How Hardware Development
Wins with SPICE
How Hardware Development Wins with SPICE

Abstract
Driven by the success from software process improvement and growing market requirements, companies like automotive suppliers start looking at systematically improving their hardware and mechanics development processes. Unfortunately no well established process frameworks exist for hardware development. SPICE being well-known in the automotive world, comes as a natural choice for hardware development improvements. Although the SPICE engineering processes are strongly focused on software our experience shows that they can provide a good model for improvement of the hardware development process, too. Circuit schematic design, PCB layout, and component selection do not have obvious counterparts in the software world. The application of the corresponding engineering base practices for hardware and mechanics development demands specific interpretations and guidance. This paper provides hands-on examples of applications from several process improvement projects. The overall experience and the strong user feedback encourage us to further pursue SPICE for hardware development.

Copyright
Copyright 2008 by Vector Consulting Services GmbH. All rights reserved. Any distribution or copying is subject to prior written approval by Vector.

Company profile
Vector Consulting Services is your partner in achieving engineering excellence. As the leading SPICE and CMMI expert we offer tailored process improvements and assessment services around the world. The consulting branch of the globally active Vector Group, we support clients from automotive, transport, aerospace, industrial systems, ICT and other domains in optimizing technical product development. Our vision: High-quality consulting with a lasting partnership with our clients. Our understanding of consulting is a results-driven implementation of hands-on solutions for your company. Our competences include ISO 15504-SPICE, Automotive-SPICE, CMMI, Six Sigma, safety (IEC 61508, ISO 26262, etc.), security and efficiency improvement. Our experience is global, but we deliver our services locally, specific to your needs. We cover all phases of successful change: Setting objectives, supporting and driving practical implementation, and assuring sustainable results.
1 “We’re not going to improve our processes only for Software”

This is a citation of one of our customers at the beginning of a SPICE-oriented process improvement project for an organization developing electronic control units (ECU) for energy management systems. They had no doubt that when we started defining and using SPICE-conformant processes we would encompass everything needed for an ECU: software, hardware and mechanics. Let’s see how we used SPICE in such environment.

In today’s modern automobiles a high degree of complexity is implemented in the electronic components. In medium-class automobiles typically around 50 electronic control units (ECU) are equipped, most with a specific microprocessor, many with micromechanical sensors and all interconnected in a huge network. This growing product complexity leads to growing development organizations and to the need of a controlled development process, as experienced in other industries like aerospace or telecommunications.

Many car manufacturers follow this experience and require a defined and controlled development process in place at their suppliers of ECUs being convinced that a high product quality can only be achieved by a high process quality. The ISO 15504 [1] and Automotive SPICE [2] model are broadly used as a common reference for evaluating the capability of the development process. The focus was originally put on software, probably because the functional impact of this new domain grew and grows massively contrary to the classical domains hardware and mechanics. The majority of last year’s car call-backs are caused by electronics – and the number is still increasing.

When suppliers start implementing and using defined processes for their software development, they quickly find that the software domain is closely related – and often interdependent – to the hardware and mechanics domains. They consequently broaden the process improvement projects to include all their engineering domains for a comprehensive process model. However, in many companies the organizational structure separates firmly the software development from the hardware development. As a result they need some common synchronization points in the developments cycles of the software, hardware, and mechanics development. Some of them also detect that their long experience in hardware development is not as systematic and transparent as they believed it to be. When introducing improved development processes for software, the need for aligning and improving development processes for hardware and for mechanics arises. The SPICE-model, however, does not yet give much orientation for these domains – historically the “S” stands for “Software” and only lately this is being replaced by “System”.

A growing number of automotive suppliers recently asked us to support their process improvement projects encompassing a mixture of software and hardware development. Our experience clearly shows that SPICE can be successfully used to also improve the development processes for hardware and mechanics. We established for example capability levels 2 and 3 in the hardware development division of a supplier for engine management ECUs. Other projects included process definition and roll-out in development divisions for energy management systems. The net win results from building upon a common model of best practices. The interpretation of the best practices is sometimes not obvious since SPICE is oriented towards software. An appro-
appropriate interpretation requires understanding equally well the principles of hardware and software development. We could achieve major advantages from the SPICE model applying it to the hardware and mechanics within an ECU development.

2 Interpretation of Base Practices

2.1 The Easy Parts

The vast majority of the SPICE processes are not focused on software development. Although SPICE has its origins in the software domain it carries a lot of valuable practices to broaden its application outside the software world. Only five of the ten engineering processes carry the name software in them. All the support and management practices are – by their very nature – independent of the processes they have to manage or support. They can be equally well applied to any development activity in any engineering domain. Figure 1 provides an overview of the processes as they are listed in Automotive SPICE, highlighting (italic font) the focus of the HIS-group (Hersteller Initiative Software, an interest group of German car manufacturers) for the most widely examined processes and the software related processes (bold font) within that focus.

Project management has to plan, monitor, and control all activities anyway regardless if someone has to program a given function or if someone has to sketch a wiring schematic. The same is true for risk management, change request management, problem resolution management, or supplier monitoring. The methods applied in quality assurance might differ, but the major ideas remain.

<table>
<thead>
<tr>
<th>Acquisition Process Group (ACQ)</th>
<th>Engineering Process Group (ENG)</th>
<th>Supporting Process Group (SUP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACQ.3 Contract agreement</td>
<td>ENG.1 Requirements elicitation</td>
<td>SUP.1 Quality assurance</td>
</tr>
<tr>
<td>ACQ.4 Supplier monitoring</td>
<td>ENG.2 System requirements analysis</td>
<td>SUP.2 Verification</td>
</tr>
<tr>
<td>ACQ.11 Technical requirements</td>
<td>ENG.3 System architectural design</td>
<td>SUP.4 Joint review</td>
</tr>
<tr>
<td>ACQ.12 Legal and administrative requirements</td>
<td>ENG.4 Software requirements analysis</td>
<td>SUP.7 Documentation</td>
</tr>
<tr>
<td>ACQ.13 Project requirements</td>
<td>ENG.5 Software design</td>
<td>SUP.8 Configuration management</td>
</tr>
<tr>
<td>ACQ.14 Request for proposals</td>
<td>ENG.6 Software construction</td>
<td>SUP.9 Problem resolution management</td>
</tr>
<tr>
<td>ACQ.15 Supplier qualification</td>
<td>ENG.7 Software integration</td>
<td>SUP.10 Change request</td>
</tr>
<tr>
<td></td>
<td>ENG.8 Software testing</td>
<td>management</td>
</tr>
<tr>
<td></td>
<td>ENG.9 System integration</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ENG.10 System testing</td>
<td></td>
</tr>
</tbody>
</table>

Legend: *Italic* = HIS-scope, *Bold* = software centric processes

Figure 1: Overview of the Automotive SPICE [2] processes with the software centric processes ENG.4 – ENG.8.
We already have to handle some challenges in configuration management when leaving the pure software world of a single system for version control and trying to apply the same criteria and processes at the management level. Many development groups omit the system and management level when considering configuration management. They don’t think about the general management documents or documents stored in a separate tool like specifications in a requirements management tool. A simple typical solution is to define a policy and corresponding naming conventions to parallelize to version control of software files and management files. Including other domains in the overall configuration management means to include yet another set of document types and other tools like CAD or PDM systems into the same policy and naming convention.

The system level engineering processes ENG.2, ENG.3, ENG.9, and ENG.10 do not require special interpretation of the standard. Leaving the typical discussions about the meaning of ‘system’ aside we can treat the requirements and tests still independently of a hardware or software solution. In our experience it comes sometimes surprising for the hardware development domains, however, that a structured and systematic approach to requirements management and system integration and system tests is necessary. The process improvement projects make all involved parties think anew about their traditional way of working. As a result the companies often developed new overall concepts for requirements management and for integration and test management.

For ENG.4 Software Requirements Analysis we see a quite simple and straightforward mapping of the software specific base practices to corresponding hardware specific practices. Simply read hardware or mechanics instead of software and the practices can be applied directly. That is, we can establish and analyze hardware or mechanics or any other subsystem requirements specification as we can treat software requirements.

In the automotive world of electronic control units, however, you will rarely find separate documents for each of the three engineering domains. Often they are maintained as one specification, maybe separated by headings or a corresponding attribute in a requirements management tool. This challenges the way specification and design documents are managed and how traceability between requirements and across different product levels is achieved and maintained once software and hardware design go their own paths.

It proved beneficial to clearly separate the software, hardware, and mechanical requirements in the requirements documents when involving the corresponding specialists (mechanical engineer, electronics engineer) in reviews or communicating ‘their’ specific requirements to them. We used tags and attributes marking a requirement to one or more of these domains. Afterwards they are connected to system and qualification test cases to ensure correct functionality and performance on the combined software and hardware level.
2.2 Challenges in Finding Equivalent Concepts

2.2.1 ENG.5 Software design and ENG.6 Software construction

The concept of a software architecture (ENG.5.BP1) translates into a top level block diagram identifying the major electronic modules of a circuit. In bigger ECUs the top level blocks are often refined in additional block diagrams. When looking at circuit schematics the customer discussed if the schematics are still part of the architecture. It makes most sense, however, to compare them to the software detailed design (ENG.5.BP6). For Software construction (ENG.6) we found other concepts in the hardware world like early and final printed circuit board (PCB) layout, design calculations, and the component selection as the most detailed level of design. We consequently use potentially different levels of block diagrams and circuit schematics. All the design has to be verified anyway (ENG.5.BP8, ENG.6.BP4) regardless of how you name it – and that’s what is important: simply by transferring the early verification principle of the software world we achieve an obvious benefit in hardware development.

One of the specific Automotive SPICE base practices is “Describe dynamic behaviour” (ENG.5.BP4). When analyzing requirements we came to the conclusion to consider the time behaviour as dynamic behaviour of the hardware, for example frequencies, modulations, signal delays, filters, control loops, etc. These are not new concepts for electronic engineers, rather daily business. The discussion showed, however, the difficulties that arise when finding matching concepts between software and hardware development.

Another specific Automotive SPICE base practice is “Define resource consumption objectives” (ENG.5.BP5). The idea in the software domain is to explicitly make programmers aware about a typical and often problematical set of non-functional requirements. In the hardware development we currently check for the non-functional requirements EMC safety (electromagnetic compatibility), heat dissipation, start-up current, power consumption in sleep mode, and product price. They were valuable for making the hardware developers aware about pricing, heat and EMC problems right from the beginning of the development. We could define detailed objectives for these non-functional requirements that also helped in deriving corresponding test cases. The additional test cases that are developed early in the project are the actual benefit of applying this software base practice to the hardware development.

When looking at the “Software construction” (ENG.6) process we had to discuss the term ‘unit’ in hardware development. It was considered to call a unit a single resistor, capacitor or diode with specific properties or some combination of elements (e.g. resistor/capacitor pair or low/high pass filter). But single electronic elements are not something that is sensibly verified separately in the sense of unit tests, they obey mere physical laws and their qualification focuses on tolerances and robustness. A good interpretation of a ‘HW unit’ is instead a ‘circuit module’, i.e. a part of the circuit fulfilling a delimited functionality, e.g. ‘voltage stabilization’. This unit-level usually corresponds to the blocks of the lowest block diagram. Also the ASICs contained in many ECUs are units from the point of view of the ECU. Due to their inner complexity they are often separate subprojects with all the necessary management and engineering processes.
In this context it was interesting to see that unit-tests in fact have an analogy in hardware development. With the help of additional ‘driver circuits’ (that are in fact comparable to software test stubs) the single circuit modules could be verified separately.

SPICE and especially Automotive SPICE require a very high degree of traceability from the requirements to the actual implementation and to the corresponding tests. We took a rather pragmatic approach for hardware development in that we apply the same method of traceability among the requirements and between requirements and tests as for software development. Traceability down to the hardware design level is achieved by a clear naming and numbering scheme of all electronic components and signals that is to be used in all documents. The bills of material are not only helpful for configuration management, but also help to identify specific components and their values in a schematic and in the layout.

How do you verify hardware design? We selected verification methods for block diagrams (reviews), wiring schematics (reviews, circuit simulations), design calculation (recalculations, circuit simulations), PCB layout (review, prototypes; EMC simulations were considered too expensive), and component selection (failure tree, control calculations). They are completed by breadboard mock-ups (called “hedgehog wiring”) to check circuit design. The agreed naming rules for the electronic components and for signals for traceability are verified by reviews.

Although we did not find a generic one-to-one mapping of the software design levels architecture, detailed design, and software construction to respective hardware design levels, we found within projects always handles to achieve such a recursive and stepwise approach to design. The benefits of applying SPICE and its principles to hardware development are manifold:

- different levels of design allow to zoom into specific issues while still maintaining an overview,
- all results of design activities have to be verified resulting in higher product quality,
- naming schemes for electronic components and signals permit a clearer communication.

2.2.2 ENG.7 Software integration test and ENG.8 Software testing

The purpose of an integration step is to bring together all the modules and components that had been identified in the design phase. Verifying interfaces and correct interaction between the modules can be done for hardware modules as it is required for software components and software units. In a new development the block diagrams, potentially at multiple levels, lead the required integration. Whereas in smaller projects only the complete PCB could be tested, more complex ECUs require explicit stepwise breadboard wirings or even special stripped down test-PCBs and corresponding tests between the separately designed modules. The PCBs often contain special test pins to ‘debug’ signals on specific points of the circuit.

Other cases of hardware integration are the integration of variants or the integration of new or modified functionality over time (e.g. direct wiring to sensors later replaced by dedicated LIN-controllers). These different ways of integration require the same
kind of strategic planning as corresponding integration strategies in the software world.

Pure hardware testing on a hardware-level is almost impossible as all reasonable projects include a microcontroller that only works if some software is provided. In some circuits we found that special test software has to be developed for electronic tests, which artificially generates stress situations that cannot be generated by the application software (e.g. processor overload, defined read/write of all registers, generation of high frequency patterns, etc.). These tests are interpreted as hardware tests even though some software is incorporated.

A central concept of SPICE is to verify the software before passing it on to system integration and system test. When talking with system testers, many could tell some ugly stories about immature software being handed to them on Friday afternoon. The logical consequence in the hardware world is to require the same explicit releases for the PCB. This is a concept that is not always ingrained in the hardware development departments. But it showed a similar benefit as we know it from the software world.

### 2.2.3 Generic Practices

The generic practices for the process attributes Performance Management (PA 2.1) and Work Product Management (PA 2.2) are mostly addressed by a systematic application of project management, quality assurance and configuration management. No special interpretation of these processes is necessary when managing hardware development as had been explained above.

Tools are an essential part in the hardware development. These tools for hardware design (schematics and circuit design programs, simulators, PCB track router) and test (automated test environments, HiL, etc.) have to be listed and maintained as process resources for GP 2.1.5. The expected quality of the work products GP 2.2.1 is partially checked automatically in some of the tools, e.g. in an autorouter. In all other cases we developed classical checklists (see section 3.3).

### 3 Experiences and User Feed-Back

We will show in this section concrete solutions to implement Automotive SPICE within an ECU development organization. Our customer – a traditional supplier of automotive wiring systems – saw a new market segment in energy management systems and created a division for development of new, innovative products. These systems are needed to optimize the energy flow in today’s automobiles, where huge amounts of energy are needed for short periods of time at some specific place of the automobile, e.g. during the frequent engine starts of a car equipped with start-stop-automatic.

Our example customer’s small organization had it’s origins in hardware development and showed up with experienced people in this domain. They were facing the challenge to build up similar in-house experience for software development. And they knew that good processes are the basis for efficient product development and that - by the way - these processes help to fulfill the requirements of car manufacturers who require SPICE level 2 or better for any development of components containing software.
3.1 One integrated process for the entire ECU development

When managing a process improvement project, one usually focuses on getting a leading coalition, defining realistic targets, producing quick wins, ensuring stakeholder involvement and so on. In this project we also focused from the beginning on defining and introducing pragmatic and integrated processes. This helped us to win quick acceptance and to settle the improved processes firmly in the organization. Figure 2 shows an integrated process landscape with three equal, parallel development branches for hardware, software and mechanics development in the middle.

![Integrated Process Landscape](image)

**Figure 2: The process overview**

3.2 A detailed view: Process definition for hardware development

When defining our customers’ processes for hardware and mechanics development – corresponding to ENG.5 to ENG.8 – we applied the same principles as for software: explicit design phase, early fault detection, thorough quality assurance before any transition of responsibility, neat documentation facilitating traceability and communication.

In the left part of the “V” in Figure 2 a process called “System design” is described. This comprises not only the System architectural design (ENG.3), but also the design of hardware, software and mechanics (ENG.5). We found it useful to keep all design information together in one document, having one chapter for each of the design domains (system, hardware, software, mechanics). The detailed hardware design comprises an architecture, definition of functional blocks, interfaces, EMC safety, heat generation and dissipation, selection of core components and design alternatives.

Traceability to the system level is realized by cross references inside the document. In the other direction from the specifications to the design, the naming convention of parts in the wiring schematic and bills of material allows a mapping to the functional blocks, permitting enough traceability to the single electronics component without too much overhead.
The advantage of this approach is that one single document gives a very good overview over the product and that an acceptable level of traceability could be easily realized without expensive tools. The disadvantage of concurrent editing of different specialists was not (yet) problematic because the organization is still small and some developer roles are still unified in one person (e.g. one developer does both software and hardware development).

The single document “System Design” is the main input for the implementation process, depicted in Figure 3. The figure shows a very simple description of a development process, covering all the needs of the small organization. The three work products wiring schematic, PCB layout and bill of material were - obviously - already in use before the process improvement started. What was added were several verification steps listed in Figure 4. Wiring schematic and PCB layout are verified by reviews (see next chapter for details). Hardware tests are explicitly performed before the PCB is released.

Figure 3: Document flow of hardware implementation

<table>
<thead>
<tr>
<th># Process step</th>
<th>PL</th>
<th>HW</th>
<th>ST</th>
<th>QA</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Create wiring schematic</td>
<td>I</td>
<td>R</td>
<td></td>
<td></td>
<td>System design</td>
<td>Wiring schematic</td>
</tr>
<tr>
<td>2 Verify wiring schematic</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>Wiring schematic</td>
<td>QA-report, simulations</td>
</tr>
<tr>
<td>3 Create PCB layout</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>Wiring schematic, System design</td>
<td>PCB layout, PCB-specification</td>
</tr>
<tr>
<td>4 Verify PCB layout</td>
<td>I</td>
<td>R</td>
<td></td>
<td></td>
<td>PCB layout, Designrules</td>
<td>QA-report</td>
</tr>
<tr>
<td>5 Create bill of material</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>Wiring schematic</td>
<td>Bill of material</td>
</tr>
<tr>
<td>6 Create PCB and parts assembly</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>PCB layout, Bill of material, PCB-specification</td>
<td>PCB</td>
</tr>
<tr>
<td>7 Execute electronic tests</td>
<td>R</td>
<td></td>
<td></td>
<td></td>
<td>PCB, testplan</td>
<td>Testplan, test metrics</td>
</tr>
<tr>
<td>8 Release PCB</td>
<td>I</td>
<td>R</td>
<td></td>
<td></td>
<td>Test metrics</td>
<td>QA-report</td>
</tr>
</tbody>
</table>

Figure 4: Document flow and process steps of hardware implementation. (Legend: PL = Project leader, HW = Hardware developer, ST = Tester, QA = Quality assurance, R = Responsible, I = Information)

The biggest benefit and quick win for product quality was the introduction of the verification steps (see process steps #2, #4, #7 and #8 in Figure 4), i.e. the application of PA 2.2. During the reviews in the first pilot project there could be detected half a dozen critical or important findings. In another pilot project one originally planned PCB layout iteration could be saved.
3.3 A detailed view: Quality assurance for hardware development

In the detailed view on the process in the previous section we saw that quality assurance (QA) activities are foreseen at neuralgic steps of the process:

1. Review of the design document
2. Review of the wiring schematic
3. Review of the PCB layout
4. Release of the PCB

All of these QA activities follow the same principle: They use a predefined set of questions which is checked and documented in a peer-review manner. Our experience is that investing a good effort in finding and formulating the right questions resulted in a very efficient review process. The effort per review session could be limited to around 1-2h, which for the engineers was an acceptable timeframe.

The ‘right’ questions were mostly deduced from the experience of our customers’ specialists – we asked them “where did you have problems in the past?” or “imagine that you receive the wiring schematic from your foreign colleague – where would you look at before you accept it?” On the other hand we could leave out questions which were already checked by the engineering tools – this is important to avoid overloaded checklists with too many questions. Figure 5 shows an excerpt of a checklist for a wiring schematic.

<table>
<thead>
<tr>
<th>Question</th>
<th>Critical aspects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does the wiring schematic correspond to the hardware design?</td>
<td>Disturbance sources identified (e.g. for clock, noise)?</td>
</tr>
<tr>
<td></td>
<td>Crosstalk (ohmic, capacitive, thermal)?</td>
</tr>
<tr>
<td></td>
<td>Circuit resistivity or maximum length defined?</td>
</tr>
<tr>
<td></td>
<td>Critical nodes (e.g. high resistive, highly sensitive)?</td>
</tr>
<tr>
<td></td>
<td>Thermal sensibilities?</td>
</tr>
<tr>
<td>Documentation</td>
<td></td>
</tr>
<tr>
<td>Is the title block filled correctly?</td>
<td></td>
</tr>
<tr>
<td>Is the naming systematic?</td>
<td></td>
</tr>
<tr>
<td>Layout requirements</td>
<td></td>
</tr>
<tr>
<td>Power wiring defined?</td>
<td></td>
</tr>
<tr>
<td>Layer of wires defined?</td>
<td></td>
</tr>
<tr>
<td>Critical distances defined?</td>
<td></td>
</tr>
<tr>
<td>Dummy elements foreseen?</td>
<td></td>
</tr>
<tr>
<td>Cooling pads defined?</td>
<td></td>
</tr>
</tbody>
</table>

Figure 5: Review checklist for a wiring schematic (excerpt)

The SPICE idea to verify all relevant work products according to specified criteria was new in the hardware development of our customer. The critical aspect is the ‘relevance’ of each work product. In discussions with the experts we scrutinized for each of the main work products (wiring schematic, bill of material, PCB layout) whether foreseeing a review is useful or not. In their experience there was no need for a review of the bill of material, as it was automatically generated and not problematic in the past.
It is good practice to perform a QA activity when the responsibility between two process steps changes to another role. In our previously depicted hardware development process this happens at the end (process step #8 in Figure 4) – the following integration and testing steps are under the responsibility of the role ‘tester’. A formal release of the PCB should assure that Exit/Entry-criteria are fulfilled. Figure 6 shows that this formal step can be very short, because it is limited to check for preceding reviews, the completion of derived measures and a neat documentation of faults and changes. Having checked this, it’s the right time to create a baseline consisting of exactly those documents which just have been inspected.

This ‘formal’ release step during piloting in the first projects quickly evolved to an active ‘handover’ step between the hardware developer and the tester, thus improving the communication and the understanding for the product. The creation of a baseline was an easy task even though it was a new experience for the hardware developers.

<table>
<thead>
<tr>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Has the wiring schematic been reviewed? Have all measures of the review been implemented in time? (QS-report, ToDo-list)</td>
</tr>
<tr>
<td>Has the PCB layout been reviewed? Have all measures of the review been implemented in time? (QS-report, ToDo-list)</td>
</tr>
<tr>
<td>Have all planned electric tests been passed successfully? (test specification)</td>
</tr>
<tr>
<td>Have all faults detected during the electric tests been documented? (change list)</td>
</tr>
<tr>
<td>Have all changes applied to the PCB during the electronic tests been documented? (change list)</td>
</tr>
<tr>
<td>Is the baseline created consisting of: wiring schematic, PCB layout, bill of material, QS-report, ToDo-list, change list and test specification?</td>
</tr>
</tbody>
</table>

Figure 6: Checklist for the PCB release

### 3.4 User feedback

Our customer liked the new, comprehensive development processes. We found the department leader proudly presenting it to neighbored departments and earning respect during an internal process audit performed by an auditor known to be very critical. Another improvement initiative running at that time at the level of the company’s central project office was highly interested in taking over some good solutions – and the department leader gained even more reputation, drawing him a broad smile on the face. He also used this argument when convincing his boss to continue the effort to further maintain and optimize the processes.

Concretely, the department leader and the developers see a clear advantage to check for defects early in the ECU development process and appreciate the practicability of the checklists. They will, however, have to maintain these, adjusting them to changes of defect root causes. A small process management organization has been set up and was prepared for this mid- and long-term task.

The need for an explicit design phase preceding the ‘classical’ development required some change of mind – especially during acquisition activities. The involvement of software and hardware specialists in the early requirements analysis and system design required some effort, being sometimes painful. The spending of this effort however resulted in product proposals with better reusability, higher maturity and – last but not least - which are well documented.
4 Summary

Our experience clearly shows that the collection of good engineering practices in SPICE can be successfully applied to the hardware development processes. Although we did not always find a one to one mapping, e.g. for unit tests or specific design documents, we could use the main ideas of SPICE with very good results. Mainly the ideas of a staged design, a systematic verification of all design steps and the improved test cases that are linked to requirements led to the improved efficiency and project transparency. We got encouraging feedback from the electronic engineers who started the improvement project sometimes reluctantly.

Considering the described difficulties of mapping the SPICE base practices to the hardware development it can be assumed to be very difficult to use the processes ENG.4 – ENG.8 generally for process assessments of hardware development. One of the advantages of SPICE over other process models is the detailed requirement for specific work products and their characteristics in the software domain. The specific requirements cannot easily be interpreted for hardware development. It is not obvious for assessors and assessed development groups what the corresponding work products in hardware development should look like (e.g. unit tests or dynamic behaviour).

It is necessary to develop a set of comparable best practices including typical work products that are accepted as an industry standard for hardware development in similar way that ISO 12207 is accepted for software development. The experiences we gained in the mentioned projects can be used as a first contribution. The upcoming part 6 of ISO 15504 that addresses the processes of systems development as described in ISO 15288 has yet to show its practical usability in such an environment. Unfortunately it was not yet available at the time of our past projects.

We are facing the challenge to map the concepts of ENG.4 – ENG.8 to yet other engineering domains like mechanics and hydraulics development (e.g. for car body or transmission systems, etc.) in several customer projects we are currently supporting. The transfer of the SPICE ideas requires quite some discussions and substantial experience in the corresponding engineering domains to find a common and acceptable interpretation among all involved parties.

Bibliography

2 Automotive SPICE Process Assessment Model (V2.3), The SPICE User Group, 2007.
Biographies

Dr. Rolf Ebert is Senior Consultant at Vector Consulting Services GmbH and SPICE Competent Assessor (iNTACS certification). He is helping clients in analyzing and improving their product development processes and methods for comprehensive and sustainable results. As a manager of big and international projects in industry automation and automotive he gained broad practical experience in several engineering domains like computer science, mechanics, fluid dynamics, energy engineering, and others.

Thomas Wunderlich is Senior Consultant at Vector Consulting Services GmbH and SPICE Competent Assessor (iNTACS certification). He is helping clients in process improvement projects with strategical consulting, process definition, project coaching, performing process audits and acting as SW quality responsible. He gained experience applying SW-CMM at Alcatel for many years, lead CMMI and SPICE process improvement projects for suppliers in the automotive industry and performed several assessments.

Affiliation:

Vector Consulting Services GmbH, Ingersheimer Straße 24, D-70499 Stuttgart, Germany. URL: www.vector-consulting.de
Ihre Ansprechpartner

Deutschland und alle Länder, soweit nicht unten genannt
Vector Consulting Services GmbH
Vector Informatik GmbH
Ingersheimer Str. 24
70499 Stuttgart
DEUTSCHLAND
Tel.: +49 711 80670-0
Fax: +49 711 806070-444

Frankreich, Belgien, Luxemburg
Vector France
168, Boulevard Camélinat
92240 Malakoff
FRANKREICH
Tel.: +33 1 4231 4000
Fax: +33 1 4231 4009

Schweden, Dänemark, Norwegen, Finnland, Island
VecScan AB
Theres Svenssons Gata 9
41755 Göteborg
SCHWEDEN
Tel.: +46 31 764 7600
Fax: +46 31 764 7619

Großbritannien
Vector GB Limited
Rhodium, Central Boulevard
Blythe Valley Park
Solihiull, Birmingham
West Midlands, B90 8AS
UNITED KINGDOM
Tel.: +44 (0)7851 435126

USA, Canada, Mexiko
Vector CANtech, Inc.
Suite 550
39500 Orchard Hill Place
Novi, MI 48375
USA
Tel.: +1 248 449 9290
Fax: +1 248 449 9704

Japan
Vector Japan Co., Ltd.
Seafort Square Center Bld. 18F
2-3-12 Higashi-shinagawa,
Shinagawa-ku
140-0002 Tokyo
JAPAN
Tel.: +81 3 5769 7800
Fax: +81 3 5769 6975

Korea
Vector Korea IT Inc.
Daerung Post Tower III, 508
182-4 Guro-dong, Guro-gu
Seoul, 152-790
REPUBLIK KOREA
Tel.: +82 2 2028 0600
Fax.: +82 2 2028 0604

www.vector-consulting.de