RWTH Aachen

Department of Computer Science
Technical Report

Report of the GI Work Group "Requirements Management Tools for Product Line Engineering"

Danilo Beuche, Andreas Birk, Heinrich Dreier,
Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen,
Isabel John, Ramin Tavakoli Kolagari,
Thomas von der Maßen (Ed.), Andreas Wolfram

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2006-14
RWTH Aachen · Department of Computer Science · December 2006

The publications of the Department of Computer Science of RWTH Aachen (*Aachen University of Technology*) are in general accessible through the World Wide Web.

http://aib.informatik.rwth-aachen.de/

Contents

1 Introduction 4
  1.1 Background 4
  1.2 Motivation 4
  1.3 Organization of Work Group 5
  1.4 Aims and Approach 6

2 Related Work 6
  2.1 Work of the Previous Working Group 6
  2.2 Other Tool Evaluations 7

3 Documentation of Tool Scenarios 8
  3.1 hp 9
  3.2 Harman/Becker 11
  3.3 Continental Temic 13
  3.4 RWTH Aachen University 15
  3.5 Variability-Oriented Reuse of Functional Requirements in the Framework of a Function Signal Network in Telelogic DOORS – Current Research Approach at DaimlerChrysler 17
    3.5.1 Variability in the Function Signal Network (FSN) 18

4 Requirements for Requirements Management Tools in the Context of Product Line Engineering 20
  4.1 Starting Point 20
  4.2 New and Changed Requirements 21
    4.2.1 New Requirements Related to Variability 22
    4.2.2 Relationships to Product and Project Management 22
  4.3 Prioritization of Requirements for the Context of Product Lines 24
    4.3.1 Particular Priorities in a Product Line Context 24
    4.3.2 Product-Line-Specific Priorities per Requirements Group 25

5 Evaluation of Tools 29
  5.1 Explanation of the Evaluation Approach 29
  5.2 Visualization 30

6 Summary and Future Work 33

7 References 35

8 Appendix 35

1 Introduction

1.1 Background

Why should we pay more attention to a product line requirements management process than to a single project environment and why is requirements management for product lines very much more difficult than for a single project environment?

The question is easy to answer. As time goes by, customer requirements change, and so the product itself will change as well to satisfy the customer's needs. A feature that was revolutionary a few years ago is now standard and may become obsolete in the future.

Like the product itself, the specifications for it will change, evolve and vary as well.

In order to keep track of the changes and to control and categorize our requirements, we need requirements management tools. Most of the requirements management tools on the market are focused on a one-project scope. But efficient product development will need to re-use as many components as possible to keep expenses low for new developments. Reuse by copying all components of a previous product to form the basis of a new one is a pragmatic approach, but it is unsystematic and short-time oriented. If the number of product derivates increases, it is almost impossible to select the best matching template for a new product in the family tree. Total re-use is one of the goals of product line development, meaning that the components must not only be copied but actually shared among several subprojects. But the product will not only consist of the shared requirements, some of them will be specific. Commonality and variability are the magic words that modern product development has to deal with.

In software development, these mechanisms are likely to be implemented in object-oriented programming languages. Classes can be derived from other classes by inheritance of the attributes and methods, with the possibility of overloading their methods to implement the differences between the derived and the base class.

But requirements management for product lines has more needs. The differences between one derivate and another may consist in:

- changed requirements
- obsolete requirements
- additional requirements

The single fact that requirements can change and different versions of the same requirement are part of two derivates of a product makes high demands on requirements management tools and their configuration management functionality.

1.2 Motivation

The report bases on research findings of a previous working group (see the next section for details) identifying essential challenges in the context of requirements engineering and software product lines. In [7] major challenges in product line engineering are presented:

- Justification of the platform approach as a process model by a cost / benefit-analysis
- Independent platform team
- Cooperation between platform and product development teams
- Proof of justification of the platform team
- Communication overhead
- Configuration management
- Influence of the architecture on requirements negotiation
- Description of variability for domain analysis
- Domain analysis and domain description
- Explicit requirements process
- Sequence of integrating requirements into the platform
- Explicit prioritization of requirements
- Realization of platform requirements in products
- Use of the architectural advantages
- Description of the generic architecture
- Effective tool support

We decided to investigate the challenge of *effective tool support* for several reasons:

- Until now there does not exist any systematic analysis of requirements for product line tools.
- Tools play the key role when installing a new approach (like product lines) because tools are a cross sectional issue. If people are convinced from the effectiveness and usability of the tools then they will also be more willing to support the overall process – which is in fact represented by the tools.

In a methodical investigation on the product line approach tools are not important, but to bring a product line approach to live in practice, tools become crucial. A bad implementation or missing requirements may result in a destruction of the whole product line approach. So we started from this point of view and gathered key requirements for requirements management tools supporting software product lines. The report is oriented at practical issues and tries to support product line approaches in industrial settings. The requirements are quite abstract and don't describe technical solutions. They are thought of as a list of demands that a tool developer can take as a starting point and that can be refined. The motivation of this working group is to initialize a discussion on this (often neglected) topic and to push the convincing product line approach one step forward into the direction of a wide-spread industrial usage of software product lines.

1.3 Organization of Work Group

The working group 'Requirements Management Tools for Product Line Engineering' was set up in January 2004. It is the successor of the working group 'Requirements Engineering for Product Lines', which completed its work successfully in December 2003.

The working group met eleven times, i.e. every two months, to find an answer to the question 'at which functionality do state-of-the-art requirements management tools lack for product line development?' Actually, requirements management tools lack for most product line techniques, which means that complicated workarounds or more flexible tools like Word or Excel are necessary to analyze and manage requirements in a product line context.

The members of the working group are:

- Dr. Danilo Beuche, pure-systems GmbH
- Dr. Andreas Birk, sd&m AG
- Dirk Janzen, Automotive Systems GmbH
- Heinrich Dreier, SYNSPACE GmbH
- Heidi Galle, Conti Temic microelectronic GmbH
- Andreas Wolfram, Conti Temic microelectronic GmbH
- Gerald Heller, Hewlett Packard GmbH
- Dr. Ramin Tavakoli Kolagari, Technical University of Berlin
- Isabel John, Fraunhofer IESE
- Andreas Fleischmann, TU München
- Thomas von der Maßen, RWTH Aachen

1.4 Aims and Approach

The objective of this working group was to define the typical features required by product line development and to analyze and select existing tools on the market that are suitable for product line requirements management. To achieve this, we organized our work into the following steps:

1. Analysis of existing publications
2. Collection of real use scenarios for requirements engineering tools in a product line environment
3. Derivation of product line requirements
4. Proposal for functionality in requirements engineering tools regarding product line needs
5. Benchmark of existing requirements engineering tools regarding these requirements

2 Related Work

2.1 Work of the Previous Working Group

This work group, which deals with tool aspects for requirements engineering in the context of product line engineering, is the successor of a previous working group, where aim was to identify the main problems in product line development. This previous work group, set up in 2000, comprised representatives of the following organizations: Robert Bosch GmbH, Hewlett-Packard, Fraunhofer IESE, University of Aachen (RWTH Aachen), and sd&m AG. These organizations shared an interest in the topic of requirements engineering for product lines and set out to identify the key problems in product line engineering practice along with potential (and proven) solutions. While the group's work focused exclusively on requirements engineering issues, it soon became clear that they would have to adopt a broader approach, given the close interconnection of requirements engineering with other issues in a product line context. They provided an overview of the main problems in product line development (cf. previous section), which could be organized to the following four main problem categories:

(1) organization and management,
(2) requirements engineering,
(3) product-specific vs. platform-specific interests, and
(4) architecture.

These categories were the result of systematic collection and clustering of known problems the members of the working group. Based on their own experience as well as their understanding of the technology, the members of the working group derived and described potential solutions for the main problems. The group concluded its work in 2003. The results are documented [7] and the report (approx. 55 pages) can be downloaded at

- *http://www.iese.fhg.de/Pulse/Activities/RE4PL.html* *or directly at*
- *http://www.iese.fhg.de/pdf_files/iese-121_03.pdf*

2.2 Other Tool Evaluations

The Software Engineering Institute (SEI) at Carnegie Mellon University has a long tradition of product line research. In 2002, it conducted a survey asking 31 companies, about their strategies for dealing with product lines, which tools they used for product lines and how they used them. The report from this survey (approx. 70 pages) can be viewed at

- *http://www.sei.cmu.edu/publications/documents/02.reports/02tn017.html*

The ‘Virtuelles Software-Engineering Kompetenzzentrum’ has made a summary of the tool aspects of the SEI survey. It shows that in 2002, the tools used in requirements engineering for product lines were mainly proprietary tools; the most widely used commercial tools were Requisite Pro (27%), Doors (19%), and Slate (3%). The Summary (approx. 1 page) can be viewed at

- *http://www.software-kompetenz.de/?16937*

In 2003, the SEI held a workshop that “explored the area of tool support for product lines with representatives from technically sophisticated organizations, having direct experience in software product lines”. They identified and discussed hot issues of software product line tools. The report (approx. 45 pages) can be viewed at

- *http://www.sei.cmu.edu/publications/documents/00.reports/00tr002.html*

Since there is a continuous development in both tool support and product line methods, the results of this survey are of limited utility for the evaluation of current tools.

The International Council on Systems Engineering (INCOSE) has a requirements management tools survey site dating back to the 1990s. They have published and still maintain a table evaluating some 30 requirements management tools along with about 15 detailed attributes. However, since the vendors are evaluating their own products, the results of this evaluation should be treated with caution. The results of the survey can be viewed at

- *http://www.paper-review.com/tools/rms/read.php*

One weakness of the INCOSE evaluation is that the tools are evaluated not by users but by their vendors. Also, the survey focuses exclusively on requirements engineering; specific product line aspects are not yet covered.

In 2003, the European research project CAFÉ provided a list of tools used for product line engineering, including some of these tools' deficiencies. They list 31 product line tools, including about ten requirements tools: "This document collects the knowledge of the CAFÉ partners with respect to the tools available for product family engineering development. It lists the tools used for product family engineering by CAFÉ partners." Two major issues were covered: first, analysis of the available tools for product family development, and second, elicitation of requirements regarding the functionalities potential tools should provide. This list can be viewed at

- *http://www.esi.es/en/Projects/Cafe/board.html*

The Atlantic Systems Guild offers "a survey of requirements engineering tools [that] gives you a review of most of the current tools." It covers a broad selection of requirements management tools and is frequently updated. However, no specific product line aspects are covered in this list. The survey can be viewed at

- *http://www.volere.co.uk/tools.htm*

In 2004, the computer magazine IX conducted a study of the eight major requirements engineering tools: Borland CaliberRM, Compuware Reconcile, IBM RequisitePro, NCH Miro.BAS, Polarion, QA Systems IRqA, Serena RTM Workshop, and Telelogic DOORS. The results of this study are not freely available, but can be downloaded for about 400€ at

- *http://www.heise.de/kiosk/special/ixstudie/05/01/*

Unfortunately, this study focuses exclusively on requirements engineering. It does not cover specific aspects of product line engineering.

At DaimlerChrysler, a research team has developed a schema for evaluating requirements engineering tools. Our work group has adopted this evaluation schema and extended it to cover product line aspects, too [2].

3 Documentation of Tool Scenarios

In this section, we describe scenarios of the adoption and use of tools in the context of product line engineering. The collected scenarios are not fictitious but describe real practice based on daily routine of members of the work group. The scenarios thus reflect real problems in the context of product line engineering. All the scenarios describe problems that arise during the modeling and management of information, with a focus on modeling and managing requirements. The problems arise because a specific tool does not fully support the desired modeling or management activities.

The descriptions of the scenarios cover the following aspects:

1. The development context in which the tool is used

2. Efficient und practicable handling of the used tool in the specific context
3. Workarounds, if the tool support is unsuitable

As all the scenarios are written by experts who use the tool in their daily work, the scenarios provide a good insight into the functionality provided by the selected tool, what functionality is lacking and what practicable workarounds can be performed.

The descriptions of the scenarios are designed to provide:

1. A documented and comprehensible description of typical use scenarios for selected tools.
2. A catalogue of requirements for tools that must be implemented in order to support product line engineering.

While the documented scenarios are valuable in themselves, they also help in deriving new requirements for tools. The derived requirements are documented in Section 4.

The documented scenarios comprise the following:

- Using CaliberRM at hp's OpenView Business Unit
- Using IRqA at ContiTemic
- Using RequiLine, a university prototype of a requirements engineering tool for product lines
- Using DOORS in a current research approach at DaimlerChrysler

3.1 hp

by Gerald Heller

hp is a technology solutions provider for consumers, businesses and institutions globally. The company offers a range of products and services from IT infrastructure, personal computing and access devices to global services and imaging and printing for individual consumers as well as for small and medium-sized businesses. For more details, visit hp's website at http://www.hp.com. The OpenView Business Unit is a part of hp's global software organization. OpenView has more than 15 years' experience in developing IT management software. The OpenView product line consists of a variety of products in the areas of network, storage, systems and service management. See http://openview.hp.com for details.

The OpenView organization develops its product line concurrently at different locations around the world. In the early years, the OpenView product line started with independently developed products in the area of network and systems management. These products proved to be extremely useful for customers and the product line was therefore extended over a period of many years. OpenView products are typically multitier products (UI clients, management servers, database servers and agents). The products support a wide range of operating-system platforms.
A suite of new products has supplemented this product line over the years, some were developed in-house, others acquired externally. With time, the following challenges became increasingly apparent:

- Products started to overlap in functionality.

- Customers who bought more than one product faced with consistency and efficiency problems.
- Development and maintenance costs rocketed.

Given this situation OpenView's management decided to reengineer the products into a more tightly integrated product family. The driving goals were:

- Time to value
  (Fast and easy deployment, common configuration)
- Cost of ownership
  (Minimal training and operation cost for IT personnel)
- Offering solutions and services
  (Provide a tightly integrated suite of products from which customers create a solution to address their specific business needs

Around 1999, the development paradigm was changed to a model in which reusable components with a shared data model are developed. Besides changes in organization, new processes and tools were also introduced to support this model.

Requirements engineering and management was one of the main improvement areas. Standard training sessions were organized at different sites to achieve a common skill base across the organization. The ***Volere Requirements Template*** (see [10]) was selected as a standard requirements structure for all projects. Borland's requirements management tool CaliberRM Borland [1] was chosen to support the distributed development needs of the organization. The tool was customized to ensure that every project had the same requirements structure by applying the Volere Requirement Template structure. Each requirement type in Volere is represented by one requirement type in the tool. The Volere requirements shell was also translated into the tool's capabilities.

Having the same structure applied in each project enables easy navigation between different projects, minimizes training needs and facilitates the sharing of requirements.
Our basic principles for the requirements process and tool were:

- allow broad sharing of information
- promote consistency to support efficiency, sharing and reporting
- enable flexibility
- allow individual empowerment

The application of these principles means that all workers with a requirements management tool account have access to all information. This includes workers from a wide range of departments, e.g. product development, product marketing, information engineering (documentation), support and testing. The entity to be managed in the tool (called project) may be a solution, product, component or a shared area. Requirements can be traced from products to shared components in order to support product line planning and monitoring. Specific projects were created to support more generic requirements which apply for a certain set of products or components. In these projects, shared requirements are specified and maintained once only; the individual projects merely need to refer (trace) to the global requirement.

The tool offers some basic functionality. hp developed additional add-ins and domain-specific reporting to increase the value for users of the requirement infrastructure. From a product line perspective, the tool should provide more efficient traceability mechanisms. Also, the management of shared requirements can be improved. The tool's limitations forced us to still use spreadsheet tables for multidimensional operational requirements.

The requirements process has constantly evolved, incorporating the experiences of past projects. hp has established a requirements process framework that is customized for individual project types. This process framework provides useful guidelines on how to establish consistent and efficient requirements processes for the whole product line.

3.2 Harman/Becker

Harman/Becker Automotive Systems GmbH develops and manufactures infotainment systems at many locations worldwide. These systems incorporate the following elements are made up of radio, TV, CD/DVD player, phone and navigation. The systems are developed and manufactured as supply parts for car manufacturers or original equipment manufacturers (OEM) as well as systems for the consumer market.

Becker has a very long tradition of developing and manufacturing car radios dating back to before 1950. In 1995, Becker was taken over by Harman International. In recent years, several mergers have brought new know-how to the group.

Since the 1980s, software has accounted for a constantly growing share of overall development effort. Today, this share is more important than hardware and mechanical components taken together.

Infotainment systems are complex, embedded systems with multiple external and internal interfaces. Requirements specifications for a single system comprise several thousand files. Requirements specifications for OEM systems are usually drawn up by the car manufacturer. such specifications differ in many ways:

- The requirements specifications of different manufacturers vary in their structure and level of detail.
- Some specifications are highly inhomogeneous.
- Different parts of the specifications are often written in different departments of the car manufacturer. This usually leads to inconsistencies.

The requirements specifications are normally delivered as a set of files (Microsoft Word, Microsoft Excel, Adobe Acrobat, etc.).

Requirements management tools for exchanging requirements specifications with suppliers are have been used by one car manufacturer for about two years now. Other car manufacturers have followed this example but use other exchange processes and formats. Telelogic DOORS is widely used by car manufacturers for requirements management. All manufacturers use DOORS in different ways. No manufacturer currently uses DOORS to manage all requirements specifications. Individual manufacturers have also developed their own specific exchange processes. At first sight, a single tool might be expected to produce a single exchange process. Unfortunately, the use of a single tool does not result in a single process in requirements management and exchange.

During development of the system, usually a period of several years, the requirements specifications undergo numerous changes. Some of these changes are submitted as explicit change requests, or implicitly as new document versions or file versions. Analyzing such changes and their impact involves considerable effort.

Traditionally, the systems for different series of different car manufacturers (OEMs) have been developed by independent projects there was. Some technology transfer from products that were developed for the consumer market. Similar developments in different projects still remained a risk, however.

To avoid parallel development, a common platform has recently been developed to make available basic functions for all customer-specific development projects. From a requirements management view, two tasks have to be performed:

- Identification and description of common requirements for the platform
- Matching customer-specific requirements to the platform requirements

The complexity of this task grows owing to the continuous changes in the requirements for both the platform and the customer-specific systems.

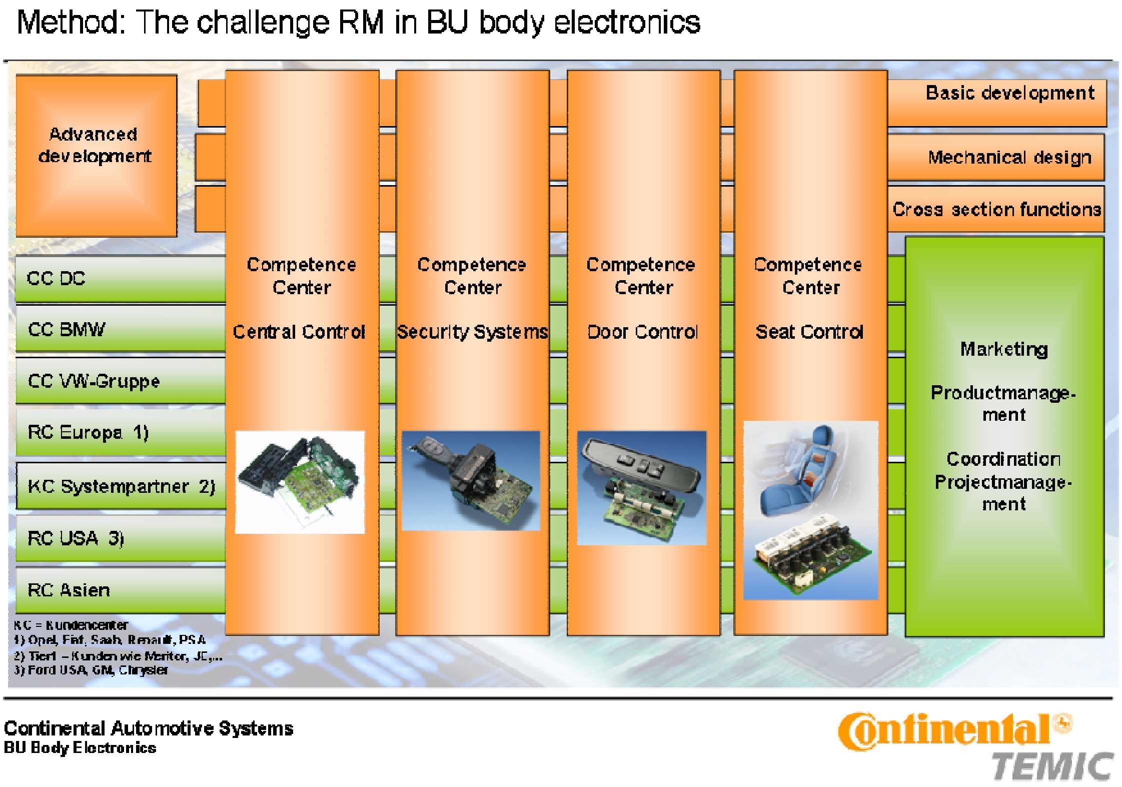
To summarize: requirements management covers the following main areas:

- Management of requirements documents
- Management of changes to requirements documents
- Exchange of requirements documents with car manufacturers
- Identification of requirements for the platform
- Matching requirements from car manufacturers to platform requirements

3.3 Continental Temic

by Heidi Galle
and Andreas Wolfram

Continental AG, based in Hannover, Germany was founded in 1871 and operates worldwide as leading automotive supplier with 80,000 employees at the end of 2005
Continental Temic, the provider of high-quality automotive electronics is today part of Continental Automotive Systems, a corporate division of Continental AG.
In 2002, the business unit 'Body Electronics' started implementation of a tool based, structured requirement management process. First of all, the needs of product line development was taken into account by changing the organisational structure into the so called 'competence centres', reflecting different product lines as shown in figure 1 below.


Method: The challenge RM in BU body electronics

Advanced development

CC DC
CC BMW
CC VW-Gruppe
RC Europa 1)
KC Systempartner 2)
RC USA 3)
RC Asien

Competence Center Central Control
Competence Center Security Systems
Competence Center Door Control
Competence Center Seat Control

Basic development
Mechanical design
Cross section functions

Marketing
Productmanagement
Coordination Projectmanagement

KC = Kundencenter
1) Opel, Fiat, Saab, Renault, PSA
2) Tier1 – Kunden wie Meritor, JC,...
3) Ford USA, GM, Chrysler

Continental Automotive Systems
BU Body Electronics

Continental TEMIC

Figure 1: organizational structure of the business unit 'body electronics'

The challenge of requirements management of several Product Lines for different customers was to handle many reusable set of requirements. Reusability not only considered for a special product line, each intersection between two sets of requirements has to be analyzed for reuse A customer project means development of several variants of the product for the customer.

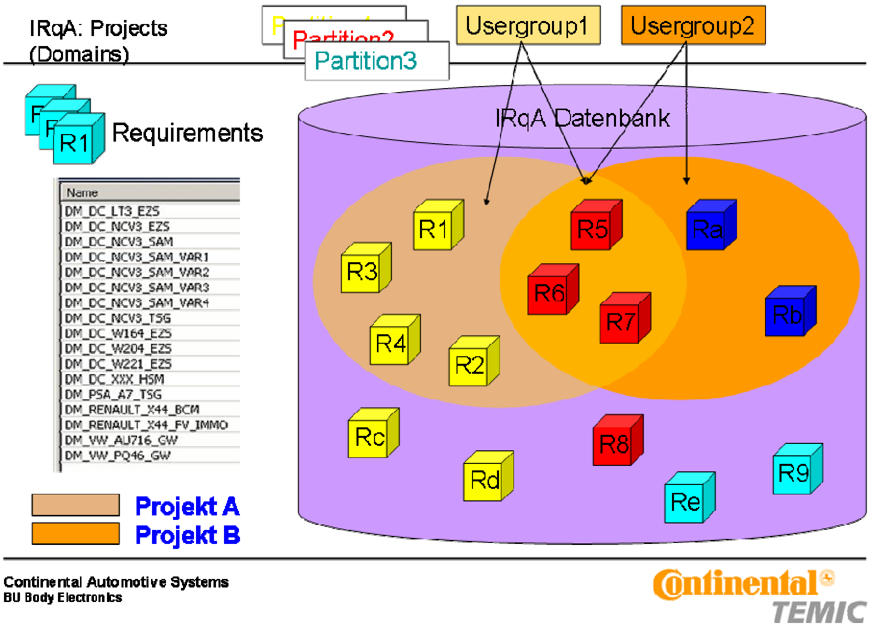
The tool should solve the following challenges:

- General Rules from customers (non functional quality requirements, environment requirements…) are valid for all projects of this customer.
- General internal rules (quality rules, development process rules, …)

- Product line requirements for each project of an product line (reusable components)
- Project specific requirements which are only valid for this product
- Variant requirements, which are valid for special variants of an product
- Type or model based requirements which are valid for all products for a special series type or model of a car (p.e. S-model of DC)

As result of a tool evaluation, IRqA [3] seemed to be the best solution for the business unit. A single database, accessible from all involved Stakeholders was the approach. All Requirements reside in this database and a project is a subset of the database content IRqA has the ability to build a graphical, standard requirement model as guidance for all projects. Requirements have a specific type with a special set of attributes. As shown in the figure below, for example the requirements R5, R6 and R7 belong to project A and project B. The yellow and red ones can belong to project A, the red ones and the dark blue ones can belong to project B. The project leaders decides over a mapping to domains which requirements are visible in the project, this is represented with the bubbles. Over several user groups and their access rights on the different set of requirements (mapped to partitions) the requirements get visible and/or writable in the project. In this example a project A developer (Usergroup 1) needs access rights on the yellow and the red requirements.

The same mechanism is used to construct attributes of project A, project B and attributes belonging to both projects.



Name
DM_DC_LT3_EZS
DM_DC_NCV3_EZS
DM_DC_NCV3_SAM
DM_DC_NCV3_SAM_VAR1
DM_DC_NCV3_SAM_VAR2
DM_DC_NCV3_SAM_VAR3
DM_DC_NCV3_SAM_VAR4
DM_DC_NCV3_TSG
DM_DC_W164_EZS
DM_DC_W204_EZS
DM_DC_W221_EZS
DM_DC_XXX_HSM
DM_PSA_A7_TSG
DM_RENAULT_X44_BCM
DM_RENAULT_X44_FV_IMMO
DM_VW_AU716_GW
DM_VW_PQ46_GW

Figure 2: database organization

This is a very simple example. To manage all of the intersections which belong to a project, it is important to decide which set of requirements are worth to be reused. The complexity of managing the shareable sets of requirements should be as small as possible and as big as necessary.

What happens if a special version perhaps version 2 of requirement R5 is valid for project A and the version 2.1 of the same requirement belongs to project B. To change a common shared requirement isn't solved really in the RM Tool, there is no possibility to branch and merge this requirement. The interface to the Configuration Management Tool doesn't offer a solution for this as well, but state of the art is that Configuration Management Tools can't manage objects; they can manage files and their changes. So at the moment, there is only the possibility to create a new requirement instead of a new version of the requirement, the red requirements become a yellow and a blue one.

Another problem to handle common requirements is that you need a person in charge for the shared set of requirements, an independent requirements engineer for the 'basic project' is a must. If there is no basic group which develops such requirements you never gets shareable project independent requirements, the project members have to develop their project and nothing else. If you look in the first picture you can see that we have no such persons or groups in our organization.

The challenge to develop and work with shared requirements in different projects for several customers can only be solved with a product line Requirements Management Tool and an adequate organizational structure.

3.4 RWTH Aachen University

by Thomas von der Maßen

One research project of the "Software Construction Research Group" at the University of Aachen is concerned with the requirements engineering of software product lines. A special focus of the project is modeling variability within requirements. As variability is a key aspect distinguishing product lines from individual products, it must be identified and explicitly modeled during the requirements engineering processes of product line development. The research group uses feature models to model variability within requirements.

A feature model captures requirements in terms of features. Typically, a feature is an abstraction from several requirements. A so-called platform feature model (PLFM) contains all identified features that are relevant to the whole product line and structures them hierarchically in a tree. Furthermore, variability is expressed by so-called domain relationships between features and feature attributes. The PLFM is part of the platform requirements specification. From the common PLFM, so-called product feature models (PFMs) can be derived. The derivation is performed by binding the variation points, i.e. selecting features that should be part of the PFM and by setting values of feature attributes. A derived PFM is part of a product requirements specification.

While much research effort has been expended on modeling of different variability types and investigating derivation processes, the quality criteria that a PLFM must satisfy have so far been neglected. A requirements specification for individual products should be correct, unambiguous, complete, consistent, ranked, verifiable, modifiable and traceable, and this also applies to PLFMs. The research group therefore investigated how variability influences the above-mentioned attributes, and how the attributes can be applied to a PLFM.

As current requirements engineering tools support neither the necessary variability concepts, multi project specifications nor the analysis of platform specifications, the Software

Construction Research Group developed RequiLine, a prototypical requirements engineering tool for product lines. RequiLine allows the modeling of requirements in terms of features and natural-language requirements. A detailed description of the provided functionality can be found in [6] and [9]. The use of RequiLine is not restricted to a special requirements engineering method or development approach. RequiLine thus supports requirements engineering for product lines using a proactive or reactive approach. A typical use scenario is described below:

1. The administrator creates a new product line project.
2. The administrator creates the required user accounts and assigns roles to them to grant privileges.
3. The analysts develop a PLFM for the product line by
    a. modeling features
    b. structuring features through domain relationships
    c. modeling variation points
    d. modeling dependencies between features
4. Optionally, the analysts write natural-language requirements and associate them with features. Natural-language requirements can be used to describe features in more detail.
5. The analysts verify and validate the PLFM using
    a. the query interface to create user-defined queries
    b. the consistency checker to check the PLFM's consistency
    c. the metric interface to reveal information about the PLFM's flexibility (variation degree) and appropriateness
6. The analysts derive PFMs from the PLFM following
    a. an explorative approach: the analysts create a new product and add features from the PLFM to the PFM using the feature selection interface and parameterize selected features by setting values of their attributes. A subsequent consistency check of the PFM must be performed to guarantee the consistency and completeness of the PFM with respect to the PLFM.
    b. a guided approach: the analysts derive a new product by using the product configuration wizard. The wizard guides users through the PLFM and requests them to resolve variation points, regarding the semantics of the variation points and modeled dependencies. The wizard ensures a consistent and complete PFM.

Typically, the steps described above are not carried out sequentially but in a highly iterative manner.

RequiLine has been evaluated in several projects in cooperation with industrial and academic partners. It supports the requirements engineering process by providing the necessary variability concepts and analysis functions.

3.5 Variability-Oriented Reuse of Functional Requirements in the Framework of a Function Signal Network in Telelogic DOORS – Current Research Approach at DaimlerChrysler

by Ramin Tavakoli Kolagari, Matthias Hoffmann, Johannes Fasolt

DaimlerChrysler is a premium car manufacturer developing automobiles for a global market. DaimlerChrysler's global presence and its innovation-oriented brand mean that development activities must meet two essential challenges of a worldwide market. First, the cars' features must be innovative and manifold and at the same time take account of current trends in the different markets. Second, the cars must be developed in short development cycles and to a high level of quality. To meet these challenges, development activities must be reuse-oriented. Since the actual development process of a car manufacturer typically takes place on the level of requirements (that are passed on to suppliers for implementation), requirements specifications for different car series must be of high quality and at the same time rapidly produced. This is only possible with systematic reuse approaches.

This section motivates, why developing requirements for electronic control units (ECUs) for a wide platform of different model ranges each comprising a whole bunch of model range specific variability is a challenging task for automotive original equipment manufacturers (OEMs). Furthermore it is shown how current shortcomings of requirements specifications can be overcome by using a product line oriented concept called Function Signal Network (FSN). Section 3.5.1 looks at a FSN in more detail and explains its use in the tool DOORS.

Variability of functions in automotive ECUs depends on different car configurations or optional equipment that can vary from one model range to another. The functions are networked and thus constitute a complex web of dependencies. Since OEMs develop most of their ECUs with the help of suppliers, their main interest is to develop the requirements for the ECUs on a concrete level consistent with the specific goals of their organization.
Requirements constitute the main interface between a supplier and OEMs and thus the right level of concretion for the description of requirements must be met such that an integration of the resulting ECUs into the vehicle can be realized. Many functions do not vary much from one model range to another, but innovative functions are often developed that are not in the focus of the presented concept. Nevertheless, the system design can differ greatly, which means that the distribution of functions over the ECUs of different model ranges changes. Analyses conducted at DaimlerChrysler show that requirements described independently of a specific hardware or software design have significantly higher reuse potential than design-oriented requirements. Typically, requirements documents are described more concretely than characteristic requirements like features, describing functions from a customer's point of view and that are the basis for management decisions. Hence, requirements documents exhibit a developer-centric technical view of functions. These functions consist of subfunctions that communicate with one another by means of signals.

Reuse can be established mainly in the dimension ECU1-ECU2, describing the variability between different ECUs with respect to time, model range or configuration. In the context of OEMs, handling variability is even more complex than in classic variability problems because the kind of variability here is two-layered here: classic variability problems encompass a set of products that share certain features and have differences. Differences must be made visible and an easy way to do so is by using feature diagrams, as proposed in [5]. In the automotive domain, this level of variability is in the automotive domain only the first layer; variability within a model range with respect to country variants, optional equipment, body variants … This kind of variability we call *model-specific variability*. But the need to reuse goes beyond model ranges – in fact, the need to reuse requirements for an OEM arises because of the necessity of managing these different model ranges. And model ranges differ from one another with respect to their different features, different technical environments and different management decisions – and they also differ with respect to different model-specific variabilities, e.g. optional equipment may become a mandatory feature from one model range to another one. This kind of variability is called *model-independent variability*.

To systematically support reuse within an OEM, it is important to develop a process to handle this two-layered variability and to have reliable tools at hand to support this process. In the context of this report, we are unable to present the process in detail, but we sketch the use of DOORS in the described scenario. Please note that the proposed ideas are still being developed and reliable practice experience was not available on publication of this report.

The basic approach to managing the two-layered variability is to construct a Function Signal Network (FSN) comprising a set of functional requirements or functions with input and output ports for signals. To facilitate reuse, functions are described independently of any specific design and subfunctions are structured in such a way that they can be easily composed to functional specifications. To support the developer of an FSN, we have developed a set of rules, including examples, patterns and instructions. Functions embrace the functionality of the selected ECUs for all model ranges, i.e. the FSN library is on the level of model-independent variability. Some subfunctions of the FSN directly represent a feature and can thus be identified as part of a model-specific specification. Other subfunctions are needed to ensure the correct interaction of already selected subfunctions and can be identified by the selected signals until all communication relationships are complete (for every selected signal there exists a generating function, which can be within the system or part of the system interface, and a consuming function, again either part of the system or of the system interface).

3.5.1 Variability in the Function Signal Network (FSN)

This section presents the elementary objects of an FSN library (the functional requirements) and the rules for combining these objects into a model-specific specification. It is concrete and very detailed and addresses readers who wish to deepen their understanding of an FSN.

Modeling with a FSN library

Functions are represented as objects with a unique identifier, textual description and relations to signals. A function may be linked to an arbitrary number of input signals and output signals. Noninstantiated functions in the FSN library include input and output ports for signals, either obligatory or optional.

Signals are described as objects with a unique identifier. A signal may be linked from an arbitrary number of functions as input as well as output. In a correct instantiation of the FSN into a model-specific specification, each signal may only be generated once as output.
Variability in the library of FSN objects is described as a property of the use of signals by functions, which is represented as a relation. A signal may be used obligatorily or optionally

by a function (with respect to its port). The influence of the presence or absence of an optional signal on a function is only visible in its textual description.

The starting point for an instantiation of an FSN into a model specific specification is a set of abstract features that are initially selected. The instantiation now becomes tricky because further variability occurring during the instantiation process must be properly managed: all the decisions made during the instantiation must be consistent and the algorithm leading a user through the possible flat tree of hierarchy in the FSN library must enable the user to jump back to each decision and undo the set of following decisions. Because of the net of relationships between the objects describing all implicit dependencies, we have the problem that there is no deterministic way through the tree of objects, which means that the decisions made may preclude possible subsequent decisions that the user wishes to make. However we don't wish to deepen the discussion about the requirements regarding an algorithm for a product line wizard because this is currently a broadly discussed problem in the community developing tools for product lines (e.g. [4]).

During the instantiation process, the features must therefore be deduced into one or more functions. There exists an Or-relationship between features and their deducible functions.
For an instantiation of a specific FSN from the library, we assume an initial feature selection. The user decides

- for each selected feature, which deducible functions are to be instantiated for the specific FSN, in which at least one function must be deduced for each selected feature
- for each optional signal usage of the selected functions, if they are part of the instantiated FSN,
- for each signal, if it is generated within the FSN (as a unique output signal) or if it is part of the system's interface and is thus generated beyond the borders of the system
- for each signal, if it is consumed within the FSN (as an input signal) or if it is part of the system's interface and is thus consumed beyond the borders of the system

The instantiation of a specific FSN (being a predecessor of a model-specific functional specification) is completed if

- for each feature and each deduced function it is decided whether or not it is selected
- for each selected function and for each linked signal it is decided whether or not it is used by a function
- for each selected signal it is decided, whether it is part of the system interface

Using DOORS

In the above section we describe how we propose to use a Function Signal Network library to derive specifications. Note that the specifications themselves include a large amount of variability. The FSN library thus includes variability on a more abstract layer, also comprising variability of variability. Currently, there are no tools on the market supporting users in such a two-layered variability setting. The tool widely used by German automotive OEMs is Telelogic DOORS [8] and we developed a first prototypical extension of DOORS realizing the above-described scenario. Since it is easy to adapt DOORS to specific needs based on a tool-specific script language (doors extension language, dxl), we were initially able to develop the prototype. We tried to implement the described FSN library with the signals representing dependencies and with the obligatory or optional ports modeling variation in such a way that the properties of DOORS were sufficient. Nevertheless, DOORS is not a tool for managing requirements of a family of products, and much less a tool for managing requirements for a set of families of products, as we need here, in the context of reusing requirements for automotive model ranges.

4 Requirements for Requirements Management Tools in the Context of Product Line Engineering

Based on the practical use scenarios described above, we wish to derive requirements for requirements engineering tools for software product lines. Our aim is to provide the requirements engineering community, and especially requirements management tool developers, with a catalogue of current requirements to adapt their tools to be make them suitable for product line engineering.

4.1 Starting Point

The basis for the requirements catalogue is a detailed description of tool requirements for single product developments, written by Hoffmann et al. in [2]. Additionally to the presented paper, they developed a comprehensible requirements catalogue for single product development tool. This catalogue comprises about 100 requirements, which are hierarchically structured. The top level serves as a grouping by stakeholders, the second level contains categories, which group related requirements:

- Requirements from tool users
    - Information model
    - Views
    - Formatting, multimedia and external files
    - Documentation of history
    - Baselining
    - Traceability
    - Analysis functions
    - Tool integration
    - Import
    - Change management and comments
    - Document generation
    - Collaborative work
    - Checking out for offline use
    - Web access
- Requirements from project administrators
    - Users, roles and rights
    - Size restrictions
    - Workflow management
    - Extensibility
- Requirements from tool administrators
    - Database
    - Encryption

Each requirement has also been prioritized by the authors. The existing catalogue has been taken and adapted in the following way, to make it suitable for product line development:

1. The documented requirements have been revised and partially reformulated
2. Requirements that become relevant in the context of product line engineering have been added to the catalogue. About 20 new requirements and three requirements categories (Configuration Management, Discussion Support and Priorities) have been additionally documented.
3. Each requirement has been prioritized. For each requirement, it has been defined whether it is equally important, more important or most important in a product line context compared with single product development.

The full catalogue of requirements and their priorities can be found in the appendix. The following subsections describe the new elicited requirements and the prioritization in detail.

4.2 New and Changed Requirements

To switch the focus from single product development to product line engineering, roughly ten percent new requirements were introduced and about the same number of requirements was changed to make them better match the product line context. Some of the new requirements are directly tied to variability handling. These will be discussed later in this section. First, we present the requirements that are not directly related to variability.

Most new requirements (11) belong to the newly introduced “configuration management” category. However, most of the requirements in this category deal with issues that are of similar importance as in single product developments. Some of these issues have been implicitly represented in the category “Documentation of History”, but since this was a mixture of configuration management issues and change management, this has been separated. The first block of new requirements deals with basic version and configuration management requirements like object identification and versioning. These requirements are relevant in any development. Support for baselining of the requirements database was considered more important in a product line context. The focus here is to be able to baseline arbitrary parts of the database in order to capture easily the state of several projects/products at once.

In a similar fashion, support for multi project/multi product status and progress reporting was considered a necessary new requirement for product line tools.

Also important, but not covered in the original requirements list, is good support for tool-based communication among users. In product line engineering, artifacts have to be discussed more intensively across a larger (and often distributed) group of people. An integrated but relatively informal type of discussion support is considered necessary in addition to the formalized change process, which the tool should support anyway. Examples are Wiki-like discussions or forum discussions which are directly linkable or linked to a specific (set of) requirement(s).

In order to break down work, the definition of arbitrary, named subsets of requirements is also a more important issue for product lines because the number of requirements is usually higher than in single product developments and some partition can be helpful.

The changes we made to existing requirements were mostly desinged to clarify their meaning, not to change their intention, so a detailed discussion of changed requirements is not considered necessary.

4.2.1 New Requirements Related to Variability

This section discusses the newly introduced tool requirements related to variability. In total, we have added only five requirements that fall into this category.

It is important that the tool supports shared multi project and multi product information models in order to provide a consistent modeling infrastructure for related projects or products that are part of the same product line development. A simple information model copy when a new derived project starts is not sufficient here because changes made afterwards have to be made individually to all related projects, which can be problematic if there are a lot of them.

Additionally, the information model provided by the tool must support the expression of variability and variation. In other words there must be ways of expressing variation points, rules for variation point instantiation and the description of relations between product variants in the tool's information model.

(Defined) variants must have a first class representation in the tool which allows the addition of "meta" attributes to the variant. Examples of such meta attributes are a list of stakeholders for this specific variant (customer, account manager, etc.) or links to other variant-specific artifacts.

Given an information model that provides support for variability definition, the tool not only offers support for variant representation but must also support the variant creation process using the stored variability information. Dependencies between variants must be expressed, e.g. it should be possible to check in which variant a requirement is realized or to compare variants with each other.

A similar requirement is the need for multidimensional prioritization of requirements in the tool with respect to different stakeholders. This can be used to help the track of individual realizations of several products using the same requirement. Use cases for such functionality are manifold. For instance, one project might decide that a feature described in a requirement is a "nice-to-have" item, while it might be essential for other projects.

4.2.2 Relationships to Product and Project Management

The management of variability in requirements is not a task in a delimited area of requirements engineering. In fact, product and project management must be considered major stakeholders requirements management. Both need special information about variability.

Product Management

One of the main tasks of product management is to plan and control the process of product development. Within this context, requirements engineering is a consumer of information produced by product management (e.g. new market-driven requirements) and a supplier of information needed by product management to make strategic decisions in product development.

The following questions therefore come to mind:

- Should the new product be developed from scratch or should it enhance an existing product?
- Which relationships exist between the new product and existing products?
- Can the basic functionality be conserved while new features are implemented?
- Should the new product be a variant of an existing product?
- How do the changes influence the product life cycle?

These are just a few of the questions that need to be considered during product management, but they are a fundamentally related to requirements management in the context of product line engineering. How can a product manager answer these questions without knowing the requirements? In addition to the number of features, the number of variants must also be considered, to make accurate cost estimation.

Furthermore, not only the development and product costs are influenced by the number of variants, but also the costs that must be considered during the product life cycle, e.g. update effort, maintenance costs and staff training.

Requirements engineering must therefore support the product management by providing information about dependencies between features. The more complex the requirements are, the more adequate tool support is needed to facilitate the management of information.

Project Management

The task of the project management in the context of software development is to plan and control the project. The main focus is on the functionality that should be provided by the new product. The change in functional requirements has a strong impact on the project plan.

If products are being developed in the context of a product line, multiple projects often use a common set of requirements concurrently. Without planned coordination of the projects, multiple implementations of the same requirement are the result, which leads to an enormous overhead in development. Since the additional effort consumes additional resources, project managers have even less time for coordination between the projects – a vicious circle.

Again, the information about dependencies between requirements across products must be provided by requirements engineering. Integration of requirements engineering can help to distribute the different tasks across projects and therefore to optimize the use of resources. The main advantage can be achieved by verification and validation. These two tasks are typically performed half-heartedly at the end of a project.

Requirements engineering must thus be based on an adequate tool to support project management by providing the necessary information. Furthermore, it is important that information in projects be swiftly available to enable effective control of projects.

To summarize, the coordination of product, project and requirements management is necessary to effectively handle variability in requirements. Adequate tool support is essential.

4.3 Prioritization of Requirements for the Context of Product Lines

This section presents the prioritization of tool requirements defined by the working group. The starting point was the list of tool requirements for single product development and the associated prioritization suggested by [2]. The working group rated each tool requirement with regard to product lines. Each tool requirement from single product development was rated as (1) equally important, (2) more important, or (3) highly important in a product line context. The working group also identified further requirements specific to product line development, as described above, and rated their importance.

The prioritization was derived by a vote among the members of the working group. It was validated through subsequent discussions. As a result of this approach, some priorities are not unique. This is due to the different backgrounds of the working group members in terms of application domain and project type. This report of the working group's results documents the variation in priorities, allowing the reader to make a well-informed evaluation of the proposed requirements list.

This section begins by presenting the key results and findings of the requirements prioritization for the context of product lines (Section 4.3.1). It then goes on to describe the peculiarities of each group of requirements in a product line context (Section 4.3.2). A complete list of tool requirements and their prioritization is given in the appendix.

4.3.1 Particular Priorities in a Product Line Context

In the context of product lines, five aspects of requirements management tools are particularly important:

- Explicit modeling concepts for requirements representation
- Changeability and adaptability of requirements representation and functionality
- Graphical presentation and visualization of requirements and their interrelations
- Collaborative work during requirements management
- Management of multiplicity in requirements definition (multiple products and projects, as well as interrelations between them)

Often, product lines pose very specific requirements on requirements management tools. It is scarcely conceivable that a tool is able to meet all these different requirements and use modes. This is due in part to the fact that the various processes, methods and organizational infrastructures applied in product line development are not yet fully understood and established. It is also due to the many different ways in which product line development is implemented throughout industry. It must therefore be possible to tailor and adapt a requirements management tool to the specific needs and characteristics of a given product line development infrastructure.

The tailoring of requirements management tools to product line development contexts requires that a tool include modeling concepts that are explicit, modifiable, extendable and adaptable. Typically, tailoring of requirements management tools is achieved by well-accessible programming interfaces, the embedding of scripts or programs into a tool, as well as extensible user interfaces (e.g. via plug-in mechanisms).

Graphical presentation and visualization are important to enable the user to keep track and control of the complex information structures involved in product line engineering.

Examples of such visualizations are dependency and traceability graphs, as well as illustration or highlighting of variability across requirements sets.

Collaborative work during requirements engineering and management is much more important in product line contexts than it is during single product development. This is due to the large number of stakeholders that a product line typically involves. Also the project-internal activities for collaboration and communication among management, platform development and product development are complex. Requirements management tools must support this collaborative work. They must also trace and document communication and communication results.

Managing multiplicity is essential in a product line context because there is always more than one product being developed, and there is always more than one project being conducted. There exist various interrelations and dependencies between the products and projects, which must be represented and managed by the requirements management tool and be understood by the users.

4.3.2 Product-Line-Specific Priorities per Requirements Group

This section describes the peculiarities of each category of requirements for requirements management tools listed in [2] in a product line context. Figure 2 visualizes the average priorities given by the group members for each requirement. To improve readability, the requirements are represented by diamonds. The sectors represent the requirements categories and the concentric circles represent the priority of the requirements in the context of product lines. The innermost circle contains requirements that are most important in the product line context, the outermost circle contains requirements which are equally important in the product line context compared to single product development.

An Analysis of Figure 2 reveals that it is mainly requirements from the categories Information Model, Views and Analysis that become most important in the product line context. A detailed analysis is given below.

User Requirements: Information Model

Priorities of tool requirements are the same as for single product development. For product lines, there are two additional requirements, both of which are highly important:

- The tool must support multi project and multi product requirements management information models (RMIs) (highly important)
- The tool must support variability mechanisms (highly important)

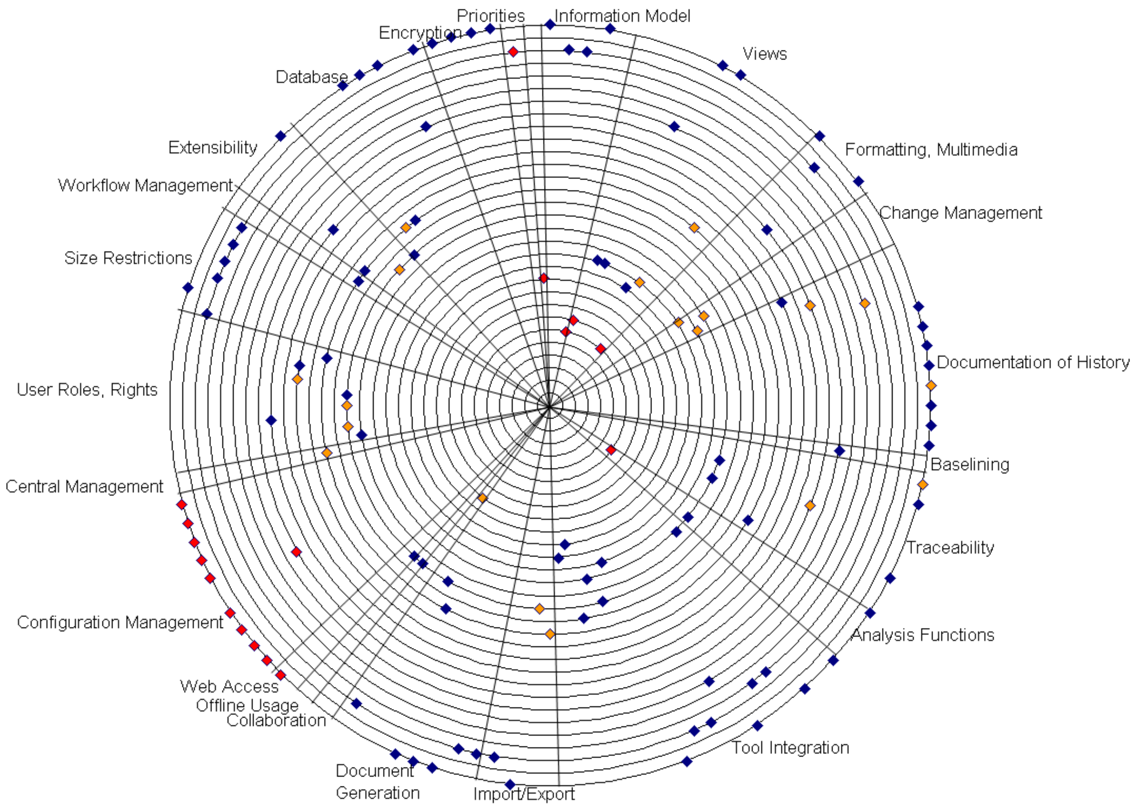


Figure 2: Prioritization of requirements in the context of product lines

User Requirements: Views

Views-related requirements are more important, but *only rarely much more* important, than for single product development. They are particularly relevant with regard to the following aspects:

- Definition of user-specific views
- Availability of graphical diagrams
- Configuration and rule-based control of views
- Modification of views in the course of a project

For product lines, there are two additional requirements:

- The tool should allow views to be predefined for user roles (important)
- The tool must support multi project and multi product requirement views (highly important)

User Requirements: Formatting, Multimedia and External Files

Mainly the same as for single product development.

User Requirements: Change Management and Comments

More important than for single product development.

User Requirements: Documentation of the History

Mainly the same as for single product development. For product line development, the following three requirements are slightly more important:

- All changes to the requirements must be tracked.
- All objects managed in the tool must be versioned.
- There must be a distinction between major and minor versions regarding objects.

User Requirements: Baselining

Mainly the same as for single product development, slightly higher importance.

User Requirements: Traceability

Slightly more important than for single product development, in particular with regard to the following requirements:

- It must be possible to define attributes for links.
- It should be possible to create rules governing what kinds of objects must have links to what other kinds of objects.
- Links must connect all objects in the database, not only in the same subset (module, project, etc.). (important to highly important)
- The tools must feature a practical, user-friendly and concise graphical representation, and navigation of the traces (e.g. matrices, trees or graphs).

User Requirements: Analysis Functions

Slightly more important than for single product development, in particular with regard to the following requirement:

- The tool should allow inconsistencies in the link structure to be analyzed (e.g. find gaps in the traces).

For product lines, there is one additional requirement:

- The tool should provide information on the status and progress of multiple projects and products. (highly important)

User Requirements: Tool Integration

Slightly more important to much more important than for single product development, in particular with regard to the following requirements:

- The tool must have open interfaces to other tools used in the development process and make information stored in them visible and linkable. (more important to much more important)
- Access rights to the external objects must be recognized. (partly more important)

Tool classes that ought to be integrated with requirements management within product line contexts are:

- Slightly much more important than for single product development: Configuration management, communication (e.g. e-mail communication), project management
- Slightly more important than for single product development: Test, validation, and verification, problem tracking, modeling and design

User Requirements: Import / Export

Slightly more important than for single product development, in particular with regard to the following requirements for requirements information import/export:

- The tools should be able to *import* existing requirements specification documents based on a predefined, customizable exchange format.
- The tools should be able to *export* existing requirements information based on a predefined, customizable exchange format.

User Requirements: Document Generation

Mainly the same as for single product development. For product line development, the following two requirements are more important:

- The tool must swiftly generate very large documents incorporating numerous external objects. A 5,000-page document with formatting and media objects should be generated overnight.
- It should be possible to run the document generation automatically as a background task.

User Requirements: Collaborative Working

Much more important than for single product development.

User Requirements: Checking Out for Offline Use

More important than for single product development.

User Requirements: Web Access

More important than for single product development.

Requirements Regarding Project-Related Tool Administration

Mostly more important than for single product development. In particular, the following aspects are more important for product line development than they are for single product development:

- Central installation and administration of projects
- Users, roles and rights
- Workflow management
- Extensibility

Requirements Regarding Technical Tool Administration

Slightly more to much more important than for single product development.

- *Database* requirements are Slightly more important for product line development with regard to scalability, availability, and backup and restoring.
- *Requirements prioritization* (stakeholder-specific) is much more important for product line development.

5 Evaluation of Tools

5.1 Explanation of the Evaluation Approach

Starting with the concrete list of requirements for product line tools, we were delighted to learn how current tools available on the market support product line engineering. Fortunately the team had some practical experience with some of the market-leading products.

Several interesting questions arose, which we hoped to answer based on the tool evaluation:

- Are there any common deficiencies across the tools?
- Do some tools clearly outperform others in terms of product line requirements?

It was clear from the beginning that these questions could only be answered very subjectively given the working group's limited time and resources.

With the requirements table described in the previous chapter, we already had a means to distinguish between typical requirements for requirements management tools and those that are more important in the context of product lines. The goal of the product evaluation was to find out which tool is better suited for product line development.

To reduce the degree of subjectivity we created an evaluation schema based on numerical values with associated semantics:

Value	Semantics
0	Don't know.
1	The requirement isn't supported by the tool.
2	The requirement isn't supported in a way that is suitable for day-to-day operations. Custom solutions (extensions) are required (and possible) to

	address the requirement.
3	The requirement isn't supported adequately but it can be addressed with organizational conventions to use the tool in a specific way.
4	The requirement is addressed well enough to do daily work.
5	The tool's implementation of the requirement is perfect (at least we cannot imagine a better way of doing it).

At first, we thought that the criteria were sufficiently well defined. However, when trying to evaluate the tools, a few problems arose.

The first was that a wide range of subjective interpretation was still possible. We believe that the situation can be improved by applying fit criteria to requirements, which we failed to do despite knowing that this is good requirements engineering practice.

Another reason for the lack of clarity is the configuration and customization capabilities of the tools. Based on above schema, values of 2 to 4 can be applied, depending on how customization is seen by the evaluator. One can argue that the tool should address the requirement out of the box but we can also argue that customization allows more flexibility.

After some discussion, we reached agreement within the group. Future work might focus on improving the evaluation schema.

5.2 Visualization

Once we had obtained the data, the challenge was to visualize the various dimensions:

1. Show the importance of the requirement with regard to product line support
2. Show how well the tool addresses the requirement
3. Be able to compare tools
4. Visualize requirement categories
   (The table has a series of categories for requirements)

We experimented with various display formats and finally decided that spider charts are best suited because they address all the dimensions described above. Bubble charts and 3D column graphics failed to visualize all dimensions.

The following charts show how well the analyzed tools address the specific requirements. The categories *View*, *Analysis Functions*, *Information Model*, *User Rights and Roles* and *Configuration Management* have been chosen because requirements belonging to these categories are most important in a product line context.

The red line indicates the product line priority of the specific requirements. Using this line, it is easy to evaluate how well the respective tools perform. Figures 4 to 8 visualize the evaluation.

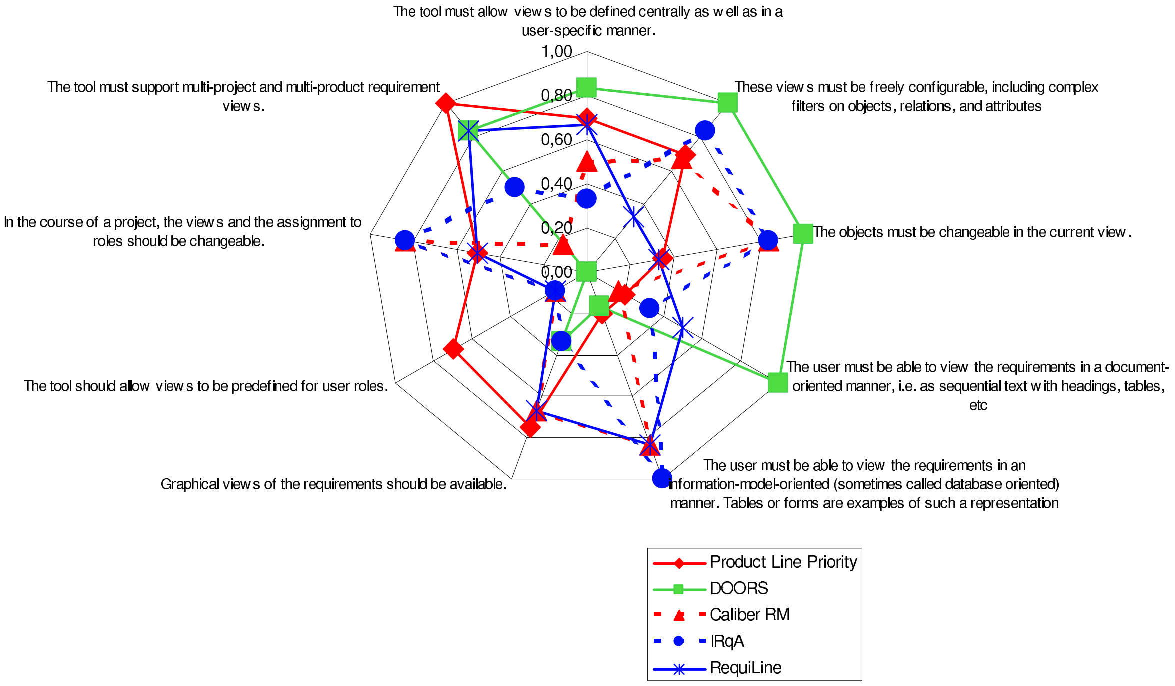


Figure 3: Category *View*

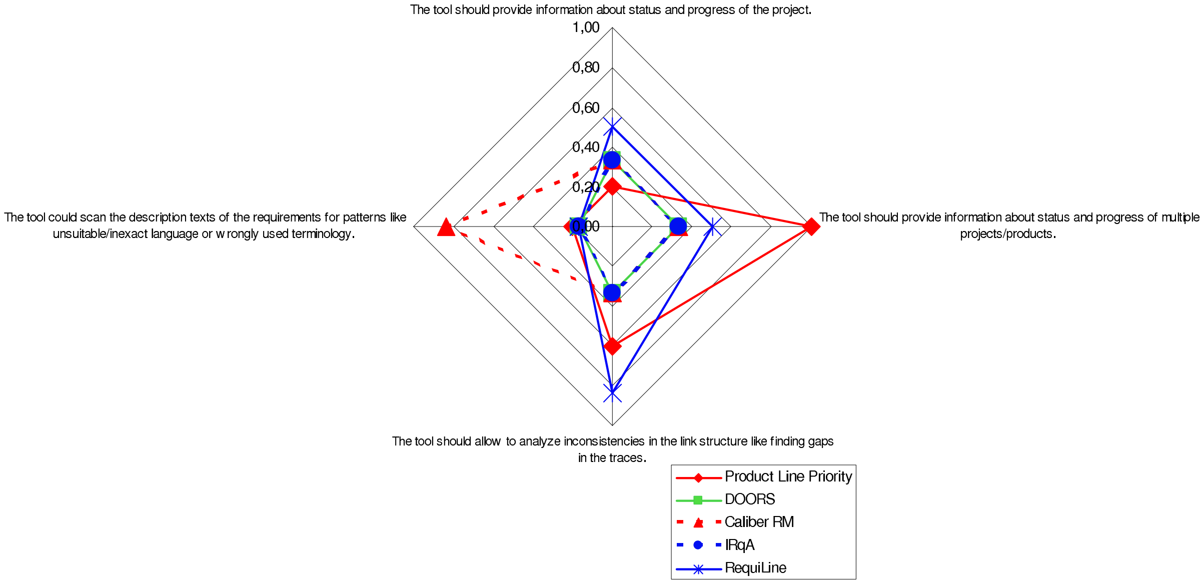


Figure 4: Category *Analysis Functions*

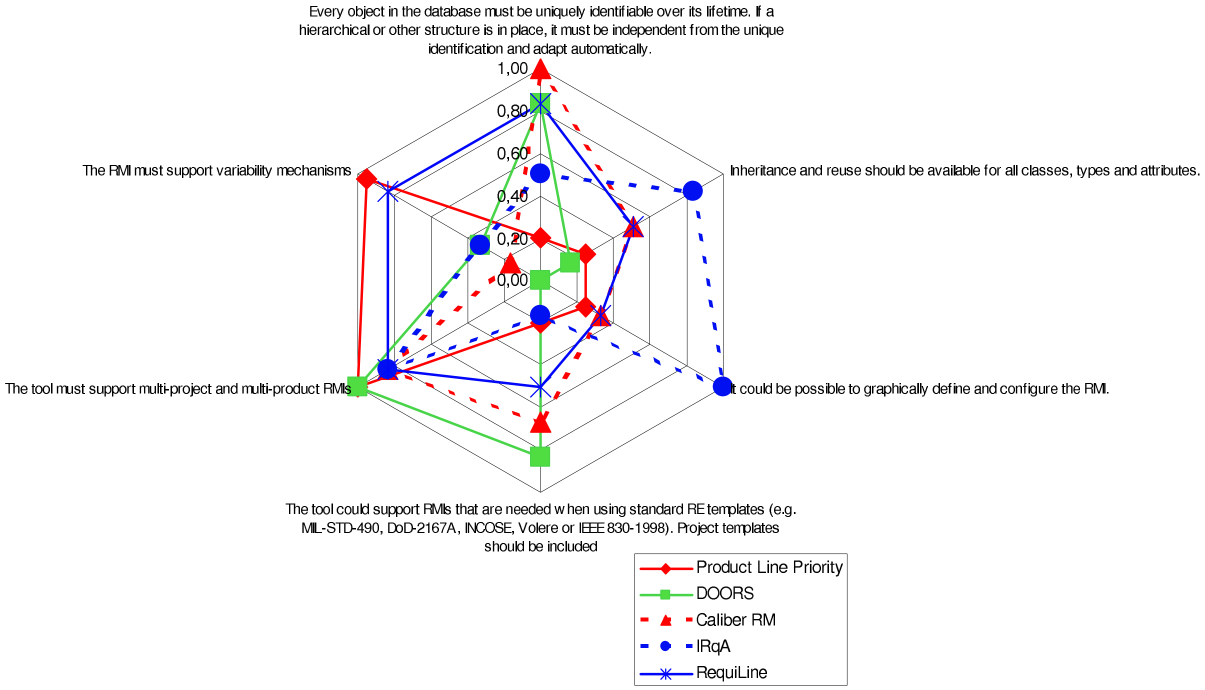


Figure 5: Category *Information Model*

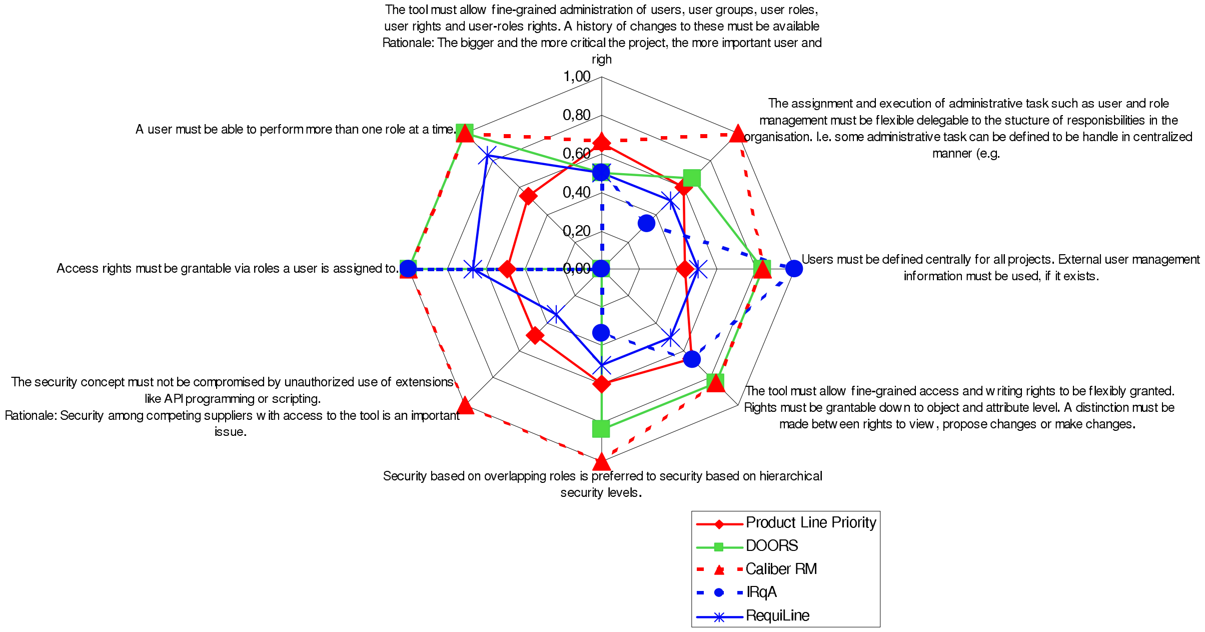


Figure 6: Category *User Rights and Roles*

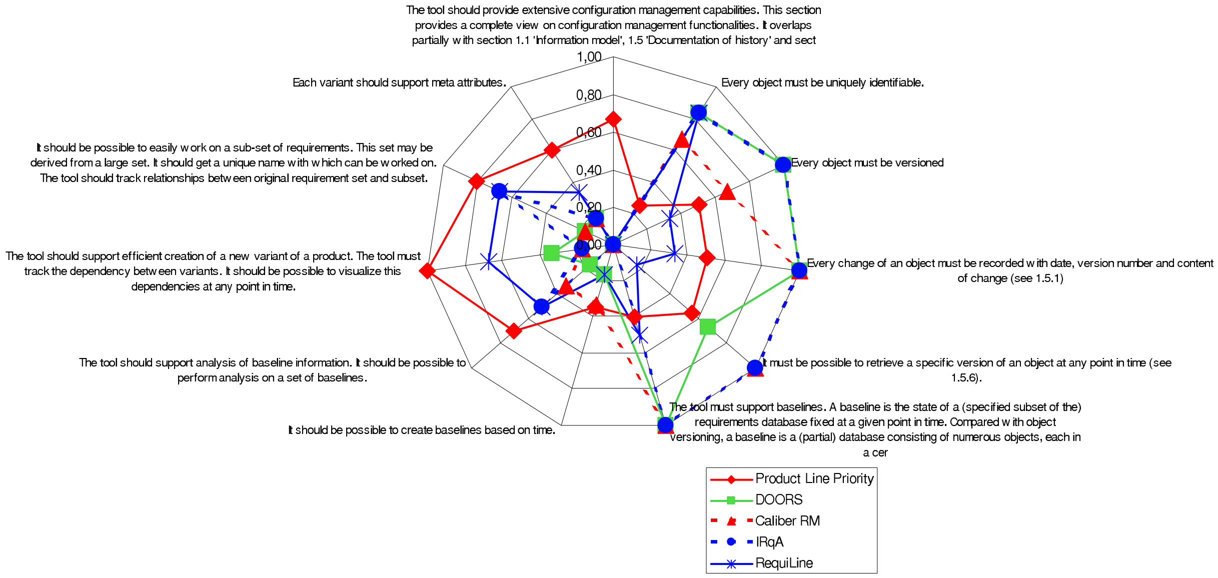


Figure 7: Category *Configuration Management*

6 Summary and Future Work

A software product-line-oriented approach to developing systems has an impact on the development process as a whole. All artifacts produced during the process have to be properly managed in order to benefit from a product-line-oriented approach. Although the basic idea of product lines is simple, systematic implementation of the approach is a challenging task. In fact, today's software system families have such complex variability that their handling must be supported by tools, otherwise a systematic approach would not be possible.

The same holds for requirements management of software product lines. The technical report presented here analyzes specific scenarios in the context of requirements management and software product lines and derives important requirements that have to be observed if requirements management tools are to be usefully applied to product lines. Current requirements management tools are evaluated on the basis of these requirements. This evaluation revealed that all requirements management tools currently used in industry need to be improved if they are to be successfully used in a product line setting. Nevertheless these tools are used in practice and are also used in product line settings and as the scenarios described in this report show they are also used successfully – but this is only achieved because workarounds are in place: either the tool itself is enhanced or related processes bypass shortcomings of the tool. But this situation is not satisfying, because especially small companies need to rely on all-embracing tool support because enhancements or heavy processes to overcome shortcomings of tools are too expensive.

The presented scenarios describe current development procedures formulated by the working group members from industry. The scenarios make it clear that a software product-line-oriented approach is possible in a development process where all produced artifacts are managed in one organizational unit as well as in a split development environment, e.g. OEM and supplier. Nevertheless, requirements management in these settings needs specific support. Recently – there being no commercial requirements management tool tailored to product lines – a whole series of flexible workarounds have emerged. Flexible tools like MS Excel or MS Word provide the means to realize these workarounds – and they are actually effective: Being born of a practical need, these workarounds are pragmatic as well as necessary and have therefore been widely adopted.

However, mature software development that is geared to product lines requires more than pragmatic, stop-gap solutions. Systematic software product line engineering calls for systematic tool support especially in the context of requirements management, mainly because one is confronted with the task of handling a huge number of requirements for multi projects and multi products. This has an impact on the information model, views, baselining, etc.

The requirements for requirements management tools were gathered by the working group and prioritized by each working group member based on his/her subjective rating of their relevance for product lines. The result is a comprehensive analysis of requirements and requirements management tools in the context of software product lines based on practical experience. The list gives an overview of the key requirements – some are newly introduced to support product lines. Especially scalability is the main driver to identify requirements or to prioritize requirements in the context of product line engineering, because in this case a huge set of data items must be handled.

None of the investigated tools (DOORS, CaliberRM, IRqA, RequiLine) supported all or most of the presented requirements best, but each tool has its strengths and weaknesses and in order to decide which tool meets best the demands of a specific organization or development approach one has to decide on the basis of the presented requirements independently.

The presented analysis and the results are so far unique. This technical report is intended to direct the attention of both researchers and tool developers to the current problem of inadequate requirements management tools for software product lines. The presented requirements indicate the future direction of tool development.

A reader with interest in tool supported requirements management for software product lines can gain the following from the report at hand:

- Current tools can be used in product line settings as the described scenarios show, but this is only possible with a huge amount of extra effort.
- The list of requirements given in this report is a basis to implement requirements management tools supporting software product lines.
- Current organizational processes can be analyzed for actual requirements and the fulfillment of these requirements in different tools can be seen in the requirements list at hand. The adequate tool can then be selected.
- The catalogue of requirements was reviewed a second time (because it bases on the published version of requirements for requirements management tools [2]) and is also interesting for single product development.

To substantiate the findings of the report, the working group has set up a website http://www.gi-ev.de/fachbereiche/softwaretechnik/re-pl/, where interested people with practical experience in using one of the listed tools are invited to evaluate them on the basis of the requirements catalogue. The website also provides more detailed information about the working group and its members.

7 References

[1]	Borland, Caliber website: http://www.borland.com/de/products/caliber/index.html
[2]	M. Hoffmann, N. Kühn, M. Weber, M. Bittner: *Requirements for Requirements Management Tools*, Proceedings of 12th IEEE International Conference on Requirements Engineering (RE 2004), 6-10 September 2004, Kyoto, Japan. IEEE Computer Society 2004, ISBN 0-7695-2174-6.
[3]	QA Systems, IRqA website: http://www.qa-systems.de/html/deutsch/produkte/irqa/irqa.php.
[4]	G. Jiménez-Pérez: *Design Wizards for Software Product Lines*, Workshop on Generative Techniques in Product Lines, First Software Product Line Conference, Ed. G. Butler, K. Czarnecki, U. Eisenecker, http://www.cs.concordia.ca/~gregb/splc-workshop.
[5]	K.C. Kang, K.C., S.G. Cohen, J.A. Hess, W.E. Novak, and A.S. Petersen: *Feature-Oriented Domain Analysis (FODA) Feasibility Study*, Software Engineering Institute of the Carnegie Mellon University, CMU/SEI-90-TR-21, Pittsburgh, 1990
[6]	Research Group Software Construction, RequiLine website: http://www-lufgi3.informatik.rwth-aachen.de/TOOLS/requiline/.
[7]	K. Schmid, A. Birk, G. Heller, I. John, S. Joos, K. Müller, T. von der Maßen: *Report of the GI Work Group "Requirements Engineering for Product Lines*", IESE-Report No. 121.03/E, November 2003.
[8]	Telelogic, DOORS website: http://www.telelogic.com/corp/products/doors/index.cfm
[9]	T. von der Maßen, H. Lichter: RequiLine: *A Requirements Engineering Tool for Software Product Lines*, Software Product-Family Engineering, Springer LNCS 3014, 2004.
[10]	J. Robertson, S. Robertson: *Mastering the Requirements Process*, Addison-Wesley, 1999.

8 Appendix

The following table lists the complete set of identified requirements for requirements engineering tools in the context of product line engineering, based on the requirements presented in [2]. The individual requirements have been prioritized for the context of product line engineering and the result of the voting by the members of the work group are presented in the last three columns. The maximum number of points reveal the maximum score, if all members would vote that the requirement is "highly important" in the product line context.

The following table shows furthermore the results of the evaluation of the analyzed tools. Each tool has been evaluated according to the procedure presented in section 5.1 using the mentioned evaluation scheme based on numerical values with associated semantics:

Value	Semantics
0	Don't know
1	The requirement isn't supported by the tool
2	The requirement isn't supported in a way that in can be used in day to day operations. Custom solutions (extensions) are required (and possible) to address the requirement.
3	The requirement isn't supported adequately, but the requirement can be

	addressed with organizational conventions to use the tool in a specific way.
4	The requirement is addressed well enough to do daily work.
5	The tools solution to the requirement is perfect (at least we cannot imagine a better way of doing it)

Finally, we analyzed how the specific tools perform with respect to the priority we defined for each requirement. We use a color spectrum from blue over green to red to indicate the level of the priority respectively the level of fulfillment of the requirement. In the following table the color of a cell border indicated the level of the priority (scaled up to 50) whereas the background color of a cell indicates the fulfillment.

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
1.	Requirements from the Users													
1.1.	Information Model		The tool must allow the user to freely define a RMI. Rationale: Since a requirements management tool must be independent of process and method, the requirements must be modeled freely in the tool. The detailed mapping of a process and its artifacts to a requirements management tool can be described using a RMI. Experience shows that especially in pilot projects this mapping varies to achieve a higher benefit from a requirements management tool.											
1.1.1.		Information Model	Every object in the database must be uniquely identifiable over its lifetime. If a hierarchical or other structure is in place, it must be independent from the unique identification and adapt automatically.	++	8	0,20	5	6	3	5	42	50	25	42
1.1.2.		Information Model	Inheritance and reuse should be available for all classes, types and attributes.	+	10	0,25	1	3	5	3	8	25	42	25
1.1.3.		Information Model	It could be possible to graphically define and configure the RMI.	-	10	0,25		2	6	2	0	17	50	17
1.1.4.		Information Model	The tool could support RMIs that are needed when using standard requirements engineering templates (e.g. MIL-STD-490, DoD-2167A, INCOSE, Volere or IEEE 830-1998). Project templates should be included	-	8	0,20	5	4	1	3	42	33	8	25
1.1.5.		Information Model	The tool must support multi-project and multi-product RMIs		40	1,00	6	5	5	5	50	42	42	42

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
1.1.6.		Information Model	The RMI must support variability mechanisms		38	0,95	2	1	2	5	17	8	17	42
1.2.	Views		The tool must support various views of the same data. A view offers the possibility to view and change a freely defined collection of parts of the data of several projects or subprojects in a freely configurable representation. Rationale: Depending on the current process step requirements management tool users work only on certain aspects of a certain part of the specification. It is therefore important, that a requirements management tool provides suitable views of the huge amount of information accessible in requirements management tools. This has a strong impact on the acceptance of the tool by the users.	++										
1.2.1.		Views	The tool must allow views to be defined centrally as well as in a user-specific manner.	++	28	0,70	5	3	2	4	42	25	17	33
1.2.2.		Views	These views must be freely configurable, including complex filters on objects, relations, and attributes	++	28	0,70	6	4	5	2	50	33	42	17
1.2.3.		Views	The objects must be changeable in the current view.	+	14	0,35	6	5	5	2	50	42	42	17
1.2.4.		Views	The user must be able to view the requirements in a document-oriented manner, i.e. as sequential text with headings, tables, etc	++	8	0,20	6	1	2	3	50	8	17	25
1.2.5.		Views	The user must be able to view the requirements in an information-model-oriented (sometimes called database oriented) manner. Tables or forms are examples of such a representation	++	8	0,20	1	5	6	5	8	42	50	42

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
1.2.6.		Views	Graphical views of the requirements should be available.	+	30	0,75	2	4	2	4	17	33	17	33
1.2.7		Views	The tool should allow views to be predefined for user roles.		28	0,70		1	1	1	0	8	8	8
1.2.8.		Views	In the course of a project, the views and the assignment to roles should be changeable.	+	20	0,50		5	5	3	0	42	42	25
1.2.9.		Views	The tool must support multi-project and multi-product requirement views.		40	1,00	5	1	3	5	42	8	25	42
1.3.	Formatting, Multimedia and External files			++										
1.3.1.		Formatting, Multimedia and External files	The tool must allow the requirements to be enriched with formatting and objects not native to the tool. Rationale: Many specifications created with text processing tools like Word contain lots of graphics or other multimedia elements. Developers expect similar means of expressions from a requirements management tool, which must be directly visible in the requirements management tool user interface.		8	0,20	6		6		50	0	50	0
1.3.2.		Formatting, Multimedia and External files	The tool should support basic text formatting. It should also support scientific and foreign-language character sets. The tool should allow mathematical formulas to be used in the description texts.	+	10	0,25	2	4	6	1	17	33	50	8

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
1.3.3.		Formatting, Multimedia and External files	Non-text objects should be saved directly in the database or at least in a configuration management tool that is tightly coupled with the tool. If they are stored in the tool database, they must be fully covered by its version and rights control.	++	16	0,40	5	6	6	5	42	50	50	42
1.3.4.		Formatting, Multimedia and External files	External objects must be viewed either through a pre-viewer inside the tool or in the native application if called directly from the tool's user interface.	++	8	0,20	5	4	6	5	42	33	50	42
1.4.	Change Management and Comments		The tool must support change management. This can either be done by the tool itself or the tool should provide a suitable interface that conforms to the following requirements.	++	28	0,70	3				25	0	0	0
1.4.1.		Change Management and Comments	Change requests must be customizable to the change process of the users		24	0,60	2	1	2	1	17	8	17	8
1.4.2.		Change Management and Comments	The change request handling must be integrated into rights management.		26	0,65	2	1	1	1	17	8	8	8
1.5.	Documentation of the History		Rationale: In the usual highly parallel development, which is needed to reduce time to market, developers need to synchronize their specifications periodically. Differences from previous versions must be easy identifiable. During such synchronization steps discussions reconcilement may be necessary. Due to cost and complexity issues it should be possible to partially return to previous versions.	++										
1.5.1.		History	All changes to the requirements must be tracked.	++	18	0,45	5	4	6	2	42	33	50	17

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
1.5.2.		History	All objects managed in the tool must be versioned	++	16	0,40	6	4	6	2	50	33	50	17
1.5.3.		History	There must be a distinction between major and minor versions regarding objects.	+	12	0,30	1	5	1	1	8	42	8	8
1.5.4.		History	The version number should be incremented automatically when certain changes occur.	+	8	0,20	6	6	6	5	50	50	50	42
1.5.5.		History	Changes must be tracked down to the smallest unit of data structures, in most cases to attributes.	++	8	0,20	6	6	6	2	50	50	50	17
1.5.6.		History	Changes and old versions must always be available.	++	8	0,20	6	6	6	1	50	50	50	8
1.5.7.		History	The tool must allow a requirement to be changed back to any previous version anytime.	++	8	0,20	6	1	6	1	50	8	50	8
1.5.8.		History	The tool should visualize the change history.	+	8	0,20	1	6	4	3	8	50	33	25
1.5.9.		History	The tool must generate freely configurable change reports. These reports should relate to views, baselines and generated documents.	++	8	0,20	2	3	2	1	17	25	17	8
1.5.10.		History	A comment should be saved with the change to enable it to be understood later on	-	8	0,20	3	6	6	1	25	50	50	8
1.5.11.		History	Changes could be categorized for analysis.	+	8	0,20	2	1	2	1	17	8	17	8
1.6.	Baselining	Baselining	The tool must support baselines. A baseline is the state of a (specified subset of the) requirements database fixed at a given point in time. Compared	++	16	0,40	6	5	6	3	50	42	50	25

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
			with object versioning, a baseline is a (partial) database consisting of numerous objects, each in a certain version. The development status saved in a baseline is the starting point for further development. Rationale: Baselines are used to save the state of a specified set of requirements objects, a document or project before a larger development step. They also serve to freeze a development object after its completion or review. Baselines are not branches. They do not copy the objects; they are a catalog of object/version references.								🟥	🟧	🟥	🟩
1.7.	Traceability		The tool must enable traceability through links between requirements. The linking must be implemented in a highly user-friendly manner because it helps only if it is relatively complete. Rationale: For years traceability has been one of the big discussion and research issues in requirements engineering. Certain standards for security-critical fields even enforce complete traceability. Unfortunately, linking is not popular among developers because it costs time, its benefit is visible mostly in later phases, and it needs discipline in linking. Good tool support could change this and enable analyses and consistency support that would otherwise require much more effort.	++										
1.7.1.		Traceability	Links must be directed and an object must be a source and target at the same time (but not of the same link). Additionally, the user must be able to create links starting from the source or the target of the directed link.	++	8	0,20	5	6	6	4	🟧 42	🟥 50	🟥 50	🟩 33

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
1.7.2.		Traceability	It must be possible to follow links directly in both directions.	++	8	0,20	5	6	6	4	42	50	50	33
1.7.3.		Traceability	It must be possible to give the links attributes, e.g. to differentiate different kinds of links for later filtering or analysis.	++	24	0,60	3	1	4	3	25	8	33	25
1.7.4.		Traceability	It should be possible to create rules governing what kinds of objects must have links to what other kinds of objects.	+	16	0,40	0	1	6	1	0	8	50	8
1.7.5.		Traceability	Links must connect any objects in the database, not only in the same subset (module, project, etc.)	++	28	0,70	6	6	2	4	50	50	17	33
1.7.6.		Traceability	Links could be n-ary.	-	8	0,20	6	6	6	1	50	50	50	8
1.7.7.		Traceability	The tool must feature a practical, user-friendly and concise graphical representation and navigation of the traces, g.g, matrices, trees or graphs.	++	20	0,50	4	4	3	4	33	33	25	33
1.8.	Analysis Functions		The tool should be able to analyze requirements. Examples are linguistic analysis, analysis of the link structure, analysis of project progress and risk management. Rationale: The enrichment of requirements in a requirements management tool with additional information stored in links and attributes allows automatic analyses that would be costly and time-consuming if done with requirements saved in ordinary documents.	+										
1.8.1.		Analysis Functions	The tool should provide information about status and progress of the project.	+	8	0,20	2	2	2	3	17	17	17	25

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
1.8.2.		Analysis Functions	The tool should provide information about status and progress of multiple projects/products.		40	1,00	2	2	2	3	17	17	17	25
1.8.3.		Analysis Functions	The tool should allow to analyze inconsistencies in the link structure like finding gaps in the traces.	+	24	0,60	2	2	2	5	17	17	17	42
1.8.4.		Analysis Functions	The tool could scan the description texts of the requirements for patterns like unsuitable/inexact language or wrongly used terminology.	-	8	0,20	1	5	1	1	8	42	8	8
1.9.	Tool Integration		The tool must have open interfaces to other tools used in the development process and make information stored in them visible and linkable. Rationale: To improve consistency between development phases and allows complete traceability over the complete product life cycle requirements management tools must be integrated tightly into existing tool environments. The expected benefit is an improved development process and improved product quality. The introduction of a requirements management tool should not result in additional major changes of the tool environment.	++	25	0,71	2				17	0	0	0
1.9.1.		Tool Integration	Linking must not lead to redundant data.	++	7	0,20	1	1	1	0	8	8	8	0
1.9.2.		Tool Integration	The connection should be transparent in both tools.	+	11	0,31	1	1	1	1	8	8	8	8
1.9.3.		Tool Integration	Links to external objects should be managed by the tool in the same way as internal links.	+	11	0,31		6	1	4	0	50	8	33

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
1.9.4.		Tool Integration	The user should be able to navigate to these objects.	+	7	0,20	1	6	3	4	8	50	25	33
1.9.5.		Tool Integration	Access rights to the external objects must be recognized.	++	13	0,37	1	0	0	1	8	0	0	8
1.9.6.		Tool Integration	The links should be able to target the smallest possible structure of the external object (like the attribute of a class in the class diagram).	++	9	0,26	1	0	0	1	8	0	0	8
1.9.7.		Tool Integration	The interfaces used for tool integration should be active, i.e. synchronization or change notification should occur automatically.	+	9	0,26	1	6	1	2	8	50	8	17
1.9.8.		Tool Integration	Tool classes that could be sensibly integrated with requirements management are:											
1.9.8.1		Tool Integration	configuration management	++	29	0,83	2	4	4	2	17	33	33	17
1.9.8.2		Tool Integration	test, validation & verification	++	23	0,66	2	4	4	1	17	33	33	8
1.9.8.3		Tool Integration	problem tracking	+	25	0,71	2	2	1	1	17	17	8	8
1.9.8.4		Tool Integration	modeling and design	+	23	0,66	2	4	3	1	17	33	25	8
1.9.8.5		Tool Integration	communication, e.g. e-mail	+	31	0,89	2	2	2	1	17	17	17	8
1.9.8.6		Tool Integration	project management	-	29	0,83	2	3	1	1	17	25	8	8

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
1.10.	Import / Export			+										
1.10.1.		Import / Export	The tool should be able to import existing requirements specification documents based on a predefined, customizable exchange format.		19	0,54	2	2	4	1	17	17	33	8
1.10.2.		Import / Export	The tool should be able to export existing requirements information based on a predefined, customizable exchange format.		21	0,60	3	1	5	3	25	8	42	25
1.10.3.		Import / Export	The tool should recognize text marks, formatting, line ends, grammatical structure or keywords to interpret them as the beginning or end of requirements texts.	+	7	0,20	1	5	6	1	8	42	50	8
1.10.4.		Import / Export	The tool should support a semiautomatic, i.e. user interactive, import of requirements from existing documents.	+	9	0,26	2	5	6	2	17	42	50	17
1.11.	Document Generation	Document Generation	The tool must be able to generate official and internal documents. To achieve this, the tool needs a document generator that uses predefined document definitions to generate documents with current data from the database. Document generation differs from document-oriented views in that the generated documents are no longer connected to the database and an independent document file is created. Rationale: A requirements management tool is of no worth without powerful document generation capabilities. The days of paperless development are still far away, especially in fields where interaction with suppliers is important. Specifications are an important part of	++	9	0,26	3				25	0	0	0

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
			the contract with the supplier, which is why something document-like is always needed, whether it is printed or just a file. Document generation can be one of the main productivity-enhancing applications of requirements management tools, if developers can generate documents at the push of a button and don't have to carry out detailed formatting before and after document generation.											
1.11.1.		Document Generation	The subset of data to be included in the document must be flexibly configurable, comparable to views. Formatting and positioning must be flexibly configurable, too.	++	9	0,26	2	5	3	2	17	42	25	17
1.11.2.		Document Generation	The document generator must be able to include all information available in the tool.	++	7	0,20	2	4	4	3	17	33	33	25
1.11.3.		Document Generation	The document generator could be able to create documents in certain standard formats. Templates for these formats could be included.	-	7	0,20	3	3	4	3	25	25	33	25
1.11.4.		Document Generation	Non-textual objects must be included in the generated documents	++	7	0,20	4	6	6	1	33	50	50	8
1.11.5.		Document Generation	The tool must generate very large documents with many included external objects quickly. A 5000-page document with formatting and media objects should be generated in one night.	++	19	0,54	6	0	0	0	50	0	0	0
1.11.6.		Document Generation	It should be possible to run the document generation automatically as a background task.	+	21	0,60	0	6	0	1	0	50	0	8
1.11.7.		Document Generation	The document generator must be extensible via a programming interface (or similar) provided by the tool.	+	9	0,26	5	1	3	1	42	8	25	8

	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
v3.6				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
1.12.	Collaborative Working on the Same Development Task	Collaborative Work	It must be possible for many users to work on the same data at the same time. Of the many users working on a single requirement object, only one must be able to apply changes in a transaction save manner. If a user changes an object, it should refresh automatically in the user interfaces of the other users. Rationale: It is a typical situation that several users work on same or adjacent parts of specifications at the same time. Managing the data using a requirements management tool can provide a single source and up-to-date state of the project for all participants, but fine-grained locks are important not to suspend each others work. Especially if users want to reconcile a part of a specification at different locations, e.g. during a conference call, they need an instantaneous feedback of performed changes.	++	35	1,00	3	6	5	3	25	50	42	25
1.13.	Checking out for Offline Use	Checking out for Offline Use	It must be possible to check out data and a license to work on mobile offline computers without sacrificing consistency and access rights. Rationale: Although mobile network access is constantly improving, it is still far from perfect and performance is not yet predictable. In addition, many organizations have security restrictions that do not allow mobile access to the databases.	++	21	0,60	2	1	0	1	17	8	0	8
1.14.	Web Access	Web Access	The tool should have a web interface or another browser-based client that makes it unnecessary to install a client application for occasional users. Rationale: Web interfaces offer a reliable and easily manageable possibility to	+	21	0,60	5	6	1	1	42	50	8	8

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
			work with the requirements. They are interesting for collaboration with external partners ("extranet") and for internal users that use the tool only occasionally. Nevertheless, in reality most users are "power users" for whom the native clients provide a smoother user experience and opening the tool to the web causes some managers and administrators headaches.											
1.15.	Configuration Management	Configuration Management	The tool should provide extensive configuration management capabilities. This section provides a complete view on configuration management functionalities. It overlaps partially with section 1.1 'Information model', 1.5 'Documentation of history' and section 1.6 'Baselining'. Baselining is considered as a component of configuration management.		20	0,67					0	0	0	0
1.15.1	basic	Configuration Management	Every object must be uniquely identifiable.		10	0,25	5	4	5	5	42	33	42	42
1.15.2	basic	Configuration Management	Every object must be versioned		20	0,50	6	4	6	2	50	33	50	17
1.15.3	basic	Configuration Management	Every change of an object must be recorded with date, version number and content of change (see 1.5.1)		20	0,50	6	6	6	2	50	50	50	17
1.15.4	basic	Configuration Management	It must be possible to retrieve a specific version of an object at any point in time (see 1.5.6).		22	0,55	4	6	6	1	33	50	50	8
1.15.5	baseline	Configuration Management	The tool must support baselines. A baseline is the state of a (specified subset of the) requirements database fixed at a given point in time. Compared with object versioning, a baseline is a	++	16	0,40	6	6	6	3	50	50	50	25

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
			(partial) database consisting of numerous objects, each in a certain version. The development status saved in a baseline is the starting point for further development.								🟥	🟥	🟥	🟩
1.15.5.1	baseline	Configuration Management	It should be possible to create baselines based on time.		14	0,35	1	2	0	1	8	17	0	8
1.15.5.2	baseline	Configuration Management	The tool should support analysis of baseline information. It should be possible to perform analysis on a set of baselines.		28	0,70	1	2	3	3	8	17	25	25
1.15.6	Product Line	Configuration Management	The tool should support efficient creation of a new variant of a product. The tool must track the dependency between variants. It should be possible to visualize this dependencies at any point in time.		40	1,00	2	1	1	4	17	8	8	33
1.15.7	Product Line	Configuration Management	It should be possible to easily work on a sub-set of requirements. This set may be derived from a large set. It should get a unique name with which can be worked on. The tool should track relationships between original requirement set and subset.		32	0,80	1	1	4	4	8	8	33	33
1.15.8	Product Line	Configuration Management	Each variant should support meta attributes.		24	0,60	1	1	1	2	8	8	8	17
2.	Requirements from the Project Administrators This section describes criteria and their requirements that cover the requirements management tool needs from the project and tool administrators' point of view. They cover issues that are not core functionalities, but essential for large scale projects.													
2.1.	Central Installation and Administration of	Central Installation and Administration of Projects	All productline-wide information must be held and changed at one place. A history of associated changes must be available. Rationale: Typically a dedicated group of	++	19	0,54	6	6	6	5	50	50	50	42

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
	Projects		persons takes the responsibility for the correct mapping of the process specification to the requirements management tool implementation. They must master and document the further development from the initial project installation. Without such a responsibility uncoordinated deviations will take place, which can be very extensive to administrate											
2.2.	Users, Roles and Rights	Users, Roles and Rights	The tool must allow fine-grained administration of users, user groups, user roles, user rights and user-roles rights. A history of changes to these must be available Rationale: The bigger and the more critical the project, the more important user and rights administration becomes. Including external and possibly competing partners in the development process increases the importance of this functionality.	++	23	0,66	3	4	3	3	25	33	25	25
2.2.1.		Users, Roles and Rights	The assignment and execution of administrative task such as user and role management must be flexible delegable to the stucture of responisbilities in the organisation. I.e. some administrative task can be defined to be handle in centralized manner (e.g. user account creation) while others (e.g. assignement to a user to a specific project&role) may handled decentralized on project level Rationale: Structure of organization, size of product-lines	++	21	0,60	4	6	2	3	33	50	17	25
2.2.2.		Users, Roles and Rights	Users must be defined centrally for all projects. External user management information must be used, if it exists.	++	15	0,43	5	5	6	3	42	42	50	25

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
2.2.3.		Users, Roles and Rights	The tool must allow fine-grained access and writing rights to be flexibly granted. Rights must be grantable down to object and attribute level. A distinction must be made between rights to view, propose changes or make changes.	++	23	0,66	5	5	4	3	42	42	33	25
2.2.4.		Users, Roles and Rights	Security based on overlapping roles is preferred to security based on hierarchical security levels.	-	21	0,60	5	6	2	3	42	50	17	25
2.2.5.		Users, Roles and Rights	The security concept must not be compromised by unauthorized use of extensions like API programming or scripting. Rationale: Security among competing suppliers with access to the tool is an important issue.	++	17	0,49	0	6	0	2	0	50	0	17
2.2.6.		Users, Roles and Rights	Access rights must be grantable via roles a user is assigned to.	++	17	0,49	6	6	6	4	50	50	50	33
2.2.7.		Users, Roles and Rights	A user must be able to perform more than one role at a time.	++	19	0,54	6	6	0	5	50	50	0	42
2.3.	Size Restrictions	Size Restrictions	There must not be an upper limit for the size of the database and the number of requirements, users, groups etc. If such limits exist, they must be known exactly. The database must be able to handle very large projects. The database fields should not have a fixed size restriction. Rationale: Large projects in particular benefit from requirements management tools. To pre-estimate the limitations of a new project is inaccurate, because it could be the starting point of a single product or a wide product family.	++	9	0,26	6		6		50	0	50	0

	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
v3.6				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
2.3.1.			Unlimited size of a requirement	++	7	0,20	6	4	4	4	50	33	33	33
2.3.2.			Unlimited number of requirements	++	9	0,26	6	4	6	5	50	33	50	42
2.3.3.			Unlimited number of users	++	9	0,26	6	4	6	5	50	33	50	42
2.3.4.			Unlimited number of user groups	++	9	0,26	6	4	6	2	50	33	50	17
2.3.5.			Unlimited database size	++	9	0,26	6	4	6	5	50	33	50	42
2.4.	Workflow Management	Workflow Management	The tool could support systems development via an administrable, organized and structured process, called workflow. Information could be provided and rights granted depending on the current phase or step in the process. The workflow must not simply restrict the users, but guide them through the process. Rationale: A workflow provides steering mechanisms which ensure that all needed steps of an activity are completed. Workflows can help to implement a certain requirements engineering process and can improve consistency and standardization of the requirements. Rigidly IT-driven workflows are very unpopular among high skilled workers and in projects with tight timelines.	-	19	0,54	1	1	2	3	8	8	17	25

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
2.5.	Extensibility	Extensibility	The tool must be adaptable and extensible to the needs of the organization or project. Rationale: Every organization has different needs and usage patterns for a requirements management tool. Often nonstandard or domain-specific development tools have to be integrated with the requirements management tool.	++	19	0,54	5		5		42	0	42	0
2.5.1.		Extensibility	The tool must provide an open and well-documented object model and an API which makes all data and functions accessible to extensions. Standard programming languages should be used.	++	15	0,43	2	2	2	3	17	17	17	25
2.5.2.		Extensibility	The object model and the API must be follow the "open-closed" principle. Existing models and functions must not change, extensions should be possible across versions of the tool, even major versions. It should at least stay downwards compatible. Rationale: Long lifetime of product-line	++	21	0,60	0	1	0	4	0	8	0	33
2.5.3.		Extensibility	The user interface of the tool must be customizable.	++	7	0,20	4	4	3	2	33	33	25	17
2.5.4.		Extensibility	The user interface of the tool must be extensible with a non tool-specific programming language.	++	21	0,60	1	2	2	1	8	17	17	8
3.	Requirements from the tool administrators This last section of the requirements catalogue covers the requirements from the IT system administrators for a requirements management tool. Reliability and data security are the most important issues for them.													
3.1.	Database		Worldwide cooperation in development projects requires a round-the-clock access to the requirements management project database. A requirements management tool database failure can be very expensive, if developers can't	++										

	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
v3.6				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
			work and deadlines are missed.											
3.1.1.		Database	The tool must use an database technology, which must be scalable.	++	19	0,54	1	5	6	5	8	42	50	42
3.1.2.		Database	The database must be available 24h a day and 365 days a year. Maintenance work on the database must be done on the running system.	++	19	0,54	1	5	6	4	8	42	50	33
3.1.3.		Database	The database system use must be transaction-safe and the tool must consistently use this feature.	++	7	0,20	1	6	6	3	8	50	50	25
3.1.4.		Database	The database must have a consistency-analysis and data-integrity check. It must be able to repair such errors.	++	7	0,20	5	6	0	5	42	50	0	42
3.1.5.		Database	To improve data security and availability, the tool must use a database that is independent of the tool and can be administered independently.	++	7	0,20	1	1	6	5	8	8	50	42
3.1.6.		Database	It must be possible to backup and restore only a part of the data in the database, e.g. just a specific project or the complete database. This must be possible while the system is running.	+	13	0,37	1	1	1	5	8	8	8	42
3.1.7.		Database	It must be possible to export all project data and to import them again at a different time or places for/with different tool.	++	7	0,20	2	1	6	5	17	8	50	42
3.1.8.		Database	The data should be stored in a universal format.	+	7	0,20	1	5	6	5	8	42	50	42
3.2.	Encryption		Rationale: Requirements specifications of upcoming products and research prototypes are the main target of industrial espionage. In highly competitive high-tech markets, this is a											

v3.6	Name	Category	Description	Prio (single systems)	Points	rel. Prio SPL	Tool-Evaluation							
				++ high + medium - low			DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line	DOORS 7.1	Caliber RM v 2005 SP2	IRqA	Requi Line
			major problem. Suppliers have a strong interest in data security, too											
3.2.1.		Encryption	The information stored in the database of the tool must not be readable to system administrators or intruders.	+	7	0,20	0	1	0	1	0	8	0	8
3.2.2.		Encryption	The tool must allow all communication between client and server to be encrypted.		7	0,20	0	0	0	4	0	0	0	33
3.3.	Collaborative Work	Collaborative Work	There could be a comments or discussion function tightly linked to the requirements, but outside formal change management. Users could add comments to requirements and changes to requirements		9	0,26	1	5	6	1	8	42	50	8
3.4.	Priorities													
3.4.1.		Priorities	Explicit multi-dimensional prioritisation of requirements with respect to stakeholders		33	0,94	3	1	1	2	25	8	8	17

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from `http://aib.informatik.rwth-aachen.de/`. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

2001-01 *	Jahresbericht 2000
2001-02	Benedikt Bollig, Martin Leucker: Deciding LTL over Mazurkiewicz Traces
2001-03	Thierry Cachat: The power of one-letter rational languages
2001-04	Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus
2001-05	Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages
2001-06	Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07	Martin Grohe, Stefan Wöhrle: An Existential Locality Theorem
2001-08	Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09	Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10	Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11	Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 *	Jahresbericht 2001
2002-02	Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems
2002-03	Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages
2002-04	Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting
2002-05	Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines
2002-06	Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata
2002-07	Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities
2002-08	Markus Mohnen: An Open Framework for Data-Flow Analysis in Java
2002-09	Markus Mohnen: Interfaces with Default Implementations in Java
2002-10	Martin Leucker: Logics for Mazurkiewicz traces
2002-11	Jürgen Giesl, Hans Zantema: Liveness in Rewriting
2003-01 *	Jahresbericht 2002
2003-02	Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting

2003-03	Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations
2003-04	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs
2003-05	Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard
2003-06	Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates
2003-07	Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung
2003-08	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs
2004-01 *	Fachgruppe Informatik: Jahresbericht 2003
2004-02	Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic
2004-03	Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting
2004-04	Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming
2004-05	Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming
2004-06	Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming
2004-07	Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination
2004-08	Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information
2004-09	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity
2004-10	Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules
2005-01 *	Fachgruppe Informatik: Jahresbericht 2004
2005-02	Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"
2005-03	Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions
2005-04	Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem
2005-05	Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots
2005-06	Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information
2005-07	Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks

2005-08	Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut
2005-09	Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures
2005-10	Benedikt Bollig: Automata and Logics for Message Sequence Charts
2005-11	Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture
2005-12	Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems
2005-13	Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments
2005-14	Felix C. Freiling, Sukumar Ghosh: Code Stabilization
2005-15	Uwe Naumann: The Complexity of Derivative Computation
2005-16	Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)
2005-17	Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)
2005-18	Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises “Features”
2005-19	Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers
2005-20	Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.
2005-21	Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited
2005-22	Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins
2005-23	Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves
2005-24	Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks
2006-01 *	Fachgruppe Informatik: Jahresbericht 2005
2006-03	Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler
2006-04	Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation
2006-05	Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F
2006-06	Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color

2006-07	Thomas Colcombet, Christof Löding: Transforming structures by set interpretations
2006-08	Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs
2006-09	Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking
2006-10	Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed
2006-11	Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers
2006-12	Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning

* These reports are only available as a printed version.
Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.