Requirements Engineering in Automotive Development – Experiences and Challenges

Matthias Weber
DaimlerChrysler Research RIC/SM
Matthias.N.Weber@DaimlerChrysler.com

Joachim Weisbrod
DaimlerChrysler Research RIC/SP
Joachim.Weisbrod@DaimlerChrysler.com

Abstract

This paper summarizes problems, solutions and challenges of requirements engineering gained from the development of software-based automotive systems at DaimlerChrysler. The technical domains covered include telematics and entertainment, body electronics and chassis control for both passenger cars and commercial vehicles. Experiences include domain-specific issues, which are related to the different content of requirements in particular application areas and its specific notations, techniques, or process steps, as well as domain-independent issues which are largely independent from requirements content and related to the administration and management of requirements. We argue that the issues presented here, while not necessarily being in the current focus of the requirements engineering research area, are of crucial importance for improving requirements engineering in the development of software for automotive systems, and very likely, to many other kinds of systems as well.

1. Introduction

In the automotive industry, especially in the high-end market, the functionality of electronic components is becoming more and more complex at a very fast rate – about one third of development costs is spent for electric/electronic development nowadays, and this amount is still increasing. At the same time, many components are developed, integrated and tested over a sequence of prototyping phases. Furthermore, components are developed in several slightly different variants and at different schedules. As a consequence, the specification activities have reached a level of complexity, that exceeds the limits of what can be reasonably supported by conventional text processing systems used by some local heroes, because these systems do not sufficiently support management and tracing functionality.

Due to these reasons, requirements engineering processes, methods, and tools are being piloted at DaimlerChrysler in various development projects over the last years. This paper summarizes some of the problems, solutions, and challenges gained from these projects.

In the following, we first briefly characterize the application area, and then summarize a number of important experiences, both technical and process-oriented ones, gained from intensive requirements engineering support in our business units. The discussion of these experiences will be done at the end of this paper.

2. Application Areas

In the last years at DaimlerChrysler, in the course of many development projects supported by DaimlerChrysler research, significant efforts have been invested into piloting new requirements engineering processes, methods, and tools.

The projects have covered the following areas:

- *Entertainment and telematics:* Due to the incredible pace of technology, to the high visibility, and to the complexity of the underlying systems, in-car telematics has become a major challenge for automotive development. Several cases have been reported in public where the telematics system turned out to be on the critical path of automotive development. At Mercedes-Benz Technology Center, about 60 engineers are dealing with telematics system design and specification of a single car series. Obviously, in this domain it is at the same time very hard and very important to define and establish appropriate requirements engineering processes, methods, and best practices with adequate integrated tool support.
- *Car interior and passenger comfort:* In this area, the overall system is compiled by some 70 features with an averaged 50 pages of specification each. There is extensive communication between those features and components. Furthermore, even an ”ordinary” feature like the windshield wiper/washer is an astonishingly complex system. From the development point of view the main challenges are to specify features in parallel as independent as possible and to separate software-specific from hardware-specific issues.
- *Driver assistance and safety critical systems:* Both in passenger cars and in commercial vehicles more and more systems are being developed that possibly intervene with the driver’s actions (ESP, ABS, drive-by-wire, e.g.). Potentially, when malfunctioning those systems are able to cause severe accidents. In this

Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE’02)
1090-705X/02 $17.00 © 2002 IEEE

domain we have strong *requirements* on the development process: System specification and system development have to be completely comprehensible, reproducible, and continuous. Top level requirements and appropriate extensive acceptance tests for technical approval have to be defined and documented at the same time. Especially the ”natural” gap between manufacturer and supplier has to be taken into account: (1) The system to be developed is handed over to the supplier by means of a system specification. (2) The system as developed by the supplier is returned and the manufacturer has to make sure the system satisfies the specification from step 1. Furthermore, development techniques like state models and code generation are becoming more and more important in this domain.

The requirements engineering work in these areas has led to the following important results:

- In DC research, we established a *requirements engineering team* supporting DaimlerChrysler business units with a common understanding and vision.
- We defined a generic requirements engineering process used both to coordinate work within the requirements engineering team and to support discussions with the business units.
- In several DaimlerChrysler business units we adopted the generic process and tailored it to the specific needs of our customers.
- We provided and applied several DaimlerChrysler specific requirements management information models (i.e. metadata model, i.e. database scheme for the corresponding requirement management tool, see below) with comprehensive documentation.
- We specified requirements management tool guidelines optimized for and tailored to DaimlerChrysler processes.
- We evaluated experiences and worked on concepts about linking requirements management tools to other tools used at DaimlerChrysler such as Matlab/Simulink and PVCS.
- We supported many pilot projects introducing the new processes and tools. In the course of this activity, we provided many extensions and add-ons to the requirements management tool functionalities itself requested by our engineers.
- We developed research prototypes and realistic case studies of best practices in requirements engineering.

3. Experiences

In this section we present and discuss our corresponding experiences as gained from our hands-on support for the business units. We differentiate between general issues, issues concerning the requirements engineering process, and technical issues. For each category, we list and discuss a couple of concrete statements derived from our work.

3.1 General Issues

- **Textual requirements are only part of the game – automotive development is too complex to be managed just by text.**

This does not mean, of course, that textual requirements are not important. However, tackling and controlling just the textual requirements is clearly insufficient.

Even more so, when it comes to complex systems, ”controlling just the textual requirements” becomes nearly impossible without additional attributes like working state (*initial*, *defined, agreed upon, released*, e.g.) or due date (*A-sample, B-sample*, ..., e.g.) reflecting characteristics and phases of specification or development process, respectively.

The importance of these and many other attributes is now well acknowledged and useful collections exists [1]-[6]. The tricky point is to tailor the right set of attributes, such that the efforts of defining and maintaining them leverages with the benefits of better process control and specification reuse. Maintenance of attributes refers both to the instance level (i.e. changing values etc.) but also on the class level (project wide manipulation of attribute definitions).

The history of requirements has to be handled too. This does not only include recording the ”local” history of manipulation of a single requirement. Engineers typically need to see in a well-structured manner the ”global” history of manipulating a defined collection of requirements relative to some previous baseline.

In addition to attributes and history, in specifications we also have to deal with illustrative pictures of any given format, tables, recommendations of various kinds, explanations about e.g. the design rationale or design decisions taken, test information, parameters and other interface information of various kinds, and accompanying background information (excerpts from standards etc.). This kind of additional information often makes up far more than 50% of the overall specification. It should be noted that most of these objects are not system requirements themselves, in the sense that they specify a required system property. However, they all need to have attributes and history.

All these different objects will have multiple dependencies (where here a ”dependency” between A and B here means that a change of A may also lead to a change of B). Some of these dependencies will remain implicit; others will be made explicit in various ways so as to be systematically traceable (by appropriate naming or

by explicit linking). We give some experiences about explicit traceability below.

The complexity of objects of different types, their attributes, and links between them quickly becomes confusing for the engineer. Therefore there is an urgent need for a metadata model that is defined or adapted beforehand and which formally defines all this information. We found such metadata models to be of central importance in requirements management projects: such a model represents the basis to discuss with engineers both what their needs are and how they should systematically manage their specifications. At DaimlerChrysler we have introduced the acronym RMI (Requirements Management Information model) for these models. The development of a RMI together with engineers has become the central means for a successful introduction of requirements management into a project [7]. For obvious reasons of consistency and reusability we have introduced a company-wide modular RMI, so that new projects can start requirements management activities by adapting and tailoring this model. On the tool support side, tools should be based on a database system that realizes the RMI. In addition, they should have a graphical editor for the RMI.

- **Engineers live in a document-oriented world -- the look-and-feel of which must be retained by requirements management tools**

System developers typically focus on document-centered requirements engineering. This is not surprising since documents are the traditional interface both internally across departments or projects and externally to suppliers. In fact, the entire development process (and especially a formal contract) is based on documents and their exchange. The introduction of requirements management tools, which introduce a database approach, can only be successful if the document look-and-feel is retained as much as possible.

There is a strong need for tool support to retain the basic functionality of a standard text processing user interface, while adding all the management and tracing functionality. This feature should not be underestimated. We found that tools which deviate from basic aspects of standard text processing are quickly discarded in real-life projects.

This does not mean that the user should not know anything about the underlying database, on the contrary, when working e.g. on filtering or tracing of requirements he needs to have a profound knowledge of the underlying logical database structures

All the additional non-textual document information poses tricky problems for tool support: for example, recording the history of pictures is tricky, since the change history of pictures in any given format can in general only be recorded on the picture level. In any case, even without the history issue, pictures cause serious problems related to size, so that tools must support compressed formats (which they typically don't). Similarly, it must be possible to interpret tables in a very flexible manner, as sometimes an entire row or column correspond to a unique object with a unique id, attribute values, and history, and at other times a single table cell corresponds to a unique object.

There will always be documents containing information outside the requirements management tool database. Hence there is a strong need to integrate requirements management tools with document management systems. This can only be achieved at the granularity of entire RM-databases.

The input of data should be made as comfortable as possible, e.g. there should be graphical support for editing specialized input forms like in standard database systems.

Requirements management tools will never be accepted unless they provide flexible, easy-to-configure and efficient document generation facilities, satisfying given standards and combining and filtering material of all kinds. Engineers must be able to quickly produce both their main specification documents as well as various reports requested by management.

- **There is no clear boundary between manufacturer requirements specifications (*Lastenheft*) and supplier system specification (*Pflichtenheft*).**

In German, there is a very evident distinction between *Lastenheft* and *Pflichtenheft*.

At first glance, the English counterpart is the pair *user requirement specification* and *system requirement specification*. The user requirements document is supposed to tell what the problem is, whilst the system requirements document should describe the solution to tackle the problem. Generally speaking, the user specification document tells *what* the system to be developed should be able to do. The system specification document defines *how* the developer plans to realize the user requirements.

At second glance, the correlation between *Lastenheft* / *Pflichtenheft* on one hand and user requirements document / system requirements document is not that easy. The *Lastenheft* is a specification document provided by the customer and tells the potential supplier *what* the orderer wants to get. The *Pflichtenheft* as the supplier's reaction to the customers wish list documents *how* the supplier plans to realize what the customer wants to get. The problem is, that it makes perfect sense not to restrict the *Lastenheft* to the problem space: Especially in our domain there are striking reasons why describing the problem space only is not enough:

- Systems are generally divided into sub systems and components provided by different suppliers, both within one car series as well as in consecutive ones. DaimlerChrysler has to make sure that parts provided by several suppliers are going to work together both within one car series but also possibly in similar

configurations of future models. This means, that in order to specify supplier specific ”problems” we also need to specify parts of an overall solution.

- When trying to separate software from hardware components or even software suppliers from hardware suppliers things get even harder: Now even for a single component we have to specify two interactive ”problems”, namely a hardware and a software one. Furthermore, the software system should be as independent from specific hardware components as possible.
- In many situations, DaimlerChrysler’s first-to-market strategy together with the demand to keep competitive expertise in house requires for specification documents that define certain features as detailed as a concrete algorithm can get, sometimes provided as black box. Often, DaimlerChrysler is supposed to set the future standard by specifying solutions instead of problems.

Some business relations between and DaimlerChrysler and corresponding suppliers are very old and have evolved over time. This sometimes leads to a very informal mode of cooperation, in the worst case to complete dependency from the supplier.

As a consequent, we sometimes find important information missing in the *Lastenheft*. Or there are requirements listed in the *Lastenheft* that should not be there, as they restrict solution space in a costly, but overdone way. Nevertheless, there are situations where even the requirements engineering expert together with the domain engineer has problems to decide whether a given requirement should be part of the *Lastenheft* or not.

Altogether this leads to a situation where it is not easy to make a clear distinction between what should be part of the *Lastenheft* the DaimlerChrysler engineer is supposed to deliver and what should not be part of this document – the boundary between *Lastenheft* and *Pflichtenheft* is far from obvious.

- **Redundancy is where good engineers are really suffering – and the resulting systems are likely to be suffering, as well.**

In very large projects such as the development of an automotive electrical system, several hundred large specifications and related documents (up to several hundred pages each) are elaborated in parallel under a tough time schedule. It is not surprising that normally these documents contain a significant amount of redundant information.

The disadvantages are obvious: redundancy leads to inconsistencies and double work. Or – in the worst case – to specification documents derived from invalid premises, leading to components that do not fulfill their requirements or do not fit into the overall system.

The dependencies between documents are very complex and remain partially intransparent; this is simply both due to the amount of material and personnel involved and due to the lack of time.

It is very difficult to deal with this problem in the presence of document structures having grown and having gained acceptance over many years. Moreover, the document structure typically mirrors organizational structures and even the entire automotive business structure. Changes in the document structure to remove redundancies are therefore very difficult to achieve. The tradeoff between efforts and results is often negative here.

From our experience, it is already a major step to obtain and communicate an overview of the redundancies within the projects. Engineers thus become aware about those parts of their document being used or elaborated in parallel some place else and they can decide to synchronize if necessary.

The ultimate very-long-term solution, of course, is an efficient database of specification-related objects, distributed over manufacturers and suppliers, from which all documents are generated.

- **Content is not where good engineers are suffering – presentation of content and local heroes, however, may lead to problems.**

In the automotive domain – but probably in most embedded system domains – systems are typically not built on the green field but in increments: the new car series inherits most functionality from already existing ones with more or less adaptations, extensions, or innovations. The new windshield wiper basically is just another (and even better) windshield wiper.

As a consequence, at this level, there is no formal elicitation and negotiation phase, only relatively few related activities are necessary concerning the small increment being developed in a new version of a component. We found that, from many years of experience, good engineers are premier league experts in their specialized domain and have obtained deep insight on the subtleties and pitfalls of the requirements content. For them, content and domain knowledge is not the main problem.

However, there can be of course problems related to the presentation and structuring of content. Often, engineers do not describe specifications in a well-structured and easily accessible manner, leading to high communication efforts inside a manufacturer or between a manufacturer and a supplier. We argue here that a well-defined RMI implemented by an requirements management tool can help to improve the structuring and presentation of content, for example by pushing engineers to break up (lengthy) specification paragraphs into atomic requirements with associatzed tests.

There is another point of improvement: the reuse of old specifications today takes place in an ad-hoc and implicit fashion, which is neither very efficient nor very desirable with respect to specification quality. The main problem

with this ad hoc approach, however, is that a good specification is completely dependent on the one engineer being the very expert in the given domain for years. It can be very useful to regard *systematic recycling of existing specifications* as an explicit step of the specification process [8], [9], [10]. From a requirements engineering process point of view we will come back to this point in discussing the next statement below.

A closely related technical problem when introducing a requirements management database is to migrate existing documents so as to make them more reusable for upcoming projects. While requirements management tools offer some automatic capturing functionality, we found that a lot of manual work is still necessary to migrate a document content into a requirements management database so that it becomes more reusable. Furthermore, different projects must use compatible requirements management information models in order to achieve cross-project reusability. This again stresses the need for a cross-project standardized RMI when developing product families.

3.2 Requirements engineering process

- **Detailed specification is typically only documented at the leaves of the system decomposition tree.**

A complex system like an up-to-date telematics system cannot be handled by just talking about and dealing with the user and system requirements that make up the low level, detailed system specification of the components involved. On the contrary, it is inevitable to develop, to synchronize, to balance, to determine, to review, and to release a complex system top down in several layers of granularity. What makes this situation in automotive development even more challenging is the fact that generally several suppliers supply the components to make up such a system.

When it comes to requirements engineering and requirements management, this observation still holds. Unfortunately, today systematic processing and documentation of higher level requirements and design decisions is insufficient. But when high level requirements and rationales are not systematically documented, detailed requirements are built on sand: some of them are challenged regularly because they are hard to justify (even though they may be of central importance); others have become completely superfluous, but not questioned (e.g., a costly requirement derived two years ago from a legal regulation that has been suspended in the meantime).

Last but not least, well-structured requirements and design decisions on several layers of abstraction with documented and typed relations are crucial for understanding a detailed specification document. From this point of view they are the first and most important precondition for becoming less dependent from local heroes and for introducing systematic recycling of specifications. There are lots of description techniques and notations around: goal trees, feature lists, use cases and scenarios, message sequence charts, state models, and the like. The challenge is to compose, to define, to explain, and to integrate such a domain specific or even organization specific model into existing specification processes and tool suites. In this direction, our own research is mainly focused on use cases [11], [12], [13].

- **Engineers fail to manage non-functional requirements, acceptance criteria, and test cases.**

In our projects, we never found an engineer, who would not wholeheartedly agree to the following two statements:

- The specification of requirements should go hand in hand with the definition of corresponding acceptance criteria and test cases.
- Non-functional requirements like maintainability or user friendliness are far more important than a lot of the functional requirements to be fulfilled by the system to be developed.

Unfortunately, even in projects which have the appropriate resources to do so, actual specification documents do not really meet the corresponding claims. This is due to the fact that it is very hard to come up with reasonable acceptance criteria and appropriate test cases for functional requirements. When it comes to non-functional requirements, things get even harder.

From our point of view, writing good acceptance criteria is just a matter of practice and personal experience. Therefore we feel the best way to improve the state of practice with respect to acceptance criteria is to provide concrete examples for given functional requirements and to have engineers exploit and re-use these examples. Which is just another argument to foster systematic recycling of requirements.

Concerning non-functional requirements, the problem is that postulating qualities like maintainability is very hard to substantiate. In the general case, corresponding acceptance criteria and test cases are either non existent or too academic (”90% of all maintenance jobs should take less than 5 minutes.”). From our perspective, we think that most non-functional requirements should be stepwise refined until they are implemented by a set of functional ones.

- **Changes and discussions about possible changes are typical daily work – and there is no chance to change that.**

All kinds of changes occur during projects, typically including very late and totally unexpected ones. One may criticize the amount of change, but we think that this is naive. As long as they are justified and technically feasible, these kind of changes will always occur during project development. As a consequent, this means we

have to make sure that late and unexpected changes are reasonably dealt with.

Possible changes span a wide range: change of system requirements, application requirements, or component requirements during development, change of the RMI, change of schedules, change of responsibilities, change of suppliers, or even change of processes and tool environment.

As a major underlying reason for this frequency of change, we see the growing complexity, parallelisation, and time restrictions of development projects. This forces developers to introduce assumptions about the system in the initial phases of development. These assumptions are then often challenged or reworked when the system development reaches more maturity.

There are two levels of granularity when it comes to change management in requirements engineering:

- *requirements management level oriented change management*, which includes small changes in requirements, including refinement and clarifications, and which is performed by the engineers on the technical level, and
- *"official" change management* between manufacturer and supplier, which includes large changes and may affects costs and contracts, and which takes place on a more political level.

This gap should be made explicit and defined, and for each level appropriate measures and methods should be provided.

A good way to reduce the number of changes to the RMI to the ones that essentially needed is to build-up, to introduce, and to maintain cross-project standard RMIs and do a very careful tailoring step at the beginning of projects. At DaimlerChrysler we gained good experiences with this approach.

Much more than changes that actually occur, proposals for changes are discussed during the project (often very late during development). As technical expert the engineer needs to be able to quickly deduce opinions on such change proposals. If requirements engineering is based on an appropriate RMI, the relations needed to evaluate the potential effects of actually realizing the change proposal will be documented. For instance, the engineer can efficiently deduce that altering the system as proposed by change proposal *CP-0815* affects 17 test cases, namely *TC-n, TC-m*, ..., *TC-q*. This kind of impact analysis supporting the decision for change proposals is a major point where qualified requirements management techniques leverage.

Unfortunately, the lack of appropriate specifications at the upper levels of system decomposition (c.f. the corresponding previous paragraph), hinders the analysis and propagation of local changes introduced late in the development process. This in turn reduces reusability of specifications in subsequent projects.

- **User-adaptable views are a powerful instrument for guiding and managing the specification process.**

In a complex domain the requirements engineering process is a complex process with lots of different activities and continuous need for modifications. In general, there exists a variety of roles within projects with complex dependencies and these roles need to read, modify, review, accept or release various aspects of the specification at various points in time. By supplying both user specific and situation specific views on the database, the requirements management tool can be a fundamental means to coordinate and support the requirements engineering process. In some projects, we found that the support for views has let the requirements management tool become some kind of workflow management support.

Automatically computed views are a major point of leverage for the engineers' investment into adopting a new tool. However, in order to make this work, the view concept must support a number of important requirements.

To be useful, the support for defining individual views must be powerful: it should not only support attribute selection and filtering, but also the automatic generation and systematic display of derived information, possibly following multiple linking steps. Furthermore, views should support user access rights.

There should be support for view management. Often there are complex role or phase dependent views which should be programmed once and then reused across projects. Standardized views will thus become part of the RMI.

It must be very easy for the engineer to adapt views or create new simple views, e.g. to argue for some point during discussion with his manager or during review meetings with his manufacturer or supplier.

Many engineers, especially in Germany, are not used to develop their documents "in public". And they do not really like this idea, as well. When it comes to user acceptance, this observation should be taken into account by considering whether engineers should be allowed to work in a private part of the database at least in early stages of their task. Consequently, requirements management tools should be able to support such a requirement if this "engineer's sandbox" is a serious option.

- **Well-organized specification reviews are essential for a successful manufacturer-supplier relationship.**

Informal and formal reviews occur frequently in the context of steadily evolving specifications. They represent an important instrument for assuring both specification quality and subsystem compatibility. Sometimes they are not only an important instrument, but the only measure to guarantee the compatibility and integrity of the overall system composed into several components.

In the automotive domain, we are talking about a complex organization, where an engineer is responsible for several versions of his or her component at a time. Normally, these versions are in different development phases, as car series are launched one after the other (which does not mean there are no tight time constraints!). To coordinate and organize effective and efficient reviews in such a setting is both very ambitious and very important.

We found that effective requirements management practices and appropriate tool support are the key to effective and efficient specification reviews. The single source approach of modern requirements management makes sure that each and every inspector reviews the actual version of the document under test. Appropriate, role specific views and attributes for requirements under review as well as inspectors' comments make sure that at any time everyone knows what to do and how to do it. In review meetings discussions, findings, and decisions are documented online using the requirements management tool. Project management is provided with special views to control the review process, especially the rework that has to be done after the review meeting. Last but not least, due to the history features of the requirements management tool, each and every decision can be traced back and questioned later on.

The basic features of requirements management support are sufficient to provide this kind of review support: single source, user defined attributes, powerful viewing and filtering mechanisms, change history and baseline facilities, as well as the generation of structured reports derived from specific views.

We found that this kind of appropriate requirements management support substantially improves and facilitates the review process. This proved to be a major argument when it came to convincing both engineers and project management of the usefulness of systematic requirements management practices and of requirements management tools.

3.3 Technical Issues

- **Today's automotive development means distributed development – today's requirements engineering needs to be distributed, as well.**

There are two types of distributed development: DaimlerChrysler internal distribution between different departments, business units, business divisions, or corporate brands as well as distribution between DaimlerChrysler as manufacturer and orderer and suppliers. As mentioned before, we cannot ask for a one way, top down, step by step specification process without backtracking, but we have to deal with a very dynamic and unstable environment, where sometimes even early design decisions have to be altered, which means that lots of derived requirements and artifacts have to be adapted, as well. Potentially, this need for adaptation effects several roles, either company internal or external ones.

First of all, we have to provide appropriate requirement change management practices based on sound methods and suitable tools. In terms of requirements management, this means, the corresponding tool needs to support distributed, asynchronous, and role specific access for the stakeholders involved (which is a basic feature of modern database systems). The corresponding database scheme (RMI), on the other hand, should implement the change process and corresponding methods by supplying the relevant views and attributes.

In general, the specifications we are talking about represent top secret information. We do not want others to know what features our future model has or even how they are supposed to work.

Internally, security is not a major problem, as a corporate network is safeguarded against public by firewalls and such. As soon as external suppliers come into play, there is a problem. Either we need to hire special high security telecommunication services to allow for the supplier to work online in our internal database, or we need to define and establish reasonable ways of exporting parts of the database, giving them to the supplier, and re-importing the adapted apart later on. Today, the second solution is problematic, since it is not clear how to deal with links from or to objects that are being exported, replication is not trivial.

Another technical solution would be a multi server concept with server synchronization, which means, there is a server both at the manufacturer's site and at the supplier's site. Both parties work locally and independently on their servers, and regularly both servers synchronize each other by means of a connection established for that purpose.

Regardless of the technical way that is used to exchange specification related information with external suppliers, often there is a political problem: Namely, to convince management to allow for the regular exchange of this information by means of modern information technology. Which is, by the way, not an unreasonable attitude.

Finally, there is another aspect of distributed work: engineers should be able to do their job riding the train, flying on a plane, wherever and whenever they want. This means, that they have to be able to take (parts of) the requirements database with them and re-import the adapted stuff later on, as discussed above. Furthermore, this means that the requirements management tool should allow for offline work, either by some hardware (dongle) or by mechanisms to allow for checking our licenses from a pool of floating ones.

- **Traceability is a great feature – the real challenge, though, is to decide which traces to maintain.**

Introducing explicit traceability promises big savings [14], [15]. But in order to pay back, introducing explicit traceability requires a careful tradeoff analysis.

As already stated below, there is a variety of links between objects. Linking between requirements and related objects is essential, however, maintaining links requires efforts and this should be balanced by the benefits gained from explicit traceability. To this end, our advice is to basically leave these discussions and decisions to engineers, because they know what they want and what they are able to accept. Generally speaking it turns out, that in the first place engineers are keen on establishing and maintaining all kinds of traces, but later on their discipline to really do so falls far short. With this observation in mind, our general message is to keep the number of explicit traces small.

Nevertheless, the relation between requirements and test cases is so essential, that we always try to establish this link type.

Technically speaking, there are two way to express the relation between two objects: using an explicit link or using an implicit one. As opposed to an explicit link represented by an explicit link object, an implicit link can be a textual reference integrated into the text (e.g. "see SR-123") or a (document) convention (e.g. "chapter X in Document XX is always linked to chapter YY in document Y").

All decisions which objects are to be traced and how to do so (implicitely vs. explicitely) are part of the RMI and should be precisely defined in the projects inception phase.

Design guidelines are an important means to master traceability by defining rules (and corresponding mechanisms) which links to establish, how to do so, and when to update them. Design rules might, for instance, specify, that only links from features to use cases, links from use cases to scenarios, and links from scenarios to architectural artifacts are allowed.

From a requirements management tool point of view, link support is still inadequate. Mainstream tools do not support links as true first-class citizens, e.g. concerning link histories, link attributes or distributed work on links.

- **Couplings between document-based and model-based tools are not seriously used.**

While the majority of specification activities is still document-based, a growing number of specifications require complex models, such as (executable) analysis models, system and software design and integration models, and HMI models. This leads to the situation of developing a specification using two or more tools, a tool for textual specifications and one or more tools for model-oriented specification, simulation and design. Tools vendors have been asked by customers to develop script-based couplings between RM-tools and model-based tools to ease the synchronization of information so as to keep the specification consistent and avoid double work.

In our experience we have found that these tool couplings (usually originating from a specific project and designed for and paid by a specific customer) generally are not sufficiently mature to be seriously used in development projects. Several projects started using these links but later dropped them, even when it was clear that this would lead to considerable manual efforts or significant process problems.

Some important shortcomings we have experienced include the following ones.

- *Speed:* The number of linked objects may easily grow to several thousand, in which case the speed of the coupling (e.g. when calculating changes) becomes unacceptable.
- *Generating integrated document generation*. This feature is not supported by existing tool couplings. Note, that this may mean generating short status reports as well as several hundred pages of documentation satisfying a given (standardized) structure. There are a few tools dedicated to integrated document generation, but only in the context of a specific ”tools suite” of some vendor, and hence largely useless outside of it. Generic XML-based approaches offer a good solution in principle, however, they are far from being widely supported.
- *User-interface.* Usually, the tool couplings lead to redundant editors for managing cross tool information, typically one for each tool involved.
- *Automation.* The degree of automation is very low, the tools offer passive administrative support only, the user has to initiate synchronization himself. There is no active support for indicating problems.

A pragmatic approach to deal with these problems was to extend a model-based specification tool to support the management of textual specifications (including formatting, attributes, views, history, baselines, etc.).

A methodological approach to alleviate these problems is to establish design guidelines that keep the text-model interface lean, allowing links between certain text and model elements and only during certain steps of the process.

We think that the long-term solution to these problems will come from two directions: technically, tool couplings will be handled more systematically, efficiently and uniformly by XML-based approaches, methodologically, further standardization on requirements and system engineering languages will help. In particular, we think that the UML unification effort should be extended from software engineering to system and control engineering languages to reduce the current zoo of overlapping and tool-vendor proprietary notations. In a further step, one could think about including requirements management

concepts in UML so as to have all specification and system architecture related information in a single model with uniform modeling concepts.

- **Requirements management tools are the number one instrument to leverage requirements engineering practices – which means, they still need to be improved.**

Many of our issues have been related to tools. Here we list some additional ones. One may criticize these experiences as tool-focussed. We would argue against that. Many engineers are experts of their field and of the needs for their work. They crave for tool support that actually helps them with their daily problems, they quickly drop tools which lack the functionality, flexibility, or user-friendliness they need.

What makes this observation so important is the role of tools to leverage improved practices and processes. If engineers ask for improved tool support, this is a great opportunity to improve processes and practices, as well and on the fly. The obvious and serious drawback, however, is that if a tool is rejected due to deficient features or performance, the corresponding process improvements are rejected, as well. So requirements management tool support represent both a chance and a risk to requirements engineering process improvement.

With this in mind, we postulate the following additional requirements to adequate requirements management tool support:

- In order to leverage process improvement and adoption of appropriate requirements engineering practices, requirements management tools should provide some basic means for workflow support like powerful filter and view capabilities and sophisticated view management.
- Appropriate requirements management tools must be easily adaptable so as to quickly include new functionality. Otherwise projects will very quickly drop the tool, because tool vendors are generally too slow to react to urgent project needs.
- Tools should be controllable offline via appropriate APIs or at least command line interfaces, so as to be able to quickly construct and adapt tool links

In addition to those vital requests, it is often the small details that cause rejection in everyday work. If, for instance, the generation of text documents from the requirements management database takes some 8 hours, the tool is not very likely to get an award by the engineer who needs to distribute documents regularly.

4. Discussion

As the above experiences are based on real-world development projects in various domains, it is not surprising that a huge and seemingly disparate variety of issues are raised. Many issues are concerned with the appropriate management of requirements, regardless of their content. Also there are many issues of appropriate tool-support for requirements management. On the other hand, one might be surprised to *not* (or at least not very prominently) find problems related to the specification of the actual requirement content, i.e. concerning requirements elicitation or expressiveness of notations, which are focus of a considerable amount of work in the requirements engineering community.

This may be due to the fact that due to the product family development approach used in the automotive domain, it is the exception rather than the rule to develop new functionality from scratch, but to extend existing functionality in an evolutionary manner. Hence the (academic) distinction between problem and solution space becomes somewhat blurred.

As a consequence, excellent human expertise has been built up on the actual components and their functionality. These experts usually do not have the most pressing problem with specifying what the system should do, but with managing the growing complexity of the specification activities and artifacts.

We think that many of the characteristics of the automotive domain apply to other domains of complex system engineering as well, especially in those areas where product family development is taking place or emerging.

From our experience, we are therefore convinced that the issues presented here, while not necessarily being in the current focus of the requirements engineering research area, are of crucial importance for improving requirements engineering in the future.

4. Acknowledgements

We thank our colleagues from DaimlerChrylser Software Engineering Research, from Passenger Car Development, and from Commercial Vehicle Development for sharing with us their insights and experiences on requirements engineering. In particular, we would like to give special thanks to Matthias Hoffmann for his excellent comments on a draft of this paper.

5. References

[1] S. Robertson and J. Robertson, *Mastering the Requirements Process*, Addison Wesley, London, 1999.

[2] K. Wiegers, *Software Requirements*, Microsoft Press, Redmond, 1999.

[3] I. Sommerville and P. Sawyer, *Requirements Engineering: A Good Practice Guide*, Wiley, Chichester, 1997.

[4] G. Kotonvy and I. Sommerville, Requirements Engineering, Wiley, Chichester, 1998.

[5] B.L. Kovitz, Practical Software Requirements, Manning, Grenwich, 1999.

[6] D. Gause and G. Weinberg, *Exploring Requirements: Quality Before Design*, Dorset House, New York, 1989.

[7] G. John, M. Hoffmann, M. Nagel, C. Thomas, and M. Weber, "Using a Common Information Model as a Methodological basis For a Tool-Supported Requirements Management Process", *Proceedings 9th International INCOSE Symposium*, 1999.

[8] J.L. Cybulski and K. Reed, "Requirements Classification and Reuse: Crossing Domain Boundaries", *Proceedings of the 6th International Conference on Software Reuse*, Vienna, June 2000.

[9] K. Wiegers, "Requirements When the Field Isn't Green", *STQE*, vol. 3, no. 3 (May/June 2001), www.processimpact.com/articles/reqs_not_green.pdf

[10] W. Lam, J.A. McDermid and A.J. Vickers, "Ten Steps Towards Systematic Requirements Reuse", *Requirements Engineering*, Vol. 2(2):102-113, Springer, 1997.

[11] F. Kiedaisch, I. Alexander, "Towards Recyclable Requirements", *Proceedings of ECBS 2002 (9th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems)*, Lund, Sweden, 2002 (to appear).

[12] A. Cockburn: *Writing Effective Use Cases*, Addison-Wesley, 2001.

[13] D. Kulak and E. Guiney, *Use Cases: Requirements in* Context, ACM Press, Texas, 1999.

[14] B. Ramesh and M. Jarke, "Towards Reference Models for Requirements Traceability", *CREWS Report CREWS-99-13*, 1999.

[15] O.C.Z. Gotel, *Contribution Structures for Requirements Traceability*, Ph.D. Theses, Imperial College, Department of Computing, London, 1995.