



The Institution of Electronics and
Telecommunication Engineers

IETE Technical Review

Volume 26 • No. 5 • Sep-Oct 2009

www.ietejournals.org

Subscriber Copy : Not for Resale

Towards Robust Data Storage in Wireless Sensor Networks

Norbert Siegmund, Marko Rosenmüller, Guido Moritz¹, Gunter Saake and Dirk Timmermann¹

Department of Computer Science, Institute of Technical and Business Information Systems,
University of Magdeburg, Universitäts Platz 2, 39106 Magdeburg,

¹ Department of Computer Science and Electrical Engineering, Institute of Applied Microelectronics and Computer Engineering,
University of Rostock, Richard-Wagner Str. 31, 18119 Rostock-Warnemuende, Germany

Abstract

Wireless sensor networks are used for sensing data if wired communication is not suitable. Connection losses and depletion of nodes, however, result in reduced data availability of such networks. This is problematic in upcoming scenarios like health care or home automation where data availability is highly important. In this paper, we present an approach to provide robust data storage for wireless sensor networks. We achieve this goal by providing FAME-DBMS, a customizable database management system which can be tailored according to the varying requirements of a sensor network. FAME-DBMS provides reliable data storage using security and integrity features, transaction management and recovery and a customizable query engine. Since data reliability in wireless sensor networks also suffers from node failures, we propose a new *S-RAID* storage layer inspired from the RAID approach of server systems. This S-RAID is integrated in FAME-DBMS to store data redundantly and distributed in a wireless sensor network and thus provides access to data even when nodes fail.

Keywords

Data, Management, Reliability, Software product lines, Wireless sensor networks.

1. Introduction

Nowadays, sensor networks are being increasingly used in real-life scenarios, where data recording is an integral part of the overall application. Wireless Sensor Networks (WSNs) are required if wired connectivity is not suitable or even impossible, e.g., in inaccessible environments. Different radio technologies like Bluetooth [1] and IEEE 802.15.4 [2] are proposed as suitable to provide the needed low-power communication interfaces. Software development for WSNs requires high effort to overcome implicit restrictions like highly constrained resources. This results in restrictions on software footprint, processing power and hardware life time.

Furthermore, unreliable communication and depleted nodes have to be considered in WSNs. This is particularly important for critical systems as found in home automation and healthcare scenarios, where vital parameters of patients and the environment have to be monitored. Considering the application scenarios and possible node failures, robust data management in WSNs gains more and more importance.

Reliable data management raises the requirement of continuous data availability, even when nodes fail, e.g., because of power loss. In order to provide the nec-

essary robustness, often gateway or proxy concepts are applied in the area of sensor networks. Data is recorded and transmitted to proxies, which are connected to mains power systems. Considering the facts of high hardware costs, the size of such systems and the lack of scalability, these solutions are inappropriate for WSNs in the long term. A more feasible approach is to deploy intelligent system functionality to compensate single node failures. In the domain of computer and server systems, concepts for reliable distributed data storage and its organization already exist. Nevertheless, resource constraints as occurring in the domain of WSNs are usually not considered in existing computer and server solutions.

Besides limited resources, the heterogeneity of hardware and different application roles in WSNs often lead to reimplementations of the data management functionality, tailored for a single system [3]. This common practice increases time to market and development costs while leading to poor quality of software. We propose to use a customizable data management infrastructure that can be tailored according to the varying requirements in WSNs [4]. This includes not only different requirements according to used hardware of different nodes but also varying requirements on reliability. For example, support for transaction management, recovery or data security increases the reliability of data in a sen-

sensor network but is not mandatory for every node and even not for every sensor network. With *FAME-DBMS* (<http://fame-dbms.org>), we implemented such a customizable Database Management System (DBMS) that can be used on embedded devices like sensor nodes [5]. *FAME-DBMS* provides functionality required by larger data storage nodes of a sensor network as well as a customizable query interface. This query interface avoids any overhead by specifying a tailor-made query language [6]. In order to increase robustness of data storage, we propose to extend such a system by adopting the concept of RAID storage [7], known from server systems. The resulting data storage layer for WSNs can be used as the underlying layer of our customizable DBMS to achieve distributed data storage. In combination, we can provide the required data robustness and integrity necessary for future application scenarios of WSNs.

2. Customizable Data Management

Resource constraints and diversity in hardware of embedded systems force developers to create tailored software that provides only required functionality.

Software Product Lines (SPLs) enable the development of software that can be customized to different use cases and application scenarios with minimized development effort [8]. Products of an SPL differ in terms of selected *features*, i.e., the provided functionality. For example, one DBMS variant might provide a feature *Recovery* to recover data after failure. Another DBMS might not provide this functionality. The features of an SPL and relationships between the features are described in a feature model [9,10]. A feature model defines if a feature is optional or mandatory as well as additional constraints between features, e.g., if a feature *requires* another feature. A typical visualization of a feature model is a feature diagram as depicted in Figure 1. Such a diagram is a hierarchical representation of all features of an SPL features, where the topmost feature, i.e., the root of the tree, represents the domain concept. Features denoted with an empty dot are optional (e.g., feature *Transaction* in Figure 1), and mandatory features are represented by a filled dot (e.g., feature *Storage*). To derive a variant from an SPL, i.e., a concrete program, a stakeholder selects the features that fulfill her requirements and a composition mechanism is used to generate an executable application.

2.1 Implementation Techniques for Customizable Data Management

There are different approaches to implement software product lines. Two prominent approaches are the *C pre-processor*, i.e., `#ifdef` statements of the C/C++ programming language, and composition of new variants based on *components* and *frameworks*. Both approaches have

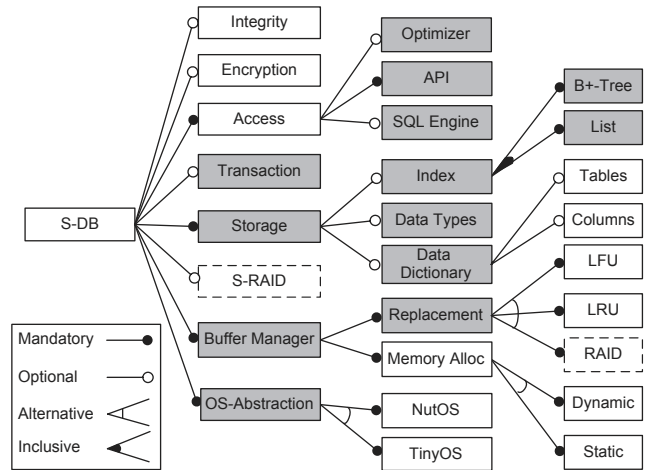


Figure 1: Excerpt of the feature diagram of *FAME-DBMS*. RAID features are currently not implemented.

benefits but also deficiencies. For example, drawbacks of preprocessor statements are degradation of readability of the source code [11] and missing modularization of features, which raises problems in software evolution and even hinders the elimination of dead features [12].

Components, on the other hand, provide good *separation of concerns* [13]; however, the achieved customizability is usually limited to parts of a DBMS [14-16]. This is caused by *crosscutting features* that are scattered all over the entire DBMS, e.g., the transaction management system, which is hard to modularize in a component. Furthermore, very small components degrade the performance due to a communication overhead [3]. The result is an overhead not suitable for embedded systems especially for WSNs.

In contrast to the C preprocessor and components, new programming paradigms such as *Feature-Oriented Programming* (FOP) [17,18] and *Aspect-Oriented Programming* (AOP) [19] are promising for implementing SPLs. For example, Nystrom *et al.* used AOP in combination with components to modularize crosscutting features that affect multiple other components [20]. Tešanović *et al.* used AOP to re-factor the C version of Oracle Berkeley DB to modularize crosscutting features that had been implemented with the C preprocessor before [21]. In the following, we will focus on FOP since it provides proper separation of concerns and at the same time also allows us to modularize crosscutting features.

2.2 Feature-oriented Programming

Feature-oriented programming (FOP) treats features of software as basic elements of the whole development process. It allows compositions of families of similar pro-

grams based on required features of a domain. *Feature modules* are code units representing a feature and implementing increments in the program's functionality [18]. To derive different variants of a program, a base program is composed with a user-defined set of features, e.g., a basic storage mechanism with additional security features. In previous work, we have re-factored and implemented DBMS product lines [4,5] to demonstrate applicability of FOP for development of customizable DBMS.

With Feature C++ [22,23], we developed an FOP language extension for the C++ programming language. It supports static composition of feature modules and allows in applying FOP to software systems intended for resource-constrained environments. Feature C++ uses a code transformation to C++ to benefit from compilers and further tool chains (e.g., profiler and debugger) that exist for most computing systems and provides performance equivalent to implementations that use the C preprocessor [5].

2.3 FAME-Data Base Management System

Using Feature C++, we have developed FAME-DBMS, a DBMS product line to be applied on highly resource-constrained embedded devices [5]. Due to the use of FOP, we are able to provide the necessary variability, which is required in the domain of embedded devices [4]. In Figure 1, we show a subset of the feature model of FAME-DBMS. Shaded features encapsulate additional features that are not shown. For example, feature *Data Types* has sub-features integer, varchar, etc. The DBMS provides access via an API or optional access via a subset of SQL (feature *SQL Engine* in Figure 1). By integrating only the required features in a specific variant, generated from the FAME-DBMS product line, we can reduce the functional overhead significantly, which results in lower application footprint and reduced energy consumption. Furthermore, dedicated variants of FAME-DBMS can be tailored and deployed for heterogeneous devices according to the specific hardware constraints and application scenarios.

3. Robust Data Storage

There are a number of factors that influence the robustness of a sensor network. Reasons are low-power wireless communication, which results in connection loss and vulnerability of transferred data. Limited battery capacity implies further restrictions concerning limited lifetime of the sensor node. In the following, we present the resulting requirements on data management to achieve robust data storage and transfer despite the mentioned problems.

3.1 Reliability Features of FAME-Data Base Management System

We have developed FAME-DBMS as a prototype for

customizable data management for sensor networks. Since different nodes of a sensor network have different requirements on reliability and customizability, a DBMS with customizable reliability features is needed. The following overview presents features implemented in FAME-DBMS whose composition depend on the reliability requirements of a concrete sensor network and application scenario. The presented features are depicted in Figure 1.

Security and Integrity: Wireless low-power communication results in vulnerability of transmissions. Invalid data and system critical commands might be injected. Furthermore, sensors and data nodes are often accessible directly and thus critical data should be encrypted. To assure integrity and confidentiality of data distribution and storage, mechanisms like checksums, authentication and authorization might be required. Features *Encryption* and *Integrity* of FAME-DBMS provide simple encryption and integrity algorithms, which can be extended according to the requirements of the concrete application.

SQL Engine: Secure query interfaces for sensor networks must be provided to allow secure data exchange with external networks. When using SQL, only the actually necessary subset of the language should be supported in order to avoid attacks that use vulnerability of parts of the query processor implementation. This reduces the possibilities of manipulation and optimizes the resulting footprint of the implementation. Furthermore, with an increased number of supported language constructs, the possibility of data corruption due to misuse is raised. However, the actually needed parts of the language depend on the application scenario and vary for different sensor networks. In order to provide a customizable interface, we use a *family of SQL dialects* that allows us to configure the actually required dialect [6]. Different SQL dialects can be created by adding functionality to the basic *select-from-where* queries. FAME-DBMS allows for including specific grouping functionalities, transactions or queries needed only for special domains. A concrete dialect is defined by a selection of features that describe the functionality needed for this dialect. Based on the chosen dialect, we can derive the query engine of a concrete DBMS.

Transaction Management: In WSNs, parallel communicating nodes in overlapping radio ranges and thereby simultaneously competing data processing is a key characteristic. In server systems, this concurrency problem is solved with transaction processing. Hence, transactions might also be used for stored data on data aggregation nodes in sensor networks. In contrast to server systems, however, a transaction management for embedded devices has to provide only minimal functionalities in order to avoid a large application footprint. Hence, serial execution of queries is often

sufficient for sensor networks.

Recovery: Features *Recovery* and *Logging* of FAME-DBMS have been designed and implemented for aggregation nodes. They enable data recovery of nodes that has temporarily failed. Recovery of data is required for aggregation nodes only because sensors usually store recently measured values non-persistently and cluster gateways only provide access to the network and do not store data.

S-RAID: The features presented above are also applied in server DBMS and are implemented as specialized and modularized variants to fulfill all constraints of sensor networks. This approach, however, is not sufficient to provide robust data storage in sensor networks, since nodes are not connected to mains power systems and may fail due to the use of batteries and faulty communication channels. In order to provide even access to data of a failed node of a sensor network, we propose to adopt the RAID approach (which is briefly described in Section 3.3) for sensor networks and store data distributed over different nodes. Feature *S-RAID* shown in Figure 1 represents functionality of the data management system used to store data in an underlying abstract storage layer. Since this feature is optional, we can create variants of FAME-DBMS that use the underlying RAID layer as well as variants that do not. This variability is needed because of different reliability requirements depending on the actual application scenario (i.e., some nodes do not need the RAID layer at all). In the last part of this section, we describe the planned RAID layer in more detail.

3.2 Data Base Management System Variants for Sensor Nodes

FAME-DBMS provides different variants of storage solutions for different node roles that can be found in a WSN. These different roles have different requirements on data management:

- Endpoint sensor: Sensor nodes require only rudimentary data management functionality. The main task of this node is to sense data and efficiently store simple key-value pairs. A data management solution for this role has to provide a minimal footprint in order to be deployed on highly resource-constrained sensor devices.
- Cluster gateway: The cluster gateway is the interface for querying data of a WSN. Therefore, a data management variant includes SQL support and a buffer manager optimized for sensor network queries. However, the variant provides only rudimentary storage functionality.
- Data aggregator: Nodes that aggregate data have to provide advanced data management functionality to collect and store the data of a group of endpoint sensor nodes. Additionally, incoming queries are mainly executed on this node. Thus, the data manage-

ment system has to provide indexes for fast data access and buffer management for data storage. Because of lower hardware limitations compared to endpoint sensors, a data management system may also include various other features, e.g., data encryption and integrity checks.

According to the requirements on data management presented above, we can deploy different variants of FAME-DBMS. As shown in Table 1, the binary size of the different variants of FAME-DBMS ranges from 10 KB for endpoint sensor nodes to 52 KB for data aggregation nodes. However, these are only the basic implementations, which might be customized according to the specific application scenario. For example, feature *Encryption* might be added to all nodes to provide secure data storage and exchange.

3.3 The RAID Layer

Improved reliability of data storage in the domain of server systems is usually achieved by using distributed DBMS. Data is stored redundantly in multiple physically distinct server systems. Each system contains a complete DBMS, which is the main reason why this solution cannot be applied to WSNs. To achieve distributed data storage on resource constrained devices, we currently develop a new abstract storage layer, which is inspired by the mature RAID concept of server systems.

The original RAID approach, presented by Patterson et al. [7], handles the issues of unreliable computer hard drives by adding intelligent system functionalities and storing data distributed over multiple hard drives. A single node in a WSN is considered by the same unreliability as an *inexpensive disk*. Due to the high number of nodes in a WSN, RAID concepts can be adopted to provide improved data integrity for WSNs too.

Due to the potential overhead of storing data on more than one node at the same time, memory usage is one of the most significant constraints when porting RAID techniques and concepts to the domain of WSNs. Hence, we introduce a new role for nodes that are responsible for storing data of endpoint sensing nodes. The storage role is ascribed to nodes with proper memory and energy resources to store data and compensate the increased communication efforts.

Using special storage roles is promising due to the

Table 1: Binary size of FAME-DBMS in three different variants

| Node role | Features | Binary size |
|-----------------|----------|-------------|
| Endpoint sensor | 5 | 10 KB |
| Cluster gateway | 8 | 36 KB |
| Data aggregator | 12 | 56 KB |

commonalities to the above presented computer storage systems. In computer networks and WSNs, specialized and optimized nodes are used for redundant data storage. According to computer networks, the number of storage nodes is small compared to the overall number of nodes in the network. In order to reduce the number of nodes and the communication effort, single nodes can be applied with more than one role. For example, the storage role can be combined with the data aggregator role.

The new approach, which is called *Sensor-RAID* (S-RAID), is integrated into the software running on a node as shown in Figure 2. It provides a layer that hides the complexity of storing data in the RAID and can be used by an application layer as well as by a DBMS. To integrate S-RAID storage in a DBMS, the FAME-DBMS product line has to be extended with functionality for using the S-RAID storage represented by feature *S-RAID* in Figure 1. Including the S-RAID feature to be concrete, DBMS provides the range of functions required for storing data records in the RAID layer. The actually provided storage mechanisms have to be selected by a stakeholder and depend on the application scenario. Thus, the DBMS may store different data records like configuration, current sensor values or aggregated sensor values independently in local storage, main memory or in the S-RAID. Such a classification of data is necessary because storing data in the S-RAID not only increases reliability but implies increased communication between nodes for data synchronization. This is critical for wireless networks due to limited energy capacities.

Miscellaneous adoptions are required, because standard RAID concepts highly rely on underlying file systems and are optimized for appropriate data access times. Our approach picks up the concepts for robust storage only, not for latency optimization. Thus, we define different classes with different levels for the S-RAID, which slightly differ from existing RAID technology. In the following, we define the needed classes *Copy*, *Data Priority*, *Fragmentation* and *Hops*.

- **Copy:** For S-RAID solutions in wireless sensor networks (WSNWs), the number of copies for a datum has to be specified. For computer RAID-1 system, one copy is used. For S-RAID, even more copies might be useful. Level: number of copies.
- **Data Priority:** The heterogeneous hardware and the changing environment in a WSNW can cause inaccessible communication routes and exhausted memory. Data prioritization is required to prefer specific data for redundant storage in case of unavailable resources. Levels: priorities and related protocols for prioritization.
- **Fragmentation:** According to computer RAID systems, data may not only be copied but also fragmented. By using specific algorithms, data can be fragmented and stored distributed over nodes. Fragmentation might result in reduced communication efforts for single nodes due to the reduced data size. Nevertheless, fragmentation implies a protocol overhead for transmission and processing. For data fragmentation, application of erasure code [24] might be applicable. Level: fragmentation protocol.
- **Hops:** The deployment of S-RAID concepts causes higher energy consumption due to the increased communication effort. To limit this energy consumption, the number of hops a node can synchronize its data with, is limited to either physical or logical values. Level: number of hops.

Separating S-RAID and DBMS has a significant benefit over an integrated solution; applications can use the S-RAID as well and can provide robust data storage without using a DBMS, e.g., for storing configuration data.

4. Conclusion

In this paper, we have presented an approach to increase the robustness of data storage in wireless sensor networks. We achieve this goal by providing FAME-DBMS, a DBMS product line, which can be tailored according to the requirements of different hardware platforms and roles of nodes in a sensor network. Reliable data storage of a concrete DBMS generated from

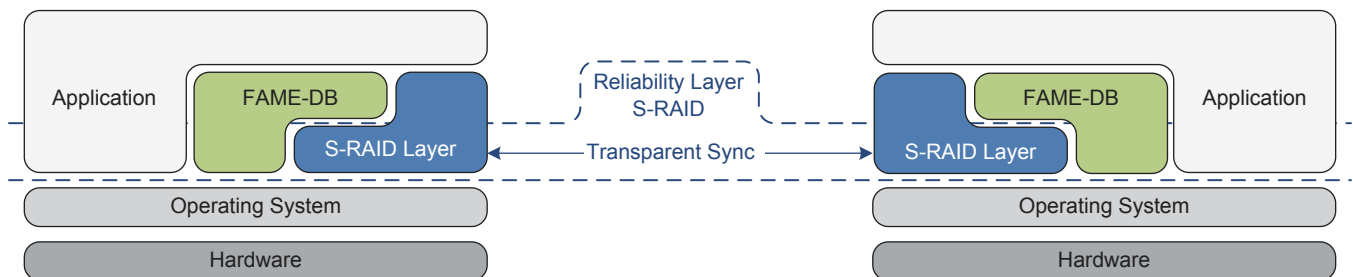


Figure 2: Integration of a reliability layer (S-RAID), FAME-DBMS and applications in nodes of a sensor network.

the product line is provided by including security and integrity features, transaction management and recovery and a customized query engine. Since data reliability in WSNs suffers from node failures, we have presented a new *S-RAID* storage layer concept inspired by the RAID approach of server systems. This *S-RAID* is integrated in a DBMS to store data redundantly and distributed in a WSN and thus provides access to data even when single nodes fail. In future work, we aim at implementing the RAID layer in order to increase the reliability of data. This includes an extension of FAME-DBMS for accessing the *S-RAID* as well as data recovery strategies required in case of node failures.

5. Acknowledgments

Norbert Siegmond and Marko Rosenmüller are funded by the German Ministry of Education and Science (BMBF), project 01IM08003C. The presented work is part of the ViERforES (<http://vierfores.de>) project.

References

1. Bluetooth SIG: BLUETOOTH SPECIFICATION. Technical report (2007).
2. IEEE Computer Society: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Technical report (2006).
3. S. Chaudhuri, and G. Weikum: Rethinking Database System Architecture: Towards a Self-Tuning RISC-Style Database System. In: Proceedings of the International Conference on Very Large Data Bases (VLDB), Morgan Kaufmann (2000) 1-10.
4. T. Leich, S. Apel, and G. Saake: Using Step-Wise Refinement to Build a Flexible Lightweight Storage Manager. In: Proceedings of the 9th East-European Conference on Advances in Databases and Information Systems (ADBIS 2005). Lecture Notes in Computer Science (LNCS), Berlin/Heidelberg, Springer-Verlag (2005) 324-337.
5. M. Rosenmüller, N. Siegmond, H. Schirmeier, J. Sincero, S. Apel, and T. Leich, *et al.*: FAME-DBMS: Tailor-made Data Management Solutions for Embedded Systems. In: EDBT'08 Workshop on Software Engineering for Tailor-made Data Management (SETMDM). (2008) 1-6.
6. M. Rosenmüller, C.Kästner, N. Siegmond, S. Sunkle, S. Apel, and T. Leich, *et al.*: Sqlà la carte - toward tailor-made data management. In: 13. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW), GI (2009) 117-136.
7. D.A. Patterson, G. Gibson, and R.H Katz: A case for redundant arrays of inexpensive disks (raid). In: Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data, ACM (1988) 109-116.
8. P. Clements, and L. Northrop: Software Product Lines: Practices and Patterns. Addison-Wesley (2002).
9. K. Czarnecki, and U. Eisenecker: Generative Programming: Methods, Tools, and Applications. Addison-Wesley (2000).
10. N. Siegmond, M. Kuhlemann, M. Rosenmüller, C. Kästner, and G. Saake: In-tegrated Product Line Model for Semi-Automated Product Derivation Using Non-Functional Properties. In: Workshop on Variability Modelling of Software-intensive Systems (VaMoS). (2008) 25-31.
11. H. Spencer, and G. Collyer: #ifdef Considered Harmful, or Portability Experience With C News. In: Proceedings of the USENIX Summer 1992 Technical Conference. (1992) 185-197.
12. I.D. Baxter, and M. Mehlich,: Preprocessor Conditional Removal by Simple Partial Evaluation. In: Proceedings of the Working Conference on Reverse Engineering (WCRE), IEEE Computer Society Press (2001) 281—290.
13. E.W. Dijkstra: On the Role of Scientific Thought. In: Selected Writings on Computing: A Personal Perspective. Springer Verlag (1982) 60-66.
14. M. Seltzer: Beyond relational databases. Communications of the ACM (CACM) 51(7) (2008) 52-58.
15. A. Geppert, S. Scherrer, and K.R. Dittrich : KIDS: Construction of Database Management Systems based on Reuse. Technical Report ifi-97.01, Department of Computer Science. University of Zurich (1997).
16. M. Stonebraker, and U. Cetintemel: One Size Fits All: An Idea Whose Time Has Come and Gone. In: Proceedings of the International Conference on Data Engineering (ICDE). (2005) 2-11.
17. C. Prehofer: Feature-Oriented Programming: A Fresh Look at Objects. In: Proceedings of the European Conference on Object-Oriented Programming (ECOOP). Volume 1241 of Lecture Notes in Computer Science., Springer Verlag (1997) 419-443.
18. D. Batory, J.N. Sarvela, and A. Rauschmayer, : Scaling Step-Wise Refinement. IEEE Transactions on Software Engineering (TSE) 30(6) (2004) 355-371.
19. G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C.V. Lopes, J.M. Loingtier, *et al.*: Aspect-Oriented Programming. In: Proceedings of the European Conference on Object-Oriented Programming (ECOOP). Volume 1241 of Lecture Notes in Computer Science., Springer Verlag (1997) 220-242.
20. D. Nyström, A. Tešanović, M. Nolin, 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, IEEE Computer Society (2004) 1-8.
21. A. Tešanović, K. Sheng, and J. Hansson: Application-Tailored Database Systems: A Case of Aspects in an Embedded Database. In: Proceedings of International Database Engineering and Applications Symposium, IEEE Computer Society Press (2004) 291-301.
22. 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). Volume 3676 of Lecture Notes in Computer Science., Springer Verlag (2005) 125-140.
23. S. Apel, T. Leich, and G. Saake: Aspectual Mixin Layers: Aspects and Features in Concert. In: Proceedings of the International Conference on Software Engineering (ICSE), ACM Press (2006) 122-131.
24. A.G. Dimakis, V. Prabhakaran, and K. Ramchandran: Decentralized erasure codes for distributed networked storage. IEEE/ACM Trans. Netw. 14(SI) (2006) 2809-2816.