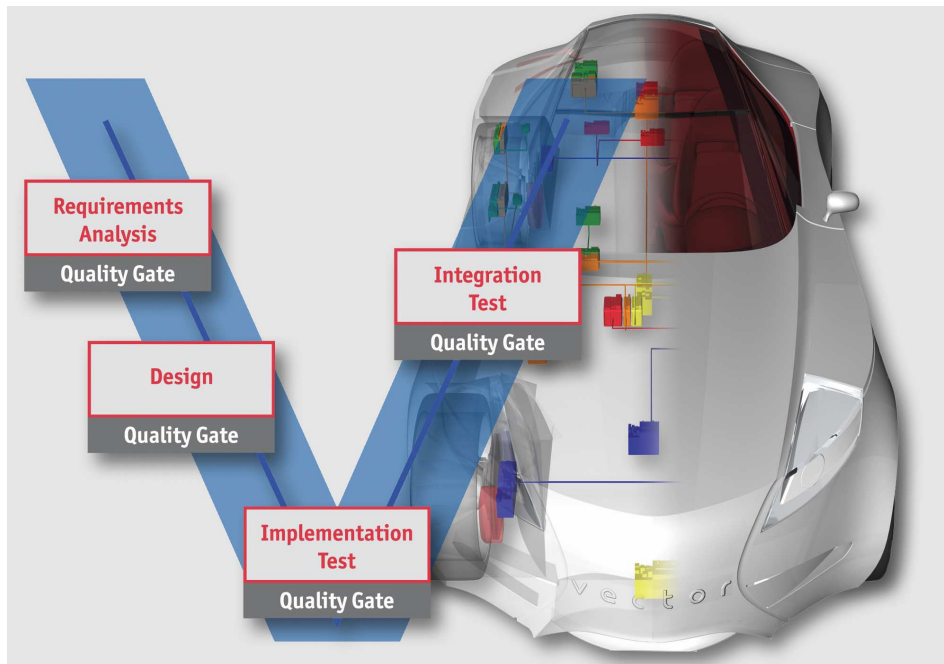


Process Improvements in Software Development

Optimizing software quality with SPICE

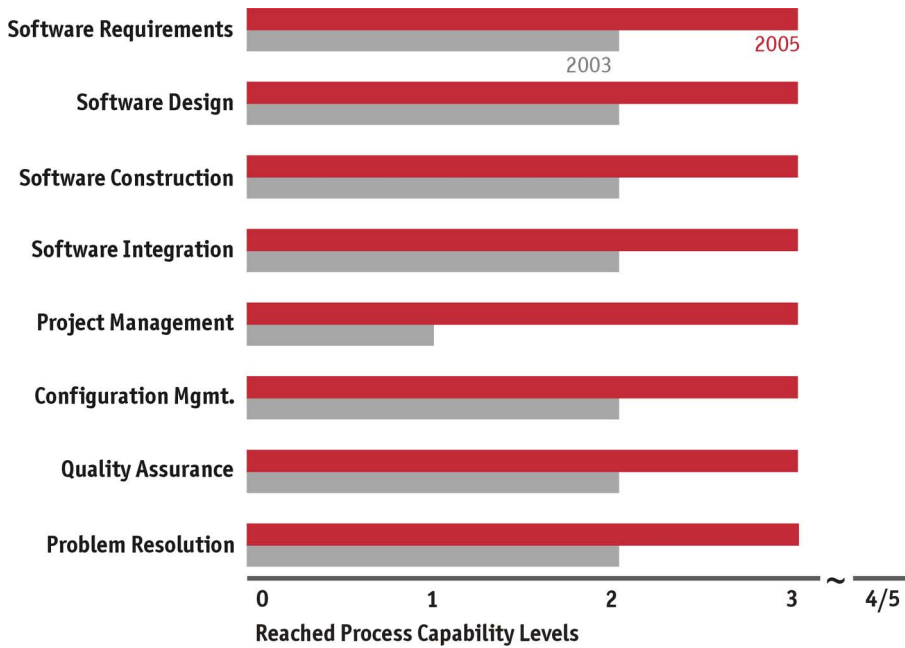
The automotive industry is making great efforts to improve software quality. In this context the "Embedded Software Components" business area of Vector Informatik GmbH successfully completed a process improvement project. One requirement of the project specification was to attain SPICE Level 3 for all relevant processes. This article summarizes the methods and experiences from practice.



Uniform standards in software development are an important goal of the automotive industry. In the HIS (Herstellerinitiative Software) interest group a number of large German automotive OEMs have come together to work out uniform standards for network software modules, process maturity determination, software testing, software tools and programming of ECUs. The foundation for monitoring processes of their ECU and software suppliers is SPICE (ISO/IEC TR 15504). In the meantime, this standard has

spawned the "Automotive SPICE" standard now used for this purpose.

In December 2001 a customer of Vector Informatik GmbH conducted an assessment in the "Embedded Software Components" (PSC) business area. One consequence was that a quality manager was hired for the organization, which at that time had 45 employees, and a project was started for improving processes. Another assessment was conducted after the project had been running for one year, in July 2003. The PSC department, which had grown to 68 employees during that period, fulfilled the requirements of SPICE Level 2 essentially. In the interest of optimal customer satisfaction, quality improvements and efficiency increases, a follow-up project was initiated for further process improvements, which was to have a running time of 2 years. This project was completed by an assessment in October 2005 in which SPICE Level 3 was successfully attained (Figure 1). At that point in time the PSC department had already grown to 82 employees. Vector is one of the first suppliers and is still one of the few suppliers of software components that applies processes at this SPICE maturity level.



[Figure 1: Assessed SPICE maturity levels of the “Embedded Software Components” business area in the years 2003 and 2005.]

Interpreting assessment results

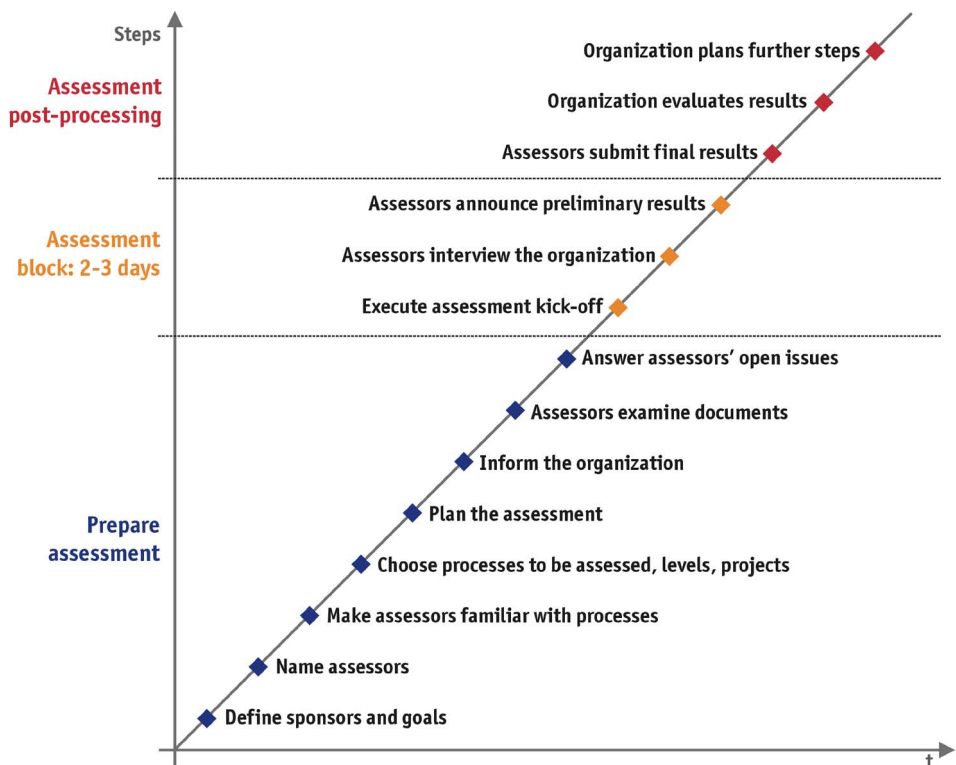
It is not possible to reduce the results of a SPICE assessment to just one level. The information on the assessed processes would be missing. Even if the scope of the HIS assessment were based on ISO/IEC TR 15504, such a simplified statement of results could be interpreted in any number of ways. Moreover, the results do not represent a certificate with an “expiration date”. Instead they should be understood as a snapshot of capabilities at the time of the assessment.

Preparing and performing assessments

To ensure that assessment results are representative of the organization, the assessed projects must cover various facets of the organization. Of course, before the assessment it is necessary to define which processes are to

be evaluated. This may require a plausible rationale for why a particular process is not relevant to the organization. It is also necessary to define the level up to which individual processes are to be audited, and this must flow into planning for the assessment. Otherwise there is a risk of assessors seeing so few criteria satisfied for a specific process on Level 2 that they do not check Level 3. This makes it difficult for the assessed organization to undertake an improvement project, since important information on process deficits may be lacking.

An assessment is an inspecting situation in which the organization must be able to present its processes to people outside of the company in a clear and comprehensible way. Repeated collaboration with the same assessors and with a well-rehearsed assessor team simplifies communication, agreement on ratings and sticking to the assessment schedule (Figure 2).



[Figure 2: Individual stages of the assessment plan.]

Project for process improvement

To successfully pass such an assessment as a "freestyle", it is first necessary to master the more important and significantly more extensive short program – the process improvement project. Attention has to be paid to the term "project" here. Such a project will usually have a lower priority than paid development contracts. Therefore it is important to practice risk management, which can clearly depict decisions and their consequences for the improvement project.

For organization purposes, the improvement project is decomposed into parts with little dependency, the processes. Organizational-specific requirements and SPICE requirements are defined for each process. A team is assigned to each process that defines the improved structured process. The teams come from all parts of the organization that will later live the process. This ensures that the processes are created according to the needs of the organization. Where it is not possible to accommodate the needs of certain sub-organizations "under one roof", modified process variants are provided.

To achieve consistency of interfaces between processes, milestones are needed for synchronization. These milestones serve to identify and resolve interface conflicts. The leader of the process improvement project participates in team meetings and identifies the project status. Within this circle, intermediate results are discussed, further tasks are defined and assigned, and the overall project is tracked.

Reasons for improving processes

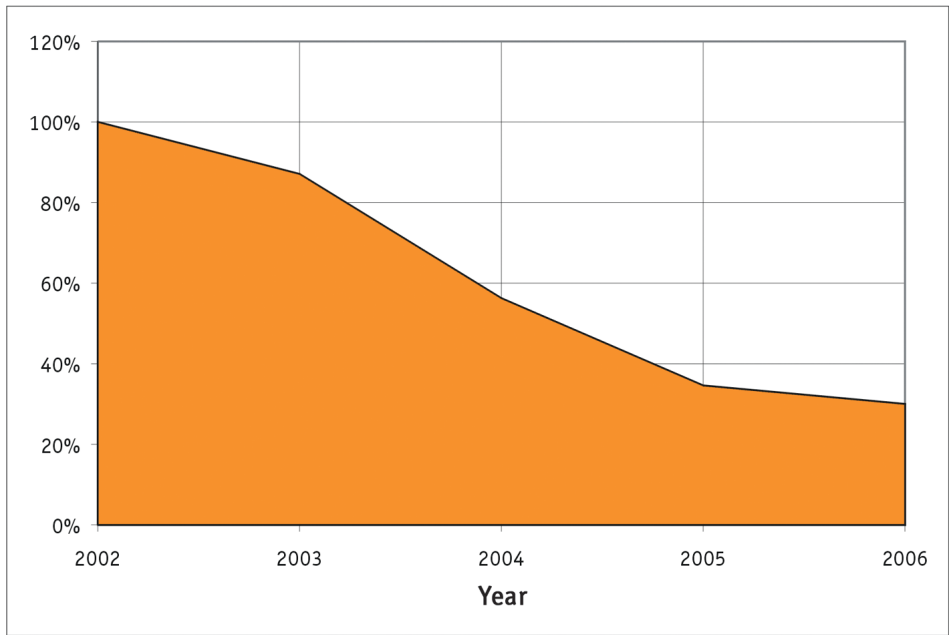
Once the processes have been defined and evaluated by pilot implementations and are ready to be lived in the organization issues arise such as: "Why are work products such as a requirements specification or a design necessary?" It is an exaggerated formulation to state that the source code is available all of the information. To determine the essence of a unit of software, even if the source code exists it is still difficult to understand as to take the ingredients and derive the method for preparing a good tasting meal. The dish can only be cooked again or modified if there is knowledge of which ingredients are used and how they should be prepared. In this analogy the ingredient list represents the requirements and the food preparation steps the process of designing a software project. What becomes clear here is that the source code does not uniquely mirror the requirements and design decisions.

One often hears the question: To what degree of precision should the requirements of a software project be described. There is no fixed rule, but some tips may be helpful here. It is always necessary to check whether all necessary aspects are covered in their full breadth, e.g. functionality, error handling, diagnostics. It is even more difficult to define the depth of the requirements description. What is important is to define this: Who should understand the description? To continue with the analogy of cooking: A hobby cook or a professional cook? Based on this definition, risk management then intervenes and prescribes "as little as possible and as much as necessary", so that the audience understands the

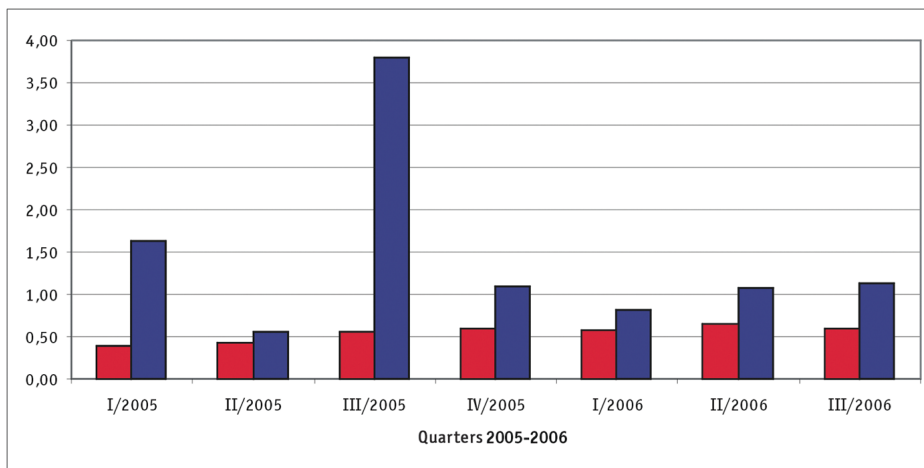
description but is not flooded with useless information. It is helpful if the recorded requirements are checked by a person who represents the language of the target group.

Measuring improvements

Metrics contribute to measuring the success of improvements. They may be used to quantify undesirable deviations from the process. The metric verifies, based on historical data, that a relevant number of deviations exist, that this makes improvement worthwhile, and that the extent of deviations is reduced by the improvements (Figure 3). Historic data and the trends derived from them (Figure 4) are well suited for measuring improvements, and they yield valuable metrics if supplemented by targets. Decisions are not made based exclusively on the results of metrics, rather they are safeguarded by more in-depth analyses such as organized discussions. This prevents factors not considered in the metric from having a negative impact on decision-making. By analogy, it does not make any sense to lump together the number of meals cooked in a cafeteria and those of a prestigious restaurant. It is best to define metrics so that they are specific to the organization, appropriate to the project, independent of specific persons and have a clear objective.



[Figure 3: Deviations from expected behavior for all product releases (Value for year 2002 serves as reference and is normalized to 100%).]



[Figure 4:
 Red: Number of deviations found per hour work on all code inspections.
 Blue: Relationship between saved error correction work to hours worked on all code inspections.]

Keys to success

Management support for the process improvement project is an absolute "must". The success of the project and the desired cultural change hinges on this. Management must actively support the project and bring it into the organization. Even after project completion, management must demand fulfillment of the new processes. This includes

promoting further improvements in processes. Under no circumstances may the organization be allowed to fall back on its old culture and methods. Otherwise the prognoses for additional improvement projects are poor. Employees might come to expect that each of these projects will just pass by, and then the "old routine" will move back in. Besides management support there are other critical factors for success (see text in box). If these factors are taken seriously, nothing stands in the way of sustained improvement in software quality.

- Management genuinely supports process introduction
- Employees define their processes
- Try out processes and then introduce them
- Demonstrate benefits of improvements
- Improve processes to become better and not just to satisfy standards
- Management makes process-conformant decisions in daily business too
- Processes should not be described in too much detail
- Monitor defined processes
- Avoid falling back into "old" business culture

[Text in box: Factors in the success of a project for process improvement.]

Revised: 3/2007

Word count: 1,572

Character count: 10,100

Figures:

Figure 1: Assessed SPICE maturity levels of the "Embedded Software Components" business area in the years 2003 and 2005.

Figure 2: Individual stages of the assessment plan.

Figure 3: Deviations from expected behavior for all product releases (Value for year 2002 serves as reference and is normalized to 100%).

Figure 4:

Red: Number of deviations found per hour work on all code inspections.

Blue: Relationship between saved error correction work to hours worked on all code inspections.

Text in box: Factors in the success of a project for process improvement.

All figures: Vector Informatik GmbH

Further links:

www.automotive-his.de

www.automotivespice.com

www.isospice.com

www.sqi.gu.edu.au/spice

www.sei.cmu.edu/cmmi

Author:



Marc Kipping (Graduate Engineer). After completing his studies in Computer Engineering he developed software, systems and processes for safety-related embedded transportation systems. Since 2002 he has been employed at Vector Informatik as a Quality Manager responsible for process improvement programs and quality assurance in the business area of "Embedded Software Components".

Tel. 0711/80670-408, Fax 0711/80670-399,

E-mail: marc.kipping@vector-informatik.de

Vector Informatik GmbH

Ingersheimer Str. 24

70499 Stuttgart

Germany

www.vector-informatik.com

Editorial contact person: Holger Heit

Tel. +49-711/80670-567, Fax. +49-711/80670-555,

E-Mail: holger.heit@vector-informatik.de