An Architecture for Interoperability of Embedded Systems and Virtual Reality

Veit Köppen, Norbert Siegmund, Michael Soffner and Gunter Saake

Institute for Business and Technical Information Systems, Department of Computer Science, University of Magdeburg, Germany

Abstract

Virtual Reality enhances the development process of complex and inter-operating products due to bringing existing systems together with virtual prototypes. The modeling of existing products within the virtual reality environment and furthermore the properties of products and product combination are important factors for success in a product life cycle. A reduction of effort for modeling of existing products and simulation of properties can be achieved, when systems and their properties are transported to the virtual reality environment. In this paper, we present a service-oriented architecture for embedded systems and virtual reality. The multiplicity of interfaces, protocols, and hardware and software aspects requires an architecture that overcomes the related difficulties to increase efficiency. Service-oriented architectures make different scenarios in the product life cycle possible, whereas the implementation effort for embedded systems is reduced due to software reuse.

Keywords:
Embedded systems, Product life cycle, Service-oriented architecture, Virtual Reality.

1.   Introduction

The development of Virtual Reality (VR) is a result of the growing complexity of products, e.g., cars or medical devices. This complexity is imminent in the complete product life cycle (PLC) of complex products, where Virtual Engineering (VE) is only present in the engineering phase. Virtual Engineering is often a promising possibility to experience the products within their future environment. Cost reduction is another reason for using this technique. Furthermore, technical and business properties which are not obvious or hidden in reality can be visualized, e.g., security, safety, availability, product cost. Using existing products in the VE enhances the development of new products or improves the interaction of real and future products. A reduction of programming, modeling and simulation effort is often needed to make use of an operating virtual environment.

In the domain of embedded systems (ESs), a variety of protocols, operating systems, applications, and communication are used. For that reason, an architecture has to be adaptive and able to serve this high complexity. Software for ESs is often re-implemented due to varying constraints although there are only a few differences in the implemented functionality. Software product lines (SPLs) [1] are a promising approach to decrease the development effort due to reusing a common code base in a family of similar programs. The usage of SPLs in combination within the architecture can result into cost reduction, increased efficiency, and enables a high flexible development process, where results can be transferred to customers in an early stage.

Virtual Reality is the basis for combining existing products with real or virtual prototypes in an early stage of development and interconnection. To visualize the obvious and hidden properties of interconnecting systems, it is necessary to merge data from real systems and digital development, such as Computer Aided Design (CAD). Different data schema, formats and storage systems exist and have to be fused into a common tailor-made data schema [2]. Real-time requirements, data streams and transformation and aggregation rules for product attributes have to be considered. Another challenge is the complexity of data and restriction of storage and memory for the virtual environment. In addition, data from the Internet of things can also be represented in VR. Sensor networks collect a huge amount of data that enhances business process control, i.e. by a quick reaction on temperature changes. A holistic approach for the complete PLC that combines and makes all data available is in focus of this paper. This sums up to several advantages shared over the whole PLC.

For these challenges, we propose an architecture that brings embedded systems together with future (virtual) products. A crucial requirement for this architecture is high interoperability, to enable the cooperation of implemented real and virtual products. Virtual Reality represents the regarded environment on the one side as a

filter to differentiate between relevant and non-relevant properties. On the other side, it brings non-visible properties into focus.

The paper is structured as follows. In the next section, a VR environment is briefly described to put requirements on the architecture in foreground. This is followed by a brief overview on Service-Oriented Architectures (SOAs). Section 3 gives three application scenarios for the use of our concept. In Section 4, we present the SOA for ES within a VR environment.

2. Background

The abstract PLC, as depicted in Figure 1, consists of different phases in which the scope of the product changes. *Planning* a new product is done in the first steps completely virtual, cf. VE. Nevertheless, more and more products are not developed from scratch but are results from existing products. Combining these products with new inspirations might only be efficiently used in virtual environments. Hence, to reduce efforts and cost (cf. sales and profit curves in Figure 1), it is necessary to use and reuse previous models in the virtual environment. This yields in a competitive advantage for a company. In different domains, such as automotive and railway vehicle manufacturing, VE is already intensively used in the development phase. During production, a control of the results of the development phase should be implemented. For consistency and non-discontinuity of media, the virtual models should also be used. Furthermore, product automation is possible and previous developed and refined models can be implemented. The operation of a product can include initiation, maintenance and repair, as well as inspection. For each part measures of the product are controlled. These measures might be

obvious or hidden and VR brings them into focus. In the last phase of the PLC, the disposal of the product can lead to a disassembling of the product and the disposal or recycling. In many cases, this is also the start of a new product and a new PLC.

2.1 Virtual Reality

Virtual Reality (VR) is the depiction and perception of reality and the corresponding physical properties in a real-time computer-simulated environment at the same time. Virtual Reality can be divided into seven categories: Simulation, interaction, artificiality, immersion, telepresence, full body immersion, and networked communications [3]. In the PLC, all categories can occur. However, dependent upon the phase of the cycle one category might dominate others. For example, in the planning phase, simulations play an important role, whereas from the customer perspective in the production phase, immersion is important to increase customer satisfaction in an early stage. Using VR enables the possibility to bring measures and indicators of a product together in the context of the environment of the product. This improves production time as well as it reduces errors and therefore a more efficient PLC is possible. Virtual Reality is an essential part of VE, where an increase of the above-described categories enhances the engineering process by bringing together planners, decision makers, users, etc.

2.2 Embedded Systems

Embedded Systems denote computer systems that are integrated into a technical context, to control, to regularize or to monitor the technical system. Therefore ESs are suited and optimized for a special task. Embedded systems are increasingly used in many domains. In [4,5] it is stated that about 98% of used computer systems are ESs. In a modern car, more than 200 ESs can occur. With an increasing number of cooperating systems, it is extremely important to pay attention in the development process. This can affect the co-working of existing as well as future ESs. Within the development process of a new product, many ESs, especially the software of the systems, are reinvented, although a reuse might be useful. This is respected to the fact that requirements are different and properties, functional and non-functional, have changed. Visualization of properties reduces development cost, due to an improved identification of reusable parts.

A subset of properties of an ES in the development domain can be classified as stated in Figure 2. A differentiation is achieved by using classes and subclasses. Measures that can be assigned to a class or subclass are not visualized.

We divide properties into the classes of meta data, virtual, real, and compositional properties. Meta data give
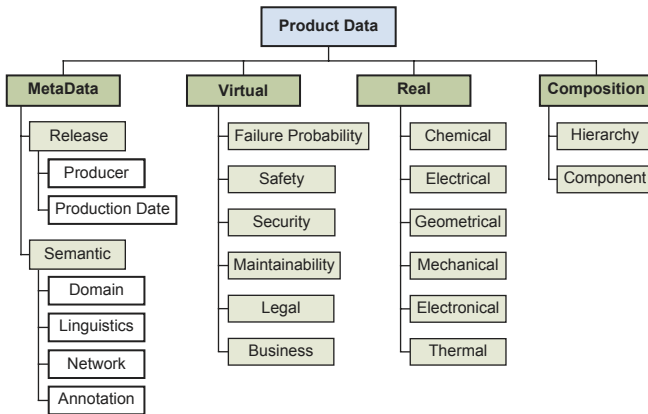

Figure 1: The product life cycle phases.

Figure 2: Hierarchy of properties for ES.

information of the version of a product as well as semantic information. Version or release information might be further categorized by details on producer or production date. Additional differentiations might be possible. Each category should contain at least one measure as a root node in the hierarchy. This facilitates the control within the PLC. In Figure 2, we only show some measures for the meta data category due to limited space. Virtual properties represent information on products that are devoted to non-physical properties, whereas real properties represent information on physical attributes. A different classification of virtual properties is possible, i.e. using quality instead of failure probability, safety, and security. However, in the product development phase, these properties have to be considered carefully and we use them as own categories to bring details into the focus. Further categories within the virtual domain are maintainability, which condenses statements for product development and future adaptiveness, legal issues such as given by law or technical orders, and business, where information on cost of a product, return on investment, and other indicators are summarized.

A product or ES does not need necessarily to be described by all measures or categories. In practice only the information is provided, that is required in the actual phase of the PLC. Note, that for some product combinations transformation rules in the composition process might be applicable, such as sum of weights. Other transformation rules particularly for virtual properties are not available. Therefore these properties have to be measured or estimated for new product combinations.

2.3 Interoperability Aspects

Cooperative ESs have to be connected via a network infrastructure. There exist a multiplicity of connection possibilities due to hardware and protocols. Besides protocols and hardware, challenges regarding networks have to be met. These network challenges are:

- Availability of network — the network might be inaccessible.
- Response time — the latency in a network is not null.
- Transmissions — transmission rate is not infinity.
- Security — the network might be untrustworthy due to an attack.
- Network structure — the structure of the network is changing.
- Administration — the network can be administrated by different users with different scopes.
- Cost — cost of network traffic have to be respected.
- Heterogeneity — network objects, e.g., clients, are mostly heterogeneous.

To meet these challenges and to enable heterogeneous devices to cooperate in such a network, it is not possible to implement the communication for each element of the network in isolation. Another important issue in this context is addressed by the dynamic structure, where devices can connect or disconnect over time. Therefore, we suggest a service-oriented approach to overcome arising problems. In Section 2.5, SOA basics are given and in Section 4 we present the current state of the implementation.

2.4 Product Lines and Software Product Lines

Product lines are increasingly used nowadays. Whereas in the last century Henry Ford stated 'Any customer can have a car painted any color that he wants so long as it is black', this changed dramatically in the last decades. In 1985, 40% of the sales of the automobile company Audi were done by the car model Audi 80 with 1.3 L and 55 hp [6]. In 2000, BMW supplied 10^{20} car alternatives for customers [6]. This development is also important and applicable to software.

An SPL is developed to create a large number of related products by reusing a set of software artifacts called core assets or simply code units [7,8]. These code units, once developed and tested, can be composed to derive different related variants of a program family. This decreases development effort and time-to-market while providing a high degree of reuse [9]. Different programs of an SPL differ in features, e.g., one service variant might have feature TCP/IP provided for communication and another not. These differences are called *variation points*. Features of an SPL and relationships between them are described in a feature model with additional information like attributes or annotations [10,11]. A feature model defines whether a feature is optional or mandatory and is typically visualized with a feature diagram which is a tree-like representation of all features of an SPL. To derive a product, a stakeholder selects the features from the feature diagram that fulfill her functional requirements. These features point to an implementation module. Depending on the implementation technique, this module can be a component, an aspect or a feature

module. This derivation process is completed with the verification of the correctness of the selection and the generation of the product. The generation of a product also depends on the used technique; aspects are weaved within a base code, components and feature modules are composed together.

2.5   Service-Oriented Architecture

Service-Oriented Architecture (SOA) is an approach to support service orientation. A service is comparable to a business activity. A service contains the following properties, c.f. [12,13]:

• produces an outcome,
• is self-contained,
• is a black box to the user, and
• might be composed of other services.

The above-stated properties of a service describe an abstraction from real-world processes. Technically, services are autonomous, often only loosely coupled, and reusable. Furthermore, a service is stateless. A requirement is the formal service contract that specifies properties, functionality, and meta data for the usage of the service. With these service contracts, services are discoverable in a service environment. The service representation utilizes business descriptions to provide context and implements services using service orchestration. Implementations of an SOA are environment-specific and require strong governance of service representation and implementation.

3.   Application Scenarios

In this section, some application areas are briefly addressed. These areas are in focus of the ViERforES project (http://vierfores.de) that deal with improvements of the connection of VR and ES. The proposed architecture increases the connection and inter-operability and reduces development effort of ESs and VR environment. Software product lines facilitate the re-usage of already implemented components furthermore.

3.1   Automotive

The development of cars is nowadays heavily supported by VE. This is not only restricted to the use of CAD but also to business or non-functional properties. The growing complexity of interacting embedded devices within a car requires methods that monitor and control these complexity issues. Virtual Reality is one promising method to face the blindness (hidden or non-visible properties). Existing elements have to be brought together with planned or virtual elements to optimize the development process. Those targets have to be checked in an analysis, and hidden information must be measured

and displayed in an early stage. Our approach of an inter-operating architecture enables a dynamic network structure, where different systems can connect and create a connection to a VR environment. However, first implementations for different ESs require a high initial programming effort for each embedded device.

3.2   Medicine

In the domain of medical devices, VR is used to train students as well as qualified personnel in the case of new devices or methods. Trainings, where only simulations are used, have often not the desired effect, due to different interaction techniques and immersion problems. The usage of medical devices in combination with sensors or embedded systems is a promising approach to overcome this. However, to improve the effects, it is not only the data collection from the utilized systems but also reactions of the systems from the VR scenario that are required. Besides trainings, the evaluation of planned or performed operations are possible. In such a case, the combination of VR and medical device improves comprehension in an easy way. Each communicating system can be seen as an element of a network. Our SOA approach promises a reduction of development and maintenance of this network.

3.3   Logistic Hub

The flow of products in a logistic hub has continuously increased. This is reasoned by global acting companies and just-in-time production. To control, monitor or check incoming, stocking, and outgoing materials, RFID chips are only one possible solution. For example, sensor networks can monitor physical phenomena of goods and their services. In many cases, further restrictions have to be respected, such as security or safety issues. In a real-time control center, all information has to be merged and filtered or aggregated. Real-time data warehousing might be a concept for such a scenario. However, the data integration and fusion have to be customized for each new device transmitting data. Using VR in this context is promising not only for bringing hidden information in foreground but also for making simulations and forecasting possible.

4.   An Architecture for Embedded Systems

In this section, we present an SOA for ESs within the VR. To be faced by the addressed challenges in Section 2, the architecture has to take different aspects into account. Communication issues might be respected by using a lease strategy for elements. This increases communication in the initial phase but increases reliability and availability issues. Security might be addressed by the use of encryption algorithms, e.g., AES. To enhance the use of the architecture, a dynamic network infrastructure is essential. This means that at

runtime, server and clients can connect to the network. Furthermore, it is not crucial which technique is used to implement the service [13], e.g., Java programming language. However, an abstract description of the service is needed to enable the selection of supplied elements.

In Figure 3, we present the general communication of services and clients within the implemented Jini architecture [14]. Registration of a service with the lookup service is directly done, (a) and (b) in Figure 3. To reduce bandwidth communication, a class server is used, where the service can deposit the required binaries that are necessary for the use of the service, (c) and (d). The client communication within the SOA is based on a lookup of available services, (I) and (II) in Figure 3. Afterwards the client connects to the class server and downloads the corresponding binaries, (III) and (IV). In the following phase, the communication between client and service is established, where the client sends its

requests (V) and the service the corresponding answers (VI). Note, that the first four steps are only for initialization purposes whereas step (V) and (VI) build the effective service execution which is often a continuous process.

Within the domain of ESs and VR, it is essential that the participants of the SOA are considered in more detail. On the one hand different heterogeneous ESs have to be handled, on the other hand a representation within the VR environment is often required. Furthermore, to enhance the PLC, consistency of the real ES and the corespondent in the VR has to be guaranteed. We propose a SOA based on Jini that combines hardware with an VR environment. In Figure 4 we show the elementary parts of the participants. The communication is carried out by the SOA. An ES has information on meta data, virtual, or real properties. One service of the ES can be the data staging for these information. The supply of sensor information is another service from the ES. If data preprocessing is possible, this can also be provided as a service. To cooperate with the VR an ES acts also as a client. In such a case results from the VR environment can directly be used.

A VR application server undertakes the task of visualizing, simulating and coordinating virtual elements. To enable a support of dynamically interacting systems, a service for visualization is necessary, which is provided by the VR. Another important aspect is to maintain the scenario. This includes entries and exits of existing systems and virtual products. In many cases, data have to be processed on the application server due to restrictions of ESs. Furthermore, simulations of features, which are not implemented yet, have to be included in the service landscape. Virtual products are not directly connected to the communication
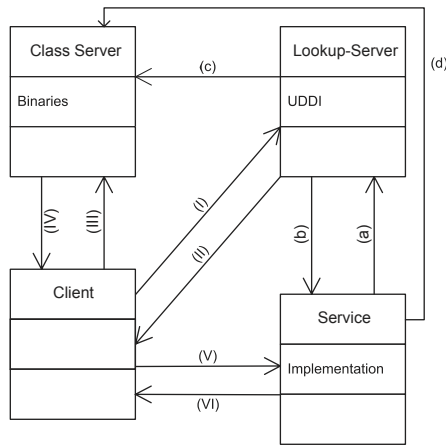

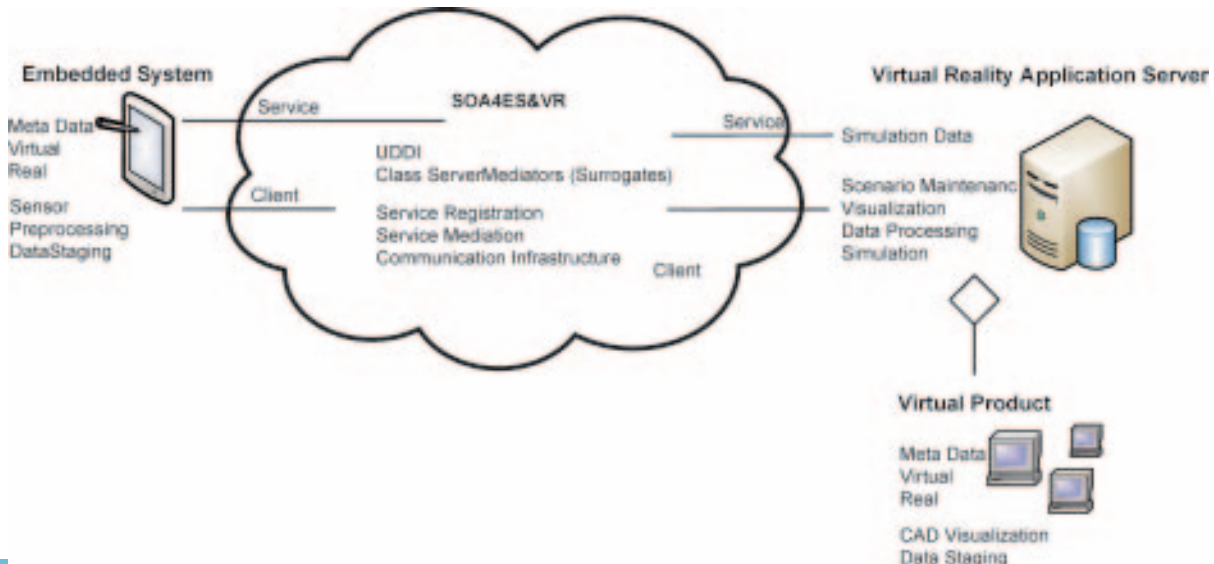**Figure 3:** Communication within the Jini [14] based SOA.

Figure 4: Participants within the VR-ES-SOA.

layer. They are part of the VR application server. However, in the context of development or VE, these virtual products provide services that are equal to services of existing products. Therefore it is mandatory to implement them with client and service features, which are initialized within the SOA by the VR. Due to similarities of virtual and real products, these implementations are very similar and therefore a software development for the SOA is only required once. In the case, that a virtual product comes to life, already implemented services and client feature can be reused. The SOA concept enables, as stated in Section 2.5, the use of small and standalone applications. One approach for an SOA is Jini [14]. This technology was developed in the 1990s by Sun (http://java.sun.com/developer/products/jini) for the Java programming language. In the focus of this project is the ubiquitous communication of small Java devices whereas the question how this communication is established should not concern. Instead of implementation aspects, easily provided services are the core idea behind Jini. Another strength of Jini is robustness and flexibility. This includes the possibility of dynamic networks, where devices are available only for a short time window. Malfunctions and failures can be recognized and a fast reaction within the network is possible. Therefore this implementation fits well into the application domain of ESs and VR.

4.1   Requirements

With the Jini Extensible Remote Invocation (http://artima.com/intv/jeri.html), the limitation of TCP/IP is abolished. A supposed drawback is the use of Java, due to the fact that ESs often only support C/C++. This is regarded by restrictions of resources like memory or CPU. Surrogates overcome this problem and make the integration of non-Java devices in a Jini network possible. These surrogates care for communication of the device as well as within the Jini network. Robustness, flexibility, and performance have to be considered in the implementation of such a surrogate.

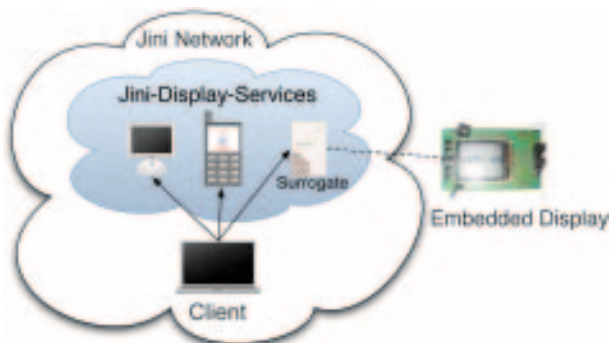In Figure 5, we demonstrate our approach to integrate the Jini SOA into the VR environment. The concept enables Java devices to communicate within the network to a client that represents the VR environment. Furthermore, communication from the client to real systems is also possible. In the case, that non-Java devices, that are the superior number of embedded devices, have to be connetced to the SOA, a surrogate acts within the network and communicates to the device.

4.2   Case Study

As a simple case study, we use the *Display 3000 D071*, as a representation for an ES, in connection with the Virtual Development and Training (http://vdtc.de) platform [15]. In our artificial example, we develop an application, where data are visualized on the ES. Within the development process, we want to monitor information on ROM and SRAM. The VR environment controls the loading of pictures and visualizes the ES, where prepared CAD information is used as well as the current state is retrieved. If the ES is connected to the network, the communication is enabled via the corresponding surrogate. The connection is established via RS232 interface. Within the VR information on ROM and SRAM of the ES are visualized. We use in our scenario data streams between the virtual client and the ES. The VR enables the visualization of data streams between different systems. Additionally, changes in the observed measures are visualized. In Figure 6, we depict a snapshot of the running system. Note, that this is only an artificial application, due to bandwidth limitations (RS232), the updates of the ES need a long time and advantages of the use of VR for this scenario is rather frustrating than motivating.

5.   Conclusions and Outlook

In this work, we proposed to use a service-oriented architecture for the intercommunication of embedded devices within the Virtual Reality. The use of services
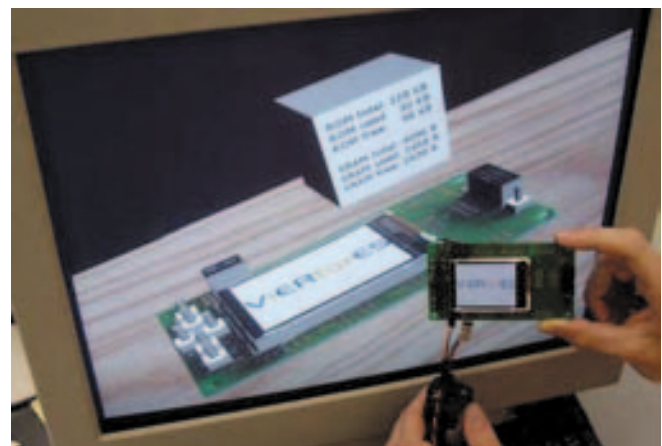

Figure 5: Jini Network for communication of ESs and VR.

Figure 6: Display 300 D071 (real and virtual).

enables a complexity reduction for different devices. Furthermore, we showed an implementation of the SOA based on Jini. Research has to be taken for dependencies of an increasing number of devices as well as influence of network traffic. Challenges of security and safety are not addressed in this work but have to be investigated in the context of the SOA approach, too.

6.   Acknowledgments

References

1.   M. Rosenmüller, N. Siegmund, H. Schirmeier, J. Sincero, S. Apel, 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.

2.   N. Siegmund, C. Kästner, M. Rosenmüller, F. Heidenreich, S. Apel, and G. Saake. Bridging the Gap Between Variability in Client Application and Database Schema. In: 13. GI-Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW), GI (2009) 297-306.

3.   M. Heim. The Metaphysics of Virtual Reality. Oxford University Press (1993).

4.   D. Tennenhouse. Proactive Computing. Communications of the ACM (CACM) 43(5) (2000) 43-50.

5.   J. Turley. The Essential guide to semiconductors. Prentice Hall Press, Upper Saddle River, NJ, USA (2003).

6.   M.S. Andres. Die optimale Varianz. Brand EINS 01/06 (2006) 65-9.

7.   P. Clements, and L. Northrop. Software Product Lines: Practices and Patterns. Addison-Wesley (2002).

8.   C.W. Krueger. New methods in software product line practice. Commun. ACM 49(12) (2006) 37-40.

9.   W.A. Hetrick, C.W. Krueger, and J.G. Moore. Incremental return on incremental investment: Engenio's transition to software product line practice. In: Proc. Conf. Object-Oriented Programming, Systems, Languages and Applications, New York, NY, USA, ACM Press (2006).

10. K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson. Feature- Oriented Do-main Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University (1990).

11. K. Czarnecki, and U. Eisenecker. Generative Programming: Methods, Tools, and Applications. Addison-Wesley (2000).

12. M. Bell. Service-oriented modeling: service analysis, design, and architecture. John Wiley and Sons, New Jersey (2008).

13. T. Erl. SOA Principles of Service Design. Prentice Hall (2007).

14. J. Waldo. The Jini architecture for network-centric computing. Communications of the ACM 42(7) (1999) 76-82.

15. A. Hintze, M. Schumann, and S. Stüring. Interaktive szenarien für die ausbildung von wartungs-und instandhaltungspersonal. In Schulze, T., Lorenz, P., Hinz, V., eds.: SimVis, SCS Publishing House e.V. (2000) 225-238.

AUTHORS

Veit Köppen received his MSc degree in Economics from Humboldt-Universit ät zu Berlin, Germany in 2003. During 2003-2008, he worked as a faculty member in the Institute of Production, Information Systems and Operation Research, Freie Universit ät Berlin, Germany. He received a Dr. rer. pol. (PhD) in 2008 from Freie Universit ät Berlin, Germany. He is now a member of the Database Group at the Otto-von-Guericke University Magdeburg, Germany. Currently he is the project coordinator in the project funded by the German Ministry of Education and Research. His research interests include Business Intelligence, data quality, interoperability aspects of embedded devices and process management.

E-mail: vkoeppen@ovgu.de

Norbert Siegmund was born in Aschersleben, Germany. He received his Master in Computer Science (degree: Diplom-Informatiker) from Otto-von-Guericke University, Magdeburg, Germany in 2007. He joined immediately the database workgroup at the Otto-von-Guericke University in Magdeburg as a PhD student. His research interests are tailor-made data management and non-functional properties in software product lines.

E-mail: nsiegmun@ovgu.de.

Michael Soffner was born in Staßfurt, Germany. He received his Master in Computer Science (degree: Diplom-Informatiker) from Otto-von-Guericke University, Magdeburg, Germany in 2005. Then he worked for two years in the software development company Q-fin GmbH, Magdeburg, Germany. In September 2008, he joined the database workgroup at the Otto-von-Guericke University in Magdeburg as researcher in a project funded by the German Ministry of Education and Research. Currently he is PhD student and his research interests are real-time data management and information generation in heterogeneous networks.

E-mail: michael.soffner@ovgu.de.

Gunter Saake received the diploma and a PhD in Computer Science from the Technical University of Braunschweig, F.R.G. in 1985 and 1988, respectively. From 1988 to 1989 he was a visiting scientist at the IBM Heidelberg Scientific Center, where he joined the Advanced Information Management project and worked on language features and algorithms for sorting and duplicate elimination in nested relational database structures. In January 1993, he received the Habilitation degree (venia legendi) for Computer Science from the Technical University of Braunschweig. Since May 1994, Gunter Saake is a fulltime professor for the area 'Databases and Information Systems' at the Otto-von-Guericke University, Magdeburg. His research interests inc-lude database integration, tailor-made data management, object-oriented information systems and information fusion.

E-mail: saake@iti.cs.uni-magdeburg.de