Appeared in:
Proceedings of **In-Vehicle Software 2001 (SP-1587)**, pp. 43-55,
SAE World Congress 2001, March, 5-8, 2001,
Detroit, Michigan, USA

2001-01-0025

A Case Study in Applying a Product Line Approach for Car Periphery Supervision Systems

Steffen Thiel, Stefan Ferber, Thomas Fischer, Andreas Hein, Michael Schlick
{ steffen.thiel | stefan.ferber | thomas.fischer8 | andreas.hein1 | michael.schlick }@de.bosch.com
Robert Bosch GmbH

ABSTRACT

Car Periphery Supervision (CPS) systems comprise a family of automotive systems that are based on sensors installed around the vehicle to monitor its environment. The measurement and evaluation of sensor data enables the realization of several kinds of higher level applications such as parking assistance or blind spot detection. Although a lot of similarity can be identified among CPS applications, these systems are traditionally built separately. Usually, each single system is built with its own electronic control unit, and it is likely that the application software is bound to the controller's hardware. Current systems engineering therefore often leads to a large number of inflexible, dedicated systems in the automobile that together consume a large amount of power, weight, and installation space and produce high manufacturing and maintenance costs.

This paper reports on an initiative undertaken by the Bosch Group in applying a product line development approach to develop CPS systems economically. Product line development represents a multi-system engineering approach which takes common and variable aspects between systems in the same application domain into account. It provides a basis to develop a line of products economically based on a common system architecture and reusable components.

A product line allows the degree of reusability to be optimized across different systems while simultaneously preserving the overall quality. This supports the need to develop more integrated and flexible multi-functional systems quickly and cost-effectively. The purpose of this paper is to report on the experiences and results obtained from a case study in developing a product line of CPS systems.

1 INTRODUCTION

The Bosch Group is a large electronics manufacturer that specializes in producing components and systems for automobiles. Bosch has several business divisions that develop systems for the automotive equipment market, ranging from braking systems, engine management and fuel-injection systems to body electronics, semiconductors and control units, starters and alternators, and mobile communication systems.

Several departments of the automotive equipment business sector are developing systems that are based on short range sensor technology. These systems are called *Car Periphery Supervision* (CPS) systems. CPS systems can be characterized as a family of automotive products that are based on sensors installed around the vehicle to monitor its local environment. Different sensor measurement methods and evaluation mechanisms enable the realization of various kinds of higher level applications which guarantee more safety and comfort for car drivers. Examples of such applications are parking assistance, automatic detection of objects in vehicles' blind spots and the adaptive control of airbags and seatbelt tensioners.

Advanced research and development in this field has uncovered a significant amount of commonality between sensor systems of the CPS application domain. For example, each CPS system typically requests, filters, and evaluates data from sensors while at the same time performing diagnostic, availability tests, and consistency checks. A common platform which would provide such services and which could be shared by every CPS system would allow a high degree of reusability. By reusing software and hardware subsystems, parameterizing generic parts, and plugging in individual functionality, the platform would serve as a basis for faster and more cost-effective development of CPS system variants. However, to achieve a system platform with the properties outlined above, one has to follow a systematic and reuse-oriented development process.

The case study presented in this paper reports on an initiative undertaken by Robert Bosch Corporate Research and Development and several business departments in applying a product line development approach to develop CPS systems economically. Product line development represents a multi-system engineering approach which takes common and variable aspects among systems in the same application domain into account. It provides a basis for economically developing a line of products based on a common system platform. In this context, a platform consists of a set of subsystems and interfaces that form a common structure from which a stream of derivative products can be efficiently created and launched. This allows the degree of reusability to be optimized across different systems in the CPS application domain while simultaneously preserving the overall quality. It also supports the need for a fast and cost-effective development.

The remainder of this paper is organized as follows: Chapter 2 introduces the various applications of a CPS product family. In Chapter 3, the motivation and objectives in applying a product line approach for CPS systems are outlined. A short overview on product line engineering is given in Chapter 4. Furthermore, major results obtained from the CPS product line case study are presented. The validation plan of the case study is explained in Chapter 5. Finally, Chapter 6 concludes with a short summary.

2 CAR PERIPHERY SUPERVISION SYSTEMS

To date, applications of the CPS domain are built as separate systems with the consequence that each one needs its own set of sensors, controllers, busses, hardware interfaces, application and device driver software, and so on. As mentioned in the introduction, a base system would give applications the opportunity to directly evaluate data from different sensors. Many applications that evaluate sensor data do not depend to specific types of sensors but are able to share the information from sensors installed around a vehicle. Considering that at present the number of sensors in a bumper is restricted, the shared usage of sensors becomes inevitable. This was one of the reasons why a product line development approach was adopted for this effort.

Before going into more technical details, let us first characterize the spectrum of applications that are addressed in this case study. Figure 1 illustrates the four underlying safety- and comfort-related application areas pre-crash detection, parking assistance, blind spot detection, and stop & go that are the focus of this case study.

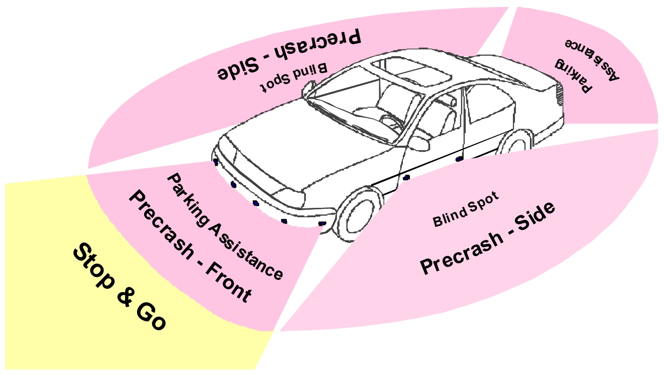


Figure 1: Applications of Car Periphery Supervision

Please refer to the following subsections for a brief description of these application areas.

2.1 Pre-Crash Detection

Current safety devices in cars are triggered by sensors that detect strong (negative) acceleration. The crash is detected as it happens, and all further actions must be taken very fast – e.g., firing of airbags. Sensors can be used for detecting an object very close to the car, its relative velocity, and its acceleration. Based on this information, it is possible to estimate the time, area, and direction of an impact even before the crash happens. These kinds of estimations are provided by *pre-crash detection* (PCD) systems. A PCD system delivers more precise information and delivers it earlier than current crash detection systems. This advantage enables these systems to be used in adaptations of airbag firing and seatbelt tensioning. For example, the trigger points of a specific airbag in different locations in the car (e.g., front and side) can be adjusted appropriately for the estimated crash situation.

Initial implementations of PCD systems use two distance barriers. Only objects entering the outer barrier will be monitored by the system. As soon as an object breaks through the inner barrier, the calculated crash impact will be reported to the air bag control unit. The appropriate proximity of the car must be supervised in order to provide this functionality, that is, the front for frontal accidents, the sides of the passenger cabin for side accidents. The measurement must be quite fast to be able to deal with relative vehicle/object speeds of up to 200 km/h. Combined with the fact that the system has to work under bad environmental conditions, the choice of possible measurement principles is restricted.

2.2 Parking Assistance

The basic idea behind *parking assistance* (PA) is to inform the driver about the distance between the vehicle and potential obstacles around the vehicle while the vehicle is moving slowly backward or forward. One goal of this application is to support the driver in avoiding scraping the vehicle against people or stationary objects. This is especially useful since most vehicles are difficult to look over and low obstacles are therefore hard to see for the driver.

Technically speaking, the car periphery is supervised using short range sensors. Usually, the different sensor signals are combined and the resulting distances to obstacles are displayed to the driver. If a vehicle comes close to an obstacle, the driver gets a warning – visually or acoustically. As a variant of PA, semi-autonomous parking systems assist the driver in backing the car into a typical parking spot. In this case, the sensors are also used to determine the size of the parking space. The system then computes the optimal track for maneuvering the vehicle into the spot and guides the driver by giving steering indications.

2.3 Blind Spot Detection

The main objective of applications based on *blind spot detection* (BSD) is to reliably detect and inform the driver of vehicles in the central blind spot region. A blind spot situation occurs when another vehicle slowly enters the blind spot region of the vehicle while passing it. The blind spot region is the spot where a passing vehicle is no longer visible with conventional mirrors.

A solution to the blind spot problem is achieved by using lateral front and rear sensors for detecting passing vehicles. The rear sensors observe the central blind spot region, whereas the front sensors discriminate irrelevant warnings. The detection of irrelevant signals is necessary to guarantee immunity of false warnings such as stationary obstacles, on-coming vehicles, and so on. For BSD, the sensor technology used must reliably supervise a proximity of about 5 m on each side of the vehicle. Visual or acoustical warnings are conceivable for informing the driver of vehicles in the blind spot. The acoustical warning signal may be intensified if the system detects that the driver wants to change lanes while a vehicle is in the blind spot. The activation of the turn signal may be used as a proxy for the intention to change lanes.

2.4 Adaptive Cruise Control (Stop & Go)

Car drivers focus a great deal of their attention on maintaining the correct distance from the vehicle ahead. Bosch has developed *adaptive cruise control* (ACC) to relieve the driver from this task. At first glance, ACC seems similar to traditional cruise control systems. The subtle difference is that ACC systems *automatically* adjust vehicle speed by measuring the distance and relative speed of the vehicle driving ahead. The additional capabilities of ACC include reacting to traffic events by accelerating and decelerating and, if necessary, by braking. This has been possible since the development of traction control and electronic stability systems. Together with the functionality of a automatic braking system, ACC systems are able to create brake pressure independently of the driver. Technically speaking, ACC is based less on conventional cruise control than on the function of brake, traction, and dynamics control systems.

If a slower-moving car appears in the lane ahead of an ACC-equipped car, the system reduces speed, first by decelerating and then, if necessary, by braking. The ACC vehicle then follows the lead vehicle at a consistent distance. If a vehicle cuts in or out of the ACC's sensing range, the system automatically selects the new relevant object. If the driving lane is open, the vehicle is accelerated to the pre-selected speed via ACC. Currently, the first generation of ACC systems have been introduced in the market. This generation covers long range applications and is suitable for high velocity traffic, as is usual on freeways. Future generations of ACC will cover medium distances with broader sensor scattering as well as short-range supervision. The latter would address stop-and-go traffic, e.g., in cities, which is sometimes called ACC *stop & go* (S&G).

3 MOTIVATION FOR A CPS PRODUCT LINE

As one can easily imagine from the preceding discussion, the different systems that can be created in the CPS application domain have much in common. One objective of the CPS case study is to identify these similarities and to build a common platform from which a set of derivative products can be efficiently developed and launched to the market. For example, consider a system platform which exploits the following reuse characteristics:

- *Hardware:* Reuse of sensors for monitoring the environment as well as reuse of electronic control units for interfacing with other hardware components and for executing the software.

- *Software*: Reuse of software subsystems for measurement, transformation, evaluation, and display of sensor data to enable higher level applications.

These reuse characteristics of a CPS system platform would count for the following benefits:

- *Lower software development costs and higher quality* due to reuse of a qualified architecture and tested software components in different product variants.
- *Lower hardware costs and integration space and less weight* due to reduction of electronic control units for processing the sensor data.
- *Faster time-to-market* due to focusing development on local, application-specific problems that do not affect the whole architectural infrastructure of the platform (delta development).

Another objective of this case study is to identify and understand the different customer requirements as well as requirements and constraints that arise from market analyses and from the opportunities and limitations of the underlying technology. As a result, the commonality and, especially, the variability of the various CPS systems should be described.

From a technical viewpoint, for example, variability arises because different sensor technologies, different numbers of sensors, different positions and alignment of sensors, and different types of measurements, transformations, and evaluation metrics are necessary to create CPS system variants. Selecting sensor technology as a sample dimension, the following range of candidate sensors may be considered for developing CPS variants:

- *Ultrasonic sensors* are suitable for short range distance measurements at low speed. This kind of sensors was the first to be used in car electronics. They do not provide speed measurements and have a restricted resolution.
- *Laser-based sensors* use either triangulation or a scanning method to derive distance information. They are not very suitable in the car electronics domain due to restrictions to a low laser emission classification and due to easy jamming by mud and dirt.
- *Microwave or radar sensors* offer many advantages in the car electronics domain. They measure faster and more accurately than ultrasonic sensors and they can be customized with different wave lengths for various measurement ranges (e.g., 60-80 GHz for long range, 20-30 GHz for short range). These kinds of sensors have different operation modes – e.g., distance, speed, or object tracking modes. Moreover, microwave sensors are reliable in bad weather situations and can be mounted behind the car body.
- *Video sensors* assist the driver with a video picture of a view that is blocked or is invisible due to darkness or bad weather.

Please note that *ultrasonic* and *microwave* are the most important sensor technologies for the current generation of CPS systems.

Finally, getting a clear picture about the required (i.e., customer-driven) and technologically feasible variants is very important for planning the introduction of CPS products in different markets. Various strategies are possible, ranging from niche-specific products to horizontal leverage and vertical scaling of products. For example, if a horizontal leverage strategy is applied, products are leveraged from one market niche to the next within a given tier of price and performance, as shown in Figure 2.

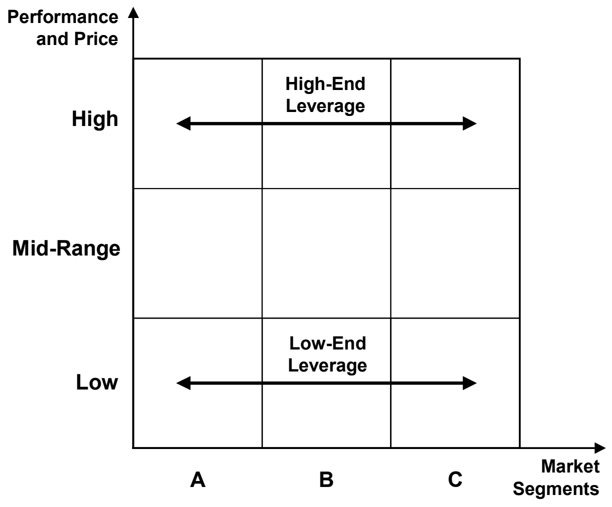


Figure 2: Horizontal Leverage

The benefit of a horizontal leverage strategy is that a company is able to introduce streams of new products across a series of related customer groups without having to reinvent the wheel for each. Further, if particular subsystems can be designed to provide a distinctive functional advantage over competitors, the entire product line will benefit. Additionally, for manufacturing, procurement and retooling costs can be minimized when new products are introduced into the line.

4 CPS PRODUCT LINE ENGINEERING

A *product line* is a set of products that together address a particular market segment or fulfill a particular mission [2]. Product lines are efficiently developed based on product platforms. Generally speaking, a *product platform* consists of a set of subsystems and interfaces that form a common structure from which a stream of derivative products (i.e., the product line) can be efficiently created and launched. Usually, a product line is part of a technical domain. A domain can be defined as an abstraction that groups a set of software systems, or some functional areas within systems according to a domain definition shared by a community of stakeholders [19]. The domain can be considered to include not only the shared terminology and definitions, but the coherent body of knowledge about domain systems shared by that community. A product line usually covers only parts of a domain. The parts that are actually included in a line of products are defined by the product line scope. The *scope* specifies the boundaries of the product line, i.e., the functionality that must be provided. Therefore, the scope describes in which context the assets will be reused in future. All products derived from the product line assets are called variants. An asset is a work product developed or reengineered *for* reuse and placed under management within an asset base. The product platform is defined by the set of core assets. Assets can be developed based on work products from any phase of the engineering life cycle, including requirements, architectural design, code, test cases, and so on.

A growing research community exists in the field of *software* product lines [1, 2, 5, 6, 8, 9, 12] although most of the underlying concepts are known from traditional manufacturing. As illustrated in Figure 3, product line engineering consists of three intertwined development processes running in parallel: domain engineering, product engineering, and management. *Domain engineering* is a development-*for*-reuse process and focuses on the analysis, specification, and implementation of assets in a particular domain for use in the development of multiple software-based products. In the context of product line development, *product engineering* is a development-*with*-reuse process. It uses the assets supplied by the domain engineering process (e.g., software components) to build new products. Product engineers typically compare customer requirements for a product with the capabilities of existing assets in the asset base and pick those assets ones that fit best. They generate a new member of the product line by combining assets. In case a requirement cannot be satisfied by the set of existing assets, the product engineer has to provide a realization for it. It is likely the case that a customer requirement is bound to a special feature in a particular product and that it does not influence the whole product line.

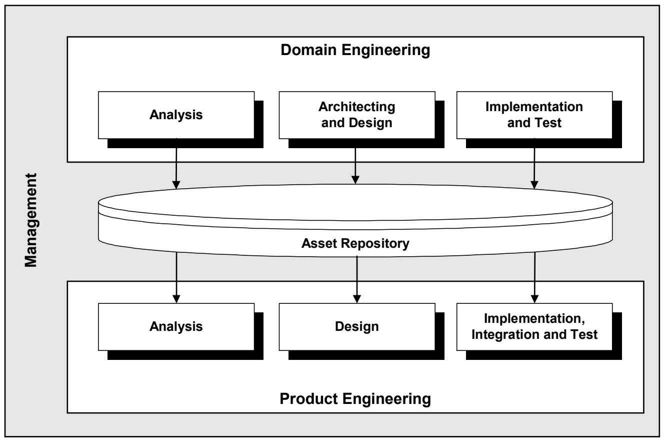


Figure 3: Product Line Process Families

In addition, support of further *management* processes is required for product line engineering. For example, project and change management are needed for domain and product engineering. Furthermore, asset management focuses on the issue of how to implement and run the interface between domain engineering and product development. Currently, the technical key to providing an interface between these processes is the asset repository.
In the following subsection we describe parts of the product line engineering process we have applied in the CPS case study.

4.1 Scoping

One of the major goals of scoping is to define the boundaries of the product line domain [5]. Scoping provides information about what capabilities are regarded to be “inside” and “outside” the product line. In our context, the objective of performing a scoping phase was to identify the business, organizational, technical, and legal requirements and constraints that are characteristic to CPS. Typical inputs for scoping are descriptions of existing products, marketing analyses, expert knowledge as well as business strategies for introducing the new products into the market. In order to define the scope of the CPS product line, we first interviewed management and investigated the business together with marketing and sales personnel. As a result, we obtained an initial sketch of the business case for the CPS product line effort. Furthermore, we described the characteristics of legacy and competitor products as well as future extensions to existing products. Finally, we analyzed standards and technology drivers that were related to the products under consideration, as defined in the business case.

As a result of these activities, we obtained an initial understanding of the functional scope for the CPS product line, as illustrated in Figure 4.

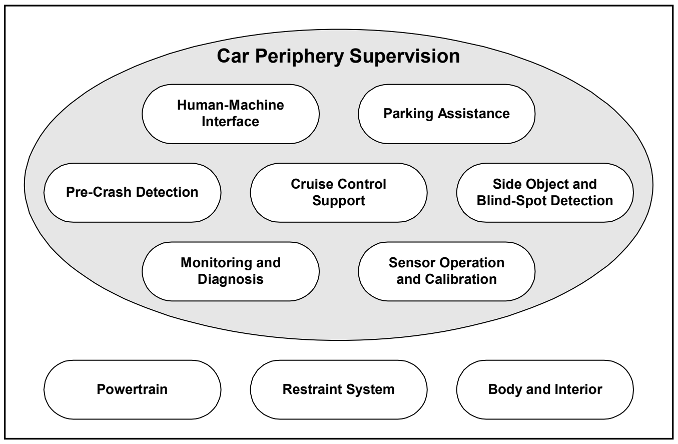
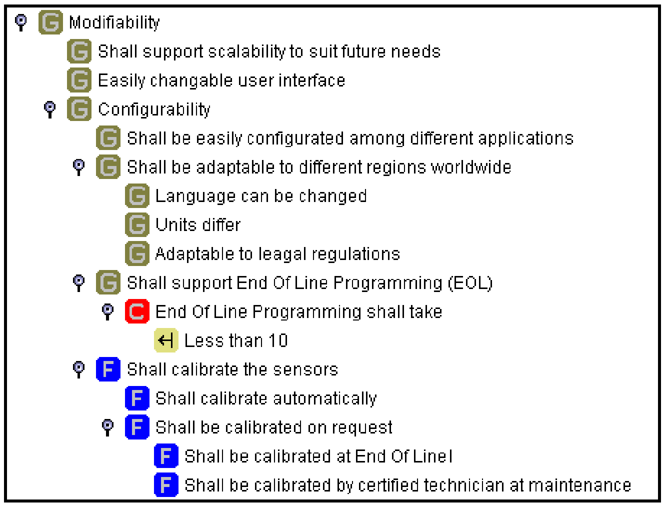


Figure 4: Preliminary Scope of the CPS Product Line

In particular, we identified seven core functional areas (human-machine interface, parking assistance, pre-crash detection, cruise control support, side object and blind-spot detection, monitoring and diagnosis, and sensor operation and calibration) and three borderline functional areas (powertrain, restraint system, and body and interior) as being part of the scope for CPS.

4.2 Requirements Analysis and Modeling

To understand the requirements for the scoped CPS product line in more depth, we went through a domain analysis phase. As a first step, we gathered, described, and classified the customer requirements for different CPS variants. We noted whether each requirement was of purely functional nature, whether it was quality-related, or if it described certain constraints on the design of a system variant. A *functional requirement* specifies actions that a system must be able to perform, regardless of taking physical constraints under consideration. *Quality requirements* are requirements that describe how certain quality attributes such as reliability, safety, performance, portability, or modifiability shall be satisfied by a system. Figure 5 shows a quality tree for the quality requirement *modifiability* which can loosely be defined as “the ability to make changes quickly and cost effectively” [10]. For each quality requirement we noted the specific goals of how it could be achieved in the CPS context as well as existing constraints and associated functional requirements. In Figure 5, goals are represented by the label “G”, whereas the labels “C” and “F” are provided for constraints and related functional requirements, respectively. Note that we applied the approach described in [20] to specify and detail the CPS quality requirements.



```
G Modifiability
    G Shall support scalability to suit future needs
    G Easily changable user interface
    G Configurability
        G Shall be easily configurated among different applications
        G Shall be adaptable to different regions worldwide
            G Language can be changed
            G Units differ
            G Adaptable to leagal regulations
        G Shall support End Of Line Programming (EOL)
            C End Of Line Programming shall take
                Less than 10
        F Shall calibrate the sensors
            F Shall calibrate automatically
            F Shall be calibrated on request
                F Shall be calibrated at End Of Linel
                F Shall be calibrated by certified technician at maintenance
```

Figure 5: Quality Tree for Modifiability

As mentioned above, a constraint specifies or constrains the design of a system. Examples of such design constraints are implementation, physical, and interface constraints. *Implementation constraints* specify the coding or construction of a system (e.g., implementation languages or operation environments). A *physical constraint* specifies a physical characteristic that a system must possess – for example, material, shape, size, or weight. An *interface constraint* specifies an external item with which a system must interact, or constraints on formats, timings, or other factors used by such an interaction.

In order to make the elicited requirements more concrete, and to explore the CPS functionality in detail, we performed a domain modeling step as proposed in [5]. To this end, we brainstormed a representative set of use cases and synthesized functional capabilities (services) and actors from the requirements. The use cases were then modeled based on the services identified which lead to a refinement of capabilities and, consequently, to a refinement of the CPS product line scope. Next you will find a list of sample services and actors we have identified as a result of our domain modeling iteration:

- *Human-machine interface* (boundary service): This service represents the interface to the user. It provides the entire functionality for all input/output interactions.
- *CPS control* (core service): The control service coordinates the functionality of the CPS system. In particular, it controls the operation modes of the sensors and determines the way in which the sensor data are analyzed. Furthermore, the service manages the resources considering the driver's intention, the state of the car, and the environment.

- *Data analysis* (core service): The data analysis service processes data from the sensors (e.g., distance and velocity). It assesses the situation and generates an environment description.
- *Output derivation* (core service): The service calculates the CPS response, e.g., the distance to closest object or the time-to-impact.
- *Sensor* (boundary service): This entity represents sensors. The CPS uses one-dimensional microwave and ultrasonic sensors that provide distance and velocity values. The measurement output of each sensor depends on different operation modes.
- *Car state* (boundary service): This service interfaces the CPS system to external systems that provide gear information, vehicle speed, steering angle, turn signal state, and ignition state.
- *Restraint system* (boundary service): Interfaces the CPS system to an external restraint system.
- *Powertrain* (boundary service): This service provides an interface to the brake and motor control.
- *Driver* and *Environment* (actors): The driver is the person who steers the vehicle, whereas the environment is a placeholder for the environmental condition around the vehicle that can be observed by a CPS system. Pedestrians and cars, for example, are part of the environment.

Each use case can now be "simulated" based on the service and actor descriptions. This leads to a better understanding of the product line functionality. It also helps in uncovering "hidden" requirements. For example, consider the generic use case "request a CPS functionality" and its associated use case model, as represented in the collaboration diagram of Figure 6. This model shows interactions between actors and services for the specific use case, thus making its functionality more concrete. The model is read as follows: A user, for example the driver, requests a CPS functionality at the human-machine interface (1). This request is forwarded to the CPS control (2). Based on the car state (3), the CPS control determines the appropriate sensor operation (4), the required analysis algorithms (5), and the output derivation mechanisms (6). Signals emitted by the sensors (7) are reflected by the environment and the responses (8) are detected. The sensors provide their data for the analysis step (9), in which the situation is assessed according to the requested application. As a result, an environment description is forwarded to the output derivation service (10) and the CPS control (11). Based on the analysis results, the output derivation service provides the restraint system and powertrain with crash estimates (12) and distance to object information, respectively (13). Finally, the derivation results are forwarded to the human-machine interface (14) and the driver is informed about the new status (15).

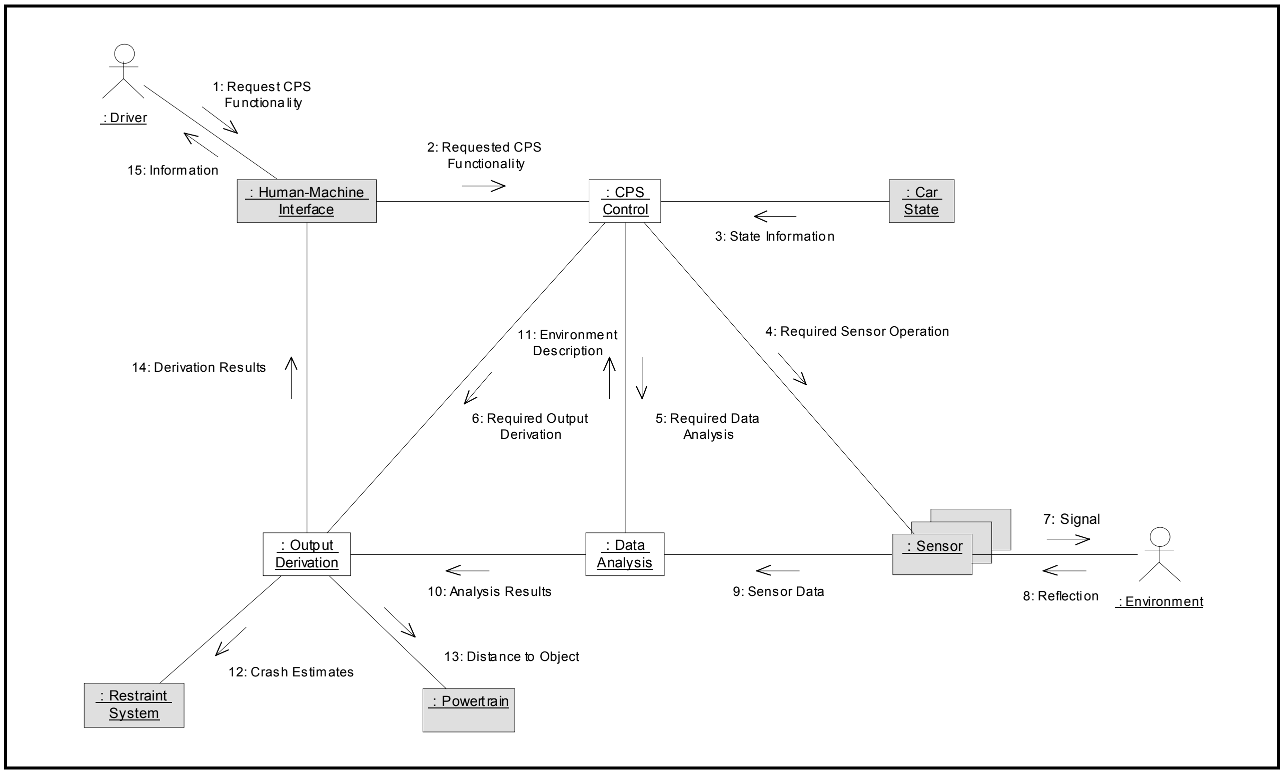


Figure 6: Collaboration Diagram Representing a CPS Use Case

4.3 Feature Modeling

To represent commonality and variability within the requirements of the CPS domain more explicitly, we used and refined the concept of feature modeling, as introduced in the FODA (Feature-Oriented Domain Analysis) method [3, 4]. In FODA, a feature is understood as a prominent or distinctive user-visible aspect, quality or characteristic of one or more software systems. Features can be interrelated by several types of links that essentially establish a tree structure with mandatory, alternative, and optional branches. Composition rules define valid combinations of optional and alternative features. In addition, we used rationales to express the concerns that are connected with feature selections. Together, the feature tree, the composition rules, and the rationales form the feature model.

One goal of domain analysis is to build an abstract model that can be used as a starting point for the derivation of specific product variants. The feature model is predestined to play this role, as it captures the decisions that have to be made to determine the individual characteristics of a concrete system. Application derivation starts with the elimination of the domain variability step by step through the instantiation of a subset of all potential features. In order to completely derive an application, i.e., one single product variant, all variability must be resolved. Nevertheless, variable products can be derived through partial derivation which leaves some selections open.

Feature modeling is advantageous for product line engineering in several ways:

- *Control over variability*. All variants of products in the domain and their relationships are represented in a comprehensive and understandable form.
- *Configuration support*. As features are linked to modeling items in later development phases, application derivation through selection of combinations of variants can be used to support configuration right up to product code.
- *Sales support*. Sales representatives have a high-level basis for discussions over product tradeoffs with customers.
- *Support for new development*. When a new product is to be developed, feature modeling simplifies the analysis of how it differs from existing ones.

A small part of the feature model for the CPS domain is illustrated in Figure 7. It shows an extract of the capabilities that can be incorporated into a product variant of the CPS product line. For clarity reasons, feature types and dependencies are not shown in the model. As an example, one could select the front supervision service to be included in the product variant in order to realize a pre-crash system. In a second step, front supervision capabilities can further be specified by, for example, defining the exact geometry of the supervision range. This can be done, for example, by choosing different kinds of measurements to be performed. Note that the quality tree and the use case models described in the previous sections provide guidance in obtaining the set of features and the associated composition rules for the products in the CPS domain.

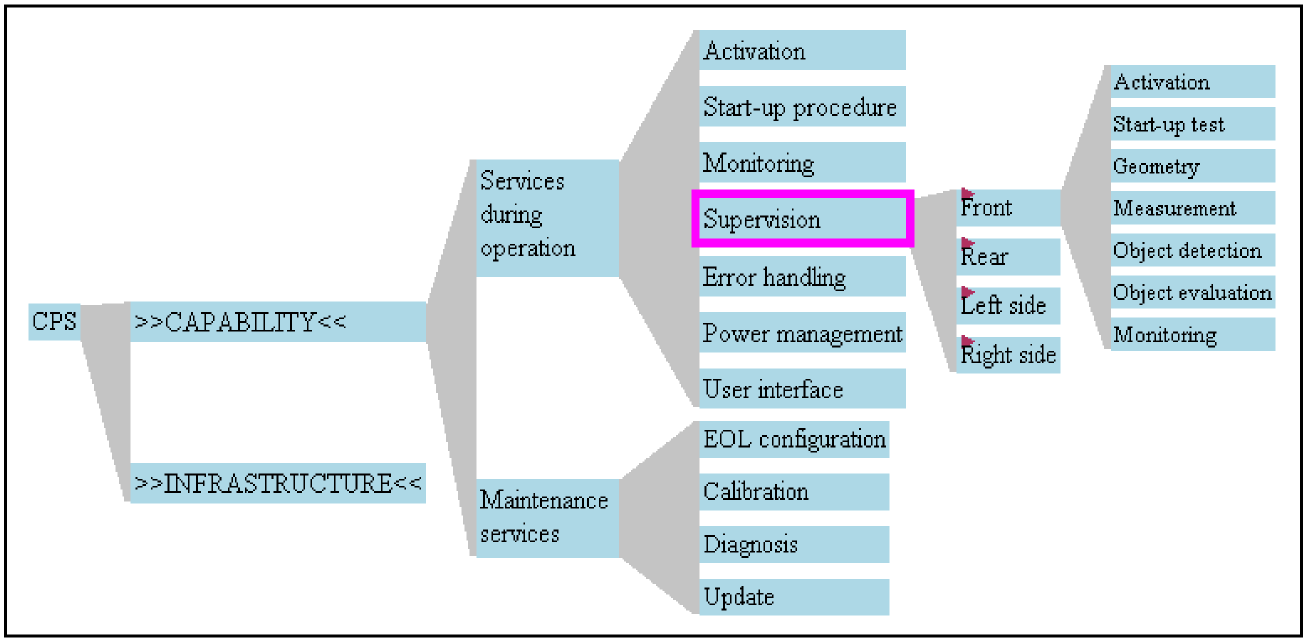


Figure 7: Partial CPS Feature Model

4.4 Architecting and Design

At the end of the first domain analysis iteration for CPS as described above, the scope of the product line had been refined. In particular, the functional and quality requirements as well as their similarities and differences were documented. This information formed the basis of the architectural design iteration.

An *architecture* describes the overall technical structure of a system. It consists of software and hardware components and connectors as well as compositions of them. An architecture represents the manifestation of the earliest design decisions about a system and is an opportunity for an early validation of the design decisions with respect to qualities [10]. We used the Architecture Based Design (ABD) method [11], developed collaboratively with the Software Engineering Institute [5], to define the conceptual architecture of the CPS product line. The ABD method considers functional, quality, and business requirements at an abstraction level which allows for the variation necessary when producing specific products of the product line. Its application relies on an understanding of the architectural mechanisms and styles [12, 13, 14, 15] necessary to achieve this flexibility. The method provides a series of steps for designing a conceptual system architecture. The conceptual architecture provides organization of functionality, identification of synchronization points for independent threads of control, and allocation of function to processing units.

Within this case study, a basic high-level architecture was developed that enables the use of one set of sensors for multiple concurrent applications as well as the control of the focus of sensors in specific situations. One restriction that arises is that only a limited number of sensors can be mounted in a vehicle which means that it will be necessary in the future to share sensors over concurrent applications. Moreover, new generations of CPS applications require improved sensor control so that specific objects can be tracked within a given supervision range.

In order to use particular measurements and provisional results repeatedly, the measurement data processing was split into several steps that are realized by subsystems with defined interfaces, as illustrated in Figure 8. Furthermore, a control subsystem was introduced to organize the measurement data processing of single or multiple applications including prioritization. The control subsystem uses a situation subsystem to determine the next steps. The latter evaluates conditions that define possible situations of a vehicle related to its environment (e.g., “normal” or “pre-crash situation”). The current state of the situation subsystem is obtained from sensor measurements and external events (e.g., shifting into reverse gear).

The repeated use of measurements and provisional results is equivalent to a repeated use of sensors and parts of algorithms. Introducing the situation subsystem also enables the specific control of resources (sensors, processor time) for both single applications and combinations of applications with different priorities. In addition, coupling the current situation to the results of the sensor measurements allows adjustments of the range of subsequent measurements (focus control) and successive objects (object tracking) for particular applications.

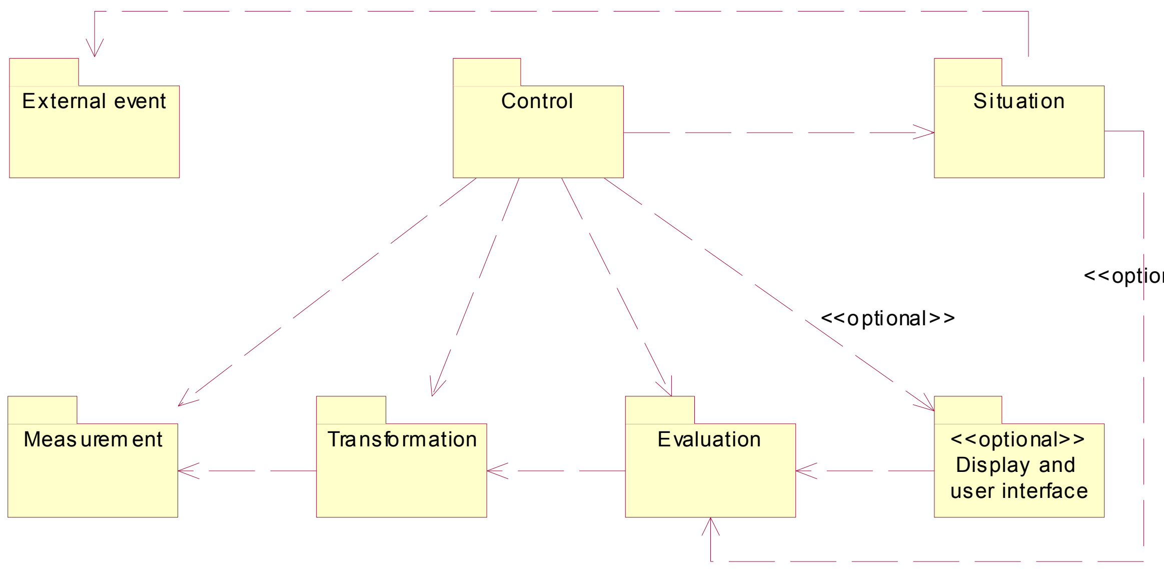


Figure 8: Conceptual Platform Architecture

The conceptual CPS architecture depicted in Figure 8 was developed by considering the CARTRONIC architectural style [17]. It consists of several high-level subsystems and dependency-relationships, represented in UML notation. The responsibilities of the subsystems can be summarized as follows:

- *Measurement:* This subsystem encapsulates all details that are related to sensor measurement. It can be thought of as a hardware capsule that provides an interface for controlling the measurement and for preprocessing the measurement data.
- *Transformation:* This subsystem transforms the measurement data into a description of the vehicle's environment.
- *Evaluation:* This subsystem analyzes the environment description with respect to the active application – for example, parking assistance or pre-crash detection.
- *Display and user interface:* This subsystem is used by applications such as parking assistance to inform the user about evaluation results, e.g., the distance to the next obstacle, and about the status of the CPS system. The subsystem can be seen as optional since it not required in every CPS product variant.
- *Situation:* This subsystem contains descriptions of normal and exceptional situations and priority regulations for resource conflict resolution between several applications. It can be represented by a state machine and uses external events and evaluation results to determine its current state.
- *Control:* According to the CARTRONIC style used in automotive systems, the control subsystem coordinates other subsystems. In particular, it determines the order of computing steps depending on the respective situation.

Figure 9 shows an example of component reuse for different kinds of products. As explained above, the measurement subsystem includes components for measuring distance and velocity. Slow, fast, and dangerous objects can be detected by the corresponding components in the transformation subsystem. As illustrated in Figure 9, these five components can be used to create different product variants. For example, the distance measurement and slow object detection components can be (re)used in parking assistance, ACC stop & go, and pre-crash detection systems (and, of course, combinations and variants of these systems).

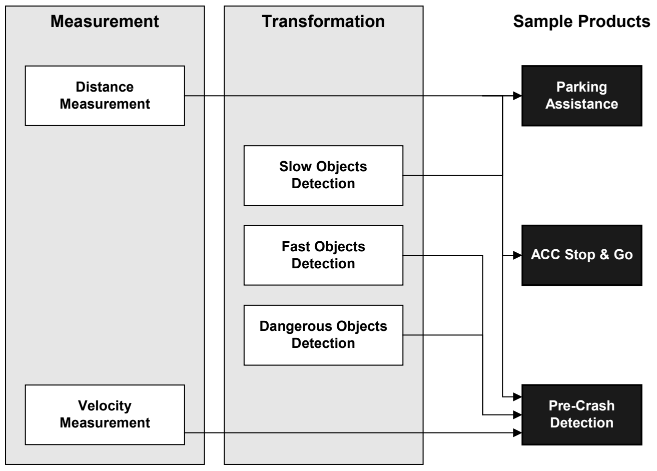


Figure 9: CPS Component Reuse Example

Altogether, the basic CPS architecture allows modular control of the measurement system. Depending on the current application, the corresponding subsystems and components are activated. Finally, it is possible to integrate other applications into the system by adding the corresponding evaluation mechanisms.

5 EMPIRICAL VALIDATION

Business needs and the technical evolution of information technology (IT) in cars stimulates Bosch Corporate Research and Development to invest in product line technology. In 1997, investigations were commenced on the application of a product line approach for automotive electronics systems [5, 7]. After acquiring the basic knowledge and processes, the approach is now used in several development projects. One of these projects is covered by this case study. Among other objectives, it shall evaluate the feasibility and benefits of product line development for Bosch in the CPS application domain. In particular, the following assumptions shall be verified:

- Product lines are appropriate to deal with increasing complexity, safety, security, reliability, performance, and cost concerns for IT in cars.
- Domain engineering supports reuse in order to shorten time to market during product development and to increase the quality of products.
- The initial investment for domain engineering pays off for a product line.

These assumptions can be used to derive finer grained measurement goals that can be verified by using the Goal-Question-Metric (GQM) approach [18]. After applying the GQM method, the measurement goals (G) are refined into questions (Q) and consecutively into metrics (M) which will supply necessary information for answering those questions. The GQM method thus provides a measurement plan that deals with the particular set of problems and the set of rules obtained for data interpretation. The interpretation gives answers if and to what extent the stated goals could be attained.

To understand the application of the GQM approach in the CPS context, consider the examples given in the following two subsections.

5.1 GQM Validation Example: Sensor Sharing

Goal
The CPS platform shall support sensor sharing to save costs, installation space, power consumption, and weight in a car.

Questions
On which system level is sensor sharing measured? What is a baseline for a comparison? Does a maximum degree of sensor sharing lead to lowest costs, volume, power consumption, and weight? Which overhead is created by sensor sharing, if any? What is the cost for shared sensors in CPS? What would be the cost to implement the same functionality if no sensor sharing could be achieved?

Metrics
For example, basic data can be provided by calculating the *sensor utilization (su)* which takes the number of available sensors and the number of sensors used by each application into account:

$$su := \frac{1}{I}\sum_{i}\frac{n_i}{n}$$

$n_i =$ number of sensors used by application i

$n =$ number of sensors installed in the car

$I =$ number of applications installed.

If the degree of sensor sharing is high, the value for *su* will be close to 1. Current engineering efforts show that sensor sharing can likely be achieved at different levels for the CPS product line. In order to gain economy, volume, power consumption, weight, and hardware costs must be lower for platform-based solutions than for separated systems which has to be proven in future investigations.

5.2 GQM Validation Example: Time-to-Market

Goal
The CPS products shall be implemented as a product line (i.e., based on a common platform architecture) in order to allow an easy and quick product construction.

Question
What is the average time required for building a CPS product variant for the high-end, low-end, and mid-range market?

Metrics
The rate for product realization can be measured by taking the degree of reusability into account. Software reusability, for example, can be tracked on functionality and product level. In Figure 10, a *hypothetical* example for reusability measurement based on code size is given. As one can see, the software size is measured in 1000 lines of code (kLOC).

	Product				
Functionality	**1**	**2**	**3**	**4**	**5**
Parking Pilot	■ 30	▨ 32	32	32	32
Parking Assistant	■ 35	▨ 39	▨ 40		40
Parking Place Survey	■ 20			22	22
Blind Spot Detection	□				■ 17
Pre-Crash Detection	■ 60		▨ 60	▨ 60	□
Total kLOC	**145**	**71**	**132**	**124**	**121**

Legend

kLOC new:	■
kLOC reused & changed:	▨
kLOC reused:	□

Figure 10: Hypothetical Reusability Matrix

Within a reuse-oriented and platform-based development effort, time-to-market is expected to be significantly shorter than in traditional product development once that the reusable asset base is established.

5.3 Special Measurement Problems

Many goals for product line engineering are the same as for traditional project and organizational management. However, regarding the decision to adopt a product line approach, a lot of obvious benefits do not have associated measures, like the institutionalization of domain knowledge from experts into assets. Special problems that make quantitative measurement difficult include the following:

- Product cost needs allocation from asset development otherwise asset development charges product development for their work. What is the associated time frame for the cost calculation?
- Aggregating measurements across multiple products may be misleading, as different scales, distributions (e.g., defect densities) or double-counting properties across products (e.g., errors in assets) might be used.

- The level of granularity of measures can vary from very fine to very coarse (e.g., features vs. collection of products). It is difficult to decide in advance which level will help to verify the goals that have to be met.
- It is hard to collect measures consistently across product line projects. Reasons for this are, for example, different products, people, processes, time, and history effects.

Note that the empirical validation plan could be sketched here only briefly. This plan is under execution and we are expecting more quantitative results in the near future. As soon as further results are available, we will cover this topic in more depth.

6 SUMMARY AND CONCLUSIONS

In this paper, we have presented results obtained from a case study in developing a product line for Car Periphery Supervision systems. In particular, we outlined the safety- and comfort-related application areas of CPS systems and demonstrated the motivation to apply a product line development approach for building such systems. Furthermore, we described results from the product line scoping phase and gave insights into requirements analysis and feature modeling. In addition, we outlined the conceptual platform architecture which has been defined in the first iteration of the product line engineering process. Finally, we presented a small part of our validation plan.

Since work of this case study is very much in progress, we expect a refinement of the results in future iterations.

ACKNOWLEDGEMENTS

This work was partially funded by the ITEA project 99005 ESAPS (Engineering Software Architectures, Processes, and Platforms for System Families) which is part of the EUREKA Σ! 2023 programme. We would like to thank our colleagues Franz Grzeschniok, Stefan Kowalewski, Erhart Lederer, John MacGregor, Martin Mergel, and Thomas Strohm for their valuable comments and suggestions on earlier versions of this paper.

CONTACT

If you are interested in more details of our work, feel free to contact us.

E-mail:	steffen.thiel@de.bosch.com
Phone:	(+49) 69 7909-518
Fax:	(+49) 69 7909-550
Address:	**Robert Bosch GmbH** Corporate Research and Development Software Technology (FV/SLD) Eschborner Landstraße 130-132 D-60489 Frankfurt am Main GERMANY

REFERENCES

[1] P. Donohoe (ed.): *Software Product Lines – Experience and Research Directions*; Kluwer Academic Publishers, 2000.

[2] P. Clements: *Software Product Line – A New Paradigm for the New Century*; CrossTalk, pp. 20-23, February 1999.

[3] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, A. S. Peterson: *Feature-Oriented Domain Analysis (FODA) Feasibility Study*; Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, November 1990.

[4] A. Hein, M. Schlick, R. Vinga-Martins: *Applying Feature Models in Industrial Settings*; In: [1], pp.47-70, 2000.

[5] S. Thiel, F. Peruzzi: *Starting a Product Line Approach for an Envisioned Market: Research and Experience in an Industrial Environment*; In: [1], pp. 495-512, 2000.

[6] S. Thiel, S. Ferber, M. Mergel: *An Overview of Methods Supporting Product Line Development*; Technical Report BOSCH-WP1-T1.2-01, ITEA-ESAPS Project, June 2000.

[7] J. Weber, T. Bertram, T. Kytölä, F. Peruzzi, S. Thiel: *Information Technology Restructures Car Electronics*; Technical Report 1999-01-0485, SAE International Congress and Exposition, Detroit, MI, March 1999.

[8] P. Clements, L. M. Northrop: *A Framework for Software Product Line Practice, Version 2.0*; Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, July 1999.

[9] D. M. Weiss, C. T. R. Lai: *Software Product Line Engineering – A Family-Based Software Development Process;* Addison-Wesley, 1999.

[10] L. Bass, P. Clements, R. Kazman: *Software Architecture in Practice*; Addison-Wesley, 1998.

[11] F. Bachmann, L. Bass, G. Chastek, P. Donohoe, F. Peruzzi: *The Architecture Based Design Method*; Technical Report CMU/SEI-2000-TR-001, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, January 2000.

[12] J. Bosch: *Design and Use of Software Architectures – Adopting and Evolving a Product-Line Approach*; Addison-Wesley, 2000.

[13] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal: *Pattern-Oriented Software Architecture: A System of Patterns*; Wiley, 1996

[14] E. Gamma, R. Helm, R. Johnson, J. Vlissides: *Design Patterns – Elements of Reusable Object-Oriented Software*; Addison-Wesley, 1994.

[15] I. Jacobson, M. Griss, P. Jonsson: *Software Reuse – Architecture, Process and Organization for Business Success*; Addison-Wesley, 1997.

[16] U. Seger, P. M. Knoll, C. Stiller: *Sensor Vision and Collision Warning Systems*; Technical Report 2000-01-C001, In: Proceedings of Convergence 2000, October 2000.

[17] T. Bertram, R. Bitzer, R. Mayer, A. Volkart: *CARTRONIC – An Open Architecture for Networking the Control Systems of an Automobile*, SAE International Conference and Exposition, Detroit, MI, February 23-26, 1998.

[18] V. Basili, D. M. Weiss: *A Methodology for Collecting Valid Software Engineering Data*; IEEE Transactions on Software Engineering, 10(6), pp. 728-738, 1984.

[19] M. Simos: *Organization domain modeling (ODM): Formalizing the Core Domain Modeling Life Cycle*; Symposium of Software Reusability (SSR-95), *ACM SIGSOFT Software Engineering Notes*, pp. 196-205, August 1995.

[20] J. Kuusela, J. Savolainen: *Requirements Engineering for Product Lines*; In: Proceedings of the 22nd International Conference on Software Engineering (ICSE-2000), pp.61-69, 2000.