A Model-Based Approach to Innovation Management of Automotive Control Systems

Mario Gleirscher, Andreas Vogelsang
Institut für Informatik, Technische Universität München, Germany
{gleirsch,vogelsan}@in.tum.de

Steffen Fuhrmann
BMW AG, Germany
steffen.fuhrmann@bmw.de

***Abstract*—Innovation management of automotive control systems is a challenging issue not least because of the need to handle short iterative life cycles and families of these systems. After collecting experience in control engineering from a three year collaboration in the automotive domain, we conclude that innovation management is only weakly aligned with feature and platform development. This situation makes it difficult to assure feasibility of innovations and to identify potentials for innovations.**

We present a novel approach to integrating innovation management with requirements and technology management by using behavioral system models. We investigate requirements-based and technology-based innovation. To the best of our knowledge, this is the first approach to using such models for managing feature and platform innovations in an automotive company. We discuss an example to illustrate how our approach can be applied.

***Index Terms*—Product innovation management, requirements engineering, embedded software, control system, mechatronics, automotive, system modeling, technology management, roadmap.**

I. Introduction

Many innovations in automotive control systems have been achieved by intensive use of software in control technology. To provide the desired vehicle features, automotive companies have technology platforms, which consist of mechanical parts, physical and control devices, their configuration, and the software that runs on the control devices. To provide hundreds of features, this software comprises several million lines of code distributed over a large number of electronic control units [1]. Such platforms are key assets for competitiveness because they enable new features in quick development cycles and at high quality. This situation challenges the *management of innovations* of features and the technology platform.

Innovation management pertains "all the activities in the process of idea generation, technology development, manufacturing and marketing of a new (or improved) product or manufacturing process or equipment" in a company [2]. *Innovation* can be seen as an iterative process initiated by a technology-based invention and leading to new markets or service opportunities [3]. The nature and course of innovations is explained by two theories, the *linear model of innovation* [4] and *open innovation* [5]. The linear model focuses on the initiation of innovations through the customer (market-pull) or through engineering research (technology-push). Open innovation defines further motivators for innovations. In the present article, we look at product innovation management. Hence, we use the terms *innovation management* to indicate the corresponding part of the product development process of a company and *innovation* to signify the result evolving from innovation management.

Current Practice: The exploitation of knowledge about innovations is decisive for innovation management, especially the estimation of the impact of innovations on features and the technology platform. Similar to [3], we decided to distinguish between *requirements-based and technology-based innovations*. Current approaches propose processes for the development and integration of both kinds of innovations: For example, the Capability Maturity Model Integration (CMMI) guides the selection and rollout of innovations [6]. CMMI instructs large steps and does not explain how to design and assess product innovations. Standards like SPICE [7], ISO 9001 [8] and ISO 14764 [9] even neglect specific tasks for innovation management. They refer to the more general term *change management* of processes and technologies. In contrast, systems engineering supports innovation management by evolutionary development techniques, architecture documentation and technology management, requirements and configuration management, creativity techniques, system modeling and value engineering [10], [11].

Problem Statement: From the challenges we perceive in innovation management of automotive control systems, we derive two research questions:

***RQ1**: How to capture models of features and the technology platform to identify innovations?* The approach performed by many automotive companies to develop control systems, particularly their embedded software, is focused on features. For each vehicle domain (e.g. driving dynamics, chassis, interior), responsible departments and roles design and implement the part of the control system that is related to their features. In this respect, these departments and roles are isolated from each other. In recent years, embedded software of automotive control systems comprises strongly coupled architectures [12]. This lack of overview of the architecture of the control system impedes the identification of innovations of embedded software,

978-1-4799-6357-7/14/$31.00

IWSPM 2014, Karlskrona, Sweden

hardware devices and mechanical parts in a flexible and extensible technology platform.

***RQ2**: How to evolve features and the technology platform to accomplish innovations?* Features are based on the capabilities (e.g. physical performance) of the currently or potentially available technology platform. It is hard to determine, how identified feature innovations drive the evolution of the technology platform and how identified platform innovations drive the evolution of features. Low collaboration of innovation, requirements and technology management as well as design and implementation of features impede strategic innovations and, thus, complicate market strategies. Moreover, [13] emphasizes the challenge of *customer integration for system families* (also known as product families) and *system family design support* for all stakeholders including architecture and platform modeling and evolution in accordance with changes in customer requirements.

Contribution: In this paper, we propose a model-based approach to innovation management of automotive systems. We describe three models of an automotive control system: a *feature hierarchy*, a *platform service hierarchy* and a *platform component model*. The feature hierarchy describes vehicle features offered to the customer. The platform service hierarchy and the platform component model describe the capabilities and constituents of the technology platform. First, we argue that the three models address **RQ1** and support the identification of feature and platform innovations. Second, we address **RQ2** by applying these models to accomplish innovations of both kinds: (A) requirements-based innovations, which introduce new and innovative features that may require new or adapted platform services to be provided by the technology platform, and (B) technology-based innovations, which introduce new and emerging technologies that may enable a number of new or enhanced features. We illustrate the two activities by an example based on our experience gained in a three year collaboration with industry.

Outline: The remainder of the paper is structured as follows: Section II introduces our model of automotive control systems. Section III describes the two innovation activities and how they are supported by this model. Section IV illustrates our approach through an example that is discussed in Section V. In Section VI, we relate our approach to existing work on innovation management and platform development before we conclude and outline future research directions in Section VII.

II. System Specification

The system model we employ to specify feature and platform innovation consists of two layers with three models. An overview is given in Fig. 1. The *feature layer* is specified by the *feature hierarchy* model. This model describes the features a vehicle offers to the customer. Advancements in this layer represent feature innovations. The *platform layer* represents the technological capabilities and constituents of a platform. It is specified by two models, the *platform service hierarchy* and the *platform component model*. These models describe the capabilities in terms of platform services and the corresponding

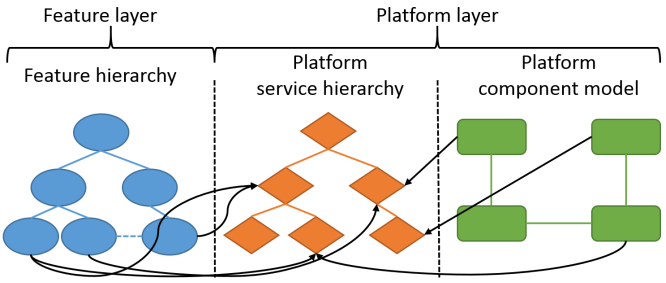

Fig. 1. Overview of the system specification models

platform components. Trace links between the three models describe, which features are specified on the basis of which platform services and which platform components provide the platform services. On the basis of this system specification, we are able to precisely document innovation activities and reason about them.

Foundations: The three models are based on earlier work [14] and the formal concepts of FOCUS [15], a theory for *discrete-event modeling of distributed, interactive systems*. Particularly, we use this theory as a basis to describe features, platform services, and platform components through a syntactic interface and an associated interface behavior that can, for example, be described by state machines.

The following terms apply to each of the models and are used throughout this paper: An *automotive control system* is a technical system comprising vehicle mechanics, electrics, electronics, and control software. *Requirement candidates* are pieces of information specifying innovative (future, potentially realized) parts of features or the technology platform.

Variation points are parameters that introduce underspecification, flexibility, and variability into a system specification. They can be used to encode simultaneously buyable product variants (*feature variation point*), to functionally distinguish alternatively planned or built technology variants (*technology variation point*) or to represent different versions of a feature consecutively rolled-out or planned in terms of a *roadmap* (*evolution point*).

A. Feature Hierarchy

The feature hierarchy is a model representing the *feature layer*. We describe the methodology of feature hierarchies by means of the main concepts and their usage:

A *Feature* (also *customer, product or user function*) is a behavioral property of a system observable (and influenceable) by its user. A feature can specify functional goals and requirements of users, it can represent use cases or model complex behavioral system properties. An example of a feature is *safety braking* that performs an emergency braking in case of suddenly appearing objects. We label a feature by a name and describe its behavior by means of a state machine.

In a *feature hierarchy*, features are hierarchically composed to a functional specification of a (multifunctional) system [16]. Motivated by *UML/SysML use case diagrams* [17], we distinguish between *parallel composition* (solid arcs in Fig. 3)

and *fragmentation* (dashed arcs in Fig. 3). The use of parallel composition aims at reducing complexity and division of labor by breaking down a specification to a set of features. The use of fragmentation aims at a comprehensible and maintainable representation of a state machine that describes the behavior of a single feature. The state machine of a fragmented feature is derived by superimposition [18].

Feature variation points allow introducing variability into the hierarchy. Feature variation points can be introduced both for parallel composition (labeled with $variation$ in Fig. 4) and for fragmentation (labeled with $extend$ in Fig. 3). We call a hierarchy without variation points *specific*, and *generic* otherwise. A generic feature hierarchy models a set of specific hierarchies. Together, composition and variation provide possibilities for reuse in large sets of specification models.

Changes/transformation: based on the notions of feature, composition and variation, the feature hierarchy assigns a clear meaning to changes of the hierarchy such as introduction, update and deletion of (sub)features.

Versions/evolution of features or a feature hierarchy can be tracked (a) by putting the corresponding set of changes under version control or (b) by using evolution points.

B. Platform Service Hierarchy

The platform service hierarchy models the capabilities of a platform. It establishes a connection between the features and the constituents of the platform on which the features shall be realized. Features are specified on the basis of *(enabling-)services* that the technology platform provides. A platform service is provided by a platform either directly by a material component, such as a sensor or an actuator (we then speak of a *basic platform service*), or by an immaterial software component that is part of the platform and may, for example, aggregate a number of basic platform services. The platform services of a platform are structured in the platform service hierarchy. The leafs of the platform service hierarchy represent the basic platform services, whereas the inner nodes represent platform services provided by software components of the platform building up on the basic platform services. The platform service hierarchy can be compared to an *application programming interface (API)* of a platform, i.e. features can be specified by referencing the platform services of the platform service hierarchy. In the following, we will describe the most important concepts in detail:

A *platform service* represents data, information or functionality that is provided or can be processed by a technology platform. A platform service has a name and is characterized by a set of *platform service attributes*. An example of a platform service is an *object recognition*, which describes the platform capability to recognize appearing objects. This platform service may be used to specify innovative features.

A *platform service attribute* provides additional information for a service. A mandatory platform service attribute, for example, is a *description*. Further platform service attribute examples are unit, resolution or derivation of the information provided by the platform service.

A *platform service hierarchy* is a hierarchical structure of platform services. Two platform services are related in this hierarchy if one platform service builds up on the other. A platform service *object recognition*, for example, might build up on two platform services *optical image right* and *optical image left*.

Since platform services provide an abstract view on the capabilities of a platform, the *evolution of platform services* is always coupled with an evolution of the platform, i.e. an evolution of a material or an immaterial component.

C. Platform Component Model

The platform component model describes and structures the material and immaterial parts (e.g. physical devices and software components) of a technology platform. These platform components provide the platform services that are specified in the platform service hierarchy. The most important concepts of the platform component model are described in the following.

A *platform component* is a *material* or *immaterial* part of a technology platform. It has a name and a set of *platform component attributes*. An example of a platform component is an *ultrasonic sensor* (material) or a *sensor fusion software component* (immaterial).

A *platform component attribute* provides additional information for a component. A mandatory component attribute, for example, is a *description*. Further component attribute examples for an electronic control unit are clock rate, size, or memory.

A *material component* is a physically present element of a technology platform. Examples are sensors, actuators, computing units, and communication buses.

An *immaterial component* is an element of a technology platform, which does not have a physical representation. Immaterial components have no physical properties such as weight or size. In this paper, we only consider software components as immaterial components.

The *platform component model* contains all material and immaterial components of a platform and relates them to each other. Possible relations between components are *consists of*, which specifies that a component consists of a set of other components, *communicates with*, which specifies that two material components communicate with each other, and *runs on*, which specifies that an immaterial component is executed on a material component.

The relations between components may be subject to constraints, for example, a computing unit, in general, is only allowed to *communicate with* a communication bus and not directly with another computing unit.

Similar to the feature and platform service hierarchies, the platform component model can contain variation points to specify platform variants. In this model, material and immaterial components may evolve (e.g. the integration of a new sensor or the implementation of a more precise algorithm). An evolution of components affects the platform services that are provided by the platform.

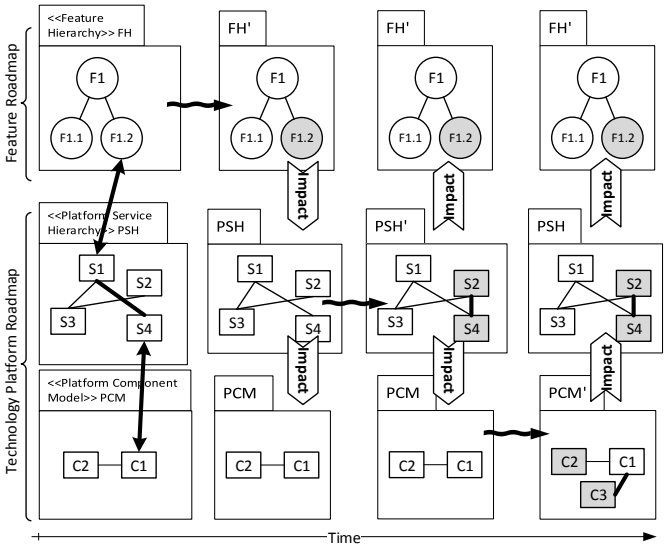

Fig. 2. Traceability between the evolving models of the system specification.

D. Integrated View

The integration of the three models is done by establishing trace links between the elements of different models. A feature in the feature hierarchy is traced to all platform services by which the feature is specified. For example, a *safety braking* feature may be traced to an *object recognition* platform service because the feature relies on the platform capability to recognize appearing objects. A platform service in the platform service hierarchy is traced to all platform components that are necessary to provide the platform service. For example, the *object recognition* platform service may be traced to a *sensor fusion software component* that aggregates several sensor values to indicate an appearing object.

Fig. 2 shows the three models together with their trace links in four stages of evolution. By means of this traceability, it is possible to assess the impact of innovations in each of the models. An evolution of a feature, for example, has an impact on the platform service hierarchy because platform services may also have to evolve in order to cope with the evolved feature. This may impact the platform component model as well because platform components may have to evolve in order to provide the necessary platform services. An evolution of the platform, in turn, may enable an innovative feature.

III. Innovation Activities

We consider two activities in innovation management:

A. Requirements-based innovation

B. Technology-based innovation

In requirements-based innovation (also known as top-down or market-pull product innovation [2]), the task is (i) to evolve and envisage potentially usable (foreseeable) functionality including system goals, requirements, and use cases, (ii) to model the variability of features and feature hierarchies by the help of *feature variation points*, and (iii) to derive *requirement candidates* for the technology platform (e.g. platform service candidates, platform component candidates) through application of platform-based tailoring criteria (see TABLE I, Cat. Tailoring).

In technology-based innovation (also known as bottom-up or technology-push product innovation [2]), the task is (i) to evolve the built-in and envisage potentially available (foreseeable) technology platforms, (ii) to model the variability of the technology platform by the help of *technology variation points*, and (iii) to derive *requirement candidates* for the feature hierarchy (e.g. use case candidates, feature candidates) through application of feature-based tailoring criteria (see TABLE I, Cat. Tailoring).

For both activities, the notion of an *ultimate instance* is important. The ultimate instance (of a feature or a platform component) describes the envisaged maximum stage of evolution. *Roadmapping* describes the activity of defining intermediate stages to be achieved on the way to the ultimate instance. Ideally, the roadmaps for the features and the platform are synchronized in a way that on the one hand, a platform's capacity is fully utilized and on the other hand, features can be specified based on appropriate platforms. Roadmapping and its integration into our model-based approach is guided by decisions based on criteria of four categories. TABLE I summarizes these categories and provides examples.

A. Requirements-Based Innovation

Requirements-based innovations are usually motivated by newly elicited requirements or needs that originate from market research (e.g. customer surveys). The implementation of these requirements may result in additional requirements for the necessary technology platform. For example, the inclusion or enhancement of a *vehicle reversing feature* may result in additional human-user interface elements and the reuse of existing radar and automated steering subsystems. The following steps can be seen as part of a requirements-based innovation activity.

Modeling Features (i): The purpose of this step is to evolve an existing feature hierarchy at a proper level of abstraction based on the modeling concepts introduced in Section II-A. We recommend the following procedure:

1) Capture information for the addition of an innovative *feature* from existing requirements documentation (stakeholder goals, market analyses, high-level feature decisions, informal use cases, etc.). This process is guided by *decomposition criteria* (TABLE I, Cat. Decomposition), which help to find an appropriate structure in the feature hierarchy.
2) Create an *ultimate instance* of this feature and specify behavioral details as a state machine (cf. [16]).

Modeling of Feature Variability (ii): The goal of this step is to identify *feature variation points* motivated by an innovation activity and to include them into the feature hierarchy. We recommend the following procedure:

1) Refine the ultimate instance into several stages of *evolution* by applying *evolution criteria* (TABLE I, Cat. Evolution).

TABLE I
CATEGORIES AND EXAMPLES OF CRITERIA FOR MODEL-BASED INNOVATION MANAGEMENT

Category	**Objective**	**Examples of Feature-based Criteria** in requirements-based innovation (Section III-A)	**Examples of Platform-based Criteria** in technology-based innovation (Section III-B)
Decomposition	guidance for feature and platform modeling in general, cf. [19].	stakeholder goals; high-level feature decisions; informal use cases; market analysis	research activities; supplier information; market analysis
Variation	guidance for variability modeling.	features specific to a system family, type or purchase; slightly differing stakeholder goals, use cases or driving situations; reengineering of existing and simultaneously sold features	capturing dependencies such as exclusion resp. requirement of other technologies while configuring the system resp. integrating an innovation
Evolution	guidance for variability modeling for innovation or evolution.	distinguish between basic and comfort features; low and high interactivity; coupling with technology change in consecutively sold product generations	improving quality of the product; reducing production and/or development costs; technological producibility of the innovation
Tailoring	guidance for requirements derivation and assessment.	current or future realizability using the available technology platform or technologies from the market; cost/ benefit ratio; preservation of functional independence for safety reasons	number of use cases or features that are improved or even created; acquisition of new customer segments; the extension of the functional range or the perceivable improvement of feature behavior

2) Identify further *feature variation points* by considering additional *variation criteria* (TABLE I, Cat. Variation).

Derivation of Platform Requirement Candidates (iii): The purpose of this step is to perform design at a high level of abstraction and to establish traceability from the result of this process to both the existing and the envisaged platform layer. We recommend the following procedure:

1) Identify *requirement candidates* for
   a) *platform services* (i.e. platform service changes and new platform services within the platform service hierarchy)
   b) *platform components* (i.e. platform component changes and new platform components within the platform component model)
2) Assess each platform requirement candidate by applying platform-based *tailoring criteria* (TABLE I, Cat. Tailoring) to decide on a, usually reduced, set of requirements to be realized.
3) Specify *platform requirements* by introducing the new platform services and platform components as well as variants or versions of existing ones. Establish trace links between the features and these platform requirements (see Fig. 2).

In Section IV-A, we will exemplify the application of this procedure to the inclusion of a vehicle reversing feature.

B. Technology-Based Innovation

Technology-based innovations are motivated by new or emerging technologies, which are reflected by specific platform components and platform services. We refer to them as "bottom-up innovations" because of their impact on and enhancement of features. For example, qualifying a radar sensor for recognizing walking persons may enable the integration of an emergency brake assistant for pedestrians as a feature of the system.

Modeling Product Technology (i): As described in Sections II-B and II-C, we model technology by means of two models: the platform service hierarchy and the platform component model. For technology-based innovations, both models must initially be set up for the built-in product technologies. They enable the evolution of platform services and platform components. We recommend the following procedure:

1) Gather information about the current system to add innovative technologies: use the platform service hierarchy and the platform component model as existing specifications of the currently built-in technologies.
2) Identify potentially available technologies for
   a) *platform services:* Define an *ultimate instance* of the platform service and specify the corresponding *platform service attributes*.
   b) *platform components:* Define an *ultimate instance* of the platform component and specify it with *platform component attributes*.

   The identification might be supported by research activities, supplier information or market analysis (TABLE I, Cat. Decomposition).
3) Integrate the ultimate instances of platform services and platform components in the models.

Modeling Technological Variability (ii): Innovations introduce technological variability into the models of the platform layer. Here, variability is used to represent intermediate stages of evolution and alternative solutions. We express the variability by *technology variation points* and integrate them into our models of the technology platform. We recommend the following procedure:

1) Refine platform services or platform components by intermediate stages of evolution for closing the gap between the ultimate instances and the built-in technologies. We apply *evolution criteria* to derive stages of evolution (TABLE I, Cat. Evolution).
2) Model additional variability relations between potentially available and built-in technologies by applying *variation criteria* (TABLE I, Cat. Variation). An innovation might exclude other technologies or depend on them.

Having variability modeled, we see which possibilities for the improvement or evolution of the platform exist and how the features of the control system may be affected.

Derivation of Feature Requirement Candidates (iii): After modeling built-in and potentially available technologies of a platform as well as the variability in it, we evaluate the benefit of identified innovations with respect to the features they enable. We recommend the following procedure:

1) For platform component innovations, derive platform services that are provided by the platform components.
2) For platform service innovations, derive *feature requirement candidates* that are addressed by the platform innovations.
3) Assess feature requirement candidates by applying feature-based *tailoring criteria* (TABLE I, Cat. Tailoring) to decide on a, usually reduced, set of the platform services and components to be utilized. A platform service innovation, for example, has a benefit if the application of tailoring criteria derives at least one use case or feature that is improved or even created through the implementation of the platform innovation. A platform component innovation must enable a platform service innovation, which provides the mentioned benefit.
4) Specify *feature requirements* by introducing new features or feature variations, or by adapting existing features within the feature hierarchy. Establish trace links between the technology-based innovations and feature requirements (see Fig. 2).

In Section IV-B, we will exemplify the application of this procedure to the inclusion of a new ultrasonic sensor.

IV. Example

For evaluation, we applied our approach to a feature example from the driving assistance domain at the BMW Group. The data shown and discussed throughout this section has originally been elaborated during our collaboration and was documented in detail in two consecutive, internal project reports. Due to reasons of nondisclosure, we will only show simplified examples in order to illustrate our approach. However, we will refer to the original application in the discussion section.

We used SysML [20] as a language and MagicDraw[1] as a tool to visualize the three models. However, our approach is not restricted to these utilities.

A. Application to Requirements-Based Innovation

According to the procedure described in Section III-A, we show how to develop a requirements-based innovation of a vehicle feature called *vehicle reversing*.

First, we consider the current feature hierarchy (light gray and dark gray parts of Fig. 3). The ultimate instance of the *vehicle reversing* feature, including *object collision warning*, *object distance display* and an integration with an *automated braking assistant*, is depicted in Fig. 4.

Second, based on the ultimate instance of the feature hierarchy for the reversing feature (denoted by S), we decide to derive three refinements (denoted by S', S'', S'''):

S' in Fig. 5 introduces an evolution point, which *constrains* the hierarchy to include *at most one* of the sub-features *object distance display* or *object collision warning*. We applied the evolution criterion "high vs. low interactivity". Note that both sub-features now incorporate an *automated braking assistant*. The exclusive choice among the two sub-features is not directly visualizable in SysML use case diagrams and hence left out in Figures 3 to 7.

S'' in Fig. 6 introduces a further evolution point by using fragmentation for the design and implementation of an *automated braking assistant* for each of the above sub-features. We applied the tailoring criterion "cost/benefit ratio".

S''' in Fig. 7 introduces the third evolution point by *extracting* a comfort variant called *object distance display for unseizable objects* from the sub-feature *object distance display*. We applied the evolution criterion "basic vs. comfort". The behavior specification of this variant is shown in Fig. 8.

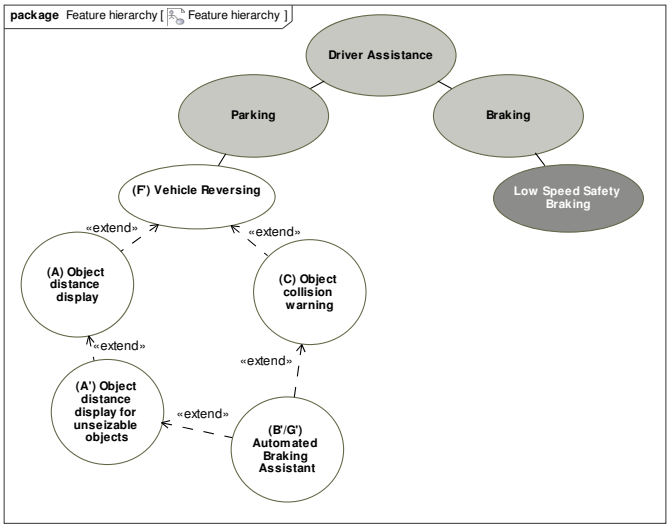

Fig. 3. Excerpt of a feature hierarchy for a vehicle

The three refinements identified by requirements-based innovation of the vehicle represent three alternative *feature roadmaps*. We select the last refinement S''' to be included into the overall feature hierarchy (white part of Fig. 3). The evolution points of this hierarchy can be interpreted and conducted as a roadmap for feature implementation.

Third, we need to derive platform requirement candidates. In our case, we annotate each of the features with a reasonably abstract state machine. For example, for the feature hierarchy S''', the state machine shown in Fig. 8 represents functional requirements imposed by the extracted sub-feature *object distance display for unseizable objects*. To reify these requirements, we establish trace links to the platform service hierarchy and to the platform component model (see Fig. 11): This includes a $uses$-link to an *environment monitoring* platform service (SysML package *information acquisition*) to measure object distance which leads to a further $provides$-link to a *sensor fusion* software component.

[1] http://www.nomagic.com/products/magicdraw.html

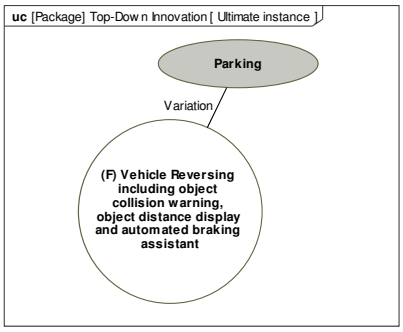

Fig. 4. Ultimate instance S of the *vehicle reversing* feature

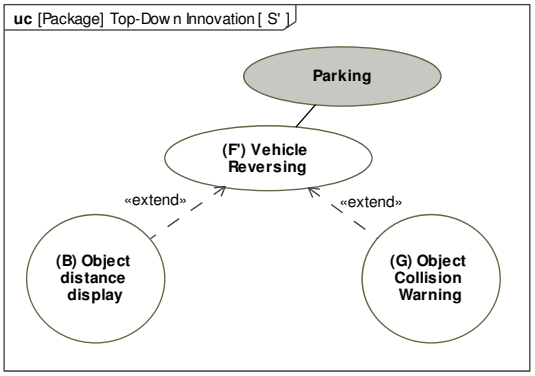

Fig. 5. Evolution points in the refined feature hierarchy S'

B. Application to Technology-Based Innovation

As described in Section III-B, technology-based innovation is triggered by a new or modified technology platform, which enables the realization of new or modified features that provide the user with additional value.

In order to illustrate technology-based innovation in our approach, we model the following situation: A supplier offers a new ultrasonic sensor that has a range of 5 meters. In the current version of a vehicle, there is an older version of that sensor mounted, which has only a range of 2 meters. According to the procedure proposed in Section III-B, we integrate the new sensor into the existing platform component model of the vehicle (see Fig. 9).

In the example, we model the new sensor as a variation, which *improves* the existing ultrasonic sensor. The improvement is characterized by the platform component attribute *range*. The new sensor is a further intermediate step on the way to the desired *ultimate instance* of an ultrasonic sensor, which is able to monitor a range of 20 meters.

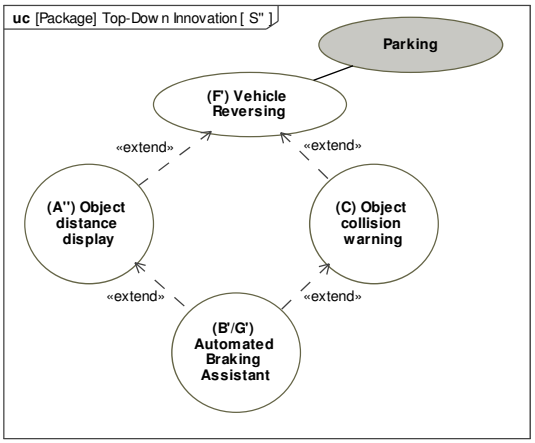

Fig. 6. Evolution points in the refined feature hierarchy S''

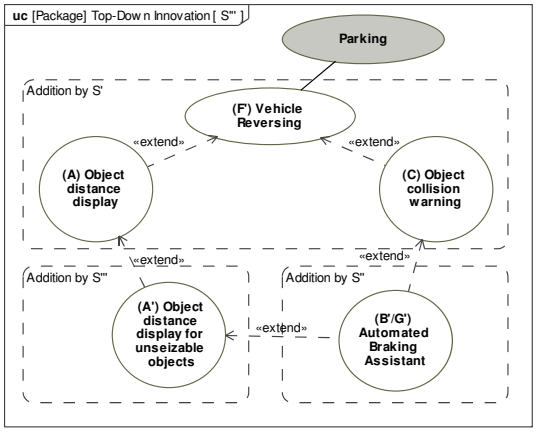

Fig. 7. Evolution points in the refined feature hierarchy S''' (includes the final set of evolution points for the feature roadmap)

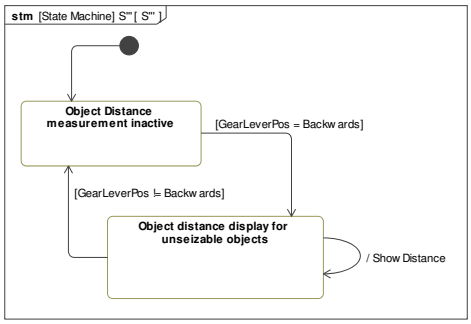

Fig. 8. State machine for sub-feature *object distance display for unseizable objects*

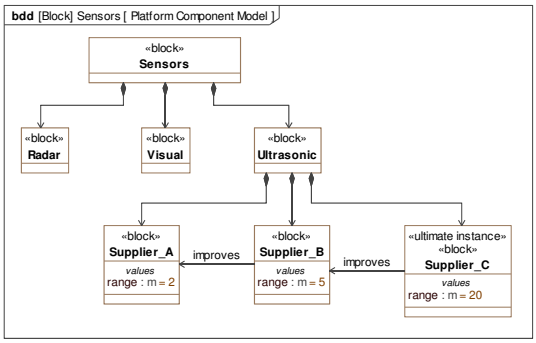

Fig. 9. Platform component model with extended sensor and ultimate instance.

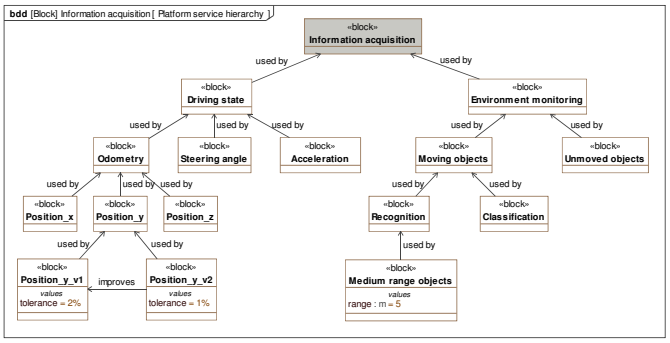

Fig. 10. Platform service hierarchy with platform services enabled by a new sensor.

The integration of the new sensor also has an impact on the platform services offered by the platform. We model this impact in the platform service hierarchy (see Fig. 10). In the platform service hierarchy of the example, the new ultrasonic sensor has an impact on two platform services. On the one hand, the sensor decreases the tolerance of the *position_y* value provided by the platform and on the other hand, we introduce a new platform service named *medium range objects*. This new platform service provides the detection of moving objects in a medium range in front of the car (e.g. a rolling ball on the street). The platform service is provided by the platform and can be used in order to realize new features. More specifically, this platform service is provided directly by the material sensor without any additional software necessary. Therefore, the new platform service is modeled as a basic platform service (see Fig. 10).

A functional requirements candidate based on this technological innovation might be a feature that prepares a full braking in case of an object detected within a medium distance from the vehicle. We assess this requirement candidate and consider it as beneficial for the functional safety of the vehicle. Hence, we introduce a new feature in the feature hierarchy named *low speed safety braking* (see Fig. 3).

As a last step, we establish trace links to the feature hierarchy (see Figures 9 to 11): The $improved$ ultrasonic sensor directly $provides$ a new platform service called *medium range objects*. The introduced feature *low speed safety braking* $uses$ this platform service as well as the platform services *environment monitoring* and *braking moment*.

V. Discussion

Strengths and Limitations: From the experiences gained during our collaboration, we conclude that our approach is aligned with the current engineering processes at the BMW driving assistance domain. In the final project workshop, we obtained positive feedback concerning the use of the feature hierarchy. In particular, the use of state machines helps in consistently establishing the fragmentation and parallel composition of features.

However, the issue to carefully argue for the additional specification effort via a clear cost-benefit analysis is still open. Moreover, we saw that neither the used tool infrastructure nor off-the-shelf tools known to us show enough capabilities to implement our approach appropriately. These capabilities include capturing the proposed models, feature and platform evolution and traceability as discussed in Section II-D.

The presented approach is motivated by issues arising in the automotive domain. However, we think that it can be a reference for other domains such as avionics, chemical process plants, industrial automation, healthcare and rail/ship transportation.

General Remarks: In our approach, we refer to the innovation activities defined by the *linear model of innovation*. The "market-pull" is addressed by our requirements-based innovation activity in Section III-A, "technology-push" is equivalent to our technology-based innovation activity in Section III-B. However, the linear model of innovations only characterizes the innovation activities and does not provide foundation for system modeling. In contrast, open innovation defines more sources for innovations in order to extend the innovation activities. Compared to *open innovation* [2], we focus on model information that can be traced back to requirements and technological characteristics of the system. Most importantly, our approach aims at supporting careful decision making for an optimal realization of features as well as utilization of technology for system family innovation as required by automotive companies.

VI. Related Work

We discuss *frameworks, platform standards and process models* related to our contributions.

We found several papers on innovation management in a software product line (SPL) context: Männistö [21] studies the interdisciplinary modeling of system families with emphasis on the evolution of system family descriptions and individual systems. His framework captures evolution as described in Fig. 2. As opposed to our approach, he applies structural rather than behavioral modeling without regarding innovations. Andreasson and Henfridsson [22] discuss the concept of *digital differentiation* to describe innovations based on SPLs. The authors studied relationships between software and mechanical engineering regarding the system architecture and the development process of an automotive company. They observed that hardware manufacturing constrains the introduction of a product-line approach. Their approach lacks details about system modeling and a procedure to follow. Böckle [23] proposes measures to prevent obstacles for innovation in product line organizations. He regards architectural and process aspects of innovations and relates the extensiveness of an innovation to performance indicators such as return-on-investment and success rate. The author describes the impact of innovations using a variability model. He does not provide a behavioral system model and an example of how the measures guide the decisions on innovations. Hanssen and Fægri [24] describe a case study on introducing open innovation to a software company that practices SPL engineering in a long-term fashion and agile development in a short-term fashion. Their case study could not confirm two known drawbacks of agile methods:

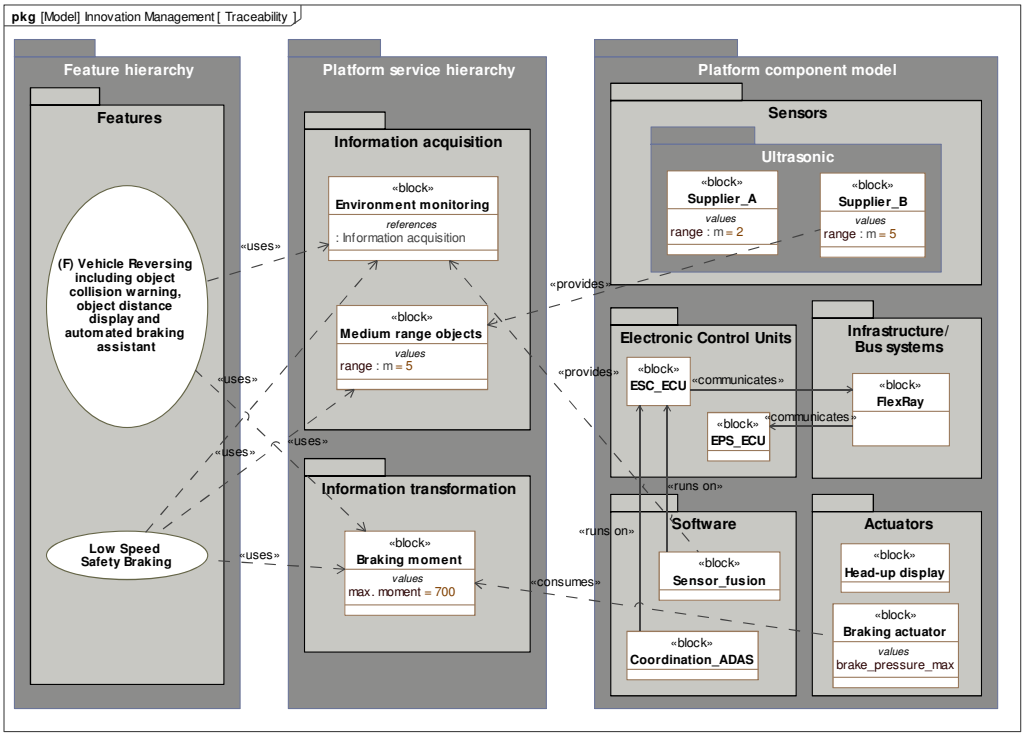

Fig. 11. Trace links connecting the entities of the different models.

loss of architectural quality of the product line and reduced applicability to large-scale projects. Their approach does not apply system modeling. As opposed to product innovation in particular, they discuss innovation in the software life cycle in general and do not discriminate between innovations and evolutions. Trujillo et al. [25] discuss the tracking of changes for evolving a family of software systems described by a feature variability model. These changes are tracked both, in the model and in the code. This setting matches our feature hierarchy and the platform component model. On the one hand, their approach is independent from innovation management but on the other hand restricted to plain software development.

Our method is aligned with the modeling concepts and abstraction layers customary to *architecture description frameworks and languages* such as AAF [14], AADL[2] [26], COLA [27], or EAST-ADL [28], *modeling notations and data exchange standards* such as FODA feature trees [29], ISO 10303 [30] AP-214/233, UML [17], or SysML [20], and *technology platform standards* such as AUTOSAR[3], or IMA [31]. However, we could not find approaches with a focus on application of these concepts for product innovation management. The same holds for application of system modeling for innovation management in a CMMI [6] context.

Haskins and Forsberg [10] regard *value engineering* [11] as one approach to bringing innovation capabilities into a systems engineering process. Value engineering is tightly related with requirements engineering and innovation management and supports functional analysis. However, the use of models as well as methodical guidance are not discussed.

The software part of automotive control system innovations can be described by *software evolution*, which denotes the adaption of software to requirements changes [32]. Meir and Ramil [33] recommend "assumptions management" to reduce uncertainty in requirements, "evolution management" for control of implementing software evolutions, "release management" to support the rollout of software evolutions, and "metrics and modeling" to improve the overall evolution process. The authors of [34], [35] and [36] show tool support for extraction, analysis, and visualization of software evolutions. These tools aim at reverse engineering, the tracking of code changes and the improvement of software maintenance. The abstraction level is low and lacks the use of models. These approaches do not meet our expectations for innovation management of automotive control systems.

In summary, there are many approaches concerned with evolution and variability in different engineering contexts and perceiving innovation management using a process-centric view [23], [22]. Only few of these combine an interdisciplinary view of innovations of automotive control systems (**RQ1**) using system modeling and providing procedural guidance on how to accomplish innovations (**RQ2**).

VII. Conclusion

Innovative ability is a critical success factor for an automotive company. From our three year collaboration, we learned about the challenges in innovation management and in mastering com-

[2]http://www.aadl.info
[3]http://autosar.org

munication between innovation, requirements and technology management of such a company. Assuming that model-based engineering techniques provide the necessary instrumentation, we proposed three models describing aspects of automotive control systems helpful for identifying innovations (**RQ1**). Based on these models, we elaborated methodical guidance helpful for accomplishing innovations resulting from two generic innovation activities. Both of these activities contain one step for identifying innovations (**RQ1**) and two steps for accomplishing innovations (**RQ2**).

The application of our approach to an example showed traceability between feature and technology platform and enabled the structured innovation in terms of roadmaps in both layers. Practicing our approach during our collaboration led to specific decomposition criteria which we consider a valuable contribution for system modeling, especially with regard to innovation and variability. Our approach is an initial step to make a variety of dependencies related to, for example, costs or quality, transparent. This way, the proposed approach establishes a foundation for comprehensive innovation management.

We focused on software-intensive control systems in the automotive domain. However, we think that our approach is applicable to technical systems of other domains as well, e.g. mechanical systems and software-intensive control systems in aerospace and medical engineering. We suggest further research in terms of the enhancement of our criteria catalogue for modeling, the customization of the modeling concepts we use, the transfer of our approach to other domains, and the assessment of its validity under different assumptions and conditions.

ACKNOWLEDGEMENTS

This article was motivated by a three year industrial collaboration with the BMW Group. We thank all our project partners, particularly, Jakob Huemer and Mariana Just, for providing us valuable insights into their challenging engineering tasks and their organization. We also thank Manfred Broy, Maximilian Junker and Stefan Kugele for helpful feedback.

REFERENCES

[1] M. Broy, "Challenges in automotive software engineering," in *ICSE*, 2006.

[2] P. Trott, *Innovation Management and New Product Development*, 5th ed. Prentice Hall, 2012.

[3] R. Garcia and R. Calantone, "A critical look at technological innovation typology and innovativeness terminology: a literature review," *Journal of Product Innovation Management*, vol. 19, no. 2, 2002.

[4] B. Godin, "The linear model of innovation - the historical ccnstruction of an analytical framework," *Science Technology & Human Values*, 2006.

[5] H. W. Chesbrough, *Open Innovation: The New Imperative for Creating and Profiting from Technology*. Harvard Business Review Press, 2003.

[6] C. P. Team, "CMMI for development," Software Engineering Institute, Tech. Rep. CMU/SEI-2010-TR-033, 2010.

[7] ISO/IEC 15504, *Information technology – Process assessment – Part 4: Guidance on use for process improvement and process capability determination*. ISO, Geneva, Switzerland, 2004.

[8] ISO 9001:2008-12, *Quality management systems – Requirements*. ISO, Geneva, Switzerland.

[9] ISO/IEC 14764, *Software Engineering – Software Life Cycle Processes – Maintenance*. ISO, Geneva, Switzerland, 2006.

[10] C. Haskins and K. Forsberg, *Systems Engineering Handbook – A Guide for System Life Cycle Processes and Activities*. INCOSE, 2011.

[11] AVE, "Society of american value engineers, international," online: http://www.value-eng.org, 2013.

[12] A. Vogelsang and S. Fuhrmann, "Why feature dependencies challenge the requirements engineering of automotive systems: An empirical study," in *RE*, 2013.

[13] J. (Roger) Jiao, W. Simpson, Timothy, and Z. Siddique, "Product family design and platform-based product development: a state-of-the-art review," *Journal of Intelligent Manufacturing*, vol. 18, no. 1, 2007.

[14] M. Broy, M. Gleirscher, S. Merenda, D. Wild, P. Kluge, and W. Krenzer, "Towards a Holistic and Standardized System Architecture Description," *IEEE Computer*, vol. 42, no. 12, Jul 2009.

[15] M. Broy, "A logical basis for component-oriented software and systems engineering," *The Computer Journal*, vol. 53, no. 10, 2010.

[16] ——, "Multifunctional software systems: Structured modeling and specification of functional requirements," *Science of Computer Programming*, vol. 75, no. 12, 2010.

[17] Object Management Group, "OMG Unified Modeling Language™(OMG UML), Superstructure - Version 2.4.1," Tech. Rep. OMG Document Number: formal/2011-08-06, 2011.

[18] S. Apel and C. Lengauer, "Superimposition: A language-independent approach to software composition," in *Software Composition*. Springer Berlin Heidelberg, 2008.

[19] B. Penzenstadler, "Exactly the information your subcontractor needs: Desyre – decomposing system requirements," in *1st International Workshop on Requirements Patterns*, Trento, Italia, 2011.

[20] Object Management Group, "OMG Systems Modeling Language (OMG SysML™)," Tech. Rep. OMG document: formal/2010-06-01, 2010.

[21] T. Männistö, "A conceptual modeling approach to product families and their evolution," Ph.D. dissertation, Espoo, 2000, acta Polytechnica Scandinavica, Mathematics and Computing Series.

[22] L. Andreasson and O. Henfridsson, "Digital differentiation, software product lines, and the challenge of isomorphism in innovation: A case study," in *ECIS*, 2008.

[23] G. Böckle, "Innovation management for product line engineering organizations," in *Software Product Lines*. Springer, 2005.

[24] G. K. Hanssen and T. E. Fægri, "Process fusion: An industrial case study on agile software product line engineering," *Journal of Systems and Software*, vol. 81, no. 6, 2008.

[25] S. Trujillo, G. Aldekoa, and G. Sagardui, "Tracking the Evolution of Feature Oriented Product Lines," *JISBD*, 2007.

[26] F. Kordon, J. Hugues, A. Canals, and A. Dohet, Eds., *Embedded Systems: Analysis and Modeling with SysML, UML and AADL*. Wiley, 2013.

[27] W. Haberl, M. Herrmannsdoerfer, S. Kugele, M. Tautschnig, and M. Wechs, "Seamless model-driven development put into practice," in *Leveraging Applications of Formal Methods, Verification, and Validation*. Springer Berlin Heidelberg, 2010.

[28] EAST-EEA, *Electronics Architecture and Software Technology - Architecture Description Language*. ITEA project 00009, 2004.

[29] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," SEI, CMU, Pittsburgh, PA, Tech. Rep. CMU/SEI-90-TR-21, 1990.

[30] *ISO 10303-214:2010/233:2012: Industrial automation systems and integration – Product data representation and exchange – Part 214: Application protocol: Core data for automotive mechanical design processes, Part 233: Application protocol: Systems engineering*, ISO Std. 10 303-214/233, 2012.

[31] E. E. Consortium, "Integrated modular avionics (ima) – project review," EU ENHANCE Consortium, Tech. Rep., 2002, EU Framework Programme Project.

[32] M. M. Lehman and L. A. Belady, Eds., *Program evolution: processes of software change*. San Diego, USA: Academic Press Professional, 1985.

[33] M. M. Lehman and J. F. Ramil, "Software evolution – background ,theory, practice," *Information Processing Letters*, 2003.

[34] D. German, A. Hindle, and N. Jordan, "Visualizing the evolution of software using softchange," in *SEKE*, 2004.

[35] S. Ducasse, T. Garba, and O. Nierstrasz, "Moose: an agile reengineering environment," in *ESEC/FSE*, 2005.

[36] M. Fischer, M. Pinzger, and H. Gall, "Populating a release history database from version control and bug tracking systems," in *ICSM*, 2003.