Requirements for Requirements Management Tools

Matthias Hoffmann, Nikolaus Kühn,
Matthias Weber
DaimlerChrysler AG Research & Technology
{matthias.hoffmann, nikolaus.kuehn,
matthias.n.weber}@daimlerchrysler.com

Margot Bittner
TU Berlin, Fak. IV
ISTI, SWT
margot@cs.tu-berlin.de

Abstract

Tools for managing systems requirements help keeping specifications consistent, up-to-date and accessible. Since the requirements for complex systems are themselves complex information structures that must be handled in complex process scenarios, there are many strong requirements concerning a tool for managing them. This paper presents a requirements catalog for requirements management tools that is based on substantial project experience (including tool evaluation experiences) in the area of automotive as well as aircraft and defense systems. The purpose of this catalog is to help users compare and select requirements management tools as well as helping tool providers to direct future tool developments.

1. Introduction

Requirements are developed based on the given facts of the environment and the mission to be achieved. In the course of a project, they become more concrete, more detailed and also more complex. The closer the specification approaches atomic requirements, the more complex the relationship between them becomes. When the project enters the design, implementation and integration phases, the work on requirements does not end; requirements changes must be handled, the status of requirements must be updated according to the project phase and tracing to other development artifacts should be established.

The requirements for a system are already a model of the system to be developed, which is mostly too complex to be captured at once by humans. To handle the complexity, a systematic tool supported requirements management (RM) must be established. For use in this paper, we define requirements management as follows:

> "Requirements management is the structuring and administration of information from elicitation, derivation, analysis, coordination, versioning and tracking of requirements during the complete product lifecycle."

The choice of a RM tool is an important and sensitive issue.

2. Requirements Management Tools

Requirements management tools help to achieve the goals of systematic RM:

- supporting acquisition, specification, grouping and attribution of the raw captured requirements
- supporting their derivation to more detailed levels, keeping and adjusting attributes
- enabling test cases and test environments to be defined and linked
- enabling relationships between requirements, design, realization and test to be tracked and traced
- enable distributed development within the organization

As all large systems development projects have to manage large sets of requirements, there is a wide need for such tools and a number of them have become commercially available in recent years. These tools use different concepts, have different capabilities and differing degrees of maturity with respect to their applicability in system engineering projects. Since the requirements are used and changed almost throughout the development process, changing tools in the middle of a project is an expensive and time-consuming business. In the case of modular product lines, changes are even harder to handle because of the many links between projects.

To allow a careful selection of tools that looks at all aspects of a product, this paper describes a catalog of requirements for RM tools. The catalog does not cover general aspects of tools selection like price, licensing, reliability of the vendor, stability, documentation, user-friendliness etc.. Instead, it focuses on functional requirements for the tools. By publishing a set of tool requirements based on substantial project expertise, we

Proceedings of the 12th IEEE International Requirements Engineering Conference (RE'04)
1090-705X/04 $ 20.00 IEEE

seek to help stratify some basic RM tool functionality and influence future tool development.

The requirements for RM tools are very diverse. They depend on various factors, most prominently product domain, project and organization size, the amount of reuse and the process maturity. The number of permutations between these factors is large and no tool can cover all requirements. Nevertheless, we believe that most of the differences can be covered by adjusting priorities and by using different requirements management information models (RMIs), that are independent of tools. See references [1] and [2] for more on RMIs. One of our most important criteria, then, is the possibility of freely modeling the requirements in the tool. Small organizations often need only a small subset of the predefined attributes and requirements types. Large organizations are interested in using the same tool in many different projects to save on licensing, installation and user support costs. Another differentiation is that there are tools with strong method and process support, and those with less strict process support. The latter are more suitable for development teams that already have a high process maturity. There are tools with a strong document orientation and others with a strong database orientation. The former are often preferred in systems engineering because the engineers are used to document-like specifications. The latter's strength is that they support highly structured specification writing.

3. Company Context and its Influence on RM Tool Selection

The detailed requirements outlined here were developed in automotive electronics development projects at DaimlerChrysler AG, based on experiences in aerospace and defense systems. They reflect the experience gained from feedback loops between tool evaluators and tool users. Automotive electronics products are complex, highly modularized and organized in product families. New variants and products contain relatively little functionality that is completely new. Tool support for systematic reuse, versioning and modularization of requirements is therefore important. The teams working on the specifications are very large. Developers work in parallel on many variants and vehicle projects. Scalability and efficient project and rights administration are therefore musts. The RM tool users are more involved in systems engineering than in software engineering. This is typical for the automotive sector, where most of the code in current vehicles is written by suppliers according to the OEM's specifications. This means that flexible document generation and rights management are important features of RM tools. On the other hand, integration with tools for software design and programming plays a less important role.

Strong market competition leads to high cost pressures and short product lifecycles. Features and requirements change often and late in the development process. The effort required to maintain complete traceability therefore often exceeds the expected benefits. It is important to decide how much and where to trace, to ensure that developers do it correctly. Traceability must be very user friendly. User-friendliness is generally an important requirement for tools in such an environment. A significant number of tools have received negative feedback from engineers because they lack a user friendly interface.

A broader and more detailed description of the current software engineering challenges facing automotive companies can be found in [3].

4. Structure of the Catalog

The ISO/IEC Standard for Evaluation of Software Products proposes a hierarchical structure of requirements for software products [4]. To improve readability and maintainability, we use exactly three levels of hierarchy, which have specific roles. The top level serves as a grouping by stakeholders. This is not perfectly disjunctive, but serves a very practical purpose when evaluating products by reducing the amount of text stakeholders must read to get an idea of the tool. The second level contains "criteria". Criteria have "requirements" (the third level), which specify them in a way that can be evaluated exactly. In an evaluation, the compliance of a criterion is a summary of the compliances of its requirements plus notes about issues that are not covered by the requirements but appeared relevant to the evaluator. The requirements must not completely cover their criterion: The catalog focuses on requirements that potentially discriminate between tools. All requirements are prioritized with our priorities. The priorities "high", "medium" and "low" are indicated by "(++)", "(+)" and "(-)" respectively.

5. Requirements from the Developers

This section describes criteria and their requirements that cover the RM tool needs from the system developers' point of view. They cover the core functions that make up an RM tool.

5.1 Information Model (++)

The tool must allow the user to freely define a RMI.

- 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. (++)

- The RMI must be changeable during the project. (++)
- Inheritance and reuse should be available for all classes, types and attributes. (+)
- It could be possible to graphically define and configure the RMI. (-)
- The tool could support RMIs that are needed when using standard RE templates (e.g. MIL-STD-490, DoD-2167A, INCOSE, Volere or IEEE 830-1998). Project templates should be included. (-)

Rationale: Since a RM 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 RM tool can be described using a RMI. Experience shows that especially in pilot projects this mapping varies to achieve a higher benefit from a RM tool.

5.2 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.

- The tool must allow views to be defined centrally as well as in a user-specific manner. (++)
- These views must be freely configurable, including complex filters on objects, relations, and attributes. (++)
- The objects must be changeable in the current view. (+)
- The user must be able to view the requirements in a document-oriented manner, i.e. as sequential text with headings, tables, etc.. (++)
- 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. (++)
- Graphical views of the requirements should be available. (+)
- The tool should allow views to be predefined for user roles. In the course of a project, the views and the assignment to roles should be changeable. (+)
- All users must be able to customize the standard views without changing the template. (++)

Rationale: Depending on the current process step RM tool users work only on certain aspects of a certain part of the specification. It is therefore important, that a RM tool provides suitable views of the huge amount of information accessible in RM tools. This has a strong impact on the acceptance of the tool by the users.

5.3 Formatting, Multimedia and External files (++)

The tool must allow the requirements to be enriched with formatting and objects not native to the tool.

- 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. (+)
- 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. (++)
- 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. (++)

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 RM tool, which must be directly visible in the RM tool user interface.

5.4 Change Management and Comments (++)

The tool must offer the possibility to handle formal change requests. This function must be customizable to the change process of the users and must be integrated into rights management. (This could be seen as an application of workflow management).

- The change requests should have public status information like pending, accepted, rejected. (+)
- 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. (-)

Rationale: A tool-based systematic change process can enable reduce errors significantly. Especially in the late phases of a project, a restrictive change management is important. The straightforward changes in earlier project phases must be possible as well.

5.5 Documentation of the History (++)

- All changes to the requirements must be tracked and kept in the database. (++)
- The objects in the tool must be versioned. (++)
- There must be a distinction between major and minor versions. (+)
- The version number should be incremented automatically when certain changes occur. (+)
- Changes must be tracked down to the smallest unit of data structures, in most cases to attributes. (++)
- Changes and old versions must always be available. (++)
- The tool must allow a requirement to be changed back to any previous state anytime. (++)
- The tool should visualize the change history in an appropriate way, e.g. colored version comparisons. (+)
- The tool must generate freely configurable change reports. These reports should relate to views, baselines and generated documents. (++)

- The tool could analyze changes to provide information about the project status. (+)
- A comment should be saved with the change to enable it to be understood later on. (-)
- Changes could be categorized for analysis. (+)

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.

5.6 Baselining (++)

The tool must support baselines. A baseline is the state of a (specified subset of the) brequirements database fixed at a given point in time. Compared 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.

5.7 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.

- The tool must be able to enforce the creation or change of certain links upon creation or change of a requirement. (++)
- 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. (++)
- It must be possible to follow links directly in both directions. (++)
- It must be possible to give the links attributes, e.g. to differentiate different kinds of links for later filtering or analysis. (++)
- It should be possible to create rules governing what kinds of requirements must have links to what other kinds of requirements, if this is not already enforced by the information model. (+)
- Links must connect any objects in the database, not only in the same subset (module, project, etc.) (++)
- Links could be n-ary. (-)
- The tool must feature a practical, user-friendly and concise graphical representation and navigation of the traces, g.g, matrices, trees or graphs. (++)

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.

5.8 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.

- The tool should provide information about status and progress of the project. (+)
- The tool should allow to analyze inconsistencies in the link structure like finding gaps in the traces. (+)
- The tool could scan the description texts of the requirements for patterns like unsuitable/inexact language or wrongly used terminology. (-)

Rationale: The enrichment of requirements in an RM 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.

5.9 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.

- Linking must not lead to redundant data. (++)
- The connection should be transparent in both tools. (+)
- Links to external objects should be managed by the tool in the same way as internal links. (+)
- The user should be able to navigate to these objects. (+)
- Access rights to the external objects must be recognized. (++)
- 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). (++)
- The interfaces used for tool integration should be active, i.e. synchronization or change notification should occur automatically. (+)
- The tool could support tool integration platforms. (-)

Tool classes that could be sensibly integrated with requirements management are:

- configuration management (++)
- test, validation & verification (++)
- problem tracking (+)

- modeling and design (+)
- communication, e.g. e-mail (+)
- performance analysis (-)
- project management (-)

Rationale: To improve consistency between development phases and allows complete traceability over the complete product life cycle RM 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 RM tool should not result in additional major changes of the tool environment.

5.10 Import (+)

The tool should be able to import existing requirements specification documents (e.g. from MS Word), while retaining the structure of the content.

- The tool should recognize text marks, formatting, line ends, grammatical structure or keywords to interpret them as the beginning or end of requirements texts. (+)
- The tool should support a semiautomatic, i.e. user interactive, import of requirements from existing documents. (+)

Rationale: Migrating from document-based to tool-based requirements management is a tedious task that can be partially automated. It can also be a chance to improve the structure of the requirements specification, which is why semiautomatic import is superior to automatic import.

5.11 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.

- 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. (++)
- The document generator must be able to include all information available in the tool. (++)
- The document generator could be able to create documents in certain standard formats. Templates for these formats could be included. (-)
- Together with the requirements data the document generator must be able to include meta information like the change history or ownership in the generated document. (++)
- Non-textual objects must be included in the generated documents. (++)
- The generated documents must be in a standard file format. (++)
- 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. (++)
- It should be possible to run the document generation automatically as a background task. (+)
- The document generator must be extensible via a programming interface (or similar) provided by the tool. (+)
- Access rights must be enforced in the document generator. (++)

Rationale: An RM 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 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 RM 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.

5.12 Collaborative Working on the Same Development Task (++)

It must be possible for many users to work on the same data at the same time. Of the many users working on an object, only one must be able to apply changes. 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 RM 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.

5.13 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.

5.14 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 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.

6. Requirements from the Project Administrators

This section describes criteria and their requirements that cover the RM 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.

6.1 Central Installation and Administration of Projects (++)

All project-wide information must be held and changed at one place. A history of associated changes must be available.

Rationale: Typically a dedicated group of persons takes the responsibility for the correct mapping of the process specification to the RM 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.

6.2 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.

- The administrator must be able to manage user accounts and group and role assignments centrally. (++)
- Users must be defined centrally for all projects. (++)
- The tool must allow fine-grained access and writing rights to be flexibly granted. (++)
- Security based on overlapping roles is preferred to security based on hierarchical security levels. (-)
- The security concept must not be compromised by extensions like API programming or scripting. (++)
- Security among competing suppliers with access to the tool is an important issue. (++)
- Access rights must be grantable via roles a user is assigned to. (++)
- A user must be able to perform more than one role at a time. (++)
- Rights must be grantable down to object and attribute level. A distinction must be made between rights to view, propose changes or make changes. (++)

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.

6.3 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.

- Unlimited size of a requirement (++)
- Unlimited number of requirements (++)
- Unlimited number of users (++)
- Unlimited number of user groups and roles (+)
- Unlimited database size (++)

Rationale: Large projects in particular benefit from RM 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.

6.4 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 RE 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.

6.5 Extensibility (++)

The tool must be adaptable and extensible to the needs of the organization or project.

- 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. (++)
- The object model and the API must be stable across versions of the tool, even major versions. It should at least stay downwards compatible. (++)
- The user interface of the tool must be customizable and extensible with a standard script language. (++)

Rationale: Every organization has different needs and usage patterns for a RM tool. Often nonstandard or domain-specific development tools have to be integrated with the RM tool.

7. Requirements from the Tool Administrators

This last section of the requirements catalog covers the requirements from the IT system administrators for

an RM tool. Reliability and data security are the most important issues for them.

7.1 Database (++)

- The tool must use an appropriate database technology, which must be scalable and reliable.
- To improve performance, the tool should use a database that uses a modeling paradigm similar to the one used in the tool, i.e. object-oriented, hierarchical, relational, etc. (+)
- 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. (++)
- The database system use must be transaction-safe and the tool must consistently use this feature. (++)
- The database must have a consistency-analysis and data-integrity check. It must be able to repair such errors. (++)
- It should be possible to run the database backend in a distributed manner. The synchronization mechanism of the distributed database should work reliably across different tool versions (major and minor versions). (+)
- To improve data security and availability, the tool must use a database that is independent of the tool and can be administered independently. (++)
- 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. (++)
- It must be possible to export all project data and to import them again at a different time or places for/with different tool. (++)
- The data should be stored in a universal format. (+)

Worldwide cooperation in development projects requires a round-the-clock access to the RM project database. A RM tool database failure can be very expensive, if developers can't work and deadlines are missed.

7.2 Encryption (++)

The information stored in the database of the tool must not be readable to system administrators or intruders. (++)

The tool must allow all communication between client and server to be encrypted.

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 major problem. Suppliers have a strong interest in data security, too.

8. Related Work

Bjarne Tvete of Thomson-CSF Norcom reports experience from a requirements management improvement project [5]. The project included an evaluation of three requirements management tools. Details of the evaluation and the requirements used were not published. The report merely states what the problems are.

Karl Wiegers published an analysis of the RM tools on the market at that time. It gives a good overview of the tools main features and characteristics [6]. He used 16 requirements in a short text summary to describe the tools. The requirements are fairly basic and more on a feature level than on that of a detailed evaluation..

In "The Future of RM tools", Finkelstein and Emmerich describe and discuss their expectations in detail [7]. Now, some 4 years later, nearly all of the functions they expected in the short and medium term are already available in some or most of the tools on the market. These functions are covered by the requirements formulated in this paper. For the long term, they expect solutions for integrating workflow into requirements management, for the interplay between architecture and requirements, for coupling literate and formal specification and for knowledge handling. All of these are aimed at more explicit process and method support in the tools, which is not our focus.

Reinhard Ploetsch developed a selection process and a requirements catalog for assessing and selecting requirements management tools [8]. On a high level, his requirements are well structured and reasonable, but the detailed requirements he lists in his examples are often hardly measurable, though he uses the term "metrics". In addition, there is no prioritization of the requirements, which makes the selection process less focused.

A relatively complete catalog was published by the ITEA DESS project [9]. Their requirements are strongly tailored to the application field of real-time embedded systems.

The most important and complete analysis of requirements for requirements management tools we looked at is the INCOSE Tool Survey [10]. The requirements used in the study are based on an INCOSE Report by Jones et al. that analyses the factors influencing requirements toolset selection [11]. They classify process maturity levels, RM tool users and RM tool architectures. Unfortunately, the authors do not map the requirements formulated later on to these classifications, so that the original intention of creating a general-purpose set of requirements is not achieved. Also, the requirements used in the study implicitly require a specific RMI and a specific terminology. There are, for example, requirements that call for specific attributes or link types. The second weakness of the INCOSE Tool Survey is that the tools were evaluated by the tool vendors themselves. This is not necessarily a flaw, but the requirements used were formulated for evaluation by an independent. Questions

are mostly of the open-answer type and not very detailed. Terminology was not defined, so the authors obtained some strange results when tool vendors interpreted a feature name in a legitimate but not intended way, making their tools look fully compliant but if one takes a closer look this is not necessarily the case.

Maiden and Ncube published an extensive report on deriving COTS software selection requirements, which was originated in the context of selecting an RM tool [12]. Although they did not publish their requirements for an RM tool, their methodology and the "lessons learned" were helpful in creating this catalog. A more general introduction to issues of software package requirements and procurement can be found in the position paper by Finkelstein et al. [13].

9. Conclusion

In this paper, we have presented a matured and complete catalog of requirements for requirements management tools. The catalog is not only for tool developers but also for those who use such tools. The catalog itself is not fixed, rather it is subject to periodical review and modification as computer technology trends or automotive development process trends lead to new or strengthened requirements as well as to weakened or outdated ones. We have described the influence of the automotive background to this catalog. However, we think that it is quite useful as a basis for RM tool requirements for other domains as well. For example, some of the priorities may reflect automotive experiences and needs, could well be different in other domains.

An evaluation template with some more detail is publicly available [14].

Acknowledgments

Our thanks go to our colleagues from DaimlerChrysler and TU Berlin whose insights and experiences have provided us with the essential input for writing this paper.

References

[1] M. Weber, J. Weisbrod, "Requirements Engineering in Automotive Development – Experiences and Challenges", *IEEE Software 20(1)*, 2003, pp. 16-24.

[2] G. John et al., „Using a Common Information Model as a Methodological Basis for a Tool-Supported Requirements Management Process", *Proceedings of the 9th INCOSE Symposium, A. Fairbairn et al., eds.*, Ascent Logic Corp., Brighton, UK, 1999, pp. 59–67.

[3] K. Grimm, „Software Technology in an Automotive Company – Major Challenges", *Proceedings of the 25th International Conference on Software Engineering (ICSE03)*, IEEE, 2003.

[4] The International Standardization Organization, *ISO/IEC 9126 : Information Technology – Software Product Evaluation – Quality Characteristics and guidelines for their use*, 1991.

[5] B. Tvete, "Introducing Efficient Requirements Management", *Proceedings of the 10th International Workshop on Database and Expert Systems Applications*, 1999.

[6] K. Wiegers, "Automating Requirements Management", *Software Development*, 7(7), July 1999.

[7] A. Finkelstein, W. Emmerich, "The Future of RM tools", *G. Quirchmayr and R.R. Wagner and M. Wimmer (ed): Information Systems in Public Administration and Law*, Austrian Computer Society, 2000.

[8] R. Plösch, "Efficient Assessment of RM tools", *Proceedings of the 4th Workshop Iberoamericano de Engenharia de Requisitos e Ambientes de Software (IDEAS 2001)*, Instituto Tecnológico de Costa Rica, Centro de Información Tecnológica, Serie manuales didácticos, no. 11, 2001.

[9] L. Bokhorst, "Task 1.8 Requirements Management Method D1.8.3 – RM Tools proposal", *Information Technology for European Advancement – Software Development Process for Real-Time Embedded Software Systems (ITEA DESS) Documentation*, 2001.

[10] The International Council on Systems Engineering, "Tools Survey: RM tools", Available from `http://www.incose.org/tools/tooltax.html` , 12/2002.

[11] D. Jones, D. York, J. Nallon, J. Simpson,. "Factors Influencing Requirement Management Toolset Selection", *Proceedings of the Fifth Annual Symposium of the INCOSE*, Vol. II, July 1995.

[12] N. Maiden, C. Ncube, "Acquiring COTS Software Selection Requirements", *IEEE Software 15(2)*, 1998, pp.46-56.

[13] A. Finkelstein, G. Spanoudakis, M. Ryan, "Software Package Requirements & Procurement", *Proceedings of the 8th IEEE IWSSD*, 1996, pp.141-145.

[14] `http://swt.cs.tu-berlin.de/lehre/eks/`