs. To minimize the PA erosion is the task of the phase Realize & Minimize. The bottom two quadrants of Figure 3 illustrate the application and development of PA changes as "Do" and the check between the PA and the APLA as "Check". The "Check" step follows the "Do" step and compares the structure of APLA and the appropriate PA using rules. These rules are extracted from the APLA via Architecture Checker (ArCh) [7], [8] in the "Check" step and then handed over the "Do" step.

In the case of rule violations architecture experts have to identify whether changes need to be applied to the APLA or the PA to ensure the compliance. If changes are only applied to the PA they can be executed in the "Do" step. If changes to the APLA need to be applied then the iteration between the steps "Act", "Plan", "Do", "Check" are processed until the compliance is reached.

4. CONCLUSION AND FUTURE WORK

We presented a methodology for a long-term manageable, plannable, and reusable product line architecture for automotive software systems. Our methodology aims at minimizing erosion by checking certain architecture quality criteria. As a first step of our methodology, we recovered a product line architecture as a means to measure architecture conformance within further developments, as well as a means to plan changes on the product line architecture.
Further quality criteria have to be defined as rules and constraints for the product line architecture. Hence, for all further developments we aim at ensuring that all rules are fulfilled and the erosion is limited to a minimum. However, we can not guarantee that a product line architecture for each set of given product architectures exists. As a consequence, separations into distinct product line architectures may be necessary, e.g., in the case of an engine control unit, one product line architecture for diesel engine variants and one product line architecture for gasoline engine variants.

The presented methodology was derived and established in the context of an industrial automotive project. A proof of concept is an essential part which is currently work in progress. Currently we aim to meet three different challenges. The first one is to identify functional aspects of the product independent architecture from the domain experts which concerns the creation of useful module groups for APLA from the different APLA candidates. The used techniques to recover an APLA candidate like the data mining based approach for itself are evaluated in different projects and domains [13], [14] and [15].

The second challenge is to realize a tool-chain which enables the architecture-description (here the EMAB is already developed as an architecture description language) of the different architectures (APLA, TPLA, PA), the measure and evaluation of quality attributes as well as the integration of the ArCh framework [7], [8].

The third challenge is the applicability of the tool-chain in a real world industrial project which targets the erosion management in the field of automotive Product Line Architectures and Product Architectures used for automotive software sharing between the OEM and TIER1.

5. REFERENCES