International Scholarly Research Network
ISRN Software Engineering
Volume 2012, Article ID 424230, 7 pages
doi:10.5402/2012/424230

Research Article

Secure and Customizable Data Management for Automotive Systems: A Feasibility Study

Thomas Thüm,[1] Sandro Schulze,[1] Mario Pukall,[1] Gunter Saake,[1] and Sebastian Günther[2]

[1] *School of Computer Science, University of Magdeburg, P.O. Box 4120, 39016 Magdeburg, Germany*
[2] *Faculty of Sciences, Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium*

Correspondence should be addressed to Thomas Thüm, tthuem@ovgu.de

Received 11 November 2011; Accepted 7 December 2011

Academic Editors: O. Greevy and Y. K. Malaiya

Custom tailor-made database management systems (DBMS) are an essential asset, especially for embedded systems. The continuously increasing amount of data in automotive systems and the growing network of embedded devices can profit from DBMS. Restrictions in terms of processors, memory, and storage require customizable DBMS that contain only the needed functionality. We present AutoDaMa, a customizable DBMS designed for automotive systems. With AutoDaMa, it is possible to generate tailor-made DBMS for different scenarios, for example, by restricting the storage size of the DBMS or adding security-related features such as asymmetric and symmetric encryption. We demonstrate the feasibility of our approach through applying different tailor-made DBMS versions derived from AutoDaMa in an automotive testing environment. Our experience is that AutoDaMa can dramatically reduce the development effort and can increase reliability using efficient reuse mechanisms.

1. Introduction

About 90 percent of all innovations in modern cars rely on electronic data processing systems [1]. Current estimates assume that more than one gigabyte of software is installed in one automotive system and is expected to increase in future [2]. Thus, there is a huge potential to keep development cost for automotive software as low as possible. Current cars have dozens of sensors and actors that gather huge amount of data. This data is used by *Electronic Control Units (ECU)* to provide intelligent, safety-related, and comfort functions. The ECUs are connected using a bus system based on the protocols CAN, FlexRay, MOST, or LIN [3]. Unfortunately, every ECU has an own implementation to locally store the data, which hinders the efficient development of new functions, since new functions usually require to adapt the implementation of different ECUs.

We need data management components in automotive systems that guarantee certain properties based on the actual data and application scenario. While standard DBMS can ensure transactions, recovery, persistence, and integrity, they are not suitable for embedded systems with limited resources. Thus, customized DBMS are needed providing only the required functionality and desired properties. While providing tailor-made data management is discussed elsewhere [4], we focus on security aspects of tailor-made data management necessary in automotive systems [1, 5]. The large amount of sensors and new technologies such as *Car-2-Car (C2C)* or *Car-2-Infrastructure (C2I)* increase the overall complexity and enable attackers to assess the system [6].

We argue that the use of DBMS in automotive systems provides substantial support to tackle both mentioned problems, unstructured and inefficient data management, and ensuring data security. Others sketched an approach for automotive database management systems (DBMS) to store data from ECUs and to provide access to other ECUs [7]. For example, depending on the required accuracy, the speed could be measured at the wheels or using a GPS unit and continuously stored at a database. Other devices can read the current speed from the database for further usage, for example, to decide whether to lock the doors automatically or just to display the current speed.

In this work, we present a prototypical implementation of a tailor-made automotive DBMS, called AutoDaMa. We ported an existing highly customizable DBMS [4] to an industrial prototyping system, designed for developing

and testing software in automotive systems. Based on the AutoDaMa implementation, we make the following contributions. First, we demonstrate that it is feasible to provide a tailor-made DBMS for automotive systems by means of a prototypical implementation. Second, we show that security mechanisms integrated in a DBMS can prevent from unauthorized manipulations and thus increase the reliability and safety of the system. Third, we outline the benefits of a customizable DBMS for efficient and flexible data management in automotive systems.

2. Tailor-Made Software

We propose the use of tailor-made DBMS in automotive systems. But, it is not obvious how to develop tailor-made software systems. While there are several implementation techniques to build tailor-made software systems, we focus on feature-oriented programming, since we used this technique for our prototype.

A *feature* is a software engineering term that is meant to represent concrete requirements of a stakeholder [8]. More specifically, a feature provides the essential or additional functionality of a software system [9]. When we consider a DBMS, a feature for example is the supported operating system, for example, Unix or Windows. Another feature can be the storage, which is, for example, based on pages or a list index. Optionally, we can add data types using features. As these small examples show, features have different relationships to each other: features can be optional (pure addition of behavior), mandatory (strictly required to provide core requirements), and exclusive (features are alternative to each other).

Feature-oriented programming (FOP) puts a new dimension to the object-oriented paradigm and supports the modularization of features [10]. As a feature represents a requirement of a certain stakeholder, it usually crosscuts the modularization into classes, that is, a transaction feature needs to change a number of methods in several classes. Therefore, feature-oriented programming provides the ability to split classes into features, that is, the user can define several class fragments consisting of certain fields and methods. Thus, a feature consists of a set of class fragments.

Tailor-made software can be build using feature composition. Based on a selection of features, class fragments are composed to tailor-made classes that only contain the desired functionality. A software system built using feature composition is called a *variant*. The set of all variants that can be built from a set of features is called a *software product line*.

We use FeatureC++ to implement a software product line of DBMS for automotive systems called AutoDaMa. FeatureC++ is a language extension of C++ with support for features and their composition based on a feature selection [11]. By using FeatureC++, we gain the ability to provide different variants of AutoDaMa, which are tailor made for respective automotive systems.

Utilizing feature-oriented programming and software product lines to build tailor-made DBMS comes with advantages. First, new variants can be easily developed by implementing a new feature or just by combining already existing ones. Good reuse opportunities allow faster and less expensive development [12]. Second, efficient algorithms exist to test or verify all variants at the same time, instead of redundantly checking each similar variant from scratch [13, 14]. Hence, building reliable and tailor-made DBMS seems worth, and we study the feasibility for automotive systems.

3. Automotive Data Management

In this section, we give an overview of automotive systems and how they work. Subsequently, we figure out current problems of data management in such systems and how a customizable DBMS can help to overcome these problems.

3.1. Automotive Systems in a Nutshell. Automotive systems are complex, software-intensive systems, where up to 80 ECUs (including corresponding sensors and actors) are connected via bus systems. The data exchanged in such a system can be used by the ECUs to fulfill a certain function (in isolation or interaction with other ECUs) or to derive new data that are needed for certain functionality. For instance, given the information on wheel rotation in relation to the time needed for one rotation and the wheel size, an ECU can compute the speed, the revolution per minute, or the distance for a certain range of time.

In Figure 1, we show an exemplary and simplified part of such an automotive system. Besides the physical connection of the several components via different bus systems (CAN, MOST, etc.), we divided the system in *logical* subsystems. According to the function an ECU contributes to, it is assigned to the *power train, infotainment,* or *comfort* subsystem. All subsystems are connected to each other by a central device called *Gateway*. Furthermore, the different sensors (S_i) and actors (A_i) are directly connected with the ECUs where the sensors collect data, and the actors act on data in a predefined way.

As one can imagine, the data used in the different subsystems have different influence on the overall behavior of the system and thus different requirements. For instance, the power train subsystem has hard real-time constraints (<10 ms), which requires high transmission rates for data. Furthermore, manipulation of data that is used within the power train subsystem can have devastating consequences for the safety of the system *and* its occupants due to high accident risks. By contrast, manipulation of infotainment data may be annoying but not critical for life and limb.

Additionally, the particular devices (ECUs) have a local view on data, that is, each ECU is responsible only for the data used by itself. As a result, data handling is highly decentralized, heterogeneous, and unstructured in automotive systems. In fact, the logic for handling data is implemented as hardware solution and thus very inflexible with respect to changing requirements.

3.2. Problem Statement. Automotive systems are expected to become even more complex in near future due to new technologies such as Car-2-Car, Car-2-Infrastructure, and

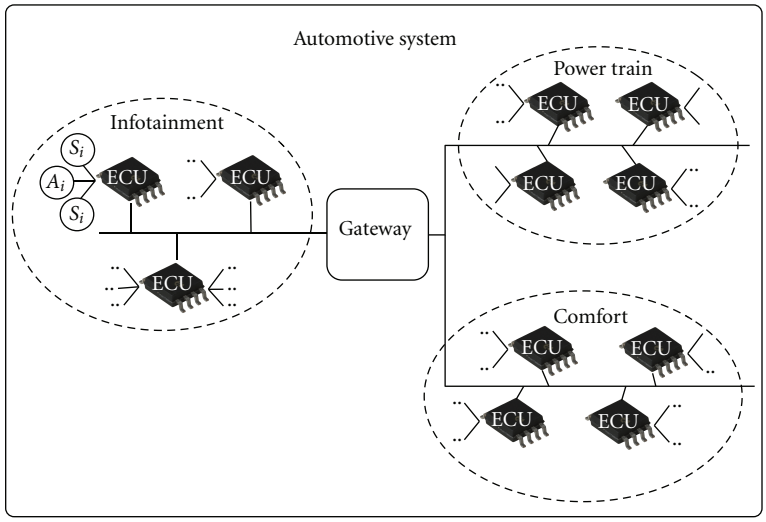

Figure 1: Exemplary part of an automotive system.

X-by-Wire [6]. This, in turn, comes along with an increase of the data that has to be managed within the system. We argue that the current solution for data "management" has reached its limit and will lead to severe problems in future. Firstly, the missing (uniform) data structure, as provided by a DBMS, increases the complexity and hinders a consistent view on the handled data. Second, the decentralized data handling leads to increased I/O operations and thus to an inefficient data access. In both cases, a data management system can mitigate the problems and even provide new capabilities such as consistent data validation and verification. Third, current solutions do not address data security aspects such as integrity, authenticity, and privacy though recent approaches show the vulnerability of automotive systems to malicious attacks [5, 15, 16]. Additionally, new technologies such as C2C or C2I even increase the risk of such attacks [17]. With a DBMS and its access management capabilities, these problems can be addressed efficiently.

However, common database implementations that provide the needed capabilities are not applicable due to the restrictive resource constraints of automotive systems. Moreover, the monolithic architecture of traditional implementations does not meet the different requirements of the different subsystems and its devices. Hence, we argue that we need a highly configurable data management that can be tailored to the specific requirements. We propose to use a software product line to address the demand for customization and present an exemplary implementation to achieve the following goals: First, improved flexibility and extensibility of data management in automotive systems by introducing tailor-made data management; second, integration of security aspects that can be customized to several levels; third, demonstration of the feasibility of automotive data management by means of an example.

4. Prototype AutoDaMa

We implemented the customizable DBMS AutoDaMa that runs on the MicroAutoBox [18]. MicroAutoBox is an embedded system provided by the company dSPACE designed for but not restricted to automotive scenarios. It can be connected to a CAN bus to communicate with other automotive devices and directly to a PC via serial port. The connection to a PC is used to program the MicroAutoBox and for debugging purposes.

AutoDaMa is a DBMS that can be customized to the needs of the application, for example, whether security features are needed, but also database management-specific options such as Indexes. Hence, it is possible to optimize the DBMS regarding several nonfunctional properties such as binary size, performance, or security issues.

Fortunately, we did not need to implement the DBMS from scratch. We based our implementation on FAME-DBMS [4], a feature-oriented DBMS for embedded scenarios which is highly customizable. In FAME-DBMS, several features can be deactivated to save resources strongly limited on embedded systems.

FAME-DBMS is written in FeatureC++ [11], a language to write feature-oriented C++ programs. The DBMS implementation is split over several feature modules each implementing a certain feature, for example, whether the DBMS is in-memory or a page replacement strategy or whether the DBMS supports the removal of tuples.

While FAME-DBMS is already intended to run on embedded systems, porting it to the MicroAutoBox was concerned with high effort due to hard restrictions for applications running on it. Hence, we only ported a subset of features of FAME-DBMS to the MicroAutoBox. In total, we ported 32 variants of FAME-DBMS, that is, DBMS variants that can be generated with the ported feature modules. For example, we omitted FAME-DBMS's support for SQL, and data can only be accessed using an API.

In Figure 2, we present all the features of AutoDaMa and their valid combinations in a feature model. AutoDaMa is decomposed into the features of FAME-DBMS such as the operating system, buffer management, storage management, and the data access operators.

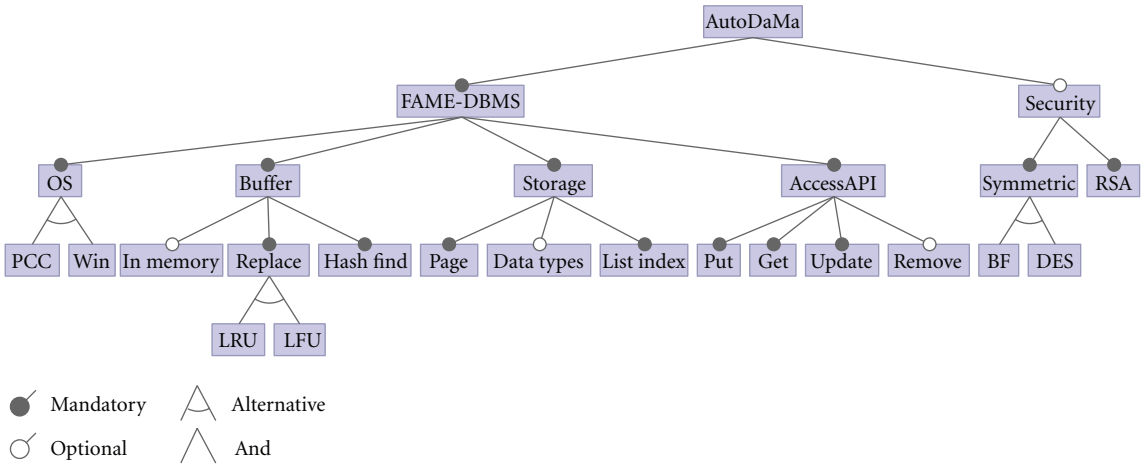

Figure 2: A feature model describing the valid feature combinations of AutoDaMa. AutoDaMa consists of the FAME-DBMS product line and some security-related features. Filled circles indicate mandatory features and empty circles optional features. An arc indicates that exactly one of the subfeatures is required, that is, the features are alternative to each other.

In automotive scenarios, security requirements play an important role. But not all data in the car should be secured as security usually comes with worse performance. Thus, an automotive security engineer should decide which data to secure and be able to generate a DBMS with an appropriate level of security. As FAME-DBMS has no support for security, we implemented some encryption mechanisms as features such that the DBMS can be customized to the intended level of security.

As shown in Figure 2, we implemented Blowfish and 3DES as symmetric encryption algorithms and RSA for asymmetric encryption. In our prototype, an asymmetric algorithm is always combined with a symmetric algorithm to gain the performance of symmetric encryption together with the security of asymmetric encryption. A total number of 96 DBMS can be generated from AutoDaMa.

Using a DBMS in an automotive context seems worth to manage the data produced by several sensors, while DBMS for desktop computers or servers have a very large footprint and have lots of unused functionality. AutoDaMa gives us the opportunity to build customized DBMS that contain only the functionality needed, and all customized DBMS have a common code base. Both help to build reliable and secure DBMS.

With AutoDaMa, DBMS can be generated fulfilling certain level of security to optimize the performance for given security requirements, that is, without encryption or a certain encryption with known performance and security properties. Furthermore, the source code is reliable because of the high reuse of software artifacts using feature-oriented programming.

5. Results and Experiences

In this section, we present two application scenarios to show the feasibility of AutoDaMa. Furthermore, we discuss our implementation and figure out future challenges that have to be addressed for establishing a fully fledged data management solution in practice.

5.1. Exemplary Application Scenarios. To demonstrate the feasibility of a (tailor-made) DBMS in automotive systems and its benefits, we have chosen two scenarios: one where data is only exchanged and stored and one where additionally security mechanisms are integrated.

Scenario 1. For our scenarios, we use a simple automotive system that measures and transmits the *revolution per minute (RPM)* data of a car. We depict the general setup in Figure 3.

The central ECU is our prototyping system that serves as gateway in our environment and contains a certain configuration of our AutoDaMa product line. For the first scenario, we configured the DBMS with the optional in-memory feature but without the `Security` feature. Due to the usage of `in-memory` storage, we are able to fulfill the real-time requirements of the systems. These requirements especially hold for data such as speed or revolution, which is stored continuously but volatile.

Additionally, we have two ECUs that exchange data via the gateway, namely, *RPM Sensor* and *RPM Display*. The first ECU is connected to a sensor that continuously delivers the RPM data to the ECU. Subsequently, this data is sent to the AutoDaMa variant (i.e., the gateway) where it is stored using an update operation. Hence, only the last data, sent to the DBMS, is stored. The latter ECU (i.e., RPM Display) requests the data from the DBMS to display it immediately to the driver within the dashboard. In case that the automotive system shuts down (e.g., switching off the car), the DBMS writes back the current value to nonvolatile storage (here flash disk).

Scenario 2. Although the first scenario provides data management capabilities to automotive systems, it does not solve the second problem we mentioned: data or IT security. For instance, with the previous scenario it is still possible to

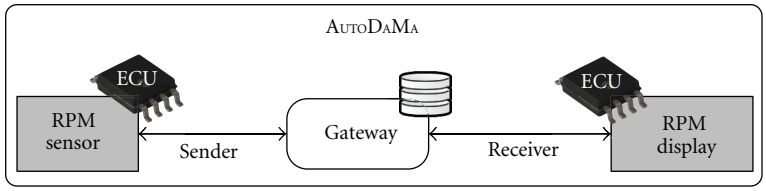

Figure 3: Basic setup of our automotive system.

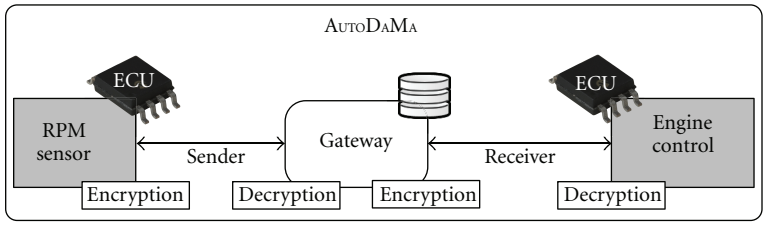

Figure 4: Setup of automotive system with encrypted communication.

connect to the bus system with an external, nonauthorized device and perform attacks such as *man-in-the-middle* in order to tamper (safety related) data [5]. Hence, with our second scenario, we aim at ensuring security aspects such as integrity or authenticity of the data. Therefore, we selected the optional `Security` feature together with its subsequent features `BF` and `RSA` (cf. Figure 2) that implement concrete encryption algorithms. As a result, we could generate a variant of AutoDaMa that includes security mechanisms.

In Figure 4, we depict the resulting scenario. Due to the security mechanisms in the DBMS that perform encryption of the data, we had to extend the other ECUs as well, in this case *RPM Sensor* and *Engine Control*. Despite this, communication and storage of data takes place as described in Scenario 1, using again the revolution data. Most notably, the data handling at each ECU changed since the data had to be encrypted after receiving and decrypted before sending. As result, we can ensure the integrity and authenticity of the data. For instance, imagine an attacker connects an external device to the system and injects malicious data. Because the attacker is not aware of the encryption key, he can not encrypt his data. Consequently, the DBMS can detect nonauthorized data and reject it. Additionally, we ensure the privacy of data because the attacker can not read the encrypted data without knowledge on the encryption key.

Results. We executed our scenarios in different time frames (e.g., 10 minutes, 20 minutes) and with different, randomly chosen values for the RPM data in the range of 0 and 4000. In both scenarios, storing and providing (which corresponds to write and read operation in common DBMS) data was possible without any problems. Furthermore, we logged the data initially sent to the DBMS and the data that arrived at the receiver and observed that they were consistent (e.g., no information get lost or changed). Finally, we were able to fulfill the time constraints of the CAN bus. Hence, the data management systems had no negative effect on the performance of the system.

5.2. Discussion. Establishing a DBMS for automotive systems with its complex, heterogeneous devices, and highly restrictive resources is not a trivial task. We have shown by a prototypical implementation that tailor-made data management, using new software engineering techniques, may overcome some of these burdens. Nevertheless, this was just a first step, and we observed different problems that have to be addressed in future.

First of all, we demonstrated the applicability of automotive DBMS with a small example that abstracts from the complexity of common automotive systems. Considering an entire system, more research and more studies have to be performed especially to establish a system-wide DBMS. A major point of interest is how we can address the different requirements to a DBMS even within a *single* system. For instance, a central DBMS with all features needed by any ECU would lead to an oversized system. Furthermore, this would definitively be a bottleneck with respect to the data flow, and thus, the performance would be unacceptably low. Alternatively, several instances of the AutoDaMa product line could solve these problems but exhibit another problem that has to be solved: how can different instances (of a DBMS) interact *efficiently* in such a scenario? Some pioneer work has been done in this field, but still many questions remain [19]. Possibly, the answer can be given by adapting *distributed* DBMS concepts and techniques to automotive systems or even investigate new ones.

Second, the limited capacity of the CAN bus, especially the restricted packet size for sending messages, could bare problems for a DBMS solution. Especially in the case of adding extra information for security or meta data, this could hinder an efficient and practicable solution. However, with new and more sophisticated solutions like the FlexRay bus, these problems are mitigated or even vanish.

Finally, the introduced security mechanisms require a corresponding infrastructure (e.g., public key infrastructure) to unfold their full potential. Unfortunately, current ECUs do not support such an infrastructure. Hence, car manufactures

have to be aware of the security risks and provide technical solutions, so that such mechanisms can be integrated by default.

Nevertheless, we are convinced that tailor-made data management is a sustainable solution to overcome the problems of managing the huge amount of data in automotive systems. With the presented prototype, we provided first insights on the feasibility of such a solution and a first step towards its realization.

6. Related Work

Tailor-made database management, especially for embedded systems, is not a new idea, and thus different work in this field exists. First, the COMET DBMS for embedded real-time systems is the most similar approach to our work [20, 21]. The focus of this approach is on component-based software development using aspect orientation. As a result, COMET DBMS is also a flexible and customizable database management system for automotive systems. However, due to the component approach, tailoring can be done only on a coarse-grained level compared to AutoDaMa. Furthermore, the authors do not consider data security as part of the DBMS.

Another approach is FAME-DBMS, which was the starting point for our implementation [4]. Although this leads to some commonalities between both systems, there are significant differences. First, FAME-DBMS is designed and implemented for sensor networks rather than automotive systems, which implies differences in the real-time behavior of the systems. Second, FAME-DBMS does not include any mechanisms for ensuring security of the managed data.

In the same way, Leich et al. present a lightweight storage manager for sensor networks using stepwise refinement and feature-oriented programming [22]. Similar to us, they argue that different nodes in the network have different requirements to the DBMS. By contrast, they focus mainly on the mechanisms used for implementation rather than the applicability in real-world scenarios. Furthermore, security issues are omitted as well, while they are included in our approach.

Beyond the mentioned systems, different special-purpose database systems exist. For instance, GnatDB [23] is a DBMS for digital right management while PICO DBMS [24] provides data management functionality for smart cards. However, these systems are different to AutoDaMa due to its specific application domain. Furthermore, these systems do not focus on customizing a DBMS.

7. Conclusion

The continuously growing number of sensors in automotive systems and the networked usage of the accrued data can profit from DBMS. Clearly, not every DBMS such as Oracle can and should be used in this context. The reason is that on embedded systems resources are restricted, and the DBMS should bring only the functionality that is actually needed.

We put an existing customizable DBMS, namely, FAME-DBMS into an automotive context, that is, we ported it to the MicroAutoBox which communicates with other automotive devices using the CAN bus. Furthermore, we extended the DBMS by new security features essential in the automotive context.

Our experience showed that while there is some effort needed to port an existing implementation to the embedded device, and the effort is worth due to several reasons. First, well-established concepts for DBMS can be reused in embedded scenarios and do not need to be implemented from scratch. Second, a customizable DBMS comes with the advantage that different DBMS are generated from a common source code, and thus the implementations are validated and well tested. Third, even for one automotive, systems different DBMS may be generated for different security requirements or requirements to the data management, for example, persistent or in-memory storage.

Future work should evaluate the benefit of DBMS in automotive scenarios using larger case studies in an industrial context. Furthermore, also other DBMS designed for embedded systems could be appraised towards their utility in automotive systems.

Acknowledgment

This work has been supported by the European Commission through the EFRE programme COMO under Contract no. C(2007)5254.

References

[1] M. Wolf, *Security Engineering for Vehicular IT Systems*, Vieweg and Teubner, Berlin, Germany, 2009.

[2] A. Pretschner, M. Broy, I. H. Krüger, and T. Stauner, "Software engineering for automotive systems: a roadmap," in *Proceedings of Future of Software Engineering (FOSE '07)*, pp. 55–71, IEEE Computer Society, 2007.

[3] W. Zimmermann and R. Schmidgall, *Bussysteme in der Fahrzeugtechnik*, Vieweg and Teubner, 3rd edition, 2007.

[4] M. Rosenmüller, N. Siegmund, H. Schirmeier et al., "FAME-DBMS: tailor-made data management solutions for embedded systems," in *Proceedings of the Workshop on Software Engineering for Tailor-Made Data Management (SETMDM '08)*, S. Apel, M. Rosenmüller, G. Saake, and O. Spinczyk, Eds., pp. 1–6, ACM, 2008.

[5] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN Networks—practical examples and selected short-term countermeasures," in *Proceedings of the International Conference on Computer Safety, Reliability, and Security (SAFECOMP '08)*, pp. 235–248, Springer, 2008.

[6] T. Ernst and A. De La Fortelle, "Car-to-car and car-to-infrastructure communication system based on NEMO and MANET in IPv6," in *Proceedings of the Intelligent Transportation System World Congress (ITSWC '06)*, 2006.

[7] S. Schulze, M. Pukall, G. Saake, T. Hoppe, and J. Dittmann, "On the need of automotive data management in automotive systems," in *Proceedings of the GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW '09)*, Lecture Notes in Informatics, pp. 217–227, Gesellschaft für Informatik (GI), March 2009.

[8] K. C. Kang, S. G. Cohen, J. A. Hess, W. E. Novak, and A. S. Peterson, "Feature-oriented domain analysis (FODA)

feasibility study," Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.

[9] C. Kästner, S. Apel, and M. Kuhlemann, "Granularity in software product lines," in *Proceedings of the 30th International Conference on Software Engineering (ICSE '08)*, pp. 311–320, ACM, May 2008.

[10] C. Prehofer, "Feature-oriented programming: a fresh look at objects," in *Proceedings of the European Conference on Object-Oriented Programming (ECOOP '97)*, vol. 1241 of *Lecture Notes in Computer Science*, pp. 419–443, Springer, 1997.

[11] S. Apel, T. Leich, M. Rosenmüller, and G. Saake, "FeatureC++: on the symbiosis of feature-oriented and aspect-oriented programming," in *Proceedings of the International Conference on Generative Programming and Component Engineering (GPCE '05)*, R. Glueck and M. Lowry, Eds., vol. 3676 of *Lecture Notes on Computer Science*, pp. 125–140, Springer, 2005.

[12] D. Beuche, *Composition and construction of embedded software families*, Ph.D. thesis, University of Magdeburg, Magdeburg, Germany, 2003.

[13] T. Thüm, I. Schaefer, M. Kuhlemann, and S. Apel, "Proof composition for deductive verification of software product lines," in *Proceedings of the 4th IEEE International Conference on Software Testing, Verification, and Validation Workshops (ICSTW '11)*, pp. 270–277, IEEE Computer Society, 2011.

[14] S. Apel, C. Kästner, A. Größlinger, and C. Lengauer, "Type safety for feature-oriented product lines," *Automated Software Engineering*, vol. 17, no. 3, pp. 251–300, 2010.

[15] T. Hoppe and J. Dittmann, "Sniffing/replay attacks on can buses: a simulated attack on the electric window lift classified using an adapted CERT taxonomy," in *Proceedings of the Workshop on Embedded Systems Security (WESS '07)*, 2007.

[16] A. Barisani and B. Daniele, "Unusual car navigation tricks: injecting RDS-TMC traffic information signals," in *Proceedings of the CanSecWest Conference*, 2007.

[17] A. Lang, J. Dittmann, S. Kiltz, and T. Hoppe, "Future perspectives: the car and its IP-address—a potential safety and security risk assessment," in *Proceedings of the International Conference on Computer Safety, Reliability and Security (SAFECOMP '07)*, pp. 40–53, 2007.

[18] R. Kraußе, *Entwicklung und evaluierung eines sicheren datenbankmanagementsystems für automotive Systeme*, M.S. thesis, University of Magdeburg, Magdeburg, Germany, 2010.

[19] S. S. ur Rahman, V. Köppen, and G. Saake, "Cellular DBMS: an atempt towards biologocally-inspired data management," *Journal of Digital Information Management*, vol. 8, no. 2, pp. 117–128, 2010.

[20] D. Nyström, A. Tesanovic, C. Norström, J. Hansson, and N.-E. Bankestad, "Data management issues in vehicle control systems: a case study," in *Proceedings of the Euromicro Conference on Real-Time Systems (ECRTS '02)*, pp. 249–256, 2002.

[21] D. Nyström, A. Tesanovic, C. Norström, and J. Hansson, "COMET: a component-based real-time database for automotive systems," in *Proceedings of the Workshop on Software Engineering for Automotive Systems*, 2004.

[22] T. Leich, S. Apel, and G. Saake, "Using step-wise refinement to build a flexible lightweight storage manager," in *Proceedings of the East-European Conference on Advances in Databases and Information Systems (ADBIS '05)*, pp. 324–337, Springer, 2005.

[23] R. Vingralek, "GnatDb: a small-footprint, secure database system," in *Proceedings of the International Conference on Very Large Databases*, pp. 884–893, 2002.

[24] L. Bobineau, C. Bouganim, P. Pucheral, and P. Valduriez, "PicoDMBS: scaling down database techniques for the smartcard," in *Proceedings of the International Conference on Very Large Databases*, pp. 11–20, 2000.