Journal of Universal Knowledge Management, vol. 1, no. 2 (2006), 112-122
submitted: 15/5/06, accepted: 15/6/06, appeared: 28/6/06 © J.UKM

The Requirements Engineering Gap in the OEM-Supplier Relationship

Christian Allmann
(Audi Electronics Venture GmbH, Germany
christian.allmann@audi.de)

Lydia Winkler
(Audi Electronics Venture GmbH, Germany
lydia.winkler@audi.de)

Thorsten Kölzow
(Audi Electronics Venture GmbH, Germany
thorsten.koelzow@audi.de)

Abstract: The OEM's requirements engineering and management process is affected by many residual encumbrances and future constraints. The encumbrances' origin comes from the OEM's strong dependency on suppliers' software development knowledge and support. In contrast, future constraints are dealing with the assembling of software engineering knowledge, especially requirements engineering, as OEM's core ability. On this account reducing the OEM's knowledge gap demands tailored engineering processes. The tailoring should take account for different kind of process knowledge, distributed development environment, changing project responsibilities, available but unstructured knowledge bases and different educational members' background. This report examines main challenges in OEM's development process to overcome the past OEM-Supplier relationship achieving common development in partnership. For clarification, three process flows are presented showing in some kind the historical evolution of the OEM-Supplier relationship. Based on the presented vital process constraints requirement methods, tools and project management guidelines are derived.

Keywords: Requirements Engineering, Automotive Software Engineering, Project Management, Knowledge Management
Categories: D.2.1, J.2, K.6.1

1 Requirements engineering in the automotive industry today

The development and production of modern cars are affected by a strong relationship between OEM (original equipment manufacturer) and its suppliers. Due to the increasing importance of software in the automotive, both OEMs and suppliers have established a requirement engineering process to stay abreast of changes. The OEM's requirements engineering process is actually tailored to the management and the integration of certain supplier products [Huhn, 03]. Therefore the OEM's goal is to provide information about the system environment into which the supplier product must be integrated. The integration of supplier products itself demands a precise

communication interface between OEM and supplier as well as between the suppliers themselves.

2 The ideal development process using the FLOW notation

To emphasize the challenges of the development, especially the requirements process, we present three different project flows using the FLOW notation [Schneider, 05]. The focus is to show the different communication channels and interfaces between the stakeholders and the developed documents, as well as the difference between the stakeholders and OEM's experience bases.
In our opinion the FLOW concept developed by the University of Hanover is well suited to model the complex project process flows with respect to the OEM-Supplier relationship. FLOW is a graphical notation to visualize the information flow in a project by modelling direct communication channels between project artefacts and project members. For the presented examples, only a subset of the basic notation is used:

- *document symbol*, representing persistent data (e.g. requirement documents written in Word or DOORS)
- *face symbol*, representing project stakeholders (e.g. requirement engineer, project leader)
- *black arrow*, representing the information flow (e.g. review meeting)
- *solid line*, information from a persistent information source (e.g. written document)
- *dashed line*, information flow from a non-persistent information source (e.g. project member)

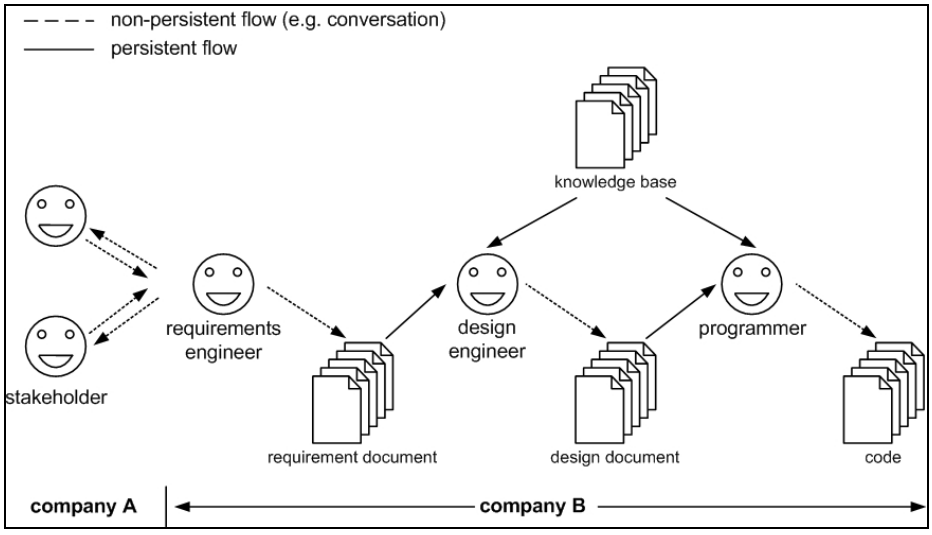

Figure 1: The ideal process flow

The first example, shown in figure 1, represents an idealized development process flow. On the left hand side the two face symbols represent an arbitrarily number of stakeholders employed by company A. These stakeholders are the main experts in a company's business for who a tool has to be built. Company A places an order with Company B for the tool development.

In our idealized view, neither the company nor the stakeholders have any imagination how software is developed. Thus, the first essential development step is the elicitation of requirements the stakeholders have, carried out by the requirements engineering team of Company B. As figure 1 shows the requirement engineer manages the elicitation process by asking the stakeholders and writing the requirements down. The resulting document is the first persistent artefact in the ideal process flow. Based on the document, ongoing developing activities are building up, see figure 1.

On the supposition that the created requirements document is well structured and the requirements quality attributes (e.g. atomicity, consistency, unambiguousness, etc.) are fulfilled, the design engineers, and later the programmers, have the chance performing their work with the help of the company's knowledge base. The knowledge base symbolizes the collected experience of the company's completed projects that can be used to perform their daily work.

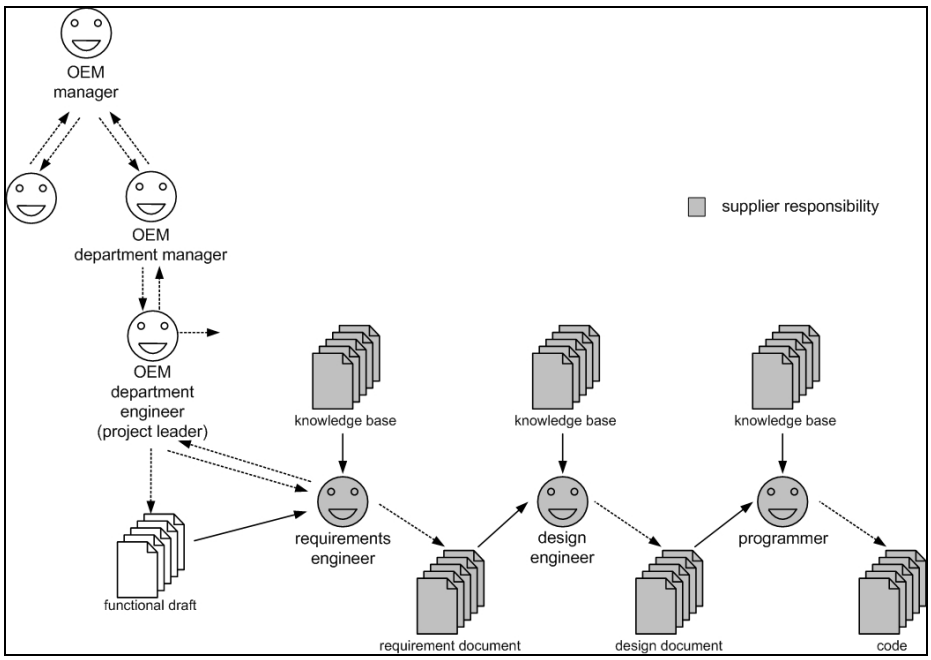

Figure 2: The OEM-Supplier process flow

3 The relationship between OEM and supplier

The OEM-supplier project example, figure 2, shows the past and in parts the present relationship between customers and contractors. Compared to an ideal process flow, figure 2 shows some important differences to the described flow in figure 1. Just like in our first example, the OEM (resp. company A) assigned a supplier (resp. company B) developing software system.

The most conspicuous distinction takes place on the left hand side. The generalized category stakeholder, like it is used in the ideal process flow, is broken down to form a distinct command structure. The exemplary command structure, shown in figure 2, represents a decision-making process starting by the OEM managers up to the department engineers. The depicted form of information exchange points out a rather generalized version of the real process flow. Nevertheless, it is important for the understanding and all further derived challenges that a written description of the information flow is only kept at one single point, the responsible department engineer. Although all relevant managers or engineers have their own vision of what the system has to perform or not, only the responsible department engineer interprets the system's requirements and writes them down. Therefore, the engineer represents a bottleneck for all requirement activities, although he is in many cases not skilled to perform requirement processes or techniques. All decisions made are based on experience he has gathered during his work life. In the past nobody would check the requirement document with respect to quality attributes like consistency, clarity, ambiguity, etc. because nobody has the experience to give evidence about the quality, especially the consistency. Generally getting feedback, the engineer presents and interprets his written text to the managers and engineers allowing a judgement about the system's functionality.

The responsible engineers cannot be fully blamed as the exclusive cause for the bottleneck. On the one hand, the engineer does not have the required experience and education to fulfill all expectations, on the other hand the project time pressure allows no further education, and adequate OEM knowledge bases on requirements engineering do not exist. In summary, all mentioned aspects lead to different quality in requirement documents. Therefore, we called the requirements document a *functional draft*. The functional draft is the basic document inviting offers. Concerning the statements above, draft and engineer represent a weak source for potential suppliers building up a system's understanding, they preferred. Managing functional drafts ranging in quality and detail has a deep impact on the supplier's side. First, the supplier cannot estimate the development cost and time resulting in imprecise offering. Especially in the field of driver assistance systems some suppliers profit from the generalized requirement description and sell their own existing system - perhaps still developed for a competitor- without any further adaptation.

Thus, the suppliers have a main interest in requirements engineering because they must manage OEM's requests and changes and perhaps have the chance to sell the developed system to another company. Accordingly, comparing figure 1 and 2, the smallest changes to the ideal process flow occur on the supplier's side. Unlike the central knowledge base of company B, the supplier's bases are more sophisticated with respect to product line development. The knowledge base assists the supplier's requirements engineers performing the preparation of the requirements document by

the usage of suitable templates, document structures and tools. In figure 2, three knowledge bases are revealed. Depending on the internal bases' structure, the diversity of companies' processes, the amount of collected experience and so on, the three shown symbols for knowledge bases represent one monolithic base (e.g. a simple ring binder) or complex distributed DBs. In summary, the requirement process is oriented to perform requirements analysis and validation regarding the specific product line information.

4 Future directions of the requirements engineering process

The OEM's disadvantages arising from the mentioned flow of figure 2 can be read out very easily. The knowledge about the integrated product (e.g. a new driver assistance system) is kept by the supplier which results in a strong dependency on the supplier concerning the integration of additional product features. Thus, the supplier can easier dominate the price for new functionalities. For the OEM, this entails the loss of two main abilities: the knowledge about the integrated product (component protection) and the possibility to negotiate with different suppliers about new product feature prices. To avoid these main disadvantages, the OEM must participate in the requirements engineering process and perhaps take a leading role developing product parts on his own.

On this account the OEM must achieve three goals:

1. the reduction of knowledge loss about his integrated systems
2. the installation of own knowledge bases by means of collecting, verifying and integrating all existing company's know-how
3. a progressive project involvement with regard to proprietary-development

All three goals have one main challenge: a *seamless transition* into daily work. The distinctions and resulting changes in the process flow are shown in figure 3.

The observable difference between figure 3 and all figures before is the omission of a clear separation between OEM and supplier part. Although the left hand part of figure 3 shows no real difference to the previous ones, a subtle distinction still exists – greater project responsibility of the department engineer in charge. One of the mentioned confusions of the OEM-Supplier process flow was the unbalanced relationship becoming manifest in the functional draft. Irrespective of the document quality and content, the supplier takes full responsibility for the success of the project. Establishing proprietary-development demands distributed responsibility like it is exposed in figure 3 by means of single workpackage leader. The scenario shows one big chance for OEM and supplier that is likewise the obstacle. Both can concentrate their efforts to their core abilities. In the case of driver assistance systems this could be the design of the sensor components and image processing as supplier's part and advanced algorithm and functionality, e.g. lane detection as part of the OEM. The obstacle would be to manage the distributed responsibility with respect to the common goal, in our case the OEM driver assistance system.

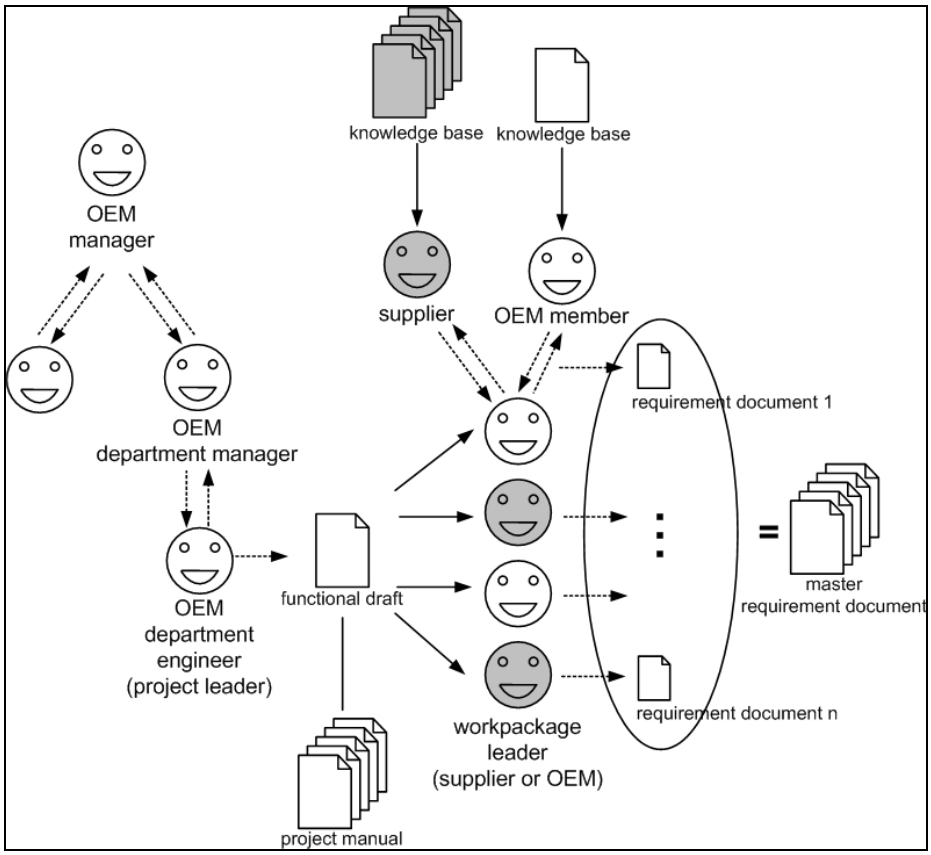

Figure 3: The potential future OEM-Supplier process flow

For future development in highly innovative fields, like hybrid technologies, x-by-wire or sensor fusion, working in partnership is inevitable. The resulting distributed process flow is very sophisticated to organize and to lead. The described process flow shows only the first part of a potential future development cycle but illustrates the main constraints. Common development means sharing knowledge, the knowledge about the actual developed system as well as the project experience companies have collected over the years.

As a consequence of the different development "philosophies", a clash of interest occurs. Now the job of the project leader is it to establish some kind of cooperate document (generally called project manual) to organise the different development activities, determine the development chain and OEM's and suppliers' fields of responsibilities. Although figure 3 shows the existence of knowledge bases on both companies' sides the usefulness of the contained information for partnership development must be questioned. Neither OEM nor supplier really has any experience in this kind of development because partnership means publication of at least essential

and confidential information. For the requirements engineering process, this implicates common:

- requirement process including elicitation, analysis and validation activities
- tool framework for distributed development
- exchange of partial development artefacts
- development and process guidelines
- definition of partnership responsibility

Time-shared and *artefact-shared* development is mandatory concerning the effort for system's development under the regular time pressure. All met implications can be subsumed to one essential point: a common development platform. To avoid misunderstandings, the development platform comprehends solutions for arising communication overhead, distributed access to the different knowledge basis, project management and tools. The realisation and establishment of this kind of partnership development demand some requisites on the OEM's side such as partially described in the following section.

5 OEM management and engineering constraints

5.1 User constraints

It must be kept in mind that although a lot of mechanical and hydraulic systems are replaced by mechatronical systems, the engineering team in charge for these components cannot be replaced in the same way. The OEM's departments are mainly staffed with mechanical and electrical engineers with associated educational background. In replacing the mentioned systems by electronic and software these engineers not only have to consider the development of the usual hardware components, but also have to take care of the electronic devices and the integrated software. The impact of this evolution is that the engineers often do not have the necessary engineering background to evolve/maintain such systems effectively and efficiently.

With respect to figure 3, the partnership provides an opportunity to coach the engineers on the job. Coaching OEM's engineers on the job by their project partners helps in finding a common discussion platform. Nevertheless, this know-how transfer can only be performed within project's limits. Hence, the goal must be to establish a requirement engineering process and workflow that is on the one hand tailored to the development process and on the other hand tailored to the engineers' abilities. Tailoring a process with respect to a special user group is necessary because nobody – as the experience in our company shows – will adhere to a process if the users can not identify with it. User constraints to process definition are:

- intuitive application handling (benchmark Microsoft Word)
- simple and seamless tool chain
- advised request and change board
- support team for process, methods and tools

5.2 The establishment of an information knowledge base

The OEM's requirement engineering process currently often lacks the knowledge of how to perform requirements engineering effectively and efficiently. The experience, gained during the development process, must be retained by building up a knowledge base for future developments. Gained experience could be workflow information, description patterns, review documents or something else affecting the requirements process. Although in most organisation standardised document templates still exist, the practice shows that project concerns, topics, time pressure or quality gates demand reengineering of the common templates. That given statement comprised an interesting point namely the information on company's exhibit engineering knowledge. The drawback: the information is scattered over the different company's departments. On this account, the company's departments reach a first evolution step by adapting the common requirement standards to their personal needs. Unfortunately, the conclusions that are drawn have no impact to company's overall engineering evolution because no feedback loop is implemented. For us, the information knowledge base is a platform to evaluate the concerns and constraints of the different evolution steps towards more applicable process flows, methods and tools. As it is mentioned at the end of chapter three, the representation of knowledge bases can range between ring binders and complex DBs. Thereby, the complexity of the knowledge base gives no evidence of the stored information. The stored "process" information could be ranged between product information, oral information flows or experience flows.

Besides this process information the knowledge base is also a platform to assemble product information such as performance or dependability analysis results, which can be used to obtain judgements about development risks at an earlier stage. Risk and cost estimation for system development is as important as new technologies are evaluated and integrated in all business areas.

5.3 Automation of user constraints using the example of traceability

The following example describes the traceability problem and shows how important automation is with regard to requirements engineering. It is assumed that usual requirements documents consist of 200 to 600 pages [Heumesser, 04] with 12 requirements per page. Let us assume that 400 pages would result in 4800 listed requirements. The management of these requirements demands to trace each of them, linking each requirement to associated requirements or test cases to validate the implementation against them. Concerning the example above, the linking of each requirement to respective test cases costs 30 seconds working time each time the link must be changed. The resulting working time by 4800 links amounts up to 40 hours respectively 1 week of work. This fictive calculation shows that although there is a need for linking requirements to refined requirements or test cases the resulting work time increases tremendously.

The actual chapter has a strong correlation to the section of user constraints. Most engineers in our company, affected by requirement engineering do not have the time for testing or evaluating new approaches or tool add-ons the company's engineering team prescribes. These engineers at the grass-root level use what is practicable for the actual project needs. Regardless of traceability advantages for them, it is more time

consuming. Bridging the gap between user prejudices and required engineering steps, the engineering rollout can only be done incrementally. Regarding test case derivation, generation and maintaining of requirement traceability in documents is essential to achieve a consistent system view. A consistent system view is as important as a distributed development environment. Both constraints must be kept in mind, especially concerning the single workpackage leader's responsibility (see figure 3). From their point of view, they can only observe the quality of the produced requirement document part (1..n). The question must be answered who guarantees the quality, mainly consistency, of the whole requirement document even if requirement changes have an impact of more than one document part. Therefore, the definition of the development platform and process must guarantee in each case a consistent system view.

5.4 Integration of formal notations

According to the close collaboration between the OEM and the supplier it is also desirable that the common development platform, see chapter 4, enables the production of consistent, unambiguous and understandable requirements documents. As shown in figure 3 requirements documents are produced independently by the OEM as well as by the suppliers, which are then merged to a master requirements document. Particularly the merging process requires a common requirements language to enable the production of a consistent requirements document. At this, the basic idea is to guide and support engineers to write high quality (clear, complete, correct, understandable and testable) functional requirements by means of a requirements specification template and/or pre-defined requirements' patterns.

Using a specification template can be a first step in obtaining more precise requirements, as it is shown in the following example. In the requirement

If the ignition is on and the engine is not running or the ABS module signals a defect, the display module shall switch on the ABS control lamp.

it is neither clear which conjunction is dominating nor if inclusive or exclusive '*or*' is meant. Therefore, the requirement can be interpreted in several ways:

- *If **(**the ignition is on and the engine is not running**)** or the ABS module signals a defect, …*
- *If the ignition is on and **(**the engine is not running or the ABS module signals a defect**)**, …*
- *If **either** the ignition is on and the engine is not running **or** the ABS module signals a defect, …*

Applying mathematically logical operators according to the specification template defined by Chris Rupp [Rupp, 04] can easily avoid these kinds of misinterpretations:

*If **(**the ignition is on **AND** the engine is not running**) OR** the ABS module signals a defect, the display module shall switch on the ABS control lamp.*

Using a specification template can help to obtain well-structured, understandable and clear requirements, but it cannot guarantee that for instance requirement 127

conflicts with requirement 305 in a specific or even worse in two different but interdependent documents.

On this account one idea is to refine the specification template suggested by Chris Rupp by means of pre-defined requirements' patterns. At this, the definition of a requirement is carried out in several phases. First of all application specific inputs and outputs must be defined in a glossary. If not all inputs and outputs are known initially, there is also the possibility to expand the glossary during the creation of the requirements specification.

Secondly the main structure of the requirement is selected:

- If … then …
- Only if … then …
- After … then …
- As soon as … then …
- …

In the third phase the conditions, like *"the ignition is on"* and the actions or processes, like *"switch on the ABS control lamp"*, are specified. At this, the nouns of the conditions and actions are derived from the glossary mentioned above and the verbs are selected from a pre-defined catalogue of automotive specific process words.

If requirements only consist of nouns, verbs and part-sentences which are defined in a global database it's easy to implement specific search routines. Therefore it is for instance possible to display all requirements corresponding to a particular input signal. This again supports the requirements engineer in locating problems between requirements in adequate time and with adequate effort.

Once a controlled natural language exists, a further step could be to map the pre-defined requirement patterns to formal/mathematical representations to obtain a formal requirements specification document in the end. Engineers can specify requirements in their familiar but restricted natural language, whereas the advantages of formal specifications and the basis for their automated support are available at the same time [van Lamsweerde, 00].

6 Future Work

Currently our scope is to implement the mentioned knowledge base and to establish a change control board to evaluate user constraints and project requirements. The change control board has the mission to evolve a so called "requirement engineering kit" tailoring the requirement process to our specific project needs with respect to product quality, user experience, time and money.

References

[Heumesser, 04] N. Heumesser, F, Houdek, Experiences in Managing an Automotive Requirements Engineering Process, 12th International Conference on Requirements Engineering. 2004, Kyoto, Japan

[Huhn, 03] M. Huhn, P.-M. Hoffmann, M. Mutz, Digitale Lastenhefte für die Softwareentwicklung vernetzter Steuergeräte, 23. Tagung Elektronik im Automobil, June 2003, Stuttgart, Germany

[Rupp, 04] Ch. Rupp, Requirements-Engineering und –Management, 2004, München, Germany

[Schneider, 05] K. Schneider, D. Lübke, Systematic Tailoring of Quality Techniques, 3rd World Congress of Software Quality, 2005, Munich, Germany

[van Lamsweerde, 00] A. van Lamsweerde A, Formal specification: a roadmap. ICSE - Future of SE Track, 2000, Limerick, Ireland