Experiences in Managing an Automotive Requirements Engineering Process

Nadine Heumesser, Frank Houdek
DaimlerChrysler AG,
Research and Technology
Postfach 23 60
89013 Ulm, Germany
{nadine.heumesser, frank.houdek}@daimlerchrysler.com

Abstract

The specification volume for all software-based systems that are built in a modern car has passed the 20,000 pages mark. Even for a single component, we find specification documents comprising several hundred pages. Clearly, such specification documents cannot be created and changed simply and quickly. We hence need a systematic process to elicit and maintain, negotiate and validate all requirements, i.e. a proper requirements engineering process.

In this paper, we present an example for such an automotive requirements engineering process and the instruments we employed to manage this process. The experiences are drawn from projects at DaimlerChrysler Passenger Car Development.

The paper sketches the requirements engineering process used, the core management instruments deployed, e.g. a feature list, and observations gained in utilizing this process.

1. Introduction

Software has become a major ingredient in modern cars. Usually, this software is not developed by the OEMs themselves but by suppliers, leaving the OEM with the duty to specify the software and its surrounding hardware (forming a complete electronic control unit, the ECU). Modern cars encompass more than 50 ECUs, which are typically developed by different suppliers. Striving to ensure smooth integration and interoperability of these ECUs, we have to provide very detailed specification documents to the suppliers before they begin with the development work (see [1, 4]). Consequently, the complete requirements engineering (RE) process for ECU specifications takes quite a while and, moreover, has to ensure proper negotiation of all relevant stakeholders.

Requirements changes may arise throughout the entire development process. Key drivers for change requests may lie in the technological, financial, or managerial fields. Other drivers may be regulation authorities or competitors. Each requirements change, however, raises the need for further negotiation and replanning.

Due to the required level of detail, specifications tend to become extremely voluminous. Specification documents consisting of several hundred pages are not uncommon.

Clearly, managing an RE process constrained by these facts cannot be done simply as a "by-product". Instead, this calls for clear management processes to be in place to plan and monitor all the specification-related activities. Here, the major challenge is to find a pragmatic level between the necessary transparency and the affordable efforts for RE process management.

In this paper, we describe our experiences in managing an RE process which we gained in a project at DaimlerChrysler Passenger Car Development. The remainder of this paper is structured as follows. Section 2 gives some insights into the application domain and elaborates the RE process followed. Section 3 explains the various management instruments we used, their implementation, and experiences gained from their application. Section 4 summarizes our findings and addresses directions for future work.

2. Instrument Cluster Development

In this section, we give a brief overview of the application context in our case study: the instrument cluster group at DaimlerChrysler Passenger Car Development. Additionally, we present the requirements engineering

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

process employed and identify problems of insufficient RE process management we observed in the past.

2.1. Instrument Cluster

An instrument cluster is a complex ECU connected to other ECUs by some communication buses (see Fig. 1).

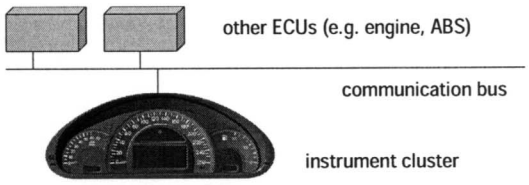

Figure 1: Example instrument cluster.

As the component is visible to the driver, it not only has to fulfill high quality and reliability demands but also needs to meet aesthetic requirements. In addition to conventional instrument cluster functions, e.g. speedometer or tank indicator, it offers advanced functions to the user via a built-in display (e.g. a travel manager). And what is more important, it provides services to other ECUs such as information about the current state of the car or data for maintenance purposes.

Specification documents (SRSes) for instrument clusters consist of anywhere from 200 to 600 pages, describing requirements on a detailed technical level. The used notation is mainly natural language, accompanied by, e.g., decision tables, state charts, data flows diagrams, signal diagrams, where appropriate. Additionally, there are many related documents that augment the actual SRS itself. Related documents might specify something as general as the overall vehicle architecture or a specific element such as dedicated protocols. Other examples for related documents are company norms and standards on, for example, electromagnetic compatibility (EMC), product warehousing or production.

2.2 Constraints and Demands

We see a number of factors that constrain instrument cluster development activities. The most important are introduced in the following.

- **Contractor–supplier relationship.** As is true for many other ECUs, an instrument cluster is developed and manufactured by external suppliers. Thus we have to run through a classical tender process: we have to initiate a call for bids, evaluate proposals, select a supplier, and monitor the ongoing development activities.
- **Engineering, not RE.** The engineers responsible for an instrument cluster consider themselves domain experts, not requirements engineering experts. This implies that they are not keen on providing high gloss specifications. Instead they are interested in finding high quality technical solutions. As a consequence, all management instruments must be easily applicable.
- **Many involved stakeholders.** As mentioned before, there are many stakeholders involved in the instrument cluster specification. For instance, human machine interface (HMI), optical appearance, or EMC requirements are defined by the departments in charge for these topics. This means that an instrument cluster engineer often acts as a requirements integrator. She has to initiate negotiation meetings and document the results in the SRS. Thus we see a significant demand to track and monitor the ongoing negotiation status.
- **Late requirements and requirements changes.** The overall development process for a new car model takes several years. Naturally changes are inevitable. And as stated in the introduction, there are many reasons for changes. Since the instrument cluster acts for many vehicle functions as a display unit, changes in these functions often influence the instrument cluster specification in some way. Thus the number of changes is even higher than for "ordinary" ECUs. Clearly, efficient change management has to be an essential ingredient in an RE management approach.
- **Requirements reuse.** Development activities for the instrument cluster of a new car model do not start on the green field. Clearly, there are existing requirements that can and should be reused. To do so, we have implemented an approach that is based on model-independent function specifications that are augmented with model-specific requirements (for more details, see [1]). This fact eases the RE process insofar that high-quality general requirements are already in place and these can be used for initial specification versions.
- **Incremental development.** The overall vehicle development process is driven by so-called quality gates (see, for instance, [3]), implying incremental development activities with defined vehicle prototype levels. Clearly, this has its impact on ECU development processes, as well. ECUs are developed as A, B, C, and D samples with each sample type having to meet certain quality criteria. Early samples might make use of slightly different technologies than the final product or may not provide the full functionality.

Given these constraints, what does managing an RE process actually mean? In our case it means that we

have to plan and monitor all the specification-related activities throughout the entire product development process. In particular, we have to pay attention to the following activities:

- **Product requirements state**, i.e. we have to plan and monitor the specification progress. We therefore first have to know the stakeholders involved for each part of the system (e.g. for a single function), the degree to which we have specified the model-independent and model-dependent requirements or whether there are open issues or not.
- **Communication and negotiation state**, i.e. we have to monitor ongoing negotiation activities with suppliers. We want to know whether a requirement has been communicated to the supplier or, if so, if it has been agreed with the supplier.
- **Ongoing changes**, i.e. we have to track all change requests.
- **Release planning**, i.e. we have to plan, negotiate and monitor the scope for each product sample. Insufficient release planning, for example, could cause extra effort (e.g. attempts to test a function that has not been [completely] implemented) or provoke improper functional behavior (because a function A relies on information from function B that has not been implemented in the given product sample).

3. Requirements Engineering Process

In this section, we sketch the overall requirements engineering process we selected to manage instrument cluster requirements. As explained, ECUs such as the instrument cluster are usually not developed by DaimlerChrysler itself but by external suppliers. Thus the complete requirements engineering process is heavily driven by activities due to contracting and supplier management. Figure 2 illustrates the overall requirements engineering process used for the instrument cluster development. Here, we see four major phases:

- **Phase 1 (internal product scoping).** In this phase, we define the scope of the product, e.g. the required functions, key requirements on communication, optical design, required hardware. Product scoping for one ECU is clearly not an isolated activity, happening instead for all the parts of the vehicle simultaneously. Clearly, this calls for a multitude of negotiation and validation activities.
- **Phase 2 (creation of call-for-tender specification").** Based on the defined product scope, a dedicated SRS that defines all the requirements to a certain level of abstraction is then built. The main emphasis is to mention all cost drivers, i.e. requirements that will have a direct impact on the ECU price. Certain functional details, e.g. algorithms or error conditions, are omitted.
- **Phase 3 (bidding and contracting).** In this phase, the call-for-tender specification is distributed to potential suppliers. They evaluate the SRS and hand in their offers. During the evaluation process, usually detailed questions that initiate further clarification and negotiation activities on the OEM side arise. Solution proposals from the suppliers also stimulate further requirements negotiations. The phase ends with supplier selection and the signing of the contract.
- **Phase 4 (product development).** In this phase, the ECU is actually developed. As set out in the previous section, this happens in several product samples. This means we have to define, at the beginning of the product development phase, which product samples are needed at which point in time (aligned with the overall vehicle development process) and which feature set has to be implemented in each product sample.

It is important to note that the complete SRS is not available at the beginning of the product development phase. In particular, new functions and features are still emerging on the detailed level. As a consequence, there are SRS updates for each iteration, adding detailed requirements for new features that are to be part of the next prototype. After a specification for a feature has been handed over to the supplier, we start a change management process for this particular feature. Requirements changes are then introduced by means of change requests. Figure 3 schematically illustrates the evolution of an SRS over time.

For each increment, we see a specification handover process (see Figure 4) where a new version of the SRS is handed over to the supplier. A single handover process is organized as follows:

According to the planned features for a given in-

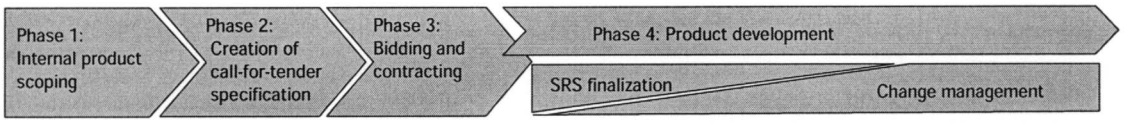

Figure 2: Requirements Engineering Process Applied.

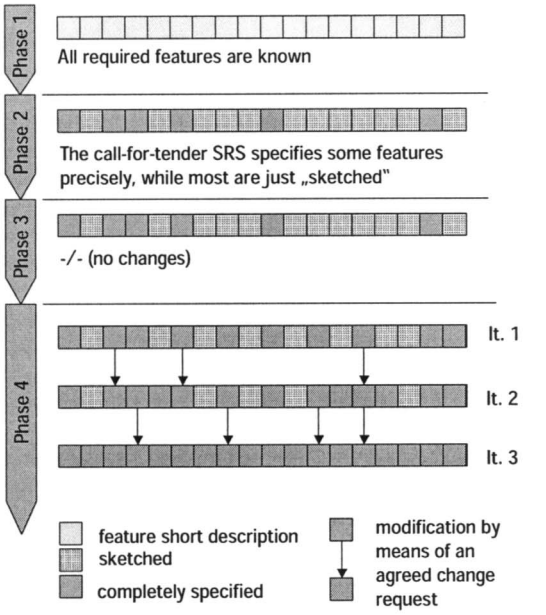

Figure 3: Evolution of an SRS over time.

crement, the relevant parts of the SRS are completed. To do so, the stakeholders concerned are asked prior to a freeze date to hand in all requirements relevant to their needs. Then, the individual requirements are integrated (and, if necessary, negotiated and modified). Before the SRS is handed over to the supplier, internal reviews are performed to assure specification quality. The suppliers then evaluate the new specification parts. All issues are reported and clarified subsequently by means of an open-issue list. Later on, the given requirements are used to check the product samples delivered.

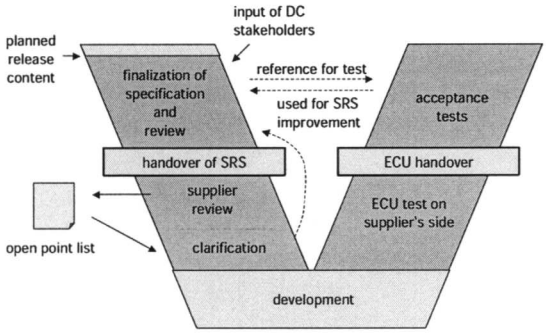

Figure 4: Handover process.

In addition to specification parts that are added in each iteration (as per the pre-planned specification stages), we see change requests affecting specification parts that have already been communicated to the supplier. They are handled in a standard change management process.

4. Management Instruments

In this section, we present the key instruments we found to be valuable for managing the RE process for our instrument cluster development projects. Over three projects, these instruments have matured and proven to be beneficial.

4.1. Feature List

The central management instrument is a so-called feature list. The feature list itemizes all the essential characteristics of the system under development. The term feature is not bound to functions only. Our feature list also mentions required hardware characteristics, HMI features, required architectural properties and so on. There are no clear guidelines to denote a feature, but a feature might be characterized by the following properties:

- A (functional) feature provides added value to the system, i.e. it makes a considerable difference whether the feature is present or not. Thus it is significantly more than, for example, an error condition.
- A feature concerns a topic we have to talk about with our supplier because it impacts the system and its costs.
- A feature is a static element of the system. We can test whether a feature is present or not.
- A feature is a term with some agreed co-notation. Thus it acts as an index to detailed specifications.

Figure 5 shows a feature list example covering functional features, hardware features, and HMI features.

In each phase of the RE process, the feature-list plays a slightly different role. In phase 1, we utilize the list to define the product scope by adding (or deleting) new features. Additionally, we assign the features to product variants. In phase 2, we use the feature list to track internal negotiation and clarification activities as well as completion of the call-for-tender specification. In phase 3, we use the feature list as a basis for discussion with our potential suppliers. They are even requested to submit their bids on a feature level. We experienced that this is not possible for all features, but for those where it is possible, it gives us a means to

- identify substantial deviations between the bids (that might indicate that one supplier has a completely different understanding of a feature than another one);

- trigger internal decision processes that might result in omission or inclusion of a "dangling" feature.

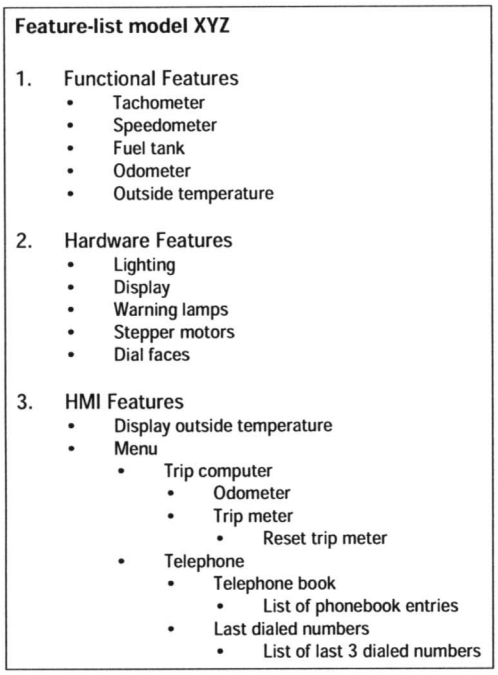

Figure 5: Feature list example (without additional columns like suppliers comments, release planning, etc.).

At the beginning of phase 4, we again plan our product releases on basis of individual features. Here, a feature might be irrelevant for a release, might be prototypically implemented or may have to be fully available. Main criteria for release planning are logical dependencies amongst features, required features according to the overall vehicle development process (a speedometer, for instance, has already to be available for the earliest prototypes), and workload. So far, no formal release planning method or tool is used

Throughout the last projects we found a feature list to be a very valuable means for planning and monitoring development activities. Sometimes, however, we still struggle with the required degree of detail. The list is definitely not homogeneous in terms of its level of abstraction. To some extent, this seems natural, as there are parts of the systems that are well known and can be identified by a single name (i.e. a coarse-grained feature) whereas other parts are quite new, resulting in more fine-grained features. In our case, the indexing relationship of a feature to specification objects (stored in the RM tool DOORS from Telelogic) is approximately 1:50, i.e. for each feature there are, on average, 50 requirements objects in the database. In total, our feature list comprises approximately 400 features.

4.2. Maturity and Negotiation State

Often, specification documents are written by different developers and merged by one person who is responsible for the particular ECU. If this person does not have the requisite knowledge about the status of the requirements, an incomplete draft version might be handed over to the supplier. The supplier than expends a great deal of time and effort to evaluate requirements that might change only a few weeks later. To avoid this kind of work, it is crucial to inform the supplier that the specification they are using is only a draft version. With this information in mind, they are able to minimize the effort to understand and capture an overview of the content. The information they need for the estimation whether to read or wait is held in an attribute on the requirements level called the modification state. The associated values are draft, in process, released, and open point.

As mentioned before, it is hardly possible to have a complete and absolutely stable specification document at the beginning of phase 4. Instead, changes and late requirements are inevitable, making it mandatory to keep track of the information and negotiation process with the supplier. We do this on the level of modification history entries. For each entry, we perform the tracking by means of a simple state model where the modification has been introduced in the specification, communicated to the supplier, or even agreed with the supplier. As some parts of our specification are model independent (and thus relevant for different suppliers), we track this information for every supplier affected.

4.3. Open-Point Lists

Along with the handover process we see an evaluation of new or modified specification parts by the supplier. Usually, this raises a number of questions and issues to be clarified. To keep track of ongoing clarification activities, all these points are collected in an open-point lists. As a response to an SRS evaluation, the supplier creates an open point list that is handed over to the OEM. All mentioned issues are integrated in an ongoing OEM open point list which is used for internal clarification activities. Therefore, we use the states Open, In process, Closed, and Refused. Clarified issues (i.e. state is Closed or Refused) are reported to the supplier who decides whether he agrees (meaning that he Closes his open point) or disagrees. In this case, the issue might stay open (which means it is reported in the next cycle again) or causes a new open point that

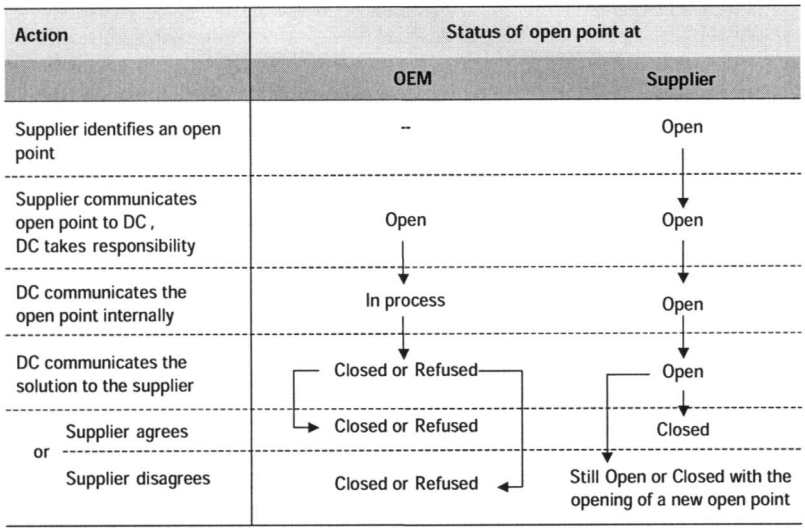

Figure 6: State model of open-points entries.

makes his issues more clear. Figure 6 illustrates this process graphically.

A crucial element in this process is the open point ID that must not be changed throughout the entire development activities. Otherwise tracking becomes unstable.

At the first glance, this process might look complicated and tedious. But the two lists help to separate concerns. For instance, both parties can manage their internal issues with their respective lists as well. And we clearly express the separate individual evaluation of all issues. By means of this process we are able to assign and track responsibilities at a fine grain level. By this, we do not lose issues (which consequently help to improve specification quality).

5. Summary and Outlook

Requirements engineering activities for complex products such as automotive ECUs require a well-defined process that is explicitly planned and monitored. Current literature provides only shallow help on this topic. For instance, status tracking is mentioned as an important ingredient (see, e.g., [5]). Instruments for planning and tracking the overall RE process are rarely mentioned. In fact, we only found the idea of feature-lists (that we exploited as well, see [2]), but without concrete usage guidelines.

In this paper, we describe an RE process we employed in the instrument cluster group at Daimler-Chrysler Passenger Car Development together with the instruments we used to manage this process. In our experience, the cost-benefit ratio of these instruments is extremely healthy. Clearly, there have been implicit instruments in use before, as well. Yet they were applied in a more ad-hoc fashion and on an individual basis. Making the instruments described here explicit and mandatory helped to streamline development activities significantly. And the feature list, in particular, was of great value in focusing development activities. The fact that suppliers also utilize this list to plan and track development activities on their side emphasizes the beneficial effect of this list.

However, there is still some potential for further improvements. In the near future, we aim at improving our reporting abilities to provide aggregated management information about the current state of the requirements.

Acknowledgements

We would like to acknowledge our colleagues in Sindelfingen, who were always willing to try something new and helped us to keep our ideas down to earth. We learned a lot and had a great time. Thanks, folks.

References

[1] N. Heumesser and F. Houdek. *Towards Systematic Recycling of Systems Requirements.* In Proc. of the 25th Int. Conf. on Software Engineering, May 2003, pp. 512-519.

[2] S. Richter. *Feature-based Programming . Lean methods in practical use.* Addison Wesley, 2003 (in German).

[3] T. Vullinghs, T. Gantner, S. Steinhauer, T. Weber. *Experiences on Lean Techniques to Manage Software Suppliers.* In Proc. of Profes 2000, June 2000, Oulu, Finland.

[4] M. Weber and J. Weisbrod. *Requirements Engineering in Automotive Development — Experiences and Challenges.* IEEE Software, Jan/Feb 2003, pp. 16-24.

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