Maximilian Wilfinger, BSc

Development Process Optimization of Mechatronic Systems in Automotive Applications

MASTER THESIS

to achieve the university degree of

Diplom-Ingenieur

Master degree program:

Production Science and Management

submitted to

Graz University of Technology

Faculty of Mechanical Engineering and Economic Sciences

Institute of Automotive Engineering

Supervisor

Assoc. Prof. Dipl.-Ing. Dr. techn. Mario Hirz

Graz, December 2017

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen / Hilfsmittel nicht benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz, am …… ……..

(Unterschrift)

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material, which has been quoted either literally or by content from the used sources.

…… ……..

date (signature)

Acknowledgement

This thesis was written at the Institute of Automotive Engineering of Graz University of Technology in close cooperation with the Department of Engineering Project Management at Magna Powertrain.

My supervisor, Assoc. Prof. Dipl.-Ing. Dr. techn. Mario Hirz, deserves special thanks for the scientific support of my work at the Institute of Automotive Engineering. His door was always open whenever I ran into a trouble spot or had a question about my research or writing.

I would also like to thank Dipl.-Ing. Irenka Mandic of the EPO Department at Magna Powertrain for the integration of the thesis into her team and the professional support I have always received.

Furthermore, I would also like to thank Dipl.-Ing. Helmut Brunner for his support at the Institute of Automotive Engineering and Dipl.-Ing. Dr. techn. Christian Kreiner for his excellent previous work on SPICE at the Institute for Technical Informatics.

Of no less importance were my fellow students of the innermost core PSM group, who have made a significant contribution to my positive and sometimes wistful reflection on the time at TU Graz. Thank you for your professional and non-professional support.

I would like to express my deepest gratitude to the people who have raised me and brought me up in many strenuous years. They have made it possible for me to pursue my personal and professional development through their versatile and generous support to this day. This accomplishment would not have been possible without them.
Thank you!

Maximilian Wilfinger

Kurzfassung

Steigende Kundenanforderungen, globaler Wettbewerb und Kostendruck erfordern reife Produkte zum Serienanlauf und robuste Entwicklungsprozesse entlang der gesamten Zulieferer-Kette mit einer proaktiven Ausrichtung zur Fehlervermeidung. Jeder Prozess muss präventiv so robust und stabil gestaltet werden, dass jede Entwicklungsaktivität, jedes Bauteil und damit auch jedes Fahrzeug, den hohen Qualitätsanforderungen entspricht. Nur so kann es gelingen das Null-Fehler Ziel zu erreichen. Diese Masterthesis behandelt die Optimierung bestehender Entwicklungsprozesse von komplexen mechatronischen Systemen in automobilen Anwendungen. Zunächst wird der aktuelle Stand der Technik im Bereich mechatronischer Systeme, internationaler Standards und gängiger Entwicklungsprozesse, sowie deren Methodik als Grundlage ausführlich dargestellt. Der Fokus liegt dabei auf der Embedded-Software Entwicklung elektronischer Steuergeräte, von der Konzeptanforderung bis zur Serienreife. Ziel der Masterarbeit ist die Entwicklung eines umfassenden Prozessmodells, welches nach der Analyse bestehender Prozesse, verschiedenster Kundenanforderungen sowie internationaler Normen, Abweichungen und Potentiale zur Prozessoptimierung aufzeigt. Grundlage der Vergleichsanalyse ist der VDA-Standard Automotive SPICE, welcher die Basis zur Optimierung vorhandener Entwicklungsprozesse darstellt. Das Tool zur Prozessanalyse soll modular so konzipiert werden, dass es das Potential hat, zukünftig um weitere Standards und herstellerspezifische Lastenhefte erweitert zu werden. Im Verlauf dieser Masterarbeit sollen, nach der Konzeptentwicklung einer funktionierenden Umgebung zur zielgerechten Darstellung der Abweichungen aller analysierten Quellen, jeder zu bewertende Prozess und dessen Basispraktiken dem genormten Pendant zugeordnet werden. Eine Herausforderung stellt dabei die Zuordnung und Verknüpfung der verschiedenen Quellen dar, bei denen es sich um komplex verschachtelte n:m Beziehungen handeln kann. Diese Umgebung soll für definierte Nutzergruppen, z.B. Prozesseigentümer, Management und Qualitätssicherung, die gezielte Analyse der Abweichungen und Bedarfe zur Prozessoptimierung ermöglichen. Erstellung von Angeboten, Anforderungsanalysen, Bewertungskriterien und eine Risikoidentifikation sind dadurch in einer frühen Phase der Produktentwicklung ebenfalls einfacher und zielbewusst realisierbar.

Abstract

Increasing customer requirements, global competition and cost pressure require mature products that are ready for series production and robust development processes along the entire supply chain with a proactive focus to prevent failures. Every process must be designed to be so robust and stable that every development activity, every component and therefore every vehicle meets the high quality requirements. This is the only way to achieve the zero-defect target. This master thesis deals with the optimization of existing development processes of complex mechatronic systems in automotive applications. At the beginning, the current state of the art in the field of mechatronic systems, international standards and development processes as well as their methodologies are presented in detail as a basis. The focus is on the embedded software development of electronic control units, from conceptual requirement definition to series production readiness. The objective of this thesis is the development of a comprehensive process model, which shows deviations and potentials for process optimization based on the analysis of existing processes, various customer requirements as well as international standards. The basis for the comparative analysis is the VDA Standard Automotive SPICE, which represents the foundation for optimizing existing development processes. This tool for the process analysis is to be modularly designed in such a way that it has the potential to be extended in the future by further standards and manufacturer-specific specifications. In the course of this master thesis, following the concept development of a functioning framework for target-oriented representation of the deviations of all analyzed sources, each process and its base practices is to be mapped to the corresponding process of the standards. One of the challenges is the mapping and correlation of the different sources, which may be complex n:m relationships. This framework is intended to allow the targeted analysis of deviations and requirements for process optimization for defined user groups, e.g. process owners, management and quality assurance. The quoting process, requirement analyses, evaluation criteria and risk identification are thus also easier and can be systematically implemented at an early stage of product development.

Table of Contents

1 Introduction

1.1 Relevance of the Topic

Automobile vehicle manufacturers – or the brand as referred to *original equipment manufacturer* (OEM) – are now transforming their vehicles from mechanical to advanced electronically controlled systems. This makes the software, with an increasing need for complexity, a major vehicle component, as it is part of embedded systems that electronically control a variety of vehicle functions. In the last generations of vehicles, the number of electronic control units and processors has risen significantly, both among low-cost and luxury models. These innovations represent safety-critical functions and therefore have to be precise and of high quality. Customers and OEMs also place high requirements on vehicle quality and the reliability of their suppliers. In order to be able to develop and produce mature products, the underlying processes must also meet high quality standards. These development processes have to be closely coordinated with the customer (i.e. OEM) in order to ensure that the product meets the customer requirements. Suppliers must be able to implement these processes according to these specifications. There are standardized guidelines and customer-specific specifications, which must be taken into account in order to acquire and develop successful customer projects.

1.2 Objectives

The objective of this master thesis is the creation of a comprehensive process model, which includes the processes for the development of mechatronic systems in automotive applications. The focus is on embedded software development according to the VDA standard Automotive SPICE. Within the framework of the thesis, a tool concept for data management and analysis is to be developed, which incorporates the process model. In addition to the still valid Automotive SPICE 2.5 standard, the new version Automotive SPICE 3.0 also comes into play. Automotive SPICE guidelines and several customer requirements are likewise included in the process model. This is to be modularly designed in such a way that it can be extended in the future by even more process scopes and customer requirements. With the help of this tool it should be possible to allocate and compare different requirements of the various sources. Based on the analysis of the deviations, potential for process optimization can then be derived in order

to cover a wide range of requirements of the automotive market by means of internal processes. This process model allows a better understanding of one's own capabilities with regard to requirements and development processes and is intended to contribute to the optimization of internal processes. This model will also be used to identify critical issues and requirements where substantial backlog is needed. Once the processes have been implemented and optimized on the basis of the analyzed requirements, this should ensure increased process capability, which can also help to save resources in an early development phase.

1.3 Structure and Methodology

This thesis is structured in such a way that the basics and the current state of the art of technology are explained in detail. First, the actual mechatronic system and its structure are briefly explained. Then the background and current topics of the entire process environment in automotive development will be explained. Process models and methodologies play a major role here, which are also the basis of many standards and customer requirements. Subsequently, the current status of standardization in the field of automotive software development is described and an explanation is given of the added value resulting from a standardization of the development processes. As the basis of the process model is the VDA Standard Automotive SPICE, this standard is explained explicitly in detail. Both the historical development and the process models are described here as well as how the process capability level is determined. Subsequently, the practical part of the work is described in detail, whereby the conceptual design of the tool for data management and analysis is in focus. Afterwards, results of the usability of the tool and analyses of defined processes are presented whose deviations offer potentials for process optimization. In the end, there is an outlook on how this process model can be further optimized in the future, e.g. by adding further standards and even more customer requirements.

2 Mechatronic Systems in Automotive Applications

Today, almost every active component in a car is a mechatronic system. Such a system allows the component to interact with its environment through the control and adaption to real-time information and the comparison to pre-defined actuating variables. This chapter provides an introductory overview of the components and functions of a mechatronic system in current automotive applications.

The word “mechatronic” is a portmanteau from the terms “mechanics” and “electronics” where the term mechanics refers to fields of the discipline of mechanical engineering (e.g. mechanics, hydraulics, pneumatics, etc.) whereas the term electronics describes the hardware and software components of such a system [1]. **Figure 2.1** illustrates the various involved disciplines in the engineering and development process of complex mechatronic systems. These disciplines have to cooperate closely to ensure a proper product development that meets the defined customer requirements.

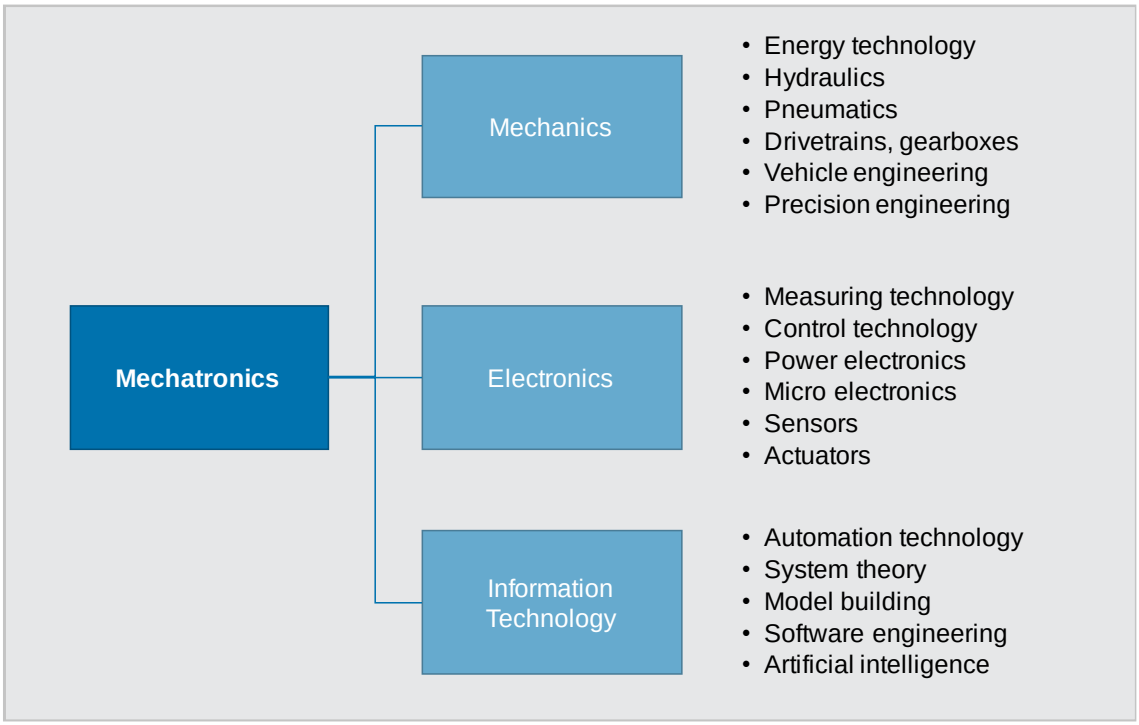


Figure 2.1 Mechatronic synergies of involved disciplines, cf. [1]

When earlier machines and precision engineered devices were characterized by the fact that they mainly consisted of mechanical components, it is now clear that the interaction of mechanical, electrical and electronic components can significantly increase the performance of these systems. However, this requires a careful adaptation of the

functions of individual components and assemblies as well as a holistic and discipline-transcending mindset in design and execution. Mechanical systems in the form of machines and devices often use the transformation of electrical, thermal, chemical or mechanical energy into the respectively required energy form. The control and regulation of the energy flow as well as of the overall process must be highly flexible due to the increasing complexity of mechatronic systems. This requires that the real-time technical measurement of process and disturbance variables is ensured as much as possible by sensors, as well as intelligent information processing technology [2].

Mechatronic systems are today present in almost the entire motor vehicle. This ranges from engine management systems and transmission control units all the way to energy management and driving dynamic modules. That also includes the communication between these systems, which is usually carried out via a standardized Binary Unit System (BUS) protocol. Typical BUS topologies in today's vehicles are for example the Controller Area Network (CAN) BUS or the more powerful but also more complex Flexray BUS. These topologies allow different components of a mechatronic system to exchange information but also enables the complex interactions between different mechatronic systems. Typical components of mechatronic systems are – but are not limited to – sensors, processors, actuators and the basic mechanical system. **Figure 2.2** provides a schematic overview of the control function of a simplified mechatronic system. These control functions generally follow the same pattern of regulating and adapting the basic system. Sensors measure the actual values of a defined variable of the surrounding environment. This result is then compared to predefined reference variables by the processor, which are either stored or calculated from different input values. The processor then makes the decision based on the comparison of the actual and the reference variable to make changes in the basic system. By the use of actuators, the mostly mechanical basic system is changed and adapted to correct the measured values to be equal to the predefined value stored in the system. Such actuators are mostly powered by auxiliary power and can be either active (e.g. electromechanical / electrohydraulic) or semi-active (e.g. electromagnetic / hydromechanics). The existence of the mechatronic system in a changing environment defines its purpose. Since the interaction with the environment may change the defined and measured variables – and thus the condition of the operations for the overall mechatronic system – constant control and adaption is necessary to prevent failures and secure the system to operate normally. That is why the control process of a mechatronic system is seen as a closed loop process [3].

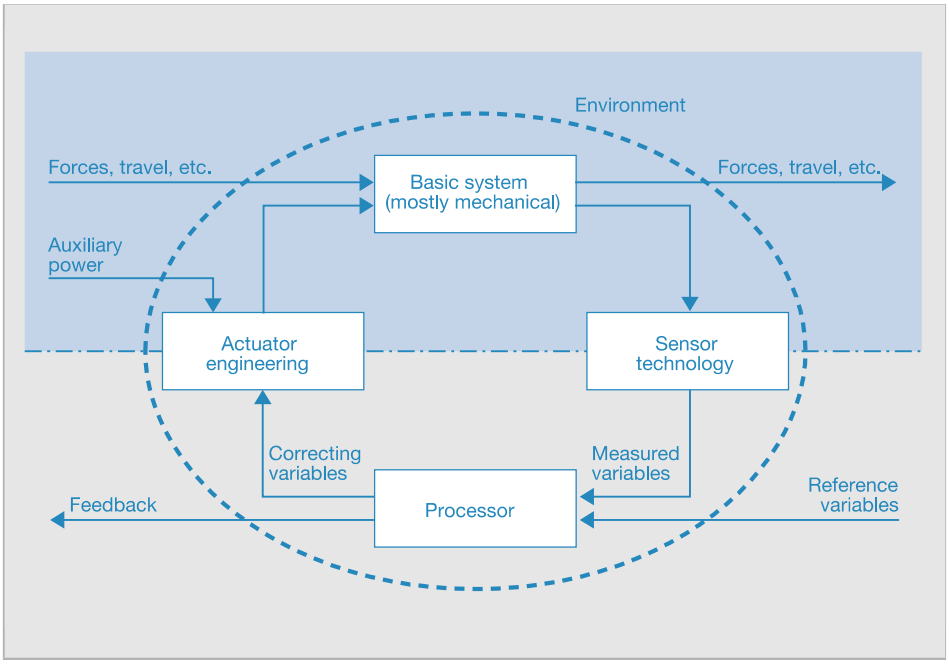


Figure 2.2 Closed loop control process of a mechatronic system [3]

2.1 Mechanics

Traditionally, most dynamic systems were based on traditional disciplines in the field of mechanics – but still today, most mechatronic systems are based on the basic mechanical principle of the behavior of physical bodies due to forces or displacement. Classical mechanics in mechatronic systems are i.e. derived from the following [3]:

- Newtonian mechanics (kinematics/dynamics),
- Hydraulics,
- Pneumatics,
- Acoustics, etc.

In mechatronic systems, these classical mechanic disciplines are mapped to components such as gears, drives, pumps or switches, just to name a few. Not only the basic system can be built out of mechanical components but also functions can be realized by traditional mechanics (e.g. sensors and actuators).
It is important to mention that innovations are nowadays less likely to occur in the field of traditional mechanics, as this discipline is the oldest one in the context of mechatronic systems. Especially in the automotive industry, innovation and patents in electronic components and software technology are increasing tremendously [3].

2.2 Electrics and Electronics

Electrics and electronics (E/E) in integrated mechatronic systems are the connection between the embedded control software and the basic mechanical system inside a mechatronic component. Electric and electronic parts are hardware components such as sensors, cables, integrated circuit boards but also microcontrollers and storage memory. Measuring systems – so-called sensors – allow the detection of states of a process. In this case, a sensor converts the physical quantity to be measured into an electrical output signal, which serves for further processing. For example, mechanical quantities such as force or torque can be converted into mechanical parameters such as length, angle, force or deformation via levers, gearing or spring elements. To manipulate and adapt, mechatronic systems make use of actuators. These are components of mechatronic systems, which convert electrical actuating signals from an information processing device (e.g. microcontroller) into mechanical control parameters required for the control of a process in the process chain. In an open loop system, the program of a microprocessor controls the action of an actuator (e.g. power currents). To change the fixed control program, a human-machine interface is used. Automated processes use closed loop systems, in which sensors measure the process state by measuring technology and provide the measured signals to the microprocessor. By comparison with set points and a control strategy, which is predetermined by calculation algorithms, control signals are then determined for the actuator, which ultimately acts as a positioning device on the process [2].

In an integrated system, most E/E components are usually located closely together on a circuit board inside an electronic control unit (ECU). The task of this integrated ECU is the whole control of the mechatronic system according to the control process described in chapter two. **Figure 2.3** provides a schematic overview of E/E components in an ECU and its data flow. The ECU has generally three interfaces to other components. These are input and output (I/O) channels for signals from sensors and to actuators, power supply and interfaces (e.g. BUS) to other systems and for diagnostics. Various input signals – either digital, analog or pulsatory – are converted, processed and provided to the microcontroller. The microcontroller, often referred to as central processing unit (CPU), is an integrated circuit board consisting of a random-access memory (RAM), fast flash memory chips and the processor itself. The CPU is the “processing unit” responsible for comparing set and actual value and calculate the necessary adjustments. These adjustment signals are then amplified and sent to the required actuators. In safety

relevant mechatronic systems, for applications in the automotive industry, there is also a monitoring module integrated inside the ECU. This module tracks all activities of the microcontroller to detect failures or irregularities in the running process. It is important to detect hazards early and prevent further fatalities that could happen in a vehicle environment. Thus, an electrically erasable programmable read-only memory (EEPROM) stores all critical information about the conditions of the ECU and the corresponding system, but also data about recorded errors and bugs. These error codes and detailed fault information can later be received via the diagnostics interface through the on-board diagnosis (OBD) interface of a vehicle, which is also connected to the vehicle's BUS system [4].

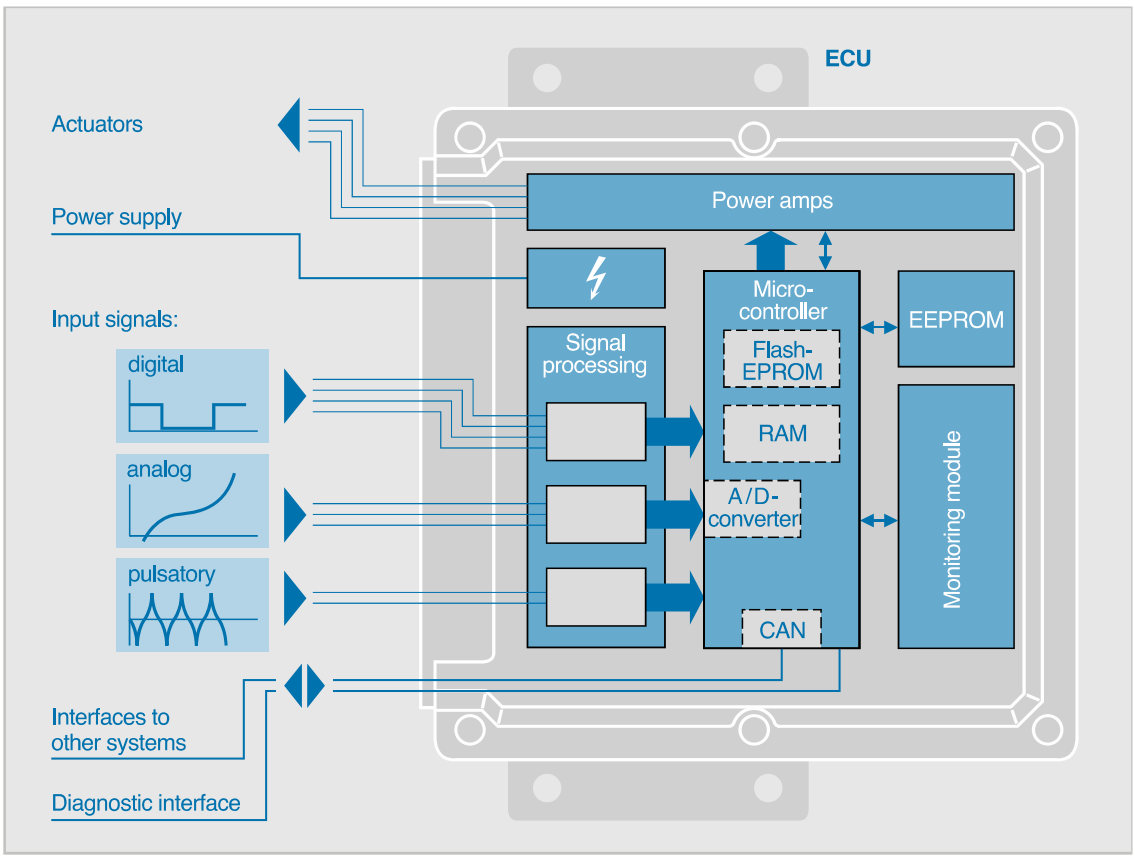


Figure 2.3 Schematic structure of an electronic control unit, cf. [4]

As previously mentioned, innovations in mechatronic components – especially in the automotive industry – happen increasingly in electronics. **Figure 2.4** shows the development of the proportion of E/E components in a motor vehicle over the last decade. A tremendous progress has been made in the development of automotive systems, enabled by the replacement of traditional mechanics by electronic systems. This has led to many beneficial innovations for passengers (e.g. vehicle stabilization systems) and pedestrians (e.g. active safety) but also for the environment (e.g. exhaust gas after treatment). A sizable amount of these innovations was enabled by the introduction of electronically-controlled systems as well as programmable electronics in the automobile and the proportion of these systems has been increasing continuously. Further requirements of active safety systems, but also in the infotainment segment of a car, will increase this development, where E/E systems will play an even bigger role as today. Customers also demand a high reliability level from their vehicle, whereas OEMs and the whole supply chain are challenged by requirements such as downsizing, cost reductions and package or weight restrictions (due to legislation). This results in an increased complexity of vehicle systems but can also be seen as an opportunity for the replacement of traditional mechanics and E/E systems towards microprocessor controlled systems. As a consequence, the development of programmable electronics and integrated control software continues to gain importance in the future [3].

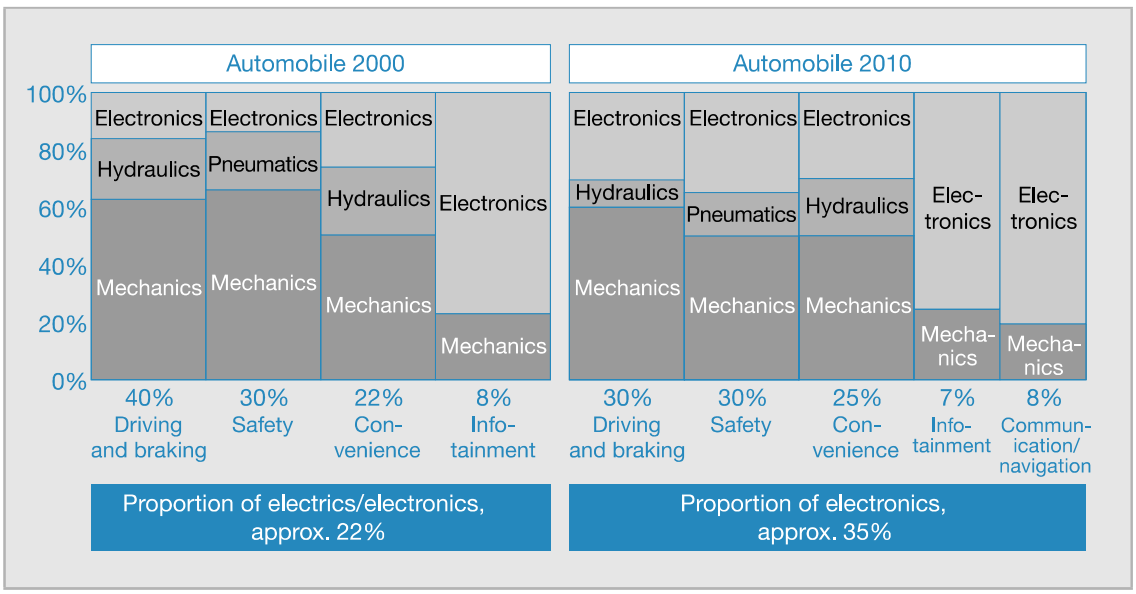


Figure 2.4 Proportion of electrics/electronics in the motor vehicle [3]

2.3 Software

In the first sections of this chapter it became clear that the design of mechatronic systems is based on the close connection of mechanical, electrical and electronic components, whereby the system properties, i.e. dynamic behavior, flexibility and learning ability, are also determined to a great extent by the software. Mechatronic systems are therefore generally characterized by a modular and clear design, which facilitates the integration of different technologies and components. This makes it possible to divide the various functions of mechanical and electronic components as well as their information technology (software) in such a way that the mechanical design is simpler and the manufacturing effort is reduced. At the same time, the interaction of the various module functions can be realized by the overall system, which would otherwise not be possible with a pure hardware design [2].

In the development of electronic systems, a general trend is from hardware to software solutions. Software solutions are ideally suited for realizing the functional aspects of electronic systems. Thus, for example, the realization of control, regulation and monitoring functions by means of software enable maximum degrees of freedom, e.g. in the design of linearizations, learning algorithms, but also safety and diagnostic concepts. No other technology offers such a large design freedom – in particular because package or manufacturing aspects have very little influence on the realization of software functions. Therefore, the implementation of vehicle functions through software provides vehicle manufacturers and suppliers a great potential for differentiation from the competition – a trend that is also observable in other industries besides the automotive industry [5].

Because this thesis focuses especially on the topic of process optimization in embedded software engineering of mechatronic systems, the general architecture, components and basic functions of embedded software is described in this chapter. Software functions in an ECU for automotive applications handle all control functions of a mechatronic process, such as the processing, comparison, offset and control of I/O signals and stored variables. To reduce the development effort for OEMs and suppliers, the software on ECUs is usually clustered in various modules. This allows a flexible adaption of the parameters and control functions, but also provides the necessary compatibility with the hardware components in the mechatronic system and the entire vehicle. In general, there is a separation of functions: either in basic software or application software. The actual software functions for the control and monitoring functions of the ECU are part of the

application software. The basic software depends on the hardware, i.e. the microcontroller used, as well as on the other hardware that can be installed on the control unit. However, the interfaces of the basic software are largely independent of the hardware and also standardized. The application software then relies on the interfaces of the basic software. It can thus be easily ported to other control units. Between basic software and application software is the interface, which connects both, the runtime environment (RTE). In the RTE, communication between the software components of the application software is realized as well as between components of the application and the basic software. This provides the necessary standardized communication mechanisms for a flexible adaption of the control unit. The advantage of this architecture is that individual software components of the application software can be ported to other control devices with virtually no change. In this case, only the communication flow within the RTE has to be changed. A standardized example of such a system in automotive applications is the AUTOSAR development environment. **Figure 2.5** illustrates the AUTOSAR software architecture of an integrated control unit for automotive applications in general. It highlights, that the RTE links together individual software functions from application and basic software on one level [6].

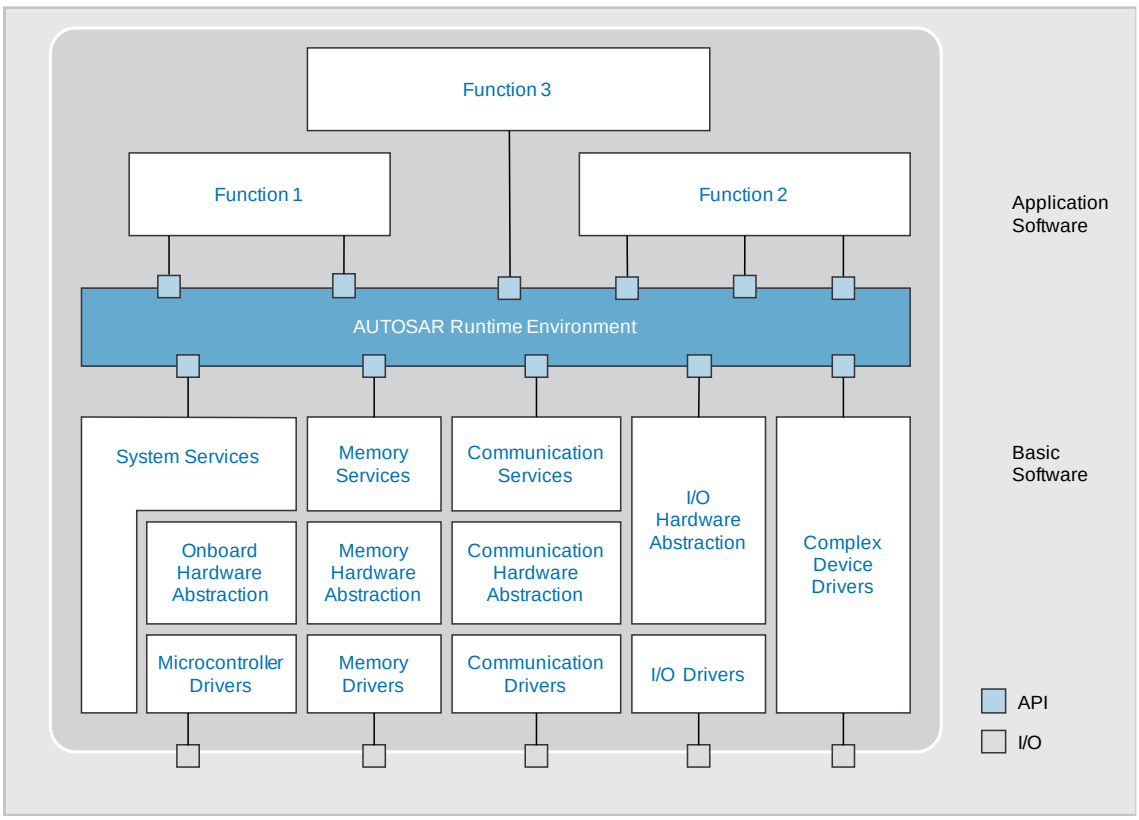


Figure 2.5 AUTOSAR software architecture for microcontrollers, cf. [6]

The advantage of such an RTE is the standardization of the transfer protocols, which allow manufacturers to create individual software modules that are able to work on different ECUs with the same RTE. This eliminates the need for translation to other protocols and allows for easy interchangeability, which also reduces costs. Complex drivers are a special case. These parts of the basic software are hardware and application-specific, which means that they are exactly matched to the ECU. These parts take on highly real-time-critical functions for which perfect interaction between HW and SW must be ensured. Since there are many such functions in the engine control system compared to other ECUs in the car (e.g. ignition, injection, tooth signal acquisition), the complex drivers take up a large part of the basic software [7].

The benefit of interchangeability of the application software with the basic software via the RTE is made possible by the integration of application programming interfaces (API). An API is a programming or software component that is made available to other programs for the connection to the system. This is a program link at source code level with defined parameters. Therefore, complex function modules are able to access the available runtime environment and through that, communicate with components of the basic software [5].

In general, the program that the microprocessor of a microcontroller executes is stored in the read-only memory and is not exchanged for different applications. An important exception is the loading of a new software version as part of a software update by “Flash programming”. In software engineering, for example, the specifications from the descriptions for control and regulation functions must be converted into a program code executable on the microprocessor and a set of parameters that can be stored in the microprocessor's data memory. For automotive applications, this process of storing the code and parameters on the ECU’s memory is only done once in the production process. Updates for ECU’s are not typical in the product lifecycle (PLC) of a car. Mostly they are done when a failure is detected in field and the manufacturer has to recall a certain car series. Over-the-air software updates are mostly limited to entertainment functions or the ability to unlock pre-defined functions (e.g. advanced driving assistance systems) [5].

In the field of automotive software development, the trend is increasingly towards the use of open source code and third-party code to prevent delays in projects. This is the result of the strong budget and time pressure that a current development project is experiencing. Greater use of third-party code helps embedded system development teams to accelerate time to market in various industries, including the automotive sector. Here, software functionalities are among the innovation drivers that generate

corresponding competitive advantages. Unfortunately, setbacks in the development of embedded systems are often amplified by the dependencies between the individual disciplines (software, mechanics, electrics). Delays that occur in one area can continue in others and lead to significantly excessive development budgets. Developers of embedded systems and enterprise software often face similar problems, so they are increasingly using similar strategies. This results in a mixed approach that combines iterative elements with the precision required for the embedded system market. The success of teams working according to agile development methods promotes further changes in the strategies of embedded system development. Today, companies throughout the industry are exploring new ways to increase efficiency through additional collaboration and the implementation of agile management processes [8].

2.4 Functional Safety of Mechatronic Systems

The demands on the safety of vehicle functions are particularly high compared to other sectors such as mechanical engineering or consumer electronics. Since the probability that in the event of an accident a person is always close to the vehicle must be assumed to be 100% (driver), the functions are usually classified into a high safety class. This is a typical example for the automotive industry and does not apply to the general mechanical engineering industries. There, the probability of people being around machines can be significantly reduced, for example, by suitable shut-off measures and similar safety devices. With mechatronic systems increasingly becoming more complex, the requirements to retain a high safety standard is vital. Failures can lead to dangerous accidents that could threat people's lives including the driver, passengers but also pedestrians and other road users. Therefore, the analysis of functional safety and the specification of suitable safety concepts has a great influence on the functional development and thus also on the software development for embedded systems in automotive applications. High safety requirements require error detection and error handling measures. One of the most powerful measures to detect and deal with errors is the redundant design of systems. This means that the trend towards distributed and networked systems in vehicles is intensified by high demands on functional safety. Additionally, this results in special demands on the development processes and development tools. Examples include the certification of development processes, software development tools or standardized software components such as the aforementioned AUTOSAR systems [5].

The concept of functional safety (FuSa) is not able to provide one hundred percent reliability against system failure. Legal requirements demand safety measures for technical systems that endanger life and property in the event of failure, so that the remaining residual risk remains below a tolerable threshold. For complex mechatronic systems with software, these are so-called safety functions. In addition, the legislation requires that safety-relevant systems must comply with the current state of the art in technology. The basic safety considerations are defined in international standards. Today, a relatively simple proof of functional safety is a prerequisite for the registration of vehicles for road traffic. In general, the compliance and application of a standard cannot be forced by legislation. But with respect of the product liability law it is in the interest of every automaker to always develop according to the current state of the art, which is defined in current standards and guidelines. The manufacturers thus protected themselves against high fines in the event of a vehicle malfunction or failure of individual system components by applying common standards [5, 7].

The most important international standard for functional safety in automotive applications is ISO 26262, a domain-specific application of IEC 61508. This standard has a significant influence on the design, qualification and production of electronic control units and has been accepted by more than 130 countries worldwide. The failure metrics of ISO 26262 and their requirements for the avoidance of random component errors, dependent errors and mutual interference have proven to be a significant influence on the safety architectures. On the basis of the procedures described here briefly, the result is a classification according to ISO 26262 into the so-called “Automotive Safety Integrity Levels” (ASIL). The evaluation basis distinguishes between different driving situations:

- *Severity*: potential hazard and severity of resulting injuries
- *Exposure*: how often are driving conditions where the fault is relevant
- *Controllability*: the controllability of the resulting situation

The classification levels range from QM (no FuSa relevance), ASIL A to ASIL D. This refers to the sum of the above-mentioned three classes, i.e. even with the highest severity potential (with possible death consequences) a QM classification can result if controllability is high and the probability of occurrence is low. As a product development based on high ASIL levels is becoming exponentially more expensive, the standard allows the decomposition into lower ASIL levels where the same tolerable residual risk is guaranteed [7]. This procedure is described in more detail in chapter 4.2.

3 Process Environment

The “process environment” describes all surrounding activities, processes, guidelines and deliverables in a development environment, independent from individual projects or customers. It is a corporation-internal approach to structure the engineering processes and align the different activities of the various departments along the development phases. Therefore, it is necessary to define clear roles and responsibilities in accordance to the time schedule but with the generic project or development target in mind. These processes and development plans are often company or industry specific and mostly derived from different sources to get a general process model that is in accordance with internal guidelines, best practices, international standards and customer specific requirements.

Fundamentally, the product development process is the central link to transform the requirements of the market, respectively the customer demands, into an appealing product. Faced with expanding markets and diversified, customer-specific mobility requirements as well as regionally pronounced certification and ecology requirements, every OEM with its product portfolio and thus its entire product development process is faced with enormous challenges worldwide. Processes can only be realized successfully if the people are working together on one goal know the processes and are able to deliver the services within the agreed time and with the agreed quality. Due to the increasing variety of vehicle types and the flexibility of digital development, product-optimized versions of the development process are created. These variations of processes require very good knowledge of the workflows as well as flexibility in the projects. Developing the individual skills of each individual employee is a prerequisite for being able to react quickly to the diversity of requirements [9].

3.1 Automotive Engineering Processes

Historically, the automotive industry is one of the industries with the most guidelines, standards and best practices. This results from the fact that a vehicle nowadays is a complex system of traditional mechanics, electric and electronics but also increasing software complexity. Additionally, the safety aspect in this industry is critical – equally the environmental impact of road vehicles – which requires progressive development but also supporting processes and the right methodologies. These guidelines change and adapt over time as technology advances.

The development time for new vehicles is currently estimated to be around three years, a production period of about seven years and a subsequent operating and service phase of up to 15 years. This results in a total product life cycle (PLC) of around 25 years. These intervals are considerably shorter in electronics due to the continuing advances in hardware technology. This poses great challenges, e.g. for the long-term supply of electronic spare parts to the market and must be taken into account during vehicle development. This also influences the software architecture. Effects include, e.g. the standardization of the software architecture and the hardware-independent specification of software functions in order to simplify the porting of the software to a new hardware generation, which might be necessary later on. A product's life cycle can be divided into three phases: development, production and operation. Product lifecycles can vary in length for the individual components of a system. For example, product life cycles for vehicles are often longer than the life or modification cycles for ECU hardware and software due to the ongoing technological advances in electronics. In addition, the system requirements can vary in the individual phases of development, production, operation and service. In this thesis, the focus is on the processes of the development phases in the PLC [5].

The increasing number of functions in a vehicle, its networking, high and increasing demands on reliability, availability and safety, as well as variant and scalability requirements lead to a complexity that can hardly be mastered without a defined development process. A method of mastering complexity that has long been used in the automotive industry is the division into systems and subsystems and their distributed development. In vehicle development, this approach initially results in a partitioning of the vehicle into the subsystems powertrain, chassis, body and infotainment. Subsequently, the further partitioning of the subsystems into subordinate subsystems and components takes place step by step. After the division of tasks and parallel development of the components, their testing is carried out, as well as the gradual integration and testing of components to subsystems via the various system levels. Finally, the subsystems powertrain, chassis, body and infotainment are integrated into the vehicle. The requirement for this is not only a clear division of responsibilities in subsystem and component development, but also the cooperation in partitioning and integration of the system with regard to packaging space, vehicle functions and production technology. In addition, the cross-company subsystem and component development between vehicle manufacturers and suppliers is very prominent in the automotive industry. The clear division of activities between vehicle manufacturers and

suppliers is therefore a fundamental requirement. A further dimension is added by the simultaneous development of different vehicles or vehicle variants. This leads to multi-project situations on all system levels at vehicle manufacturer and supplier. The cooperation of different disciplines requires a common, holistic understanding of the problem, a common understanding of the processes of problem solving, as well as a common understanding of the influences and effects of solutions on the system. Furthermore, the responsibilities and competencies have to be defined in a project. Proven holistic approaches are, for example, mechatronics on the technical side or system engineering methods on the organizational level of a development project. In the case of cross-company cooperation between vehicle manufacturer and supplier, all aspects of the business model, in particular also legal issues such as product liability or patent rights, must be clarified additionally [5].

3.2 Systems Engineering

In contrast to the development of individual components, system engineering is focused on the analysis and design of the system as a whole, not on the detailed analysis and design of its components. Systems engineering is therefore an interdisciplinary approach and includes measures to enable the successful implementation of systems. In the development process, systems engineering targets the early definition of requirements and the required functionality with the documentation of the requirements, followed by the design, verification and validation of the system. The comprehensive problem definition of a system is taken into account – such as development, scope, costs and schedule, testing, production and service up to the final disposal. Systems engineering provides a structured development process that takes into account all phases of the product lifecycle, from concept development and production to operation and service. Both technical and organizational aspects must be considered. In comparison, software development as well as the development of hardware is a discipline within system development. The precise definition of the specification and integration interfaces between system and software development is an essential prerequisite for a seamless development process [5].

During the 1960s, systems engineering was defined as an interdisciplinary, document-driven approach to the development and implementation of complex technical systems in large-scale projects, particularly in the American aerospace industry and in major military projects. From the software and electronics industry's point of view, this approach

has been continuously expanded and now offers modeling and simulation support for complex, highly networked systems. Systems engineering is based on the principle that a system is more than the sum of its subsystems. For this reason, not only the relationships of the subsystems, but also the overall context of the system in general should be considered during the development [10].
An important term in the field of systems engineering is system theory. System theory provides procedures for dealing with complexity. The common approach to manage complexity requires the following assumptions:

- The distribution of the system into components does not distort the problem
- The components are essentially identical to the components of the system
- The principles for the assembly of the components are simple and stable

The composition of a complex system in subsystems and individual components is illustrated in **Figure 3.1**. The properties of the system are derived from the relationships between the components of the system, that is, from the way the components communicate and interact. As the complexity of a system increases, the analysis of components and their interrelationships becomes complex and costly. Therefore, it is vital to define the right system boundaries and to derive the right number of components in every new development project [5, 11].

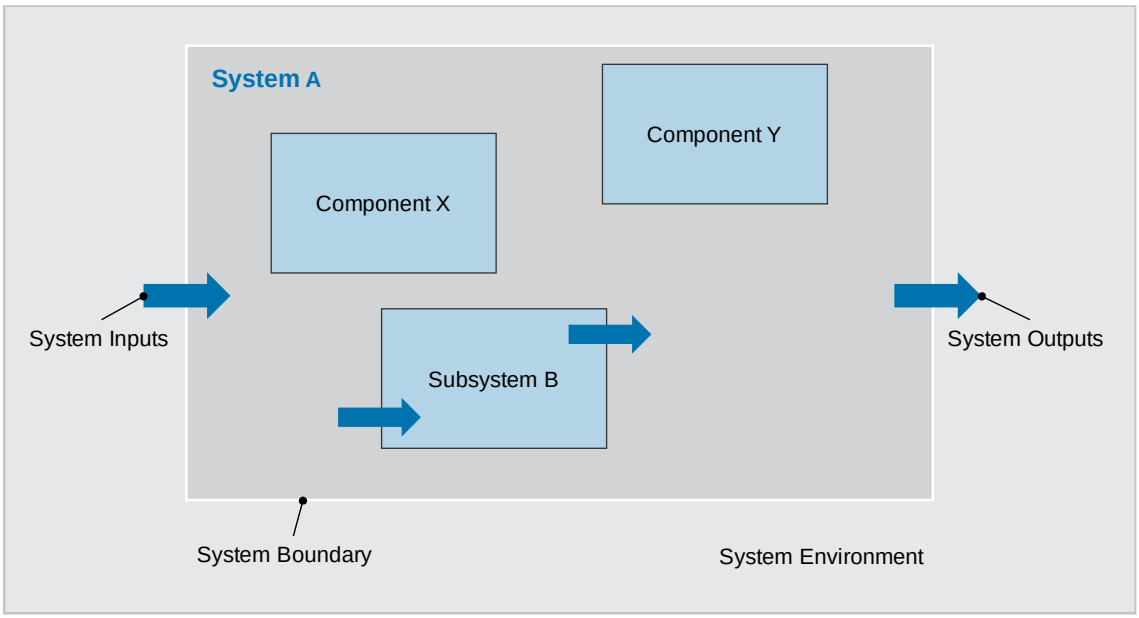


Figure 3.1 Definition of a system in system theory, cf. [11]

3.3 V-Model Development Approach

To be able to better understand, simplify and structure these efforts, various process and development models have been designed and implemented, not only in the automotive industry. Due to the numerous interactions between vehicle, electronics and software development, a continuous development process is necessary that covers all steps from the analysis of customer requirements to the validation of the mechatronic system. The V-model distinguishes between a view of the system and a view of the components and integrates quality assurance measures. It is therefore widely used in the automotive industry. This process model for development can be presented in the form of a “V”. In this way, the project phases and the interfaces between system and software development can also be represented in an adapted V-model, as well as the specific steps in vehicle development. **Figure 3.2** represents the schematic V-model according to VDI 2206. In the left half of the “V” are the processes and tasks for the design of the system. From top to bottom, the focus is shifted more and more from the overall system to individual components. On the right side, the integration of the system is visualized. To validate each integration and development step, the components and the system are tested against the defined requirements during the integration (bottom-up in the “V”) [5].

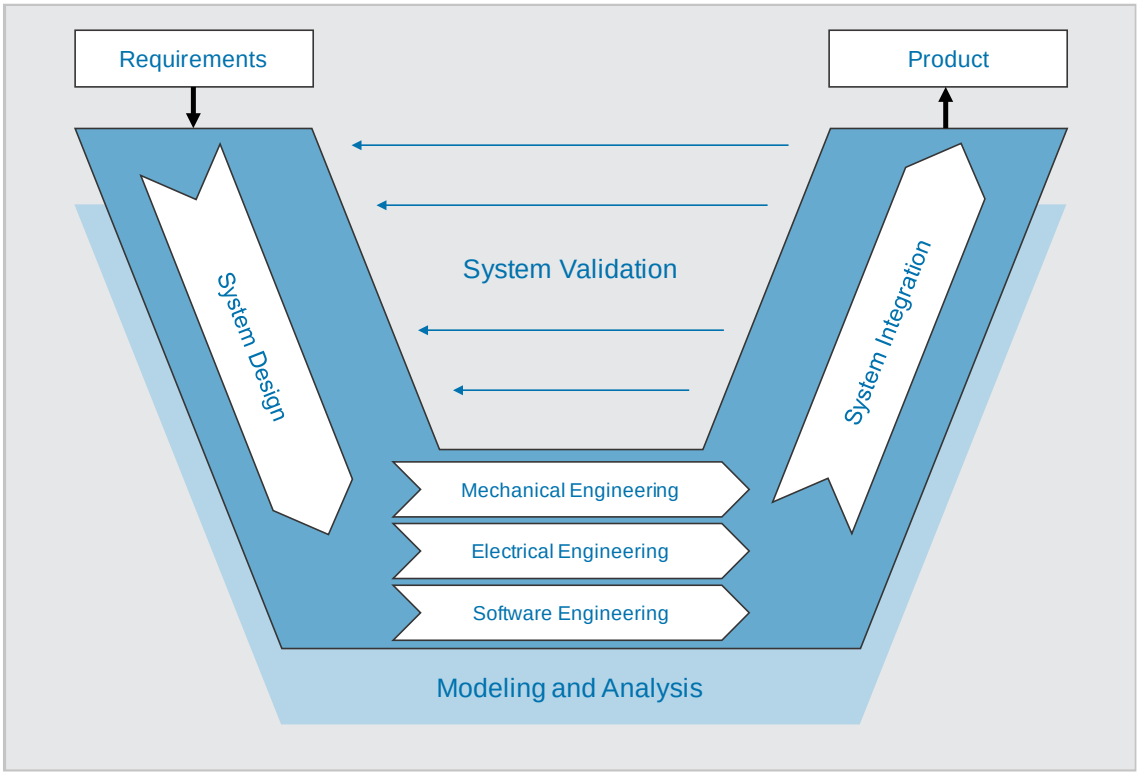


Figure 3.2 The V-model for mechatronics development (VDI 2206), cf. [10]

3.3.1 V-Model of Automotive Software Development

The aforementioned V-model is just a basic model on system level without necessary details. Multiple V-models have been developed to suit different needs and industries. In this section, the V-model in automotive development processes, especially for software development, is presented. As illustrated in **Figure 3.3**, the V-model for software development can be divided horizontally into activities regarding system development and software development. Usually, system development is done by the OEM, whereas the software is developed by component suppliers. The core process also divides the considered objectives into three different levels of abstraction. First, the logical system area – which is focused on the various functions the defined system has to carry out. After more details are defined, the technical system architecture is specified. In this architecture, it is already clear which electronic control units realizes the previous defined functions in the system of a complete vehicle. On a deeper level is the software architecture. This is the most detailed view how the defined functions can be realized and implemented. Here it is defined, which modules make up the software and what parameters are used to calculate and control the system on a software level [5].

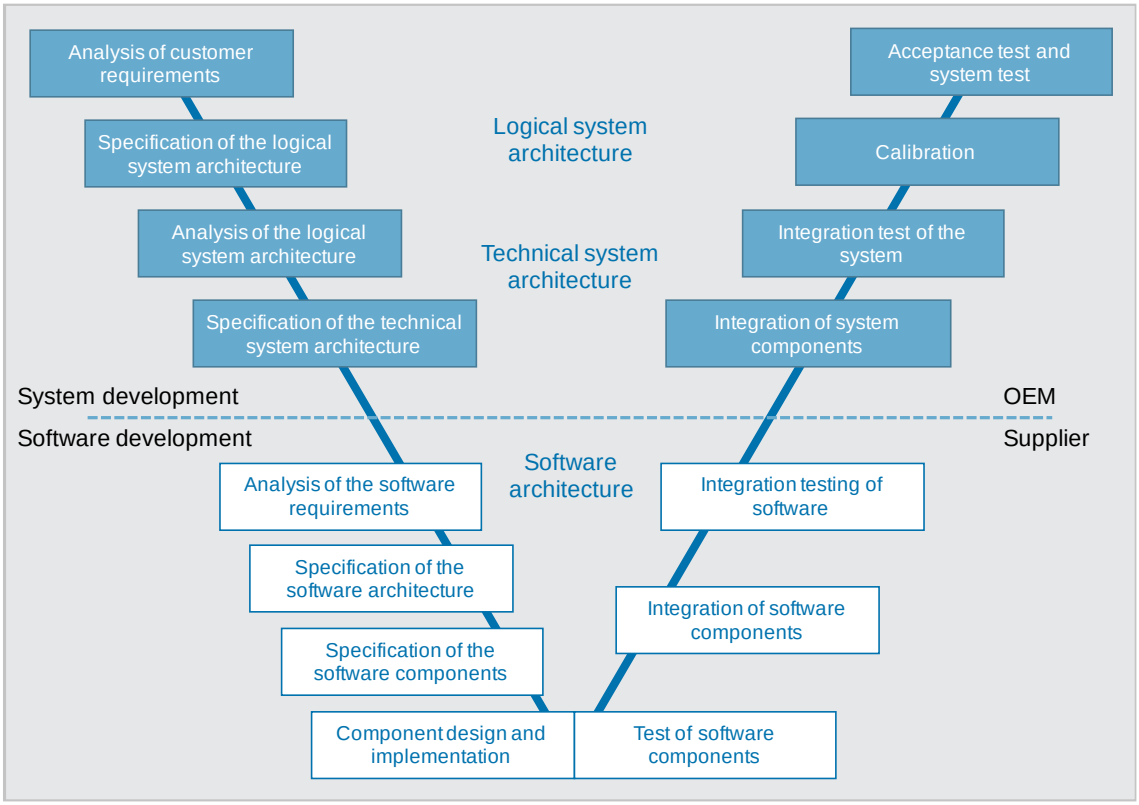


Figure 3.3 V-model process for the development of systems and software, cf. [5]

3.4 Phase-Gate Process

The “Phase-Gate Process” – also often referred to as “Stage-Gate Process” – is an exemplary model of an advanced development process in a corporation. Usually, this phase-gate process model is an integrated part of the defined development system in each company. It defines critical development phases and their corresponding gates. Each phase represents a stage of development activity with the purpose to achieve the set targets of the gate. Each gate has defined deliverables and each responsibilities and actions are defined in the corresponding phase. The status of the project is reviewed in quality or release meetings at each gate or milestone within a management team. In general, the phase-gate process model is also based on the key concept of the V-model for development projects. Usually, there are also phase gate process models for the entire product life cycle present, i.e. for the production and operation processes.

Figure 3.4 shows an exemplary phase-gate process according to Cooper. The entire process from idea and concept to production readiness is covered with several sections. Cooper defined a set of parallel activities, deliverables and outputs for each phase and gate. In the development system, quality gates mark the beginning and end of important process stages, with so-called milestones. Only if the defined product and process maturity requirements are met at these points in the process, the quality gate can be passed through. If a quality gate is not met, it is necessary to decide on and take measures that are capable of successfully passing through the “access gate for quality” in the second attempt. This is done according to a clearly defined process with clearly defined rules. Usually, the previous phase is then partially repeated in order to fulfill the requirements of the gate at the next attempt and thus reach the next phase [9, 12].

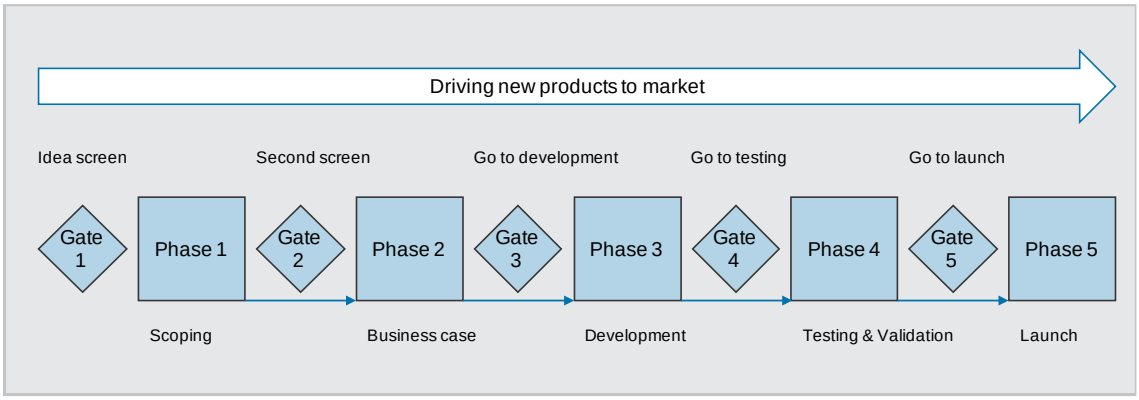


Figure 3.4 Phase-gate process according to Cooper, cf. [12]

3.5 Embedded Software Engineering

As this thesis is emphasizing the embedded software engineering part of mechatronic system development in automotive applications, this section provides a general overview of the engineering of embedded software components in the context of mechatronic systems in automotive applications.

In electrical engineering, a distinction is often made between processors, controllers and systems on chip (SoC). All these terms refer to programmable digital integrated circuits (ICs), whereby the processor as the innermost module takes over the execution of the machine commands. Embedded systems are computer systems in which processors, controllers or SoCs are equipped with supplementary electronics – power supply, protective circuits, additional components and connectors – and are usually integrated in enclosures for their dedicated task. The traditionally essential characteristic of embedded systems (and thus their differentiation against the common definition of computer systems, which is limited to PCs, smartphones, tablets, etc.) is the absence of a human-machine interface. In many cases, the driver and vehicle occupants have little or no influence on the various functions. Examples of embedded systems in automotive applications include all electronic control units. The engineering of embedded software consists of all development activities and processes in new product development of software, architectures, protocols and logics for embedded systems. This ranges from the initial idea or commissioning to development and production readiness [13].

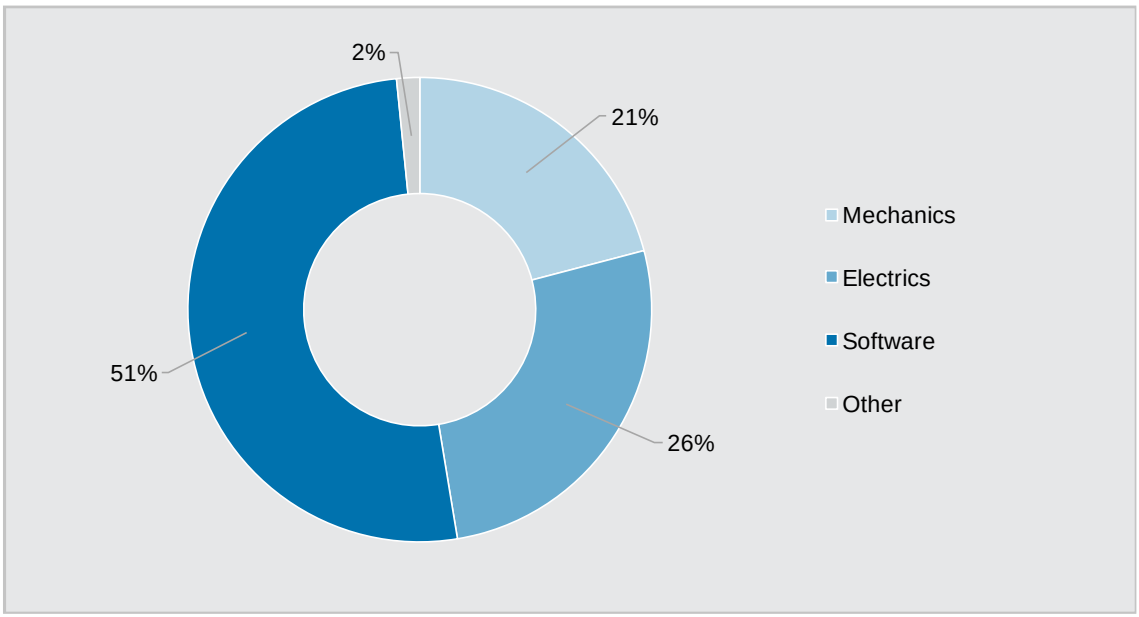


Figure 3.5 Value and cost distribution in embedded products (2016), cf. [8]

As illustrated in **Figure 3.5**, today's value of embedded systems in automotive applications is mainly the software component. The same applies to product and development costs. That is why it needs a structured and well-defined development process, which is tailored to the tasks and challenges of software engineering. A substantial reason for poor software is often the lack of interaction between systems from different suppliers. One of the requirements is to create uniform standards for all manufacturers, suppliers and programmers. Together with reduced time-to-market, all parties involved face the challenge of driving functional and software development forward efficiently. This results in the following requirements for the software development process for mechatronic systems, according to [14]:

- The complexity of both, the individual ECU and the ECU BUS network must be mastered.
- Distributed functions have to be handled in distributed systems.
- Interfaces between project partners must be adapted and defined.
- Process know-how from other industries (e.g. aerospace) can and should be taken over, especially for safety-relevant systems.
- Quality must be ensured through optimized processes and test methods.
- Testing must be an integral part of the development process as a core discipline.

Software engineering is different in many aspects compared to traditional engineering disciplines. Compared to hardware engineering, the change cycles are shorter in software development. The use of flash technology in conjunction with the network of all ECUs in the vehicle enables cost-effective software updates to be carried out in the field, for example via the vehicle's central off-board diagnostic interface – without the need for expensive removal or replacement of ECUs [5].

Due to the long product lifecycles of vehicles, the advanced development and change management of vehicle systems is particularly important. It must be possible to manage and track the effects of changes in a system. For the continuous development of mechatronic systems, project management – described in following sections – must therefore be linked to the core process together with change management, requirements management, supplier management and quality assurance as supporting processes.

Another challenge is the design and implementation of software components in the overall vehicle system in terms of availability and readiness of each modules and functions. **Figure 3.6** illustrates this development complexity over the project time. The basic idea of system theory is also a challenge in the development of a holistic system.

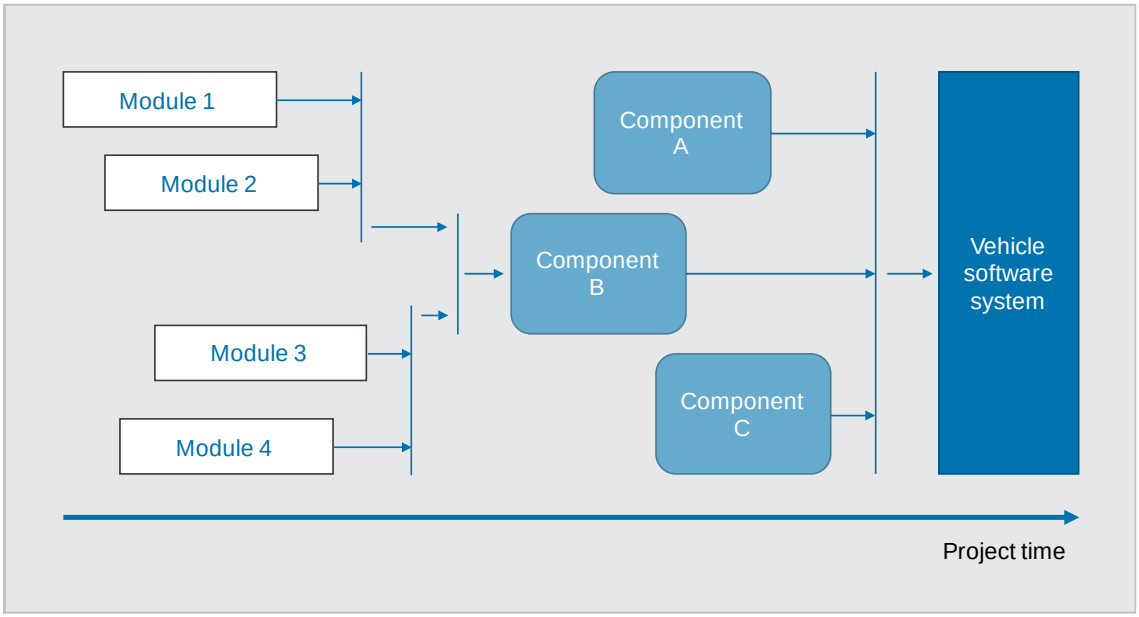


Figure 3.6 Software integration steps in product development, cf. [15]

Different components of a system are developed in decentralized departments or in a project-oriented matrix organization. Each developer must achieve the defined targets at different milestones in the development project to be compatible with corresponding functions of other developers or suppliers. At these so-called integration steps, the software versions of all components and modules must be on the same level, otherwise integration and communication among each other is not ensured. Over the course of the development project, the different modules are gradually combined into components and finally integrated into the overall system. A seamless interplay between all subsystems of different supplier is essential, which is why integration and testing is an integral part of embedded software development [15].

3.5.1 Methodologies and Development Tools

In all stages of development, appropriate methods supported by tools can contribute to improving quality and reducing risk and costs. The continuous interplay of the different tools is therefore of great importance. Possible methodologies and development tools with their significant effects on the three critical success factors quality, risk and costs are presented in this subsection.

The V-model implicitly assumes that the user requirements are captured and analyzed completely at the beginning and that a detailed specification of the technical system architecture can be derived from it. Integration is based on sequential, separated steps.

However, practice shows that in many cases these requirements are not met. The user requirements are often not fully known at the beginning and are updated during the development process. Specifications therefore initially only reflect a rough idea of the system. Only gradually are details defined. In the integration phase, component delays cause delays in integration and all subsequent steps. In cross-company development, many integration and test steps are restricted by non-existent, non-identical or current neighboring modules. Reality is therefore characterized by incremental and iterative procedures, in which steps of the V-model or the entire V-model are processed several times. Methods and tools for the early validation of requirements, specifications and realized components in the laboratory, at the test bench or directly in the vehicle can be used to support such a procedure for the development of software functions [5].

In order to be able to offer customers even more functions and even more intelligent software solutions, the path has been moving from pure code programming to model-driven software development for several years now. Model based software development takes place in large parts without programming actual lines of code. Procedures and algorithms, as well as structures and processes are modeled in different modeling languages using graphical tools and then automatically translated into program code by compilers. Various software tools are used to model the actual ECU software. This is why it is also called computer-aided software engineering (CASE) [10].

These CASE tools allow and simplify the model based development of software, especially in the automotive industry for embedded systems where the development process has an interdisciplinary approach. Interdisciplinary cooperation in software development requires a common understanding of problems and solutions. For example, when designing vehicle control functions, the reliability and safety requirements as well as aspects of software implementation in embedded systems must be considered holistically. The basis for this common understanding of function can be a graphical functional model that takes into account all components of the system. In software development, adequate model-based software development methods with notations such as block diagrams and state diagrams are increasingly replacing the traditional software specifications. In addition to a common understanding of problems and solutions, the modeling of software functions offers further advantages. If the specification model is formal, i.e. distinct and without any room for interpretation, the specification can be executed on the computer in a simulation and can be experienced in the vehicle at an early stage by the use of rapid prototyping tools. Because of these advantages, the “digital requirement specifications” have become very popular [5].

3.5.2 Structuring the Software Development Process

The combination of these different methodologies and tools in a software development project result in a structured process for software development. Software development processes consist of different tasks and activities, clustered in phases that define what the participants should do. Project members can take on various roles in software development, such as software designers, software architects, project managers and quality managers. Software development processes are organized in phases in which the respective focus is on a certain part of software development within a project. According to [15], these are summarized into the following six phases:

1. *Requirements engineering*: phase in which ideas about the software functions are generated and divided into requirements
2. *Software analysis*: phase in which the system analysis is performed, and high-level decisions are made on the assignment of functions to the logical systems.
3. *Software architecting*: phase in which the software architects define the high-level design of the software including its components and assign them to the systems.
4. *Software design*: phase in which the individual components are designed in detail
5. *Implementation*: phase in which the design is implemented for each component.
6. *Testing*: phase in which the software is tested in various ways.

The general development process is defined and often standardized for each organization, but the outcomes and milestones may vary for every project. Since not every vehicle project has the same technical complexity or characteristics when it comes to product development, the product development process must also be adapted accordingly. An essential criterion here is the degree of acceptance of already known concepts and technologies and the extent of changes or innovations, which have to be developed and secured (e.g. carry-over-parts) [9].
Developers of embedded systems and enterprise software are often faced with similar problems, resulting in an increasing use of similar strategies. This creates a mixed approach that combines iterative elements with the precision required for the embedded system market. The success of teams that work according to agile development methods promotes further changes in the strategies of embedded system development. Organizations are exploring new ways to increase efficiency through additional collaboration and new approaches to management and project organization [8].

3.6 Concurrent Engineering

Concurrent engineering – also known as simultaneous engineering – is a method to gradually reduce the development time in a project without skipping relevant phases. In the previously sketched project flow – analogous to the phase-gate process model – the activities of the individual divisions or departments always followed in sequential sequences. In practice, this often led to boundaries between departments that restricted the flow of information and caused the development process to be extended over time. Each department generally has its own conventions, description models and others. The interface is sometimes not sufficiently illuminated, which can lead to errors or unnecessary time expenditure. As the work flow in a project is divided up between different departments, the more the individual loses sight of the overall goal and feels less and less responsible for the activities of others. The sum of the optimizations in the departments is by no means equal to the optimum of the overall process [16].

Figure 3.7 illustrates the reduction of development time – and therefore possibly costs – by the application of concurrent engineering principles. The sequential phase process in the top of the figure represents the traditional approach to development projects. The second process flow is the approach with a concurrent engineering methodology applied. In general, the entire process chain is compressed in length, without shortening the individual process steps. This is possible by overlapping subsequent process phases during the development time of the project with the application of some correction loops.

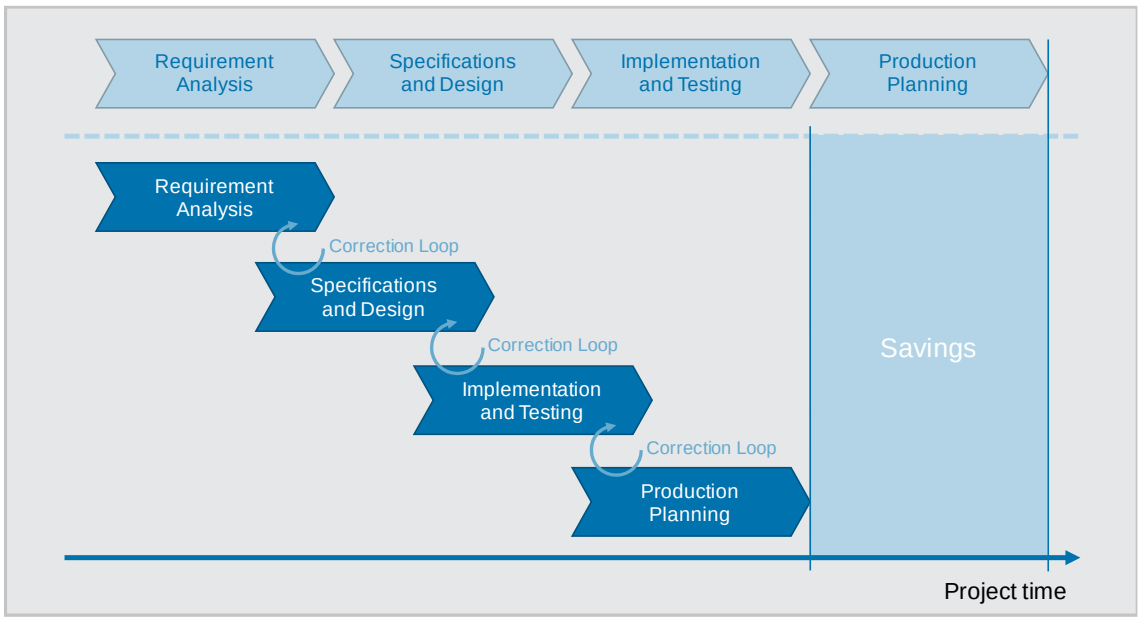


Figure 3.7 Development time savings through concurrent engineering, cf. [17]

Instead of the sequential process, it is necessary to involve the individual departments actively and simultaneously (“concurrent”) while the product is being developed. Ideally, the employees of all involved departments work full-time in the same room. Concurrent engineering therefore means a joint, simultaneous and trustworthy execution of all activities for the development of a product and the associated production facilities. Because of the intensified collaboration between the involved departments and employees, a partial parallel processing of the traditional sequence is possible. Therefore, correction or change loops are necessary at each overlapping phase to adapt the subsequent process if necessary [17].

For the development of software functions, the parallel processing of tasks means, for example, that the software function is tested and calibrated after analysis, specification, design, implementation and integration while at the same time additional new software functions are developed. Furthermore, different development environments have to be coordinated with each other. This means that simulation steps, development steps in the laboratory, on the test bench and in the vehicle, must be designed and synchronized with each other as consistently as possible [5].

To assure a successfully functioning concurrent engineering process, [17] defines the following principles that must be present:

- *Strong team leader*: the team leader gets extensive competences. As a kind of entrepreneur in the company, he is responsible for designing a product and getting it into production.
- *Close team*: team members are assigned to the team for the duration of the project. Their performance in the team is assessed by the team leader.
- *Comprehensive communication*: conflicts arising from different perspectives and interests of the various departments are resolved directly and from the beginning. The decisions taken collectively must then be actively supported by everyone.
- *Simultaneous development*: the individual development activities overlap and do not follow one another in time. This saves a great amount of time, but requires common work with visual contact, experience and foresight.
- *Sales participation*: the sales department integrates the results of market research, service and customer surveys as well as competition and error analyses directly into the development work and monitors their implementation.
- *Cooperation with suppliers*: they are generally paid to participate in the project work. It is important that a cooperation based on trust is established.

3.7 Project Management

The aforementioned benefits of concurrent engineering can be achieved with effective project management. Project management, also referred to as program management, is the organization of a project in an organization as a whole. It is one of the most important supporting processes in every company, as it is steering and controlling each project. It defines the surroundings of each development project and defines rules how each phase of the project is processed to achieve the agreed targets. A project itself is characterized by the following criteria, as described in [5]:

- *Tasks with risk and a certain uniqueness, i.e. no routine business*
- *Clear assignment of tasks*
- *Responsibility and objectives for an overall result to be delivered*
- *Time limit with clear start and end date*
- *Limited resource utilization*
- *Special organization tailored to the project*
- *Often different, interconnected and interdependent subtasks*

The objectives of a project itself are usually set in the very beginning of the project and can usually be summarized into one of these groups, according to [5]:

- *Time target*: when should the overall result be available?
- *Cost target:* how much can it cost to achieve the overall result?
- *Quality target*: which requirements should the overall result meet?

On the one hand, project management covers all aspects of project planning, i.e. planning the implementation of project goals. This means that quality, cost and schedule planning must be carried out, which is supported by organizational planning, resource planning and risk analysis. On the other hand, project management also includes project tracking and control, i.e. the monitoring of quality, costs and timing deadlines during implementation until project completion. This also includes risk management, the monitoring of emerging risks and definition of corresponding countermeasures [5].
Another important aspect is the type of organization around a project. In the automotive industry, it is common to use a so-called matrix organization, specialized for projects. The OEM as well as its supplier uses this form of organization throughout the entire line-organization of all departments. The project manager is assigned employees from the individual departments involved, for a specific project and for fixed periods of time. He is

empowered to give instructions to the employees in matters of the project. The line manager remains responsible for all other organizational questions. But he has no authority for the project. In the event of a conflict of interests, the next highest manager is responsible. In the course of intensive work teams, the team leader is more likely to be given more responsibility today, i.e. he will be involved in the decision on performance assessment. This allows for a flexible organization with a lot of knowledge exchange and participation, as the employees in the individual technical area are working on different projects with maybe similar problems and solutions. **Figure 3.8** highlights the composition of the described project organization as a matrix. Another challenge are multiple projects inside an organization, especially if the same employees are involved in several parallel projects. This can lead to overstressing of the individual employees, which can result in poorer project results [17].

Another form is the organization in a so-called pool organization. Employees are not assigned to specific groups or departments, but rather form a pool from which they are assigned to individual projects. The advantages are increased flexibility, a good use of personal skills and a better flow of information (because of no one-sided specialists). A drawback is the member's lack of feeling of belonging, as they are temporarily in different projects with no connection to other pool members. Companies are currently trying to develop this idea further. One form of this is swarm organization, whereby the decision making in projects is achieved by the swarm intelligence of the entire pool without the need of a leader but with support of mentors and mediators [17].

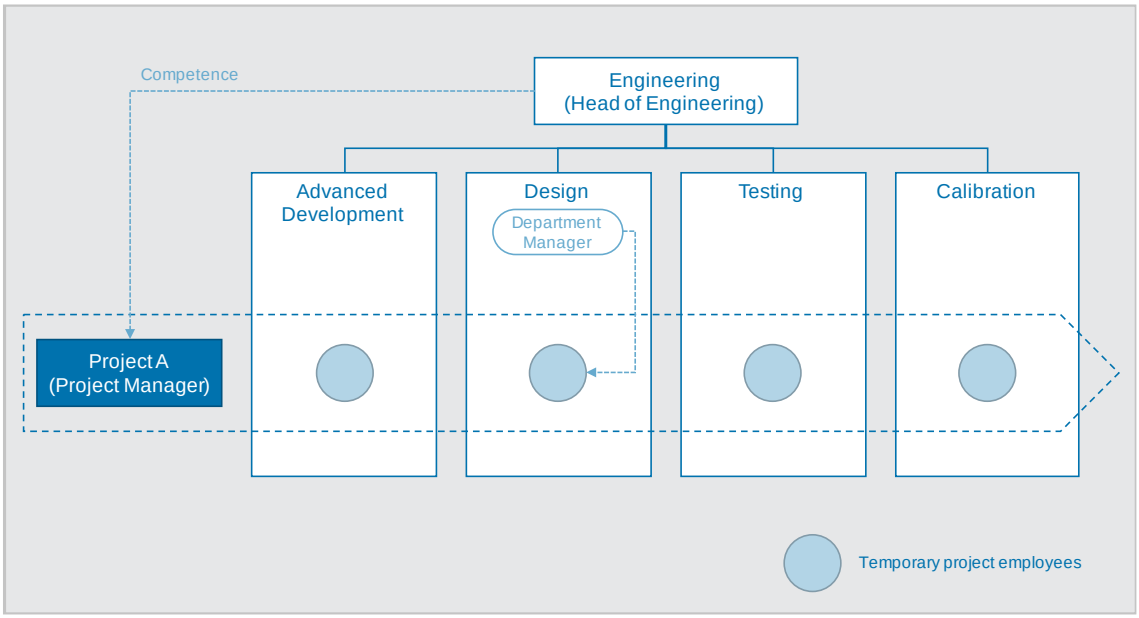


Figure 3.8 Matrix organization in an automotive development project, cf. [17]

3.7.1 Traditional Management Practices

Traditional project management practices are focused on planning and controlling of the development project to achieve the defined and well-known objectives. These were set in the beginning by a so-called product requirement document (customer specifications) and the supplier's functional specification document. Based on the description of the requirements, the project manager start to plan all resources according to the quality, costs and time targets. Usually, this is done in a waterfall method. First of all, the subtasks of a project must be defined. A milestone is an event for which subtasks of the project have to be completed. The achievement of milestones is a typical point in time for partial deliveries, tests or partial payments by the customer. The period in which a subtask is processed is called the project phase. These phases are usually subdivided into additional phases. This is especially necessary if several organizational divisions and different companies are involved in a project, which is usually the case in vehicle development. As shown in **Figure 3.9**, this results in a process sequence with subsequent tasks and processes. By applying the aforementioned concurrent principles, even more tasks can overlap with each other (e.g. software and hardware development). This sequential process is a structured approach and is well-established in the automotive industry. Because of the resulting appearance of the planning shape, this is called the waterfall method. This assures that the customer gets the right product he wanted [5, 17].

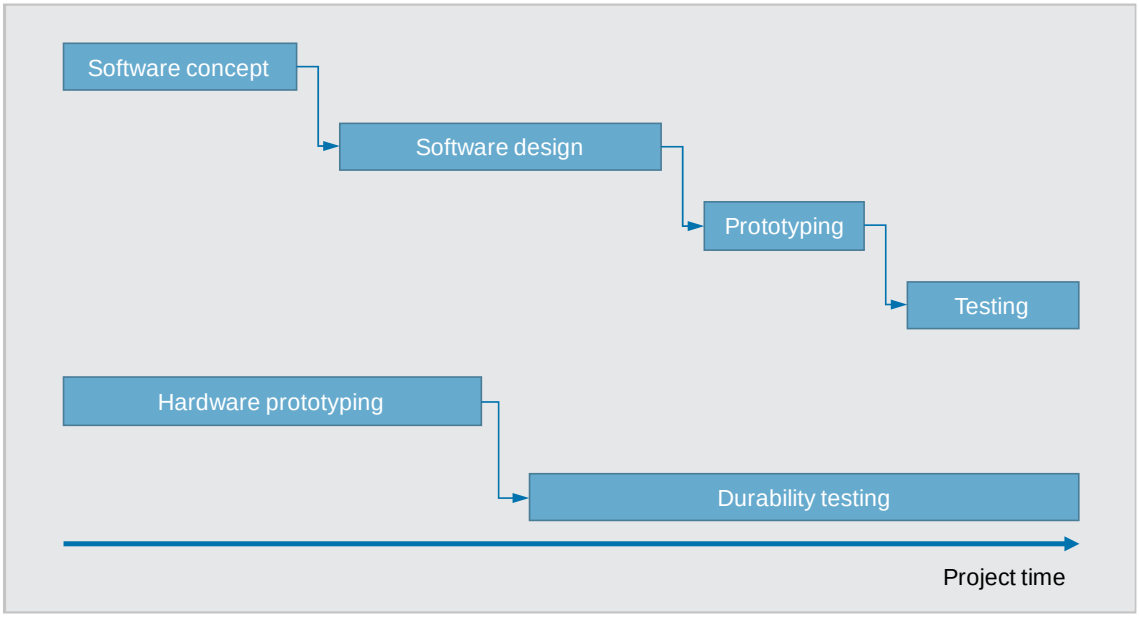


Figure 3.9 Waterfall planning principle in project management

3.7.2 Agile Management Approaches

The traditional project management has been proven to be successful in the automotive industry. But what if the environment changes so quickly that it is not yet clear what the final product should look like at the beginning? Digitalization brings a fast pace of change that forces a rethinking. It places the customer in the focus of the process. Products are tailored directly to the customer and no longer universal. That's why one no longer refers to projects but to products. Short development cycles are the key, because it allows to react quickly to the flexible market and get direct feedback from the customers or involved project partners.

In software development, the term "agile" was introduced in 2001 when the agile manifesto was formulated. Agile methods are defined as an incremental software development method, in which small software releases are developed in fast cycles with cooperative collaboration between customers and developers. The methodology itself should be easy to learn, modifiable and highly adaptive. Methodologies that were described as agile proved to be one way of making software development processes flexible. The trend towards agility is a fact in software development today. More recently, the lean paradigm was highlighted as an alternative to increasing the efficiency of software development processes. Lean is based on the fundamental principles of industrial engineering and is characterized by a philosophy of maximizing value and reducing waste. Lean is often seen as a continuation of agile in software development when agility is not sufficient enough. Unlike other software engineering topics that are designed in science and then transferred to industry, agility and lean are usually developed directly in the industry. Agile software development practices have been widely accepted by the software sector as an instrument to improve flexibility and create innovation. However, this approach is relatively new to the automotive industry and today still in the progress of implementation [18].

Examples of agile development methods in software engineering are i.e. Kanban, Lean, Scrum or extreme programming. Scrum is the most popular method of the named ones, which is already being used in the field of software development in the automotive industry. **Figure 3.10** illustrates the development process according to scrum methodologies. It is a very flexible and adaptable approach to development. The scrum team is moderated by the so-called "scrum master" which is not a project manager according to traditional management practices. The expert team meets on a daily basis

and targets individual and flexible goals to solve current problems in a technical but also creative way. According to [19], the process is structured in four loops:

1. *Pre-Production*: identifying use cases
2. *Vision*: focus on product backlog and product vision
3. *Sprint*: enables developers to construct product according to backlogs
4. *Validation*: use case verification and validation

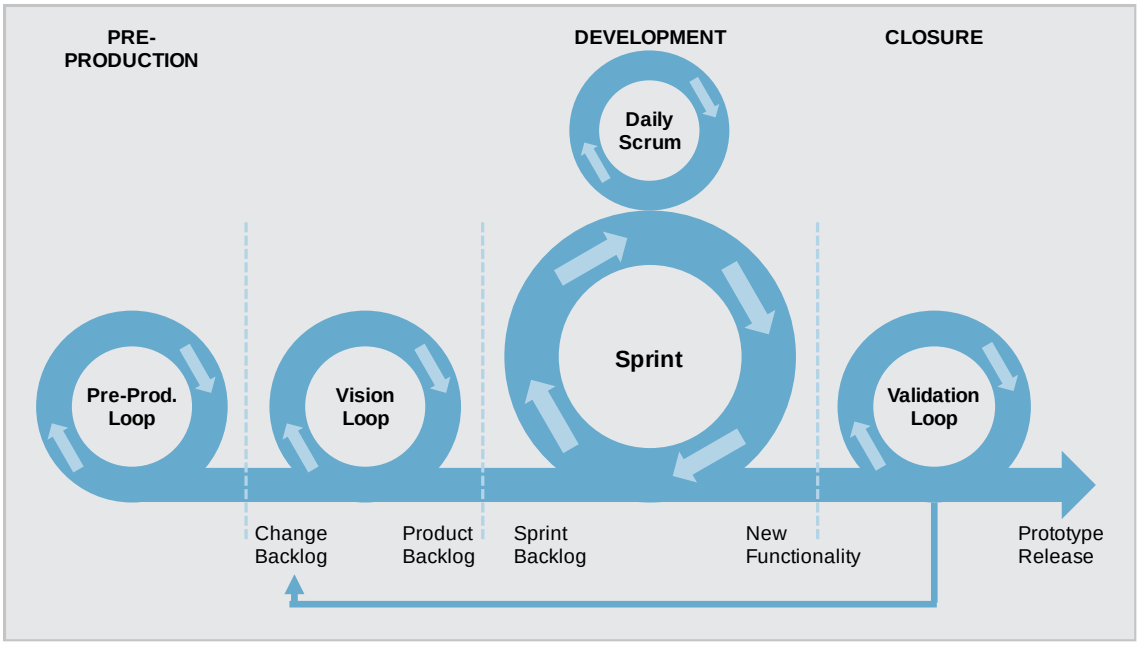


Figure 3.10 Scrum development process model, cf. [19]

To contrast traditional and agile management approaches, **Table 3.1** provides an overview of the two terms. Architecting is mostly used to describe agile development methods, while project management is the traditional way to organize and structure development projects (as described in the previous subsection).

Table 3.1 Architecting versus project management [15]

Architecting	**Project management**
Done by technical experts	Done by management experts
Technology in focus	Scope in focus
Quality focus	Cost focus
Focus on requirements	Focus on work products
Maximize functionality	Minimize costs

4 Relevant Guidelines

This chapter provides an overview of relevant guidelines and their necessity for new product development. Guidelines can be international standards, individual group standards, customer requirements or internal best practices. These structure i.e. the development processes and are of great importance for suppliers to secure commissions and be able to submit offers. That is why guidelines of various sources are the basis for the process analysis and optimization work done in the course of this thesis. In short, guidelines reflect a standardization of the development activities and processes described in the previous chapter.

4.1 Introduction to Guidelines

In the globalized world economy, products are hardly ever developed in isolation by individual companies. Companies are increasingly forced to develop their businesses in a network of global development sites, suppliers and partners. The decisive factor in this is the constantly increasing cost pressure that is driving companies to create low-cost production sites and strategic partnerships. Since at the same time the products are becoming increasingly complex and demanding and development times are shortening, two critical issues have emerged, as described in [20]:

- How to master the complex cooperation and value chains?
- How can quality, cost and schedule compliance be ensured?

This has become an essential challenge for many companies, with a direct impact on market success and growth. Systematic and controlled processes, especially for management, development and quality assurance, are a decisive success factor for these issues. Guidelines from all possible sources try to address these challenges through standardization. Processes, workflows, development outcomes and other activities are recorded in written form by guidelines for specific industries or products, thus forming a basis for successful cooperation and quality assurance. In principle, guidelines serve to standardize the development processes and phases described in the previous chapter, whereby the benefits are not only of enormous added value for development and quality assurance in the globalized world. Thus, cooperation, complex development interfaces as well as project and target compliance can be ensured.

4.1.1 Necessity of Guidelines

In comparison to laws and legislation, standards and other guidelines are not a legal requirement for companies. Nevertheless, or precisely for this reason, standards play a very important role today when it comes to the development of new products, especially with distributed global product development with a complex value-chain. Standards and company-specific guidelines and requirements are intended to document the current state of the art in technology. This is therefore merely a technical recommendation, which is not legally prescribed for the company. **Figure 4.1** illustrates this difference between laws and regulations and the technical recommendations, i.e. international standards and specific guidelines, for companies in the automotive industry. Laws are legally binding, and the company and its developed products must comply with all necessary directives and regulations to get new developed vehicles approved for the road. On the one hand, standards and guidelines provide a basis for development activities and quality assurance. On the other hand, these technical recommendations are of great importance when it comes to product liability issues. In the event of a failure of the product in the field – especially with associated physical injury – the product liability law applies nationally. In this context, it is important for the respective company to have developed and produced its own products to the best of its knowledge and, to a certain extent, according to the current state of the art. This state is usually recorded in standardization processes. For this reason, these topics must also be considered in advance in the new product development process, so that no legal proceedings or claims for damages have to be paid later on in the event of incidents.

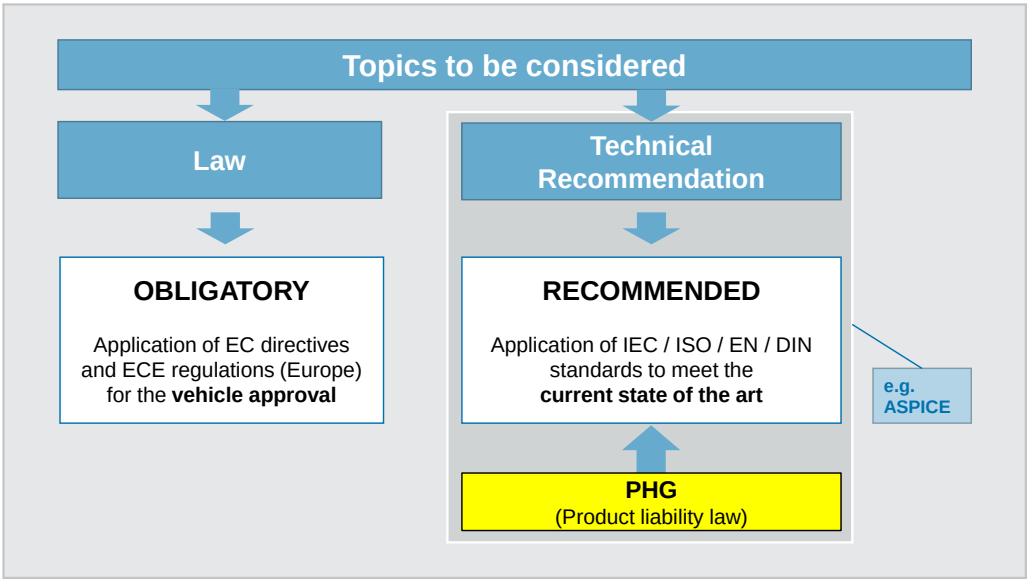


Figure 4.1 Difference and importance of laws and standards for companies

In addition to these important issues, which have to be taken into account due to product liability and vehicle approval, the necessity of guidelines in distributed product development is also essential. OEMs and all other involved companies in the supply chain must adhere to the objectives of the OEMs' standards, because only then can the developed product meet the objectives of the defined standards. In addition, each manufacturer, especially at the top of the value-added chain, has its own proprietary manufacturer-specific requirements and guidelines, which must be applied both internally and by the corresponding suppliers. Companies require suppliers to act in accordance with their own developed standards, so that development activities can be coordinated with their own project structure and development work. If implemented successfully, this facilitates a smooth project progress in development and also secures a standard for corresponding product quality, communication and data exchange. In today's globalized world, such a seamless process and exchange between equal development partners is essential to develop and launch a successful product under increasing time and cost pressure.

In addition to standardization, the dissemination of cross-disciplinary expertise for general economic growth and innovation is a further necessity for standards. For continuous economic growth, it is not enough to create new knowledge through research and development, but it must also be spread widely so that it can be applied by as many companies as possible. Standards developed in consensus by companies are particularly suitable for disseminating technical knowledge. Standardization experts document the current technological level in standard documents and thus enable broad diffusion on the market. In contrast to patents, which are subject to intellectual property rights, the knowledge codified in the standards is freely accessible to everyone and thus its distribution is not restricted. The dissemination or diffusion effect of the standards on technological knowledge and the associated contribution to continuous economic growth have already been highlighted in past studies [21].

4.1.2 Structuring Approaches

Typically, all agreements and regulations can be set out in guidelines. However, international standards define exactly what is meant by standardization work. In this context, standardization describes the activity of defining specifications for general and recurring applications that relate to current or foreseeable problems and aim to achieve an optimum degree of order in a given context. The main advantages of standardization are the improvement of the applicability of products, processes and services for their

intended purpose, the elimination of barriers to trade and the improvement of technical standards. The basis for this is a developed stage of the technical possibilities at a certain point in time, as far as products, processes and services are concerned, based on the corresponding reliable knowledge of science, technology and experience [22].

A structured overview and appropriate division is important for a systematic standardization and documentation of the necessary information. Only then is a successful adaptation and implementation into existing processes of a company possible and effective. This is precisely why many international standards are based on one another or complement each other. Normally, familiar models are also the basis for structuring standards and guidelines. In particular, the V-model of the development process presented in chapter 3.3 is the basis of many international technical standards, in particular Automotive SPICE and ISO 26262 in the automotive sector. This makes it easy to understand and implement in the company, as well as a combination or adaptation of existing processes in order to meet as many guidelines as possible.

In particular, customer requirements or group standards of the manufacturers are based on existing national or international standards. To this end, it is important to know that such national or also international standards often emerge in working groups in a long process, with representatives of participating companies, industry representatives and members from politics taking part in standardization work. Nevertheless, the results of the standards are interpreted and implemented differently in the individual companies. Precisely for this reason, manufacturers formulate their own requirements and guidelines for themselves and their suppliers in order to implement their own expectations.

Companies and organizations work together to standardize current topics and approaches for problem solving in order to be able to represent a state of the art in technology. This means that standards can either apply nationally, internationally or regionally, e.g. standards by the *International Organization for Standardization* (ISO) or the *Deutsches Institut für Normung* (DIN); be industry-specific, e.g. by the *International Electrotechnical Commission* (IEC); or be sector or product-specific, e.g. by the *Verband der Automobilindustrie* (VDA) or by the more general *Verein Deutscher Ingenieure* (VDI). These organizations usually consist of many industry-specific representatives from the individual companies – which are also competing on the free market – or of associations representing the interests of the whole industrial sector. Often, these groups also exert influence through lobbying to enforce the interests of companies in national or international legislation.

4.1.3 Assessment of Guidelines

In order to ensure that companies – and in particular their suppliers – work according to applicable standards, guidelines and requirements, the implementation and enforcement is evaluated by the manufacturer or a third, independent authority. Such evaluations and ratings are conducted during so-called assessments, which can take a day to a few weeks and are usually ordered by the OEM respectively the customer of the supplier. In some cases, suppliers also undergo an assessment by themselves to obtain certification that they have implemented current industry standards and are working according to their requirements. These assessments are usually done by others, independent institutions, customers or sometimes also competitors. As a rule, all cases are specified in the individual standards as to who can or must assess what and when. Not only the entire application or implementation of standards can be assessed but also defined work products or outcomes, especially in relation to functional safety (ISO 26262). Suppliers in particular require such certifications and ratings from assessments as a prove of capabilities, to get customer assignments and to be able to quote offers for development or production projects. This is acknowledged by either a certificate or a certain rating to which level the supplier is capable or reliable.

4.2 Applicable Standards

This section provides a brief introduction to relevant standards for product development in the automotive industry. This work is focused on the optimization of development processes based on Automotive SPICE, but for the sake of completeness the most important standards are presented, as an implementation of these standards can further improve the processes in the future.

Table 4.1 provides an overview of main approaches to evaluation in the fields of mechatronics and software development. Safety-critical products can be evaluated from different perspectives. The key issue, the software product, can be evaluated using a predefined set of quality requirements, for example. The safety assessment can examine both the product and the processes used in the development and use of the software, which are often based on domain-specific standards. The process evaluation usually focuses on the product development phase. All of these approaches provide valuable information to build confidence in the safety of the product and are key for the company to provide and demonstrate the capabilities and trust to the customer [23].

Table 4.1 Comparison of main approaches in evaluation [23]

Topic	Product evaluation	Safety assessment	Process assessment
Main purpose	To analyze and show compliance of product	To demonstrate compliance with a selected reference	To demonstrate capability to develop, deliver and improve
Main focus in safety-critical domain	Product quality, especially reliability metric and data	Compliance with generic or domain specific safety standard	Process evidences to demonstrate achievement of safety management and engineering
Specifics	Internal, external, in use metric	Inspections, reviews, V&V evidences, technical practices and methods	Professional practices, work products, capability levels
Typical standards	ISO/IEC 25000 family	*ISO 26262*, IEC 61508, IEC 60880	*ISO/IEC 15504 (SPICE), Automotive SPICE*

4.2.1 IATF 16949 (Quality Management)

The IATF 16949 – known before 2016 as ISO/TS 16949 – combines existing general requirements for quality management systems of the (mostly North American and European) automotive industry. It is based on ISO 9001 (Quality Management) and has been adapted and expanded for specific industries. Based on the ISO 9001 standard, several industry-specific standards were developed at the end of the 1990s, taking into account the supplementary requirements of the respective industries. These niche standards were mostly the result of quality agreements that dominating market players (e.g. car manufacturers) demanded of their suppliers. The development was favored by the fact that, based on such individual agreements, industry associations also issued quality standards parallel to or complementary to ISO 9001. The IATF 16949 is the most basic standard for quality management and every company in the automotive industry must comply with these requirements in order to work with customers and suppliers. It

lays the foundation of quality management in the industry and defines how products must be developed and produced in order to meet the high demand of quality in this industry. This standard is well implemented and also a basis of more complex and newer standards [24].

4.2.2 ISO 26262 (Road Vehicles – Functional Safety)

As already descried briefly in chapter 2.4, ISO 26262 was published in 2011 as an international standard for the development of safety-critical electronic systems for automobiles, and is being used more and more widely all over the world. It is based on the more general IEC standard 61508, but contains car-specific refinements. This risk-based standard recognizes that a risk cannot possibly be reduced to zero. But it requires that the risks be qualitatively assessed, and action taken to reduce them as far as is reasonably practicable. It is also based on the successful implementation of the IATF 16949 (Quality Management) and structured analogous to the V-model development approach. This standard is becoming increasingly important because it focuses on the development of safety relevant mechatronic systems in automotive applications. The most important term to understand is the Automotive Safety Integrity Level (ASIL) as a risk classification of an element of electronics. Level D stands for parts with the highest risk, A for the lowest risk. Additionally, the QM level indicates a risk level below ASIL A.

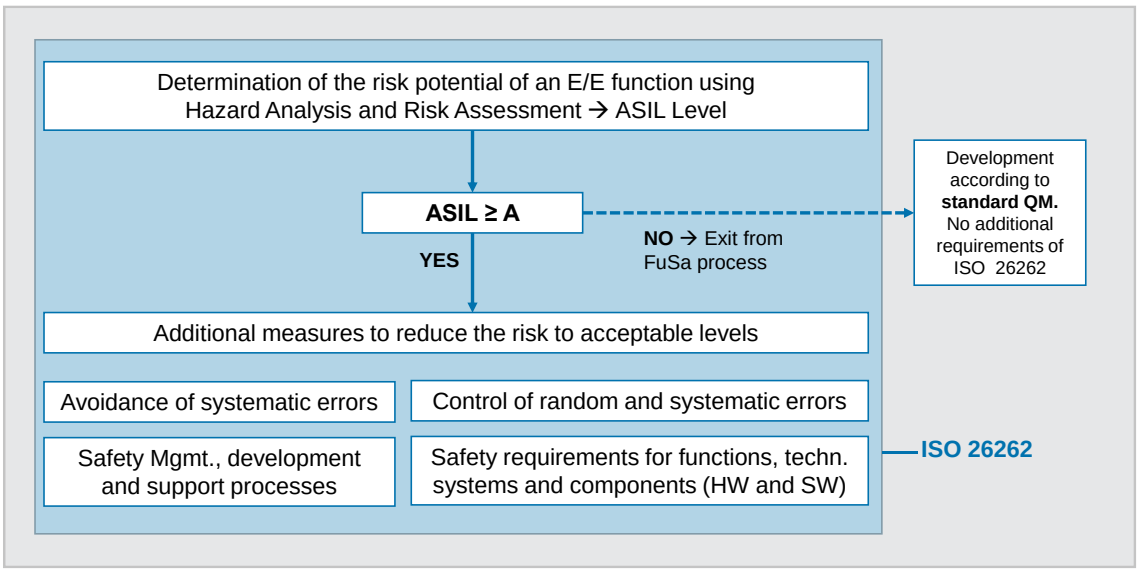


Figure 4.2 Application process of ISO 26262, cf. [25]

Figure 4.2 provides an overview of the application process of ISO 26262 for the assessment of functional safety, in particular the ASIL evaluation. The allocation of the risk class is based on an evaluation process to which the hazards are subjected. Every potentially dangerous event is classified according to *severity* (S) – the severity of the injuries it can cause. S0 stands for “no injuries” and S3 for “danger of death”. The other important factors in the evaluation are the *exposure* (E) with a range from E0 (“extremely low probability”) to E4 (very likely) and the *controllability* (C). The latter indicates the extent to which the driver can intervene to prevent injuries (C0 stands for “simple” and C3 for “difficult or impossible”). The ASIL value is determined using the common consideration of all these factors by creating the total sum. It goes without saying that a hazard with high values for S, E and C is classified in the ASIL D category. Nevertheless, a hazard with a high S-value can be classified in the category ASIL A if it occurs with a very low probability. If an ASIL value is determined for a hazard, this value is used to the safety goals that reduce the hazard and the safety requirements derived from them. The ASIL value then determines the minimum test requirements for system verification. If during the hazard analysis and risk assessment process, the values for S, E and C are so low that no ASIL level can be assigned, the QM level applies. This means that a “normal” development according to quality management is allowed for this failure. The IATF 16949, which was described in the previous section and is also a prerequisite for the implementation of ISO 26262, then applies [26].

4.2.3 ISO/IEC 15504 (SPICE)

The ISO/IEC 15504 standard – often simply referred to as SPICE – is a five-part international standard for the evaluation and improvement of a company’s software development processes. Concepts and vocabulary, requirements for carrying out process assessments are defined as the basis for process improvement and maturity and capability level determination. The most important part of the standard is the exemplary process assessment model (PAM). This PAM uses ISO/IEC 12207 as a process reference model (PRM). Most SPICE assessments are carried out using this assessment model. Further PAMs exist for certain domains, e.g. in the automotive sector as Automotive SPICE, which is described in detail in the next chapter and forms the basis of the process optimization described in this thesis. In general, the process model in the SPICE standard describes a variety of development and supporting processes for the development of systems and software. Each process, grouped according to work packages from the V-model, is described in detail with the purpose and defined

outcomes. Additionally, practices and work packages are described, which are relevant or necessary for the corresponding process phase [27].
In addition to determining the process maturity, the model also enables the identification of potential improvements for individual processes. For each process, a development path is given along the capability levels from level 0 to level 5. For each capability level, the model includes practices that must be present for the process in order to achieve a higher capability level. Furthermore, however, the model gives no indication of how these practices can be implemented. By unambiguously assigning the processes of the maturity model to the processes of the process reference model from the ISO/IEC 12207 standard, further information on the required practices is available to the model user. In addition to the maturity model, ISO/IEC 15504 includes another guideline for conducting an assessment, which also addresses the possibility of self-assessment. Moreover, the standard also offers the possibility of integrating other process reference models and thus developing individual maturity models [28].

4.3 Customer Requirements

Customer requirements can be group standards, general guidelines of the respective OEMs (customers), or product-specific customer specifications. Generally, these are binding for the contractor and all deviations must be approved by the client (OEM). Customer standards, in particular the so-called group standards, are generally based on international standards that have already been published and indicate how the manufacturer interprets the requirements of the standard. This is interesting because the major manufacturers have usually been actively involved in standardization work on the new standards, but they still interpret them differently. Group standards and customer requirements often tighten the requirements in the published standards, which poses a further challenge for suppliers. In the case of project tenders, all product specifications as well as the applicable group standards and requirements are distributed to the tenderers (suppliers), who are then given the opportunity to submit an offer for a specific project. In order to be considered as a supplier for a company, all requirements must be able to be fulfilled and all applicable group standards must be implemented in the company. The certificates and ratings are necessary for this purpose through the aforementioned audits or assessments.

In the following sections three used specifications respectively group standards are presented, which have been included in the work to optimize the development processes in the process model. As a matter of principle, suppliers try to cover as many different standards and customer requirements as possible by means of their own processes and guidelines. Typically, such customer requirements are structured in a similar way to standards or laws having defined IDs for the individual requirements to ensure a reference and traceability.

4.3.1 OEM A

The customer requirement *A* is a group standard of OEM A. It sets out the basic requirements that the entire group imposes on the software installed in the vehicle and close to the vehicle and its development processes. The described standards are aimed at defining and specifying the requirements of OEM A with regard to software quality in the vehicle. These basic requirements contain the minimum requirements for the software product and its development process that apply to all types of software (e.g. application software, drivers, standard software etc.). The defined verification measures are intended to provide a transparent representation of the degree of fulfilment of the requirements by the contractor. This standard is structured analogously to the V-model in terms of content and is therefore compatible with all relevant processes described in automotive spice. It is an important requirement standard, because it is highly relevant for suppliers in terms of unit quantities.

4.3.2 OEM B

The group standard *B* applies to the development of embedded software at OEM B and its suppliers. It deals with the development of safety-related and non-safety related embedded software in vehicles, basic and application software as well as embedded software over the entire lifecycle, i.e. before production start and after that until the end of maintenance. This standard is said to be a recommendation. However, if reference is made to this standard within a binding document (e.g. a customer requirement for a development project), this group standard becomes binding. In contrast to OEM A's standard described previously, OEM B's standard is not structured in the same way as Automotive SPICE. This makes direct mapping in the process optimization more challenging. The current version is based on the standard ISO 26262 – Functional Safety described above. A successful implementation and operation according to ISO 26262 is

also a prerequisite for a successful implementation of this standard. According to OEM B, ISO 26262 is essential for the application of this group standard. This means that less of the contents and requirements can be allocated directly to the Automotive SPICE requirements. A future extension of the optimized process model to include ISO 26262 has the potential to integrate the entire standard of OEM B.

4.3.3 OEM C

OEM C's group standard is also a supplementary document to product-specific component specifications. It describes OEM C's cross-component general requirements for the provision of services within the scope of component development or series production by the supplier. The component-specific requirements for the development or series production of parts, modules, software or components are part of the respective technical component requirement specifications. This standard is a very comprehensive document, as it sets requirements for the E/E and mechanical components and their development processes in addition to software requirements. The standard also exists in different variants. In addition to the present standard for E/E, software and mechanical components, it is also available for pure software scopes, pure mechanics scopes and pre-assembly scopes. Due to the large coverage, only a small portion of the given specification on the topic of software is relevant for process optimization and mapping of processes and base practices respectively requirements. In the future, however, other aspects can be added to the process model when it is expanded, which is taken into account in the process model concept phase.

4.4 Best Practices

Within the framework of this master thesis, all internal activities, regulations and processes of the company or supplier on the automotive market are defined under the term of *best practices*. Similar to the OEM group standards, these are based on existing standards, but are adapted to the respective company and its subcontractors. Usually, these internal processes have emerged and evolved over a long period of time by adapting to new standards, customer requirements or results from the field of research and development. These best practices are very industry and product-specific, as a result of the company's know-how. Best practices are not only all internal processes, but can also include common practices that are followed as if they were unwritten laws in the company. Usually best practices also include the knowledge of past projects and

lessons-learned. Best practices are relevant for process optimization in so far as on the one hand it is possible to identify the need for optimization on the basis of the internal processes, and on the other hand, the knowledge of one's own best practices also allows an estimation of the optimization effort. Internal processes of automotive companies are usually also structured analogous to the V-model for the development of system and subsystems. Besides the base processes for the development activities, there are also supplementary processes along the entire product lifecycle to control, steer and support the main activities.

5 Automotive SPICE

Automotive SPICE (ASPICE) is a domain-specific variant of the international standard ISO/IEC 15504 (SPICE) described in the previous chapter. The purpose of Automotive SPICE is to evaluate the performance of the development processes of electronic control unit suppliers in the automotive industry. SPICE is the abbreviation for “Software Process Improvement and Capability Determination”. This chapter provides an overview of the VDA standard Automotive SPICE, which is the basis for the following process optimization.

5.1 Introduction to Automotive SPICE

The Automotive SPICE standard, derived from the ISO/IEC 15504 (SPICE) standard, is an international standard that is used worldwide in large automotive companies as a framework for the assessment of processes. Automotive SPICE can be perceived as a representative software process evaluation model because assessors evaluate indicators and metrics that measure the performance of software processes. It is a reference for the maturity models, which specifies requirements for process reference models and process evaluation models similar to SPICE. This reference model comprises several key components, namely: some lifecycle processes from several process categories for the process dimension and six skill levels for the capability dimension. The basis for the creation of products by the company is the process. The capabilities associated with their process attributes relate to the company's ability to produce these products both predictably and consistently. It includes a series of process performance and process capability assessment indicators on the basis of which objective assessments are collected that enable an assessor to assign ratings [29].
In principle, the process assessment model is a collection of best practices for the automotive industry. However, this is a model and not a simple listing of the processes. It provides the user with a tool to evaluate and compare their processes and those of their suppliers. Methods, workflows and outcomes are commonly used for this procedure. Automotive SPICE refers mainly to the development of mechatronic systems in automotive applications. The content of Automotive SPICE focuses fundamentally on system and software applications. The basic idea is the standardization and definition of all processes, their outcomes and work packages to successfully develop a product. In each process, the individual base practices are presented, and further generic practices

are introduced. In addition to the classic development processes in the V-model, the standard Automotive SPICE also offers a framework for the entire product development process. This includes defined processes for project management, requirements management, configuration management, risk management, supplier qualification and acquisition. In order for a vehicle manufacturer to award a project to a supplier, these suppliers must certify a certain degree of maturity in terms of process quality. This is determined by assessments of the OEM or third parties based on defined processes and metrics [30].

5.1.1 Relevance for Automotive Applications

Since the publication of a first process reference model by the Automotive Special Interest Group (AutoSIG) in 2005, Automotive SPICE has established itself among companies engaged in the development of software-based systems in the automotive sector. The software development for mechatronic systems is traditionally characterized by the mentioned V-model. In contrast, Automotive SPICE is specifically tailored to the needs of the development of ECUs in the automotive sector. It has been developed by users in the automotive industry and is recognized as a definitive ISO/IEC 1554 compliant process model. For the evaluation of software development processes in the supply chain, European automotive manufacturers rely on Automotive SPICE. Practical application in supplier assessments has resulted in procedures that have been developed by a VDA working group on a guideline for conducting assessments. Here, the procedure of assessments is modeled as a kind of own process and thus simplifies the implementation of the standard. In practice, Automotive SPICE is part of the quality management system of car manufacturers and suppliers. On the OEM side, there is often a supplier evaluation strategy that is coordinated with the procurement department and a strategy for monitoring and safeguarding ongoing projects that is coordinated with the engineering department. An embedding of supplier assessments according to Automotive SPICE within the framework of a comprehensive degree of maturity assurance makes sense in many cases. The aim is to determine the process maturity of the respective supplier in the complete chain of the development process of the respective component – from the first idea to series production readiness – and to ensure that it is fully operational. On the supplier's side, there is usually already a coordinated quality assurance strategy for software based systems development. Results from Automotive SPICE are also embedded in the process improvement measures. An additional challenge for safety-critical systems includes the functional safety

requirements of ISO 26262, which nowadays have a very high priority and must also be considered [31].

5.1.2 History of the Standard

The first maturity model that was widely used was CMM in the early 1990s. CMM has never played a significant role in the automotive industry, even though a car manufacturer tried supplier evaluation approaches for a short period of time. The SPICE compatible BOOTSTRAP was used by a few pioneers among automotive suppliers, but was never able to prevail over SPICE and was discontinued in the year 2003. SPICE originated from an ISO project of the same name and was published in 1998 as ISO/IEC TR 15504, whereby TR (Technical Report) represents a preliminary stage to a later international standard. The various parts of the International Standard ISO/IEC 15504 have been published successively since 2003. In 2006, the most important part of ISO/IEC 15504 was published, and in 2012 a new version was released. In 2008, part 7 “Assessment of organizational maturity” was published, which defined the normative basics of organizational assessments. In contrast to the usual project assessments, the maturity of an organization can be assessed by a larger number of random samples. Several pilot assessments have so far been carried out successfully on this basis. This methodology has not yet established itself widely. ISO/IEC 15504 has been successively transferred to the ISO/IEC 33000 family since 2015 [20].

As described previously, the industry-specific standard Automotive SPICE was developed on the basis of the mentioned international standard ISO/IEC 15504 (SPICE). Starting in 2001, Automotive SPICE was developed by the Automotive Special Interest Group (AutoSIG), which includes the automobile manufacturers Audi, BMW, Daimler, Porsche, Volkswagen, Fiat, Ford, Jaguar, Land Rover, Volvo, the SPICE User Group and the Procurement Forum. Today, Automotive SPICE is a registered trademark of the VDA. The starting point was an increased complexity and functionality in the automobile, which resulted from the increasing use of software functions. As a result, OEMs were forced to evaluate their suppliers on the basis of software capability and quality. To this end, each OEM initially had its own guidelines and approaches to evaluate the ability and maturity of its suppliers. The resulting problem was that the individual manufacturers had to meet the sometimes very complex requirements of several manufacturers, but at the same time they had to be able to cope efficiently and with well-coordinated internal processes and best practices. This quickly led to the need for a standardization of

software capability and process maturity requirements in order to achieve a uniform evaluation and implementation in the industry [30].

The breakthrough for the use of maturity models in the automotive industry came in 2001 with the decision of the German software manufacturer initiative “Herstellerinitiative Software” (HIS) to use SPICE for supplier evaluation in the software and electronics sector. Members of HIS are the German OEMs Audi, BMW, Daimler, Porsche and Volkswagen. From this time on, SPICE spread throughout the entire automotive industry. One of the great advantages of SPICE is its ability to develop industry-specific models under a common normative framework. In 2005, AutoSIG published the Automotive SPICE model, replacing the former generic SPICE standard. Automotive SPICE is now being further developed by the working group 13 of the Quality Management Center (QMC) within the VDA. Members of this working group are comprised by employees of the OEMs of the Volkswagen Group, Daimler Group, BMW Group, Bosch Group, ZF Friedrichshafen, Continental, Brose, Ford, Schaeffler and Knorr Bremse [20].

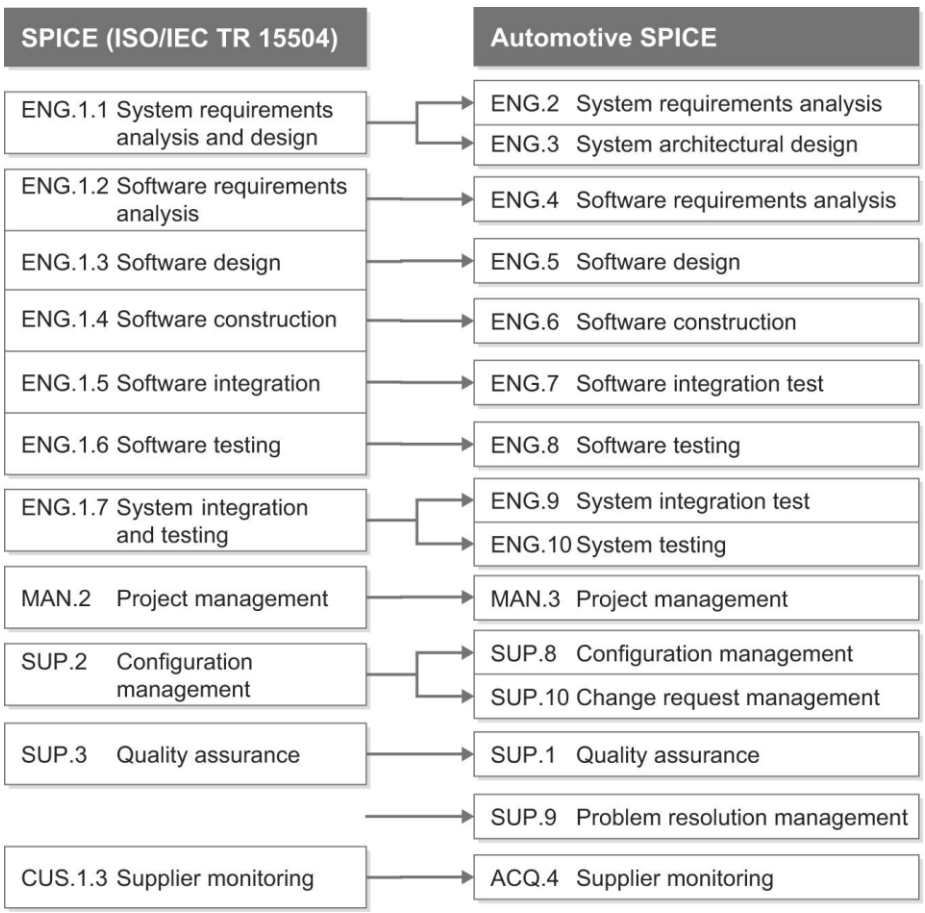


Figure 5.1 Derivation of HIS process scopes for ASPICE from ISO 15504 [20]

Figure 5.1 provides an overview of the mapping between the defined processes from the original SPICE standard ISO 15504 to the first version of Automotive SPICE. Most processes are directly related to the successful SPICE standard with additional notes and changes for the domain specific variant. Other processes (e.g. ENG.2 and ENG.3) are the result of a divided process to define each phase with their respective outcomes and work products in detail. On the other hand, processes like SUP.9 (problem resolution management) were newly added to the Automotive SPICE standard. There are more processes and contents of Automotive SPICE than illustrated in this figure, which is described in the following subsections.

After a few years of version maintenance, version 3.0 was released in 2015, which brought with it a number of structural changes in addition to further developments in terms of content and adjustments which increased the project effort. **Table 5.1** contrasts the major overall changes between Automotive SPICE 2.5 and Automotive SPICE 3.0. The most important innovation was the splitting of all engineering processes (ENG) into the two groups of systems engineering processes (SYS) and software engineering processes (SWE). This makes the process assessment model even better suited to the common V-models and a separation between these areas is recognizable. The second important innovation was the splitting of the processes that affect the unit, the smallest software element. In version 3.0, the former process was split into two new ones. One process for the unit construction and one process for the unit verification. The new plug-in concept allows the integration of hardware and mechanical processes, which are not provided by the ASPICE standard. The content of the HIS-Scope, apart from minor name changes, remains largely unchanged. Some changes cause additional effort from projects, e.g. the evaluation of alternative solutions for architectures [20, 32].

Table 5.1 Overview of major changes between version 2.5 and 3.0, cf. [33]

Automotive SPICE 2.5	**Automotive SPICE 3.0**
ENG (Engineering Processes)	SYS (System Engineering), SWE (Software Engineering)
One process for unit construction and unit verification	Unit construction process and additionally a unit verification process
No integration of HW and mechanical processes	Plug in concept allows integration of HW and mechanical processes
Known process names	The names of some processes have changed

A further change was the division into separate base practices of the individual processes for consistency and traceability. This means that there is a clear traceability for every process, similar to the described V-model. It is now clear that the designed software units must be verified and evaluated on the basis of previously defined requirements. Such a model is presented in more detail in the course of this chapter.
Another improvement is the refinement of the requirements for the work packages and their characteristics. It requires a further independence, but the objectivity is now given in ASPICE 3.0. Since individual reviews of the work packages cannot be carried out independently, this increases quality assurance. The dynamic behavior of the software architecture was also addressed in architectural design. In this context, alternative solutions or concepts must also be evaluated on the basis of defined criteria. A further increase in traceability and consistency is reflected in the test processes. These now require that a selection of the test cases must be made on the basis of the test strategy defined in advance for the individual test steps. In the course of this master thesis, the developed process model contains a mapping of all processes and base practices of Automotive SPICE 2.5 and 3.0 and highlights the deviations of the HIS-scope [33].
In order to limit the scope for interpretation of the new Automotive SPICE 3.0 standard, a new gold/blue edition was developed by the VDA. The first draft version was published in February 2017 as a yellow volume entitled “Automotive SPICE Guidelines”. The final version was released in April 2017 and contains binding rules and recommendations for the transition to Automotive SPICE 3.0 as well as its interpretation guide and instructions for assessors. This guideline was approved by the VDA Quality Management Board and is therefore binding. Further details of the assessment of Automotive SPICE will follow in the course of this chapter [32].
Despite the fact that version 3.0 of Automotive SPICE was already released in the year 2015, both versions (ASPICE 2.3 and 2.5) may still be used and Automotive SPICE 2.3 is still the version, which is considered mandatory by the VDA [33].

5.1.3 Process Capability Level

The process capability level is a central idea of the SPICE standard. This level enables the user to evaluate the process capability and improve it accordingly. Especially in the automotive industry, Automotive SPICE offers a comprehensive and standardized evaluation method for OEMs and Tier 1 suppliers. The suppliers attest to the customer a certain degree of process capability or maturity in previously defined processes from Automotive SPICE. This is necessary in order to get an assignment from the OEM for a

development project or part of it. This process capability level is determined in supplier assessments and put to the test. In this context, compliance with the selected processes is decisive. In order to achieve a certain level of process capability, defined model elements must be implemented in the own process environment. Automotive SPICE defines six levels of process capability – from level 0, being the lowest, to level 5, the highest level of process capability [30].

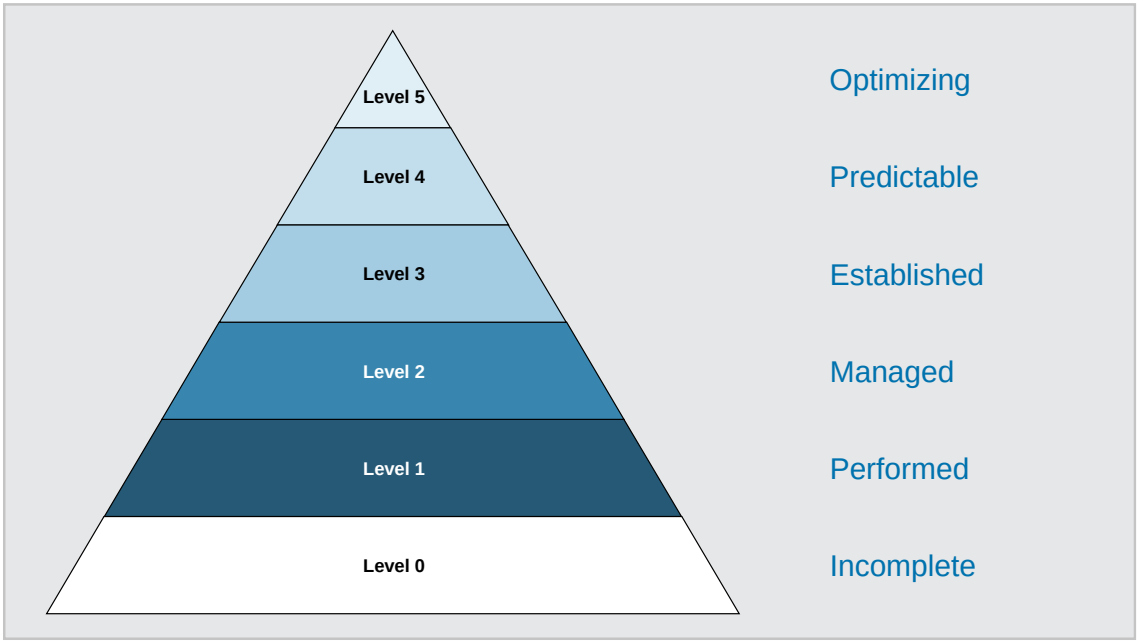


Figure 5.2 Process capability levels of ASPICE acc. to ISO 33020, cf. [34]

Since Automotive SPICE is not only concerned with adherence to required processes, but also with maturity and process capability for improvement and process innovation, the degree of fulfillment is described in analogy to **Figure 5.2** according to [35]:

Level 0 describes an incomplete process. The process is not implemented, or the purpose of the process is not fulfilled. Project successes are quite possible, but are a result of coincidence.

Level 1 is defined as a state in which everyone knows what to do. This can be described with the base practices. Each process has its own base practices that should be mastered at level 1. The implementation of these practices produces intermediate results, the work products (WP). As a rule, these work products are individual and traceable documents.

Level 2 means that everyone knows what is good and what is not good. In particular, there are document templates and checklists for the work products to be created in the

project. The preliminary results are reviewed and placed under configuration management. There are some ideas on how the process can be improved during the course of the project. The responsibilities have been clarified and the team members receive the right training in terms of their role in the project. All measures and work steps are planned, and their observance is checked. These are no longer individual base practices of the individual processes, since a large number of generic practices must be fulfilled from level 2 onwards.

Level 3 describes the realization that it is good to write down positive experiences made. This is done in the form of process definitions. This enables improvements to be made across project boundaries. The process definitions are continually adapted for use in projects, so that there is also room for improvement here. In particular, lessons learned from past projects are included.

Level 4 assumes that the defined process has already been executed many times and therefore figures and data are available that describe the process execution. Statistical considerations can be made about the implemented processes and defined upper and lower limits for the individual process parameters exist. This enables preventive process maintenance.

Level 5 describes the continuous improvement of processes. Technological innovations are explored, usability is evaluated with the applied process and implemented for optimization. This level has the highest requirements and is rarely attested.

The determination of the performance capability, and thus the assignment of the individual levels described previously, is based on so-called process attributes. Process attributes are characteristics of a process that can be evaluated on a service scale and represent a measure of the process's performance. They are applicable to all processes related to Automotive SPICE. Process capability determination using a process assessment model is based on a two-dimensional model. Processes defined in a process reference model provide the first dimension. In the second dimension, the ability levels are divided into process attributes. These process attributes return the quantifiable properties of the process capability. A detailed overview of the assessment and classification into the above levels is presented at the end of this chapter [30, 34].

Based on experience and discussions with the industrial partner, most companies achieve a maximum of process capability level 3 at the assessment. Everything else requires a disciplined and elaborate design of the processes. In addition, ongoing customer requirements for the assignment of a project require at least one assessment,

in which level 2 is achieved and an effort for improvement is evident. The results of the assessment may differ depending on the customer or OEM concerned. The work in the context of this thesis is limited to the comparison and deviation analysis at process capability level 1 of Automotive SPICE. From the higher levels onwards, generic practices would also be required, which would exceed the scope of this master thesis.

5.2 Scope of Automotive SPICE

Basically, Automotive SPICE covers the entire scope of the product development process, analogous to the V-model. Processes are grouped by category and, since Automotive SPICE 3.0, are sorted into process groups according to activities. There are three process categories in the scope of Automotive SPICE according to [34]:

1. Primary Life Cycle Processes
2. Organizational Life Cycle Processes
3. Supporting Life Cycle Processes

In these categories, the individual processes are further subdivided into the following application-related groups, according to [34]:

- Acquisition Process Group (ACQ)
- Supply Process Group (SPL)
- System Engineering Process Group (SYS)
- Software Engineering Group (SWE)
- Management Process Group (MAN)
- Reuse Process Group (REU)
- Process Improvement Group (PIM)
- Supporting Process Group (SUP)

Figure 5.3 illustrates the entire scope of Automotive SPICE as well as the mentioned categories and process groups using the V-model for system and software development processes. Key in version 3.0 is the clear separation between system engineering and software engineering, the processes at the tip of the V-model and the concept of bidirectional traceability and consistency. It becomes clear, that every process of the left side of the V-model (design), has a corresponding process on the right side (verification).

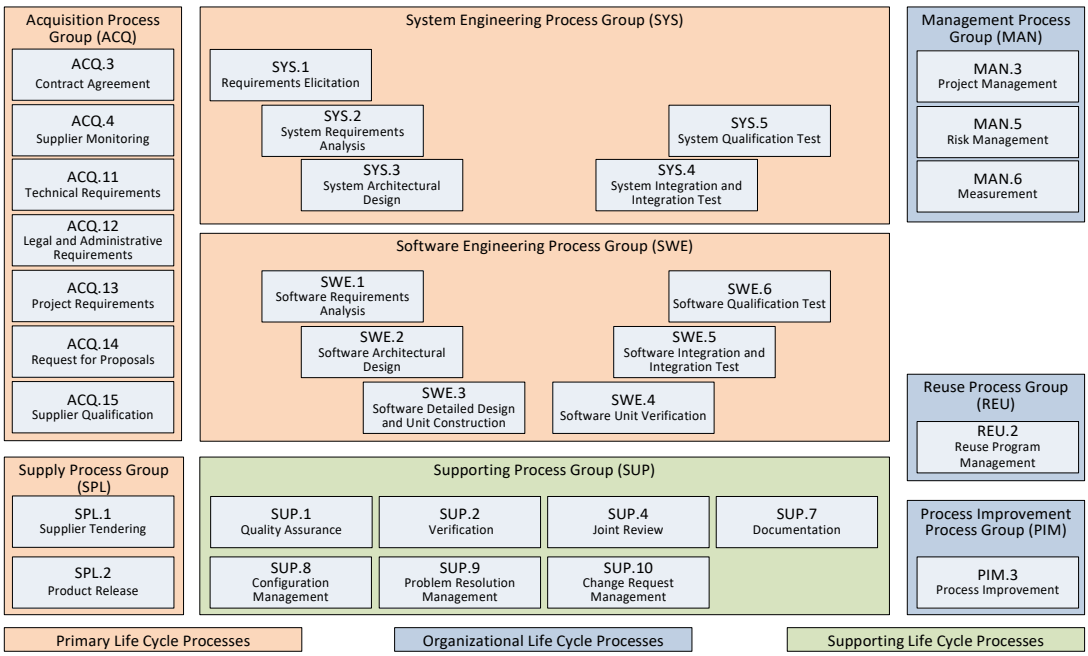


Figure 5.3 Automotive SPICE 3.0 process reference model [34]

Knowledge of terminology is also important for understanding the process model of Automotive SPICE. For this purpose, there is a separate appendix in the standard, which deals only with the terms and their definitions. **Figure 5.4** contrasts the most important central terms of the standard: *element*, *item*, *component* and *unit*. They are used consistently during the entire development process and in the PRM.

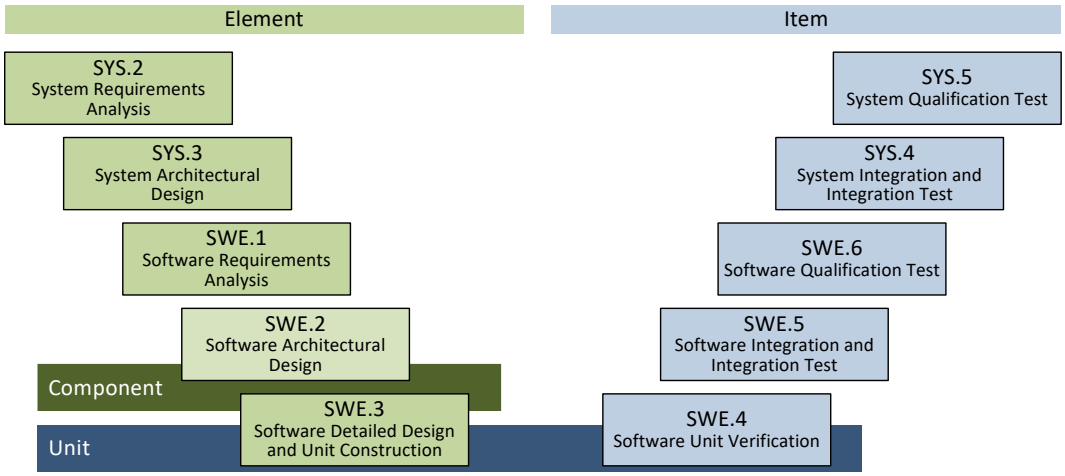


Figure 5.4 Important terminology of Automotive SPICE [34]

5.3 Components of Automotive SPICE

As already described, the standard Automotive SPICE contains individual processes. These processes are described in detail. **Figure 5.5** shows schematically how such a process is structured and defined in Automotive SPICE. Each process can be addressed individually with a unique ID and name. The purpose of the individual process is then defined. This describes the goals and the situation around the activities of the development phase according to the V-model. From this, the outcomes can be derived directly. The ID, name, process purpose and outcomes define the PRM. The outcomes are recorded and documented in the so-called work packages, individual documents. It is important to know that several outcomes are relevant for several work packages and vice versa. This is also cross-process. The work packages are described in detail at the end of Automotive SPICE and can also be uniquely assigned by means of ID. Subsequently, the base practices for each process are described. These base practices and the corresponding output work products are the performance indicators, as they provide measurable metrics for the capability determination during assessments [34].

Process reference model	**Process ID**	The individual processes are described in terms of process name, process purpose, and process outcomes to define the Automotive SPICE process reference model. Additionally a process identifier is provided.
	Process name	
	Process purpose	
	Process outcomes	
Process performance indicators	**Base practices**	A set of base practices for the process providing a definition of the tasks and activities needed to accomplish the process purpose and fulfill the process outcomes
	Output work products	A number of output work products associated with each process *NOTE: Refer to Annex B for the characteristics associated with each work product.*

Figure 5.5 Schematic template for an Automotive SPICE process description [34]

5.3.1 Process Outcomes

The outcomes of the process are derived from the purpose of the respective process. These are numbered separately for each individual process (outcome 1 to n). These outcomes can be mapped to the corresponding output work products and are related to the defined base practices [34].

5.3.2 Base Practices

The base practices (BP) are a central part of the Automotive SPICE model. They define for each process what needs to be done to meet the required outcomes. They specify defined tasks and activities. These results of the base practices are directly derived from the purpose and the defined outcomes, whereby the results of the individual base practices are directly reflected in the output work packages. For each process, the base practices are numbered consecutively in the same way as the outcomes (BP 1 to n), but are recognizable across all processes with a unique ID (e.g. SWE.3.BP1). If the base practices are fulfilled and all output work packages are available, the defined outcomes are fulfilled. This corresponds to a process capability of level 1. In order to achieve a higher rating in an assessment, it is not enough to simply implement and comply with the required base practices for each process. The generic practices (GP) must also be fulfilled. Irrespective of the individual processes, these are described in detail at the end of Automotive SPICE. However, this increases the complexity considerably, which is why the process model, which is created in the course of this master thesis, is created on the basis of the individual base practices [34].

5.3.3 Work Products

The work products (WP) are the results of the individual base practices. Once the work products have been created on the basis of all base practices for each process, all required outcomes of the process are usually fulfilled. The work packages are documents of any kind. The creator or the company is free to choose how these documents or work products look like and how they are filed or archived. At the end of Automotive SPICE, all work products are listed, and a description of mandatory contents is provided. Each work product has its unique ID, independent from processes (e.g. 05-10). It is important to know that several processes or practices can access the same work product and vice versa. Additionally, a work product can be the result of several practices and processes. A document can also contain several work products, if the author or company perceives it as necessary or beneficial. It should only be noted that the documentation and distribution of the documents is regulated inside an organization. This means that all employees involved are always working on the latest versions and traceability and consistency in the naming and referencing is guaranteed [34].

5.4 Structure of the Standard

As already mentioned, Automotive SPICE contains individual processes and their activities and outcomes. Since these are arranged in the same way as the V-model, a simple and clear structure is possible, which promotes the bidirectional traceability and consistency throughout the entire standard and process model. This is another central element in Automotive SPICE, as it provides a transparent PRM layout. **Figure 5.6** illustrates this concept by means of the relevant systems engineering processes in the V-model. All applicable base practices are highlighted. This concept also applies analogous to the software engineering processes. The term traceability refers to the presence of references or links between work products, which further supports the impact analysis, coverage, and status tracking of requirements implementation. Consistency, on the other hand, addresses content and semantics. In addition, in Automotive SPICE, bidirectional traceability was explicitly defined between the following contents, according to [34]:

- Test cases and their corresponding test results as well as
- Change requests and work products, which are affected by these requests

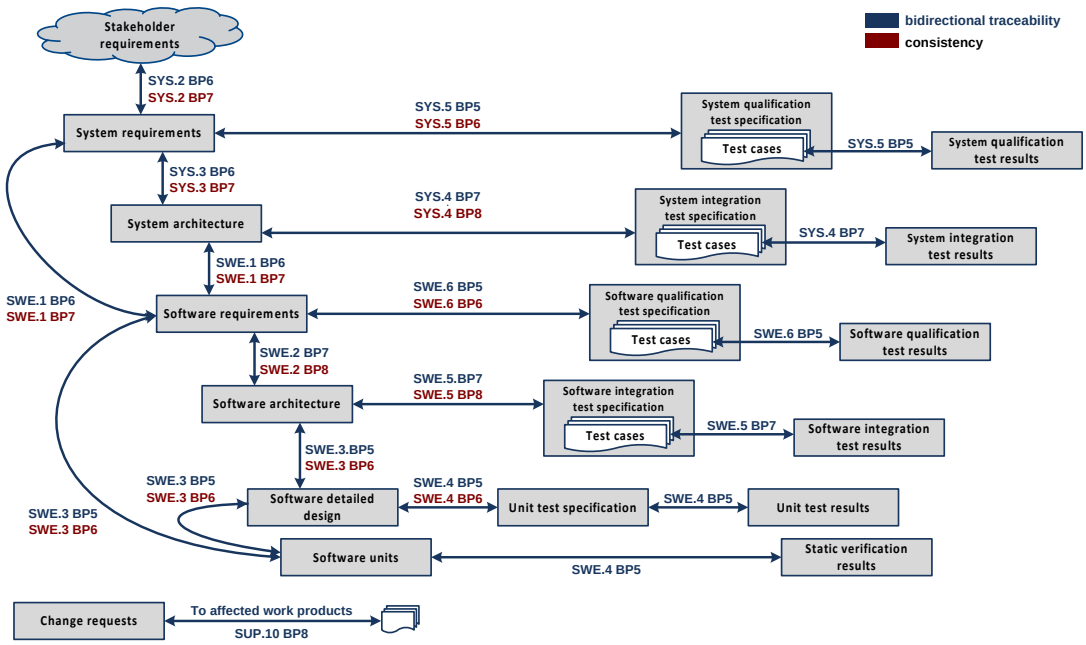


Figure 5.6 Traceability and consistency throughout Automotive SPICE [34]

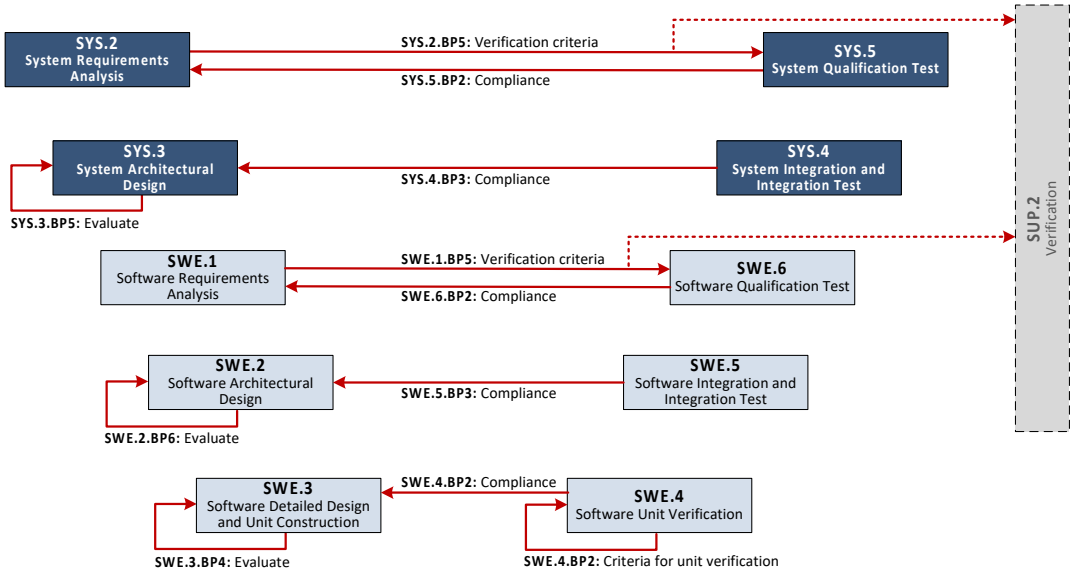


Figure 5.7 Evaluation, verification and compliance throughout the PRM [34]

Consistency can also be monitored and applied to the terms evaluation, verification criteria and compliance. **Figure 5.7** provides an overview of the dependencies and relationships between these important concepts and the structure within the process model. The evaluation of alternative solutions for system and software architectures as well as for the detailed software design is a prerequisite. The evaluation must be carried out according to defined criteria. The result of the evaluation with reasons for the architecture and design choice must be documented [34].

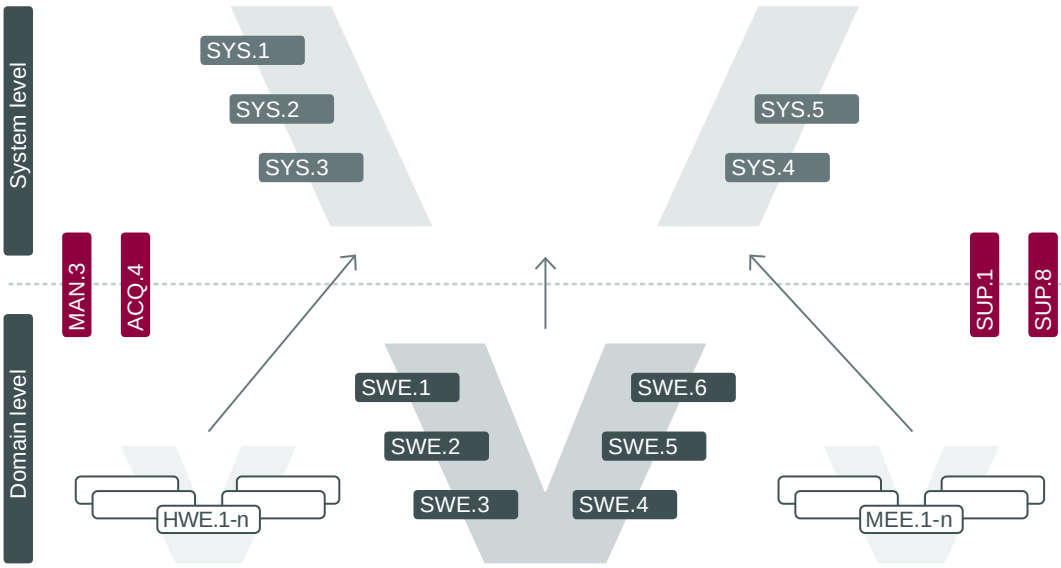


Figure 5.8 Plug-in concept of Automotive SPICE 3.0 [36]

The verification criteria are used for the development of test cases to ensure compliance with the defined requirements. The criteria for the individual testing ensure that the code complies with the detailed design and non-functional requirements of the software. For the unit tests, these criteria must be specified in a product inspection specification. Compliance with a software architectural design means that the specified integration tests demonstrate that interfaces and relevant interactions between software units, items and system items meet the requirements of the designed software architecture [34].
Another structural approach that has been implemented in Automotive SPICE 3.0 is the so-called plug-in concept. **Figure 5.8** shows this schematically, also using the V-model for development processes. The upper half shows the complete system engineering processes, which are structured according to the known V. Depending on the product being developed, domain-specific processes can be included in the lower half of the figure. Automotive SPICE 3.0 only contains the software engineering processes. By splitting up in version 3.0, further processes for hardware development (HWE) or mechanical development (MEE) can now be implemented according to the system engineering processes in analogy to the SWE processes. The remaining processes, such as support and management processes, are not domain-specific and are therefore designed to be applied to both, the system level and the specific domain levels [34].

5.5 Assessment of Automotive SPICE

The assessment is the method for determining the capability of a process. For this purpose, the processes of a specific project or company are compared with different Automotive SPICE processes. The assessment is mainly used for two reasons:

- *Process improvement* (internal assignment): determination of the need for internal process improvement.
- *Assessment of the capability* (external assignment): assessment of the risk for supplier selection

Regardless of the purpose of the evaluation, the results – in addition to the process capability level – are statements about the strengths and weaknesses of certain processes and recommendations for action. It is important to note that an assessment is always based on the evaluation of measurable metrics of individual processes and is not targeted specifically at personnel or products [30].

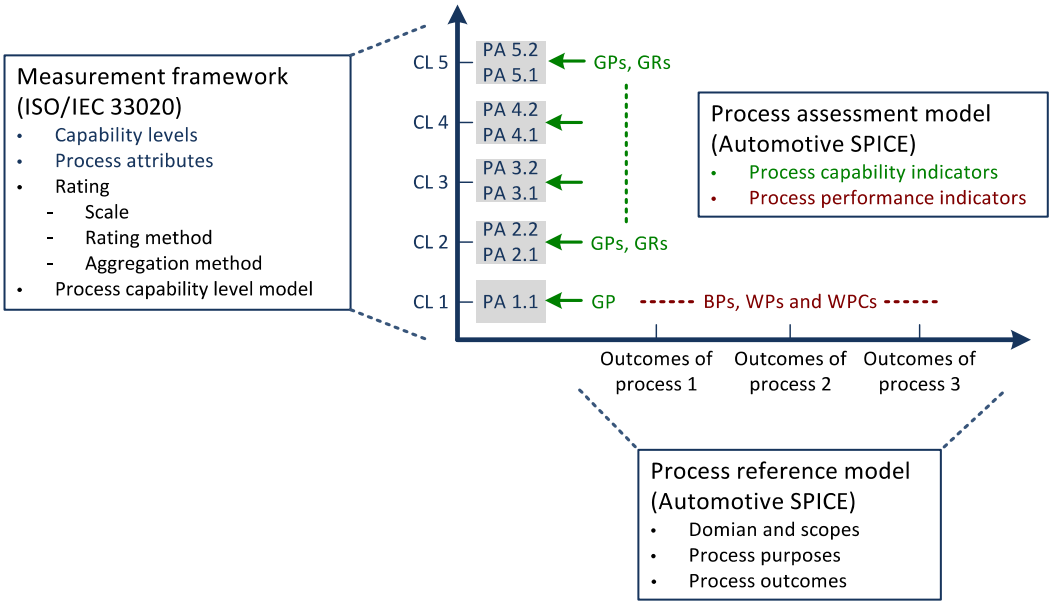


Figure 5.9 Process capability assessment dimensions [34]

Figure 5.9 shows the two-dimensional evaluation, which results in the already described process capability level. On the one hand, the process capability indicators and also the process performance indicators are used as a benchmark. Whether a certain level is achieved or not is determined on the basis of the defined process attributes. In principle, the process attributes are valuated using a four-level scale [34].

Table 5.2 Rating scale according to ISO 33020 [34]

N	Not achieved	0 to ≤ 15%	There is little or no evidence of achievement of the defined process attribute in the assessed process.
P	Partially achieved	> 15% to ≤ 50%	There is some evidence of an approach to, and some achievement of, the defined process attribute in the assessed process. Some aspects of achievement of the process attribute may be unpredictable.
L	Largely achieved	> 50% to ≤ 85%	There is evidence of a systematic approach to, and significant achievement of, the defined process attribute in the assessed process. Some weaknesses related to this process attribute may exist in the assessed process.
F	Fully achieved	> 85% to ≤ 100%	There is evidence of a complete and systematic approach to, and full achievement of, the defined process attribute in the assessed process. No significant weaknesses related to this process attribute exist in the assessed process.

Table 5.2 shows the rating scale for process attributes in the Automotive SPICE assessment. In order to support the evaluation of process attributes, the ISO 33020 framework offers a defined rating scale with an option for refinement, different rating methods and depending on the rating class. In addition, the Automotive SPICE standard defines a finer scale with additional subdivisions (*P*-, *P*+, *L*- and *L*+). This allows a finer graduation for the final evaluation of process capability. The final process capability level depends on which rating level is reached in which evaluated processes according to the process attributes. **Table 5.3** shows this entire result based on the required process attribute fulfillment levels. This process capability level model is based on the ISO 33020 standard and defines the rules on how achieving each level depends on the rating of process attributes for the assessed and all subordinate levels. Generally speaking, reaching a certain level requires that the corresponding process attributes must be reached to a large extent and that all lower-level process attributes are fully reached [34].

Table 5.3 Process capability level model according to ISO 33020 [34]

Scale	Process attribute	Rating
Level 1	PA 1.1: Process Performance	Largely
Level 2	PA 1.1: Process Performance	Fully
	PA 2.1: Performance Management	Largely
	PA 2.2: Work Product Management	Largely
Level 3	PA 1.1: Process Performance	Fully
	PA 2.1: Performance Management	Fully
	PA 2.2: Work Product Management	Fully
	PA 3.1: Process Definition	Largely
	PA 3.2: Process Deployment	Largely
Level 4	PA 1.1: Process Performance	Fully
	PA 2.1: Performance Management	Fully
	PA 2.2: Work Product Management	Fully
	PA 3.1: Process Definition	Fully
	PA 3.2: Process Deployment	Fully
	PA 4.1: Quantitative Analysis	Largely
	PA 4.2: Quantitative Control	Largely
Level 5	PA 1.1: Process Performance	Fully
	PA 2.1: Performance Management	Fully
	PA 2.2: Work Product Management	Fully
	PA 3.1: Process Definition	Fully
	PA 3.2: Process Deployment	Fully
	PA 4.1: Quantitative Analysis	Fully
	PA 4.2: Quantitative Control	Fully
	PA 5.1: Process Innovation	Largely
	PA 5.2: Process Innovation Implementation	Largely

Although well-known manufacturers have been able to position their interests as content in Automotive SPICE through standardization work, there are differences in the perception of some requirements. This is one of the reasons why some manufacturers have specified their exact interpretation in binding customer requirement specifications. In this way, OEMs want to keep the freedom of interpretation to a minimum and emphasize their own focus. Although Automotive SPICE is generally applicable, manufacturers place different emphasis on the parts that are important to them. For example, OEM A gives more importance to definition and specification of the detailed design, while OEM C sets the priority on verification and validation. In discussions with the industrial partner, it has also emerged that this behavior of the OEMs is directly reflected in the process capability level that has been achieved. If a supplier achieves a level of 1 for OEM A, it has happened before that OEM C confirms a level 3 for this supplier.

The interpretation of assessment results often focuses on the question of their comparability. On the one hand, an organization that commissions an internal quality assurance team or an external service provider to carry out an assessment will ask itself whether this result also corresponds to the possible evaluation by a customer. On the other hand, a customer who wants to carry out an assessment of a supplier project will be confronted with the question of whether the results of an assessment not carried out with their own assessors allow the necessary conclusions to be drawn on the qualification of the supplier. Practical application in supplier assessments has resulted in procedures that have been developed by the VDA working group 13 into a guideline for conducting assessments. These are the aforementioned yellow ribbon Automotive SPICE Guidelines. Here, the procedure of assessments is modelled as a process. This guideline developed by manufacturers and suppliers provides recommendations for the application of Automotive SPICE in the software development process. In particular, process improvement is taken up as a core element of an assessment and modeled as an integral part of the overall process. Rules and recommendations for the evaluation of each practice are also mentioned. The rules are much more important for the assessors, since a written documentation is necessary in case of a deviation. It also stipulates that, among other things, a process must be down-rated if a previous process does not meet certain requirements [31].

Since the entire process scope of Automotive SPICE would considerably increase the effort of an assessment and would not necessarily improve the informative value of the results, a common scope was defined. This basic subset of processes (also known as

the HIS scope) has been defined by the VDA QMC as a selection of standardized assessment methods. This selection reflects as accurately as possible the processes by means of which the supplier and its software processes can be evaluated as suitably as possible in relation to process capability. These include in particular all sub-models of the V-model. Studies conducted by the VDA have also shown that good process capability in the HIS Scope in question is directly reflected in increased product maturity. Unfortunately, a recent study by the Volkswagen Group revealed a deficit in these very bottom processes of the V-model. Processes ENG.5, ENG.6 and ENG.7 (from Automotive SPICE 2.5) are the weakest. These relate in particular to software design and integration. On the other hand, all supporting processes performed best (SUP). In particular, MAN.3 (project management) was seen as a critical factor, since the entire process capability level can be overturned depending on its rating. Another result of Volkswagen's statistical analysis was that, in addition to correlations in the rating of the individual processes, there are also strong dependencies between the individual base practices. This is also reflected in the requirements of the Automotive SPICE guidelines for assessments. In an experiment, it was then possible to predict the assessment of a process on the basis of a few individual base practices (four out of eight) using a regression model with high accuracy. The proposal for a new approach to assessments provides a direct input for future versions of the Automotive SPICE standard. The focus should therefore be on the effectiveness of the individual processes rather than the consistency of the individual base practices. They suggest, that these BPs should also be consolidated and optimized in future versions of the standard [37].

6 Concept Development for Process Analysis

This chapter describes the entire concept development as well as the necessary prior requirement definition for the data storage and analysis, which was carried out in the context of this master thesis. The chapter is structured according to the methodology used in the practical part of the work. Firstly the approach to process definition and the associated selection of suitable processes, base practices and relevant sources is explained. Subsequently, a tool environment is to be developed in which the comprehensive process model can be used in a clear and easy to use manner. In addition to the first ideas and concepts, this chapter also highlights all problems and challenges that have arisen in the course of concept development. In particular, the complex n:m relationships (cardinalities in the database model) between the individual requirements of the respective sources should be mentioned. The chapter concludes with the result of a working concept for data storage and analysis within the process model. For this purpose, all steps necessary for the development of this tool concept are presented. A special focus is on the creation of the VBA code for mapping base practices for the necessary comparison of the various included sources. Subsequently, this tool concept is filled in the following chapter and allows conclusions to be drawn about deviations and optimizations for the respective user groups.

6.1 Introduction to the Corporation

This thesis is done in close cooperation with Magna Powertrain in Lannach, Austria. Magna Powertrain is incorporated into the management structure of Magna International, one of the leading automotive Tier-1 suppliers in the automotive industry. The Lannach plant is the European Headquarter of Magna Powertrain, home of the Driveline Systems business unit. There, production and manufacturing, but also the engineering and program planning of customer projects, take place. The most important products in the portfolio include for example all-wheel drive (AWD) systems, transfer cases, power take-off units, axle drive modules and disconnect systems. Most of these products are complex mechatronic components with integrated actuators and electronic control units. Depending on the customer requirements, specific software for driving dynamics and handling is developed and applied in the vehicle. Nevertheless, for every active and controllable system the modular base software has to be developed and adapted to the vehicle specific environment.

6.2 Current Situation and Challenges

At the beginning of this master thesis, all requirements for the results of the master thesis were determined and specified together with the industrial partner. The aim of the work is the development of a comprehensive process model, which indicates deviations and potentials for process optimization based on the analysis of existing processes, various customer requirements as well as international standards. The comparative analysis is based on the VDA Standard Automotive SPICE, which is the basis for optimizing existing development processes. This standard in version 2.5 is already implemented and well-known in the internal processes of the industry partner.

The need for the creation of a comprehensive process model for deviation analysis and potential analysis resulted from the conflicts that are constantly growing in industry, between suppliers and OEMs. On the one hand, the supplier has a fixed process environment based on the still valid standard Automotive SPICE 2.3 (or 2.5). Since a new version of the standard has now been released, it must be implemented in the near future, in the transitional period. The current process environment does not comply with the upcoming changes in the standard Automotive SPICE version 3.0. The Automotive SPICE guidelines for assessors are also to be taken into account here in order to bring changes in the processes in line with the valuation guidelines. Furthermore, the shortening of development times on the part of manufacturers (customers) again and again calls for quality problems in product development at suppliers. In order to stop these and develop more mature products, the lived development processes must also have a certain degree of maturity. Another challenge is the backlog or lack of knowledge about one's own process capability and, even more importantly, the lack of knowledge about the most diverse customer requirements. This wastes resources unnecessarily in critical processes at an early stage of development. In particular, the preparation of quotations and the analysis of requirements should be mentioned here. If these activities are carried out without the necessary know-how, in the worst case this will lead to incorrectly assumed development costs and also to a non-compliant specification.

On the other side of this conflict are the car manufacturers (OEMs). They place ever higher demands on the quality of their suppliers' products. For these reasons, they therefore require a high level of process capability with transparent development activities and cost reduction at the same time. As a supplier usually does not only serve one OEM, it has to adapt its processes to many different customer requirements. The variability in the scope of interpretation in the assessment of Automotive SPICE

described in the previous chapter plays also an important role. The differing focus on the interpretation and implementation of the standard is reflected in the staggering rating levels between different OEMs. The OEMs also require the fulfilment of further product-specific specifications in addition to the general requirements (Automotive SPICE).
For these challenges it is now necessary to develop and test a tool concept for the data handling and analysis and to fill this tool with information from the relevant sources. Subsequently, a direct comparison of the different requirements should be possible after mapping the individual processes and base practices among each other, whereby potential for process optimization and improvement can be derived by means of a deviation analysis. This environment should be as modular as possible in order to be able to incorporate even more standards into the process model in the future. This concept is also to be developed in such a way that it can be transferred to a database (e.g. an existing requirement management system) at a later point in time. The creation of this tool concept is now described in the course of this chapter. In order to meet this challenge, the division into five work packages (**Table 6.1**) was carried out beforehand at the beginning of the project including the respective scheduling.

Table 6.1 Defined work packages for the project

No.	Name	Description / Content	Outcome
1	Familiarization with the topic	Introduction to Automotive SPICE and internal processes. Research of publications on differences in ASPICE	Discuss literature / open questions
2	Requirements for data storage and analysis	Information requirements. Which data (sources) should be included? Assessment of the requirements in terms of feasibility and risks.	Summary of requirements including prioritization
3	Development of the concept for data storage and analysis	Verification that a subdivision according to base practice is sufficiently fine. Conceptual ideas for data storage and analysis. Development of the concept for data management and analysis. Detailed concept for the implementation in the selected tool environment.	Concept for data storage and analysis, experimental with SWE.3 of ASPICE version 3.0
4	Concept implementation / prototype	Implementation of the concept. Insertion of the defined data (sources).	Prototype implementation of the concept incl. analysis possibilities
5	Test and documentation	Validation and verification of the prototype. Creation of user documentation for use and extension.	Test of the prototype acc. to different user groups and documentation

6.3 Approach to Process Analysis

After coordinating the individual phases and work packages as well as the overall scheduling, the scope of the process model was first determined. This should consist of several processes, the HIS scope from Automotive SPICE. Due to the existing process knowledge, version 2.5 of Automotive SPICE is used. Additionally to the HIS scope, MAN.5 (risk management) has to be implemented to the model. **Table 6.2** provides an overview of the defined scope of processes from Automotive SPICE 2.5. This scope acts as the basis for the process model, the base practice mapping and the deviation analysis.

Table 6.2 Defined processes for the analysis (Automotive SPICE 2.5 [38])

Process ID	Process Name
ENG.2	System requirements analysis
ENG.3	System architectural design
ENG.4	Software requirements analysis
ENG.5	Software design
ENG.6	Software construction
ENG.7	Software integration
ENG.8	Software testing
ENG.9	System integration
ENG.10	System testing
SUP.1	Quality assurance
SUP.8	Configuration management
SUP.9	Problem resolution management
SUP.10	Change request management
MAN.3	Project management
MAN.5	Risk management
ACQ.4	Supplier monitoring

After defining the process scope for the comprehensive process model, the relevant sources were defined in the initial phase which provide data, especially base practices and requirements, for the model. Since Automotive SPICE 2.5 is the basis on which the further requirements are assigned to the standardized pendants, this source has been set since the beginning. As Automotive SPICE already exists in version 3.0, as described in this thesis, it must be implemented in the current transition period. ASPICE 3.0 is also an integral part of this process model, especially when it comes to mastering this challenge and identifying the changes and required change actions by means of deviation analysis. In order to improve process capability especially with regard to

assessments, the Automotive SPICE guidelines are also included, which apply to the Automotive SPICE 3.0 version. This is where rating rules and recommendations are defined. If these are already known in advance, the optimization of the processes can be fine-tuned even more precisely to the requirements. In order to extend the process model in a customer-specific manner, three customer requirement specifications are also implemented. For reasons of confidentiality, these are referred to as OEM A, B and C. Last but not least, the existing internal processes will be included in the tool concept in order to make deviations clear. In general, the process model should contain all relevant guidelines that have already been briefly presented in chapter 4. **Table 6.3** shows all relevant sources which are to be included for the process model in the tool concept to be developed for data management and analysis. The sources are ranked according to their priority in the model. In addition to the entire contents of the individual guidelines, their relationship to each other and individual deviations, the process model should represent the largest overlapping intersection of these sources. To this end, the relevant processes of all sources are to be assigned and compared against each other. The basis for this is Automotive SPICE 2.5 at process capability level 1, with the exception of the Automotive SPICE guidelines, which refer to the latest version 3.0. A subsequent assignment to Automotive 2.5 is therefore only possible after successful mapping both Automotive SPICE versions. Since a mapping at process level is too imprecise, the granularity is to be checked in the course of this process analysis. This is especially important for the creation of the tool concept. As a result of the granularity analysis, the process model will be based on the base practices of Automotive SPICE. This has advantages because they are explicitly decisive for the assessment at level 1 of process capability. It has also been identified that the comparison at process level between Automotive SPICE versions and customer requirements is therefore feasible. Since certain customer requirements are in German and binding, a translation is not necessary.

Table 6.3 Defined sources for the process mapping

Priority	Source	Name	Published	Language
1	VDA	Automotive SPICE 2.5	2010	English
2	VDA	Automotive SPICE 3.0	2015	English
3	VDA	Automotive SPICE Guidelines	2017	English
4	OEM A	Customer requirements A	2015	German
5	OEM B	Customer requirements B	2015	English
6	OEM C	Customer requirements C	2016	German
7	Magna	Internal engineering processes	-	English

Ultimately, customer requirements differ from the VDA standards in terms of terminology. The process model should also enable a comparative comparison. Especially the terms element, component, unit and item mentioned in the previous chapter are the most important ones. A clear definition and thus comparability eliminates the inconvenience of using the tool in advance.

6.4 Approach to Concept Development

The concept development is about the approach to develop a tool for data storage and analysis. First of all, the choice of a suitable system for data preservation is important. This is to be chosen with regard to modifiability and extensibility. The aim is to design the process model in such a flexible way that it can be supplemented in the future with additional content (sources) and process scopes. The information should also be presented in a clear and concise way so that all the necessary information can be accessed quickly and easily. The functionality for defined user groups should also be provided. User groups using this tool include management, quality assurance, engineers and process owners. Therefore, a search or filtering mechanism must be included in the tool. Compatibility with all employees involved should also be ensured. For precisely these reasons, *Microsoft Excel* was chosen as a suitable system for data management and analysis of the process model.

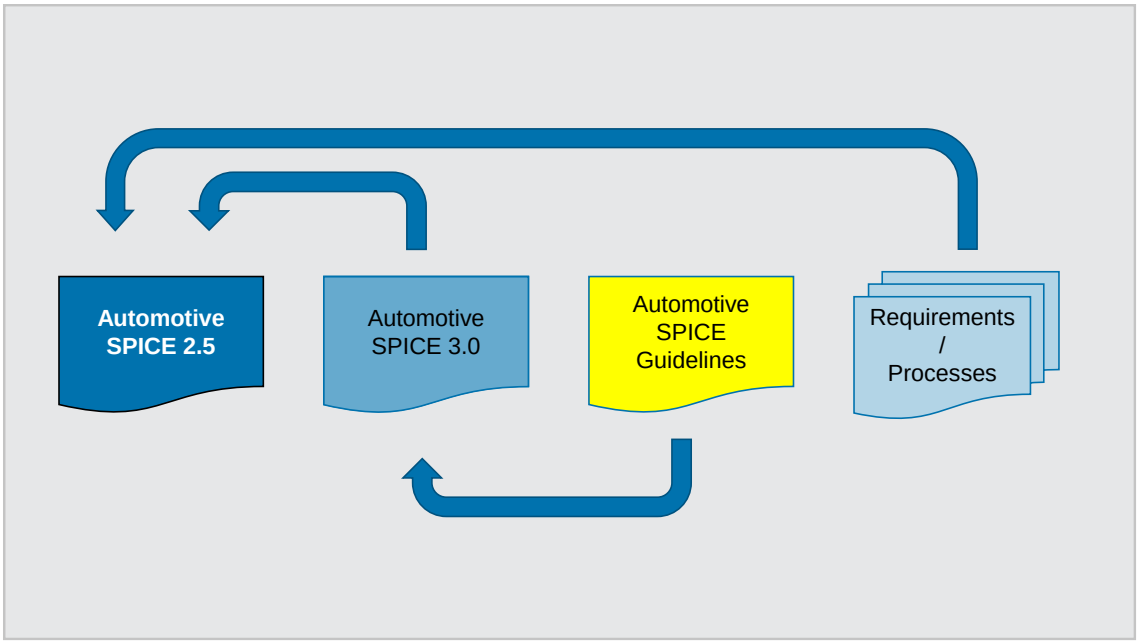


Figure 6.1 Relations and references of the various sources

Another point to consider when planning and developing the concept is the future use and handling. The process model should have a modular structure so that it can be easily integrated into a database at a later point in time, for example an existing requirements management system.

Following the review of all relevant sources for process analysis, Automotive SPICE 2.5 was defined as the basis for comparing and mapping base practices based on the still valid version of Automotive SPICE 2.5. **Figure 6.1** illustrates the relationships and references of the individual considered sources. One of the challenges is the relationship between the Automotive SPICE Guidelines published in 2017, which refer to the latest version, Automotive SPICE 3.0. Both Automotive SPICE 3.0 and all other customer requirements and internal processes are to be mapped directly to Automotive SPICE 2.5 and can then be analyzed. Since the Automotive SPICE Guidelines imply to which processes the individual rules and recommendations refer, they must first be allocated to the corresponding Automotive SPICE 3.0 counterparts. The guidelines can then be assigned to the individual base practices and processes of the basis via an existing relationship between Automotive 3.0 and Automotive 2.5.

After the initial analysis of the sources, it has also become apparent that there are inconsistent content-related correlations between the various sources. This has to be taken into account especially in relation to cardinality, if the model is to be implemented later in a database.

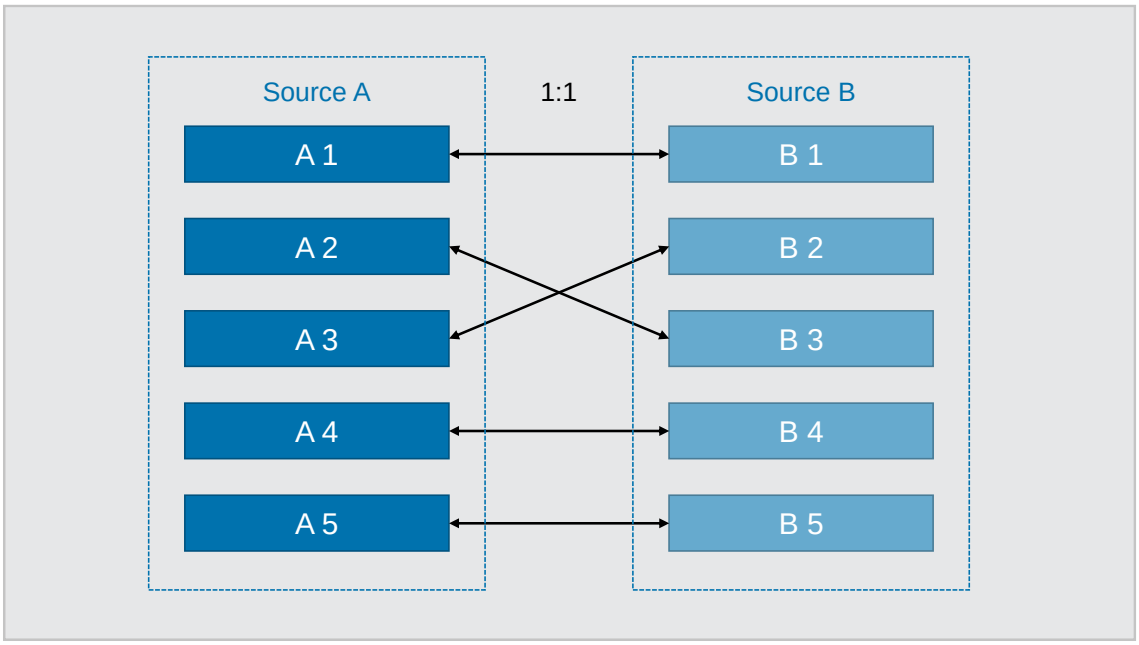


Figure 6.2 Cardinality of one-to-one between BPs of two sources

Figure 6.2 visualizes the one-to-one (1:1) relation of several exemplary base practices from two different sources. Each base practice from source A can be assigned exactly one single base practice from source B (and vice versa). It does not matter whether the base practices can be assigned parallel (A1:B1) or randomly (A2:B3), as long as each entry can be assigned to a maximum (or exactly) one counterpart of the other source. Such an allocation (1:1) would be relatively easy to implement and understand in any tool environment. However, the first examination of the individual sources has already revealed that these are not all one-to-one relationships, rather they are the exception in the HIS scope.

Figure 6.8, on the other hand, shows a one-to-many (1:n) relationship between the same base practices of two sources. Thereby, several different base practices of the second source B refer to the same one base practice of source A. This is particularly the case if customer requirement specifications have divided a required base practice from the Automotive SPICE standard into several individual requirements. In practice, this is also the case if the rules described in the Automotive SPICE Guidelines (in the respective chapter) all refer to a base practice. The same relationship can also be exactly the other way around, many-to-one (n:1), whereby in this case several base practices of source A would refer to the corresponding equivalent of source B. This cardinality between two sources is a rather more complex to implement and also occurs more frequently in the processes analyzed between individual base practices and customer specifications.

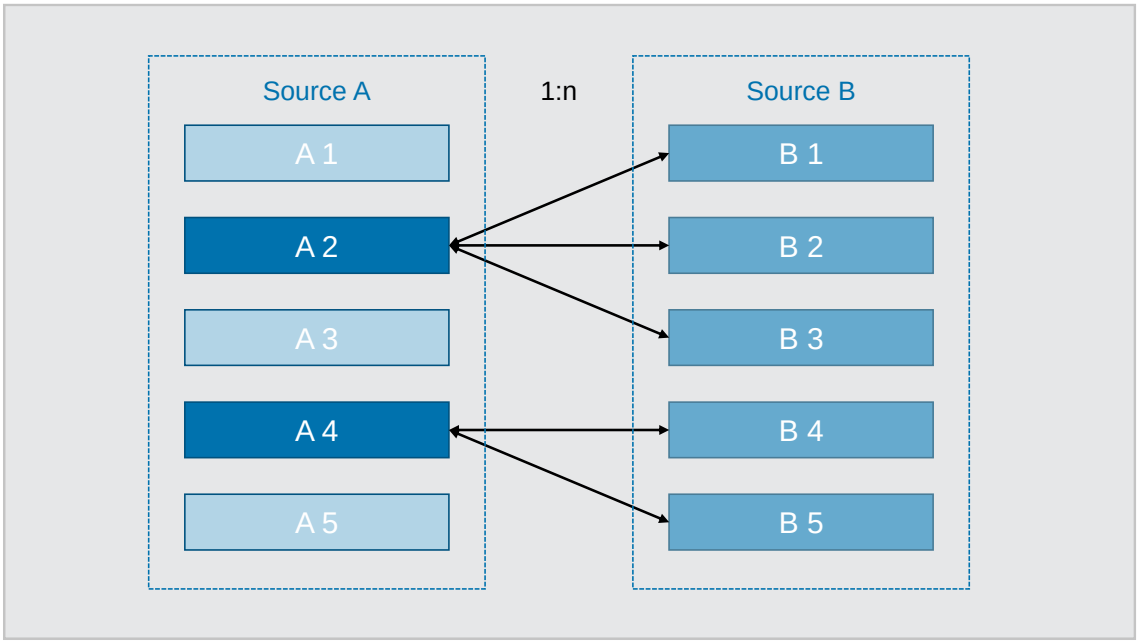


Figure 6.3 Cardinality of one-to-many between BPs of two sources

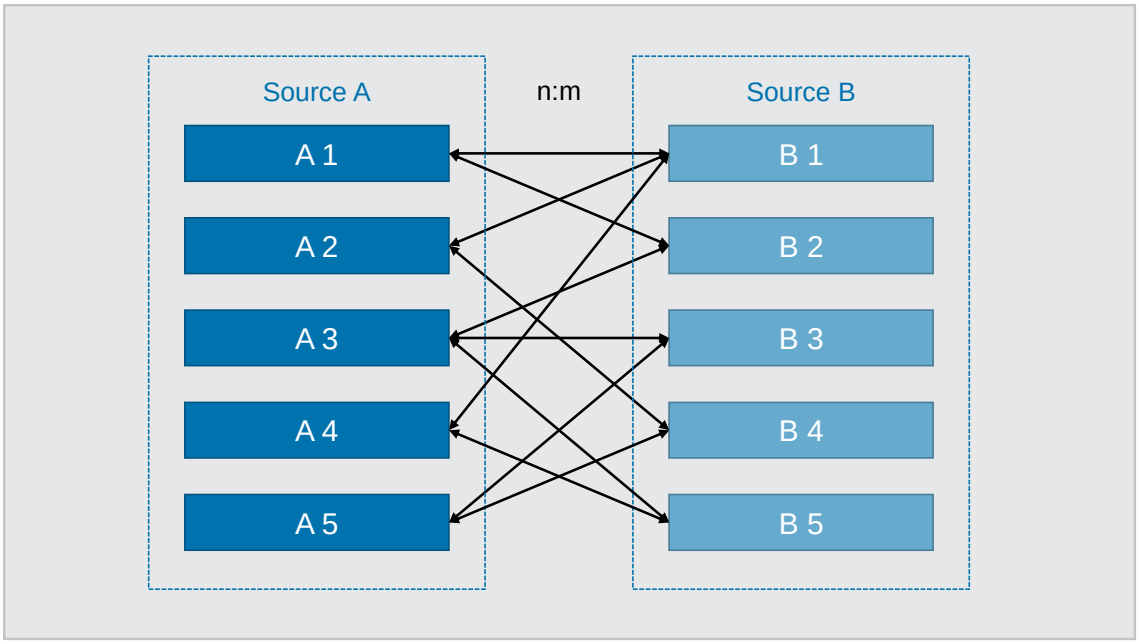


Figure 6.4 Cardinality of many-to-many between BPs of two sources

However, if one of the base practices of source B no longer refers to a single base practice of source A, which has further relations with other base practices of source B, it is called a many-to-many (n:m) relationship. **Figure 6.4** visualizes such a n:m relationship between the two known sources. These relationships and assignments are highly complex, both in terms of identification and in a clear and target-oriented way of representation. This type of relationship between the individual sources occurs in practice when, for example, generic rules of the Automotive SPICE guidelines refer to different base practices from the Automotive SPICE standard. This also frequently occurs when allocating the individual requirements from customer specifications. There are both cross-sectoral requirements and explicit requirements analogous to the base practices. These n:m relationships are very difficult to implement in Excel without losing the overview and ease of editing. A more suitable approach would be a database, but the flexibility is considerably simpler during the filling and testing of the tool concept for data storage and analysis in Microsoft Excel. A solution must therefore be found to present the complexity in a reasonable way. The number of sources also increases the complexity, since there are not only n:m. relations between two sources, but also multiple sources with different references (compare to **Figure 6.1**).

6.5 Outcomes and Tool Concept

This section describes the development of the tool concept for data storage and analysis based on the considerations described before.

The integral part of this tool is the aggregation of all base practices of Automotive SPICE 2.5, which serve as a reference for the deviation analysis from which future optimization needs can be derived. The second most important point is all the base practices of the new Automotive SPICE 3.0 standard. This is followed by Automotive SPICE guidelines, customer requirements and internal processes, which should be allocated to the respective Automotive SPICE 2.5 counterparts.

At the beginning, a simple sheet was created in Excel, which could be used to test further considerations for the concept. **Figure 6.5** shows an excerpt from the concept of what emerged from the first ideas. A separate set of columns has been created for each source (e. g. Automotive SPICE 2.5, 3.0, etc.). These include the unique ID of the base practice, the name of the base practice as well as the full text description of the base practice according to the standard. For each source there is also a column for the discrepancy, i.e. the deviations of the different requirements to the base pendant from Automotive SPICE 2.5 as well as a column for the evaluation of the deviation or the necessary modification effort. For each unique base practice or requirement from other sources there is a separate line in Excel. At first glance, this led to a clear classification of the individual sources. The filtering of the individual columns, full-text search and grouping of individual row sections or column groups is also provided in this concept in Excel. However, this concept quickly reached the limits of clearness, manageability and possibility of modification, which is described next.

ASPICE 2.5			ASPICE 3.0					Next Source
ID	Name	Description	ID	Name	Description	Discrepancy	Rating	ID
BP1	Name1	Description1	BP1	Name1	Description1	xyz		ID1
BP2	Name2	Description2	BP2	Name2	Description2	xyz		ID2
BP2	Name2	Description2	BP3	Name3	Description3	xyz		
BP3	Name3	Description3	BP4	Name4	Description4	xyz		ID3
BP3	*Name3*	*Description3*						ID4
BP4	Name4	Description4	BP4	Name4	Description4	xyz		
			BP5	Name5	Description5	xyz		

Figure 6.5 Challenges of allocating n:m relations of three sources in Excel

All critical contents of the figure are highlighted in yellow and can be summarized into the following groups:

- Duplicates of the reference base practices
- Duplicates of the source base practices
- Empty cells
- Unnecessary rows

Duplicates represent the greatest challenge. In addition to the poor representation of duplicate entries, the biggest disadvantage is the difficulty in changing individual base practices or allocating them to the standardized counterparts. This represents an extreme additional effort for the filling and maintenance of the tool concept for data management and analysis. The empty cells also impair usability and clarity in the chosen tool environment, which Microsoft Excel offers for such a many-to-many representation. Another problem that needs to be solved is the avoidance of unnecessary cells, as this also greatly restricts the transparency of the entire document.

After analyzing the first ideas and taking into account the problems that have arisen, it was possible to agree with the industrial partner on an important premise: each base practice should only be included in the tool once and also be able to be changed without having to change all links and allocations to their matching counterparts manually.

For this purpose, a separate spreadsheet has been created for each source (including the Automotive SPICE 2.5 reference) within the same file. In addition to a better overview, this also allows for the required modularity and expandability by new sources. Afterwards, the contents of the individual sources must be mapped to each other. With this idea, the use of macros and scripts in Visual Basic for Applications (VBA) in Microsoft Excel came into use for the first time. In consultation with the industrial partner, a presentation in the reference view (Automotive SPICE 2.5) is desired, in which a maximum of one line is provided for each base practice of the standard. This also addresses the aforementioned problem with duplicates or unnecessary rows. It also allows for easier integration into a later database.

Accordingly, the other table sheets of the other sources were also created, with a maximum of one line per requirement with a unique ID. **Figure 6.6** shows the exemplary structure of the new concept. In addition to a single row for each base practice, separate columns have been created on the reference sheet for each source, which are supposed to contain the contents and deviations of each counterpart of the respective BP.

Automotive SPICE 2.5 REFERENCE				Automotive SPICE 3.0	ASPICE 3.0 Guidelines	OEM 1	OEM 2	OEM 3	Internal Processes
P ID	BP ID	Base Practice	Base Practice Full Text Description						
ACQ.3	**ACQ.3.BP1**	Negotiate the contract/agreement	Negotiate all relevant aspects of the contract/agreement with the supplier. [Outcome 1] *NOTE 1: Relevant aspects of the procurement may include* • *system requirements* • *acceptance criteria and evaluation criteria* • *linkage between payment and successful completion of acceptance testing* • *process requirements, process interfaces and joint processes.*						
ACQ.3	**ACQ.3.BP2**	Specify rights and duties	Unambiguously specify the expectations, responsibilities, work products/deliverables and liabilities of the parties in the contract/agreement. [Outcome 2]						
ACQ.3	**ACQ.3.BP3**	Review contract/agreement for supplier capability monitoring	Review and consider a mechanism for monitoring the capability and performance of the supplier for inclusion in the contract/agreement conditions. [Outcome 3]						
ACQ.3	**ACQ.3.BP4**	Review contract/agreement for risk mitigation actions	Review and consider a mechanism for the mitigation of identified risk for inclusion in the contract/agreement conditions. [Outcome 3]						
ACQ.3	**ACQ.3.BP5**	Approve contract/agreement	The contract/agreement is approved by relevant stakeholders. [Outcome 1]						

Figure 6.6 Exemplary overview of the concept tool structure (ASPICE 2.5 [38])

But this also means that in the case of an allocation of 1:n and n:m relationships, several entries must be displayed in a single cell when mapping to the reference sheet. As this manually means a very high additional effort for the allocation and the modification, an automated solution had to be found by using VBA. This script should map the individual requirements (unique ID and full text description of each requirement) for each additional source to the respective base practices of Automotive SPICE 2.5 in the reference sheet. These scripts are used to automatically fill the column groups of the individual sources (colored marked in the figure).

Figure 6.7 uses three base practices of Automotive SPICE 3.0 to show how this automated mapping should be done. In the first column “A”, the reference IDs for Automotive SPICE 2.5 are entered. This means that each base practice of the source is only entered once into the tool, and the allocation to the Automotive SPICE 2.5 counterparts can be changed at any time without the need to adjust the content.

Automotive SPICE 3.0			
Ref. AS2.5	BP ID	Base Practice	Base Practice Full Text Description
ENG.5.BP4,	**SWE.3.BP3**	Describe dynamic behavior	Evaluate and document the dynamic behavior of and the interaction between relevant software units. [OUTCOME 3] *NOTE 1: Not all software units have dynamic behavior to be described.*
ENG.6.BP2, ENG.6.BP3,	**SWE.3.BP4**	Evaluate software detailed design	Evaluate the software detailed design in terms of interoperability, interaction, criticality, technical complexity, risks and testability. [OUTCOME 1,2,3,4] *NOTE 2: The results of the evaluation can be used as input for software unit verification.*
ENG.5.BP10, ENG.6.BP8, ENG.6.BP9,	**SWE.3.BP5**	Establish bidirectional traceability	Establish bidirectional traceability between software requirements and software units. Establish bidirectional traceability between the software architectural design and the software detailed design. Establish bidirectional traceability between the software detailed design and software units. [OUTCOME 4] *NOTE 3: Redundancy should be avoided by establishing a combination of these approaches that covers the project and the organizational needs.* *NOTE 4: Bidirectional traceability supports coverage, consistency and impact analysis.*

Figure 6.7 Allocation of base practices from ASPICE 3.0 [34] to ASPICE 2.5

On the basis of these considerations for the unambiguous assignment between the individual sources, a flowchart was created prior to programming, on the basis of which the process is to be illustrated. **Figure 6.8** shows the flowchart for the overall mapping process for mapping base practices of one single source to the Automotive SPICE 2.5 reference.

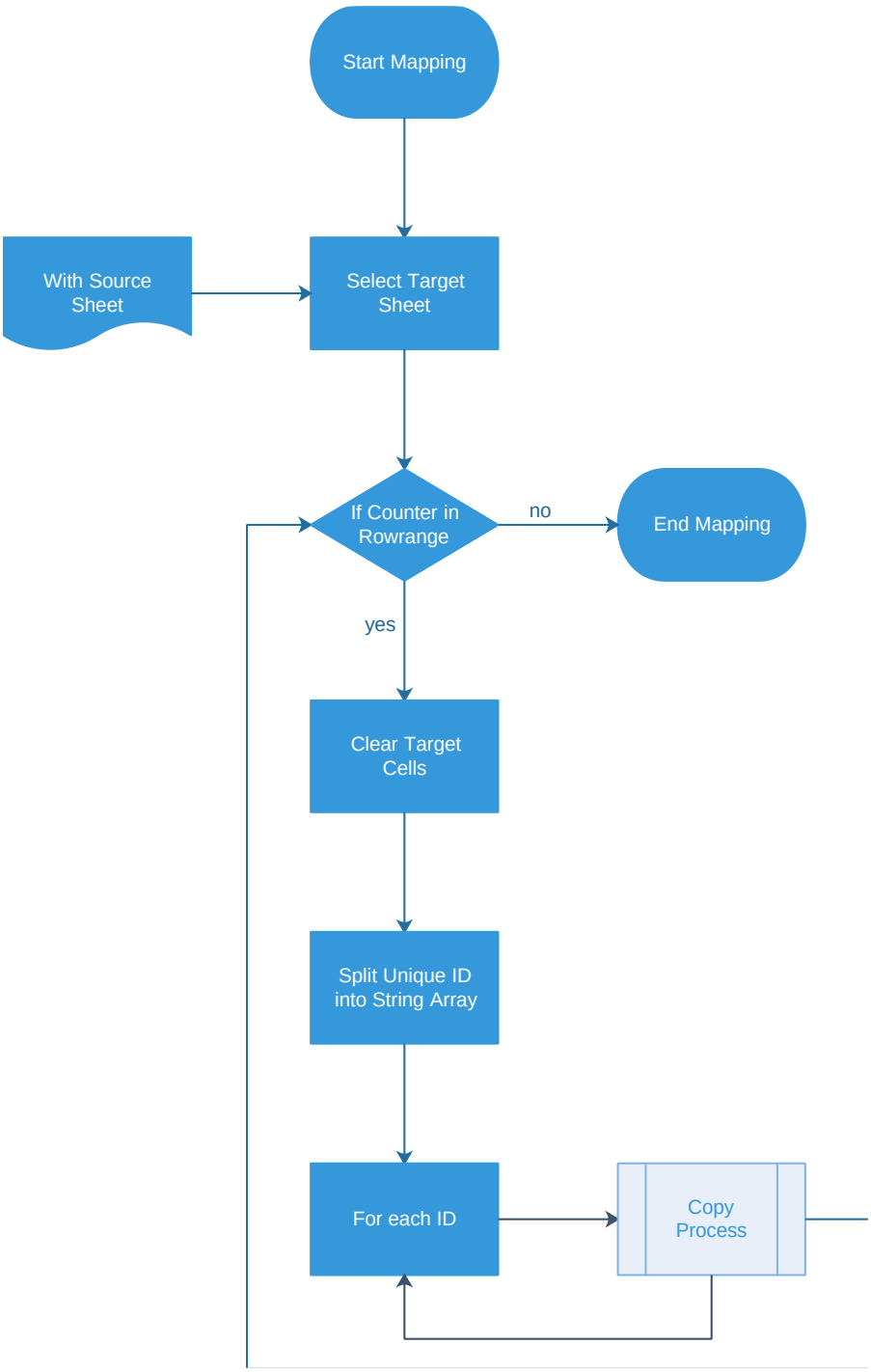


Figure 6.8 Flowchart of the overall process for mapping a single source

First, the contents of both spreadsheets must be retrieved. This is in any case the reference sheet with the individual base practices of Automotive SPICE 2.5, as well as the entries of the source to be mapped in the respective process. A range of rows is then defined, which should be large enough to contain all the entries of the respective sheet. In a query “If Counter in Rowrange” it should now be checked if the process is still in the defined range. The counter should be increased after each loop. If this is the case, all cells on the reference sheet into which the entries of the sources are to be mapped should be erased first. This allows a clean start for the copying process. The unique IDs of the first column should then be split into separate IDs, as this can be more than one. For each unique idea the sub-process of copying should be executed. When the row counter exceeds the defined row range, the mapping process ends.

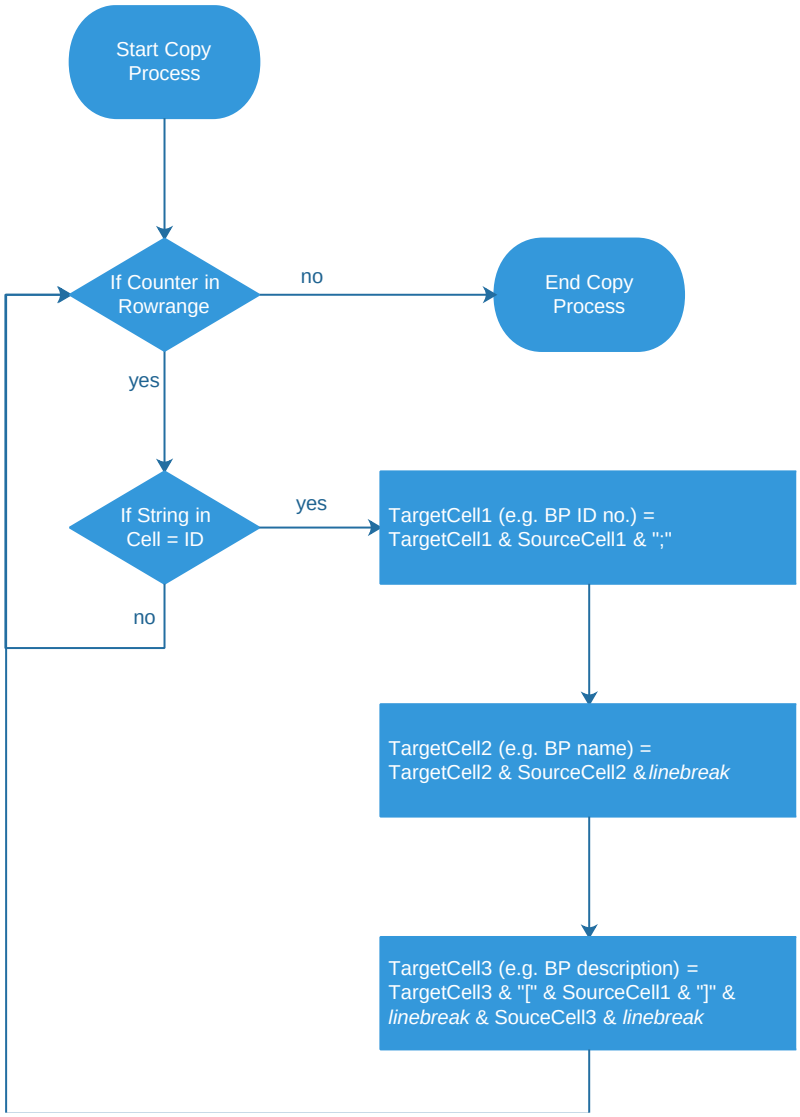


Figure 6.9 Flowchart of the individual copy sub-process for a single source

Figure 6.9 illustrates the aforementioned sub-process for copying the individual contents of the source sheets to the reference sheet. This is to be started for each unique ID that is within the defined row range. For this purpose, another counter is queried in the same way as the first example. If this counter is within the defined range of rows, the process should be continued, and the counter increased by one, otherwise the process is terminated (and returned to the overall mapping process). A further if query will then be used to find out whether the cell under consideration contains a string, which in turn contains a unique ID. If this is the case, the actual copying process starts. In order to keep this as modular as possible and to adapt it individually for each source, a separate process was developed for each content to be copied. In this example, this is the ID of the base practice, its name and the full text description of the base practice. As a cell in 1:n and n:m relationships contains the contents of several base practices as described above, the cell is not overwritten every time during the execution of the copy process. For this reason, the cell's content is redefined in each cycle: existing content plus the new content and a comma or line break, which acts as a separator.

The copied contents or their formatting can be adapted separately for each source. For example, the individual requirements from the OEMs' specifications do not usually have a name for each requirement, but instead merely a unique ID and a description or definition of the requirement. Furthermore, it should be mentioned that in the figures and in the further course of this chapter, the designation *TargetCell* refers to the target cells in the Automotive SPICE 2.5 reference table (see color-coded areas in **Figure 6.6**). The designation *SourceCell* accordingly refers to the cells on the individual spreadsheets of the various sources, which are to be mapped to the reference sheet (e.g. BP, ID, full text, etc.) analogous to the contents in **Figure 6.7**.

Based on these considerations, the VBA code was then developed with reference to the described flowcharts. This development is explained briefly as follows.

The considerations from the flowcharts concerning the counter for the if-condition have been replaced by a loop for practical reasons. For this purpose, two global public constants have been defined for the beginning and end of the line area, as follows.

```
1.  Public Const row_start As Integer = 3
2.  Public Const row_count As Integer = 500
```

Since all sheets are structured identically, the first entry (BP or requirement) is in the third row. On average, the sources contain between 250 and 450 individual entries, which means that a run-through of up to 500 rows is currently sufficient.

Subsequently, the individual tasks (subs) for the mapping processes begin. Here, such a process is to be described on the basis of the exemplary process of mapping from Automotive SPICE 3.0 to Automotive SPICE 2.5 (reference). After the start of the subs, all necessary private variables for the sub are defined. *Application.ScreenUpdating* is set to false to prevent flickering of the screen during the mapping process. Everything is cached in the RAM and only output at the end. The variable *value()* is an array containing the individual IDs for each process cycle. A string, containing the title of the worksheet representing the source, is defined as source. This is declared as extra variable to easily customize the code for a new source when creating a new mapping action.

```
1.  Sub MapAS30toAS25()
2.  Application.ScreenUpdating = False
3.  Dim value() As String
4.  Dim value_count_from As Integer
5.  Dim value_count_to As Integer
6.  Dim source As String
```

The variable source is then filled with the exact title of the worksheet and the target spreadsheet is also defined. This is usually always the reference Automotive SPICE 2.5, except for the mapping of the Automotive SPICE guidelines, which refer to Automotive SPICE 3.0 and have to be mapped to the reference from the Automotive SPICE 3.0 worksheet.
"With the source worksheet" now allows the entries in the source sheet to be included in the current target sheet. These entries are called up in the following with a prefixed dot (“.”). An example would be “.Cells(row, column)”.

```
7.  source = "ASPICE 3.0"
8.  Worksheets("ASPICE 2.5").Select
9.  With Worksheets(source)
```

For safety reasons, a query was introduced using a message box, which the user has to confirm with “yes” when calling the mapping process. This prevents unintentional overwriting of the target cells on the table sheet. Only if the user clicks on “yes” will the process continue to run, otherwise the mapping process will be aborted. The following code fragment illustrates the realization of the query using *MsgBox* and its defined properties. Therefore, another variable for the answer is needed.

```
10. Dim answer As Integer
11. answer = MsgBox("Map " & source & " to this sheet?", vbYesNo + vbQuestion)
12. If answer = vbYes Then
```

Now the actual mapping process starts. For this purpose, each row in the target sheet (Automotive SPICE 2.5 reference) is sequentially looped through using the following for expression. This runs over the previously defined range. Afterwards, each value in column “B” of the target sheet (compare with **Figure 6.6**) is first separated by commas and passed to the value array as a string. Thus, the array contains every single base practice ID of all Automotive SPICE 2.5 processes after the entire loop has been executed. In parallel, the variables “value_count_from” and “value_count_to” are filled with the lowest available field index for the specified dimension of the array respectively the highest one. These variables are needed for another for loop afterwards.

```
13. For I = row_start To row_count
14.     value = Split(Cells(I, "B"), ", ")
15.     value_count_from = LBound(value)
16.     value_count_to = UBound(value)
```

Afterwards, all target cells are emptied row by row in the same for loop, to ensure a clean start for the entire mapping process.

```
17. Cells(I, "F") = ""
18. Cells(I, "G") = ""
19. Cells(I, "H") = ""
```

Afterwards, two more for loops are started as described in the flowchart. The first runs through the *value()* array, which contains all individual IDs of the reference spreadsheet, the second one runs again over the entire row range.

```
20. For j = value_count_from To value_count_to
21.      For k = row_start To row_count
```

Now the core of the mapping process begins, the actual comparison and duplication of cell contents from the source sheet to the target sheet. The content is first compared by means of an if statement. If in the string (text content) of the source cell (with the row of the innermost for loop’s run variable and the defined column “A”) matches the entry in the *value()* array (at the place of the run variable), the function *InStr* returns the value 1. This is checked in the if statement with “If InStr > 0”.

In the test phase during the development of the tool environment, a problem occurred here at first. The VBA function *InStr* only compares if a defined text element exists in another part of the text. This is also the case if it is not exactly identical. Wrongly “ENG.5” was also mapped with “ENG.6”. Therefore, the *InStr* function has been extended with “& ‘,’ ”. This means that for the result 1, a text component including comma at the end must be identical. Accordingly, all individual IDs must be entered in the source sheets in

the reference column “A”, separated by commas. This is also the case if only one base practice is the target cell. Correspondingly, after this example, the reference cells are entered in the source sheets as follows: “ENG.5.BP1, ENG.5.BP2, ”.
The following code fragment returns the individual IDs of the source cell (here: Automotive SPICE 3.0) coma-separated into the target cell as described above (e.g. “SWE.3.BP1, SWE.3.BP2, ").

```
If (InStr(.Cells(k, "A"), value(j) & ",") > 0) Then
     Cells(I, "F") = Cells(I, "F") & .Cells(k, "B") & ", "
End If
```

Similar to the same principle, the next code snippet copies further parts of the source sheet into the target sheet. In the case of Automotive SPICE 3.0, this is first the name of the base practice and then the full text description of the individual base practices. The names are separated by a line break (using the expression *vbCrLf*) and added to the target cells. For the description of the base practices, the unique ID of the base practice in square brackets was once again placed before the full text as a requirement of the industrial partner. A line break is inserted after the full text if several base practices from the source sheet apply to the same target sheet base practice. Analogous to this example, a mapping of the description will look like this: “[ID of BP] Description of BP”.

```
If (InStr(.Cells(k, "A"), value(j) & ",") > 0) Then
     Cells(I, "G") = Cells(I, "G") & .Cells(k, "C") & vbCrLf
End If
If (InStr(.Cells(k, "A"), value(j) & ",") > 0) Then
     Cells(I, "H") = Cells(I, "H") & "[" & .Cells(k, "B") & "] " & .Cells(k, "
   D") & vbCrLf
End If
```

After the copy process the three for loops are terminated with the command next respectively the counter of the loop is incremented by one step. In this case the expression “Step 1” can be skipped at the beginning of the for loop. The else of the if statements of the message box is called when the user clicks on “No” in the popup window at the beginning of the query. In that case, nothing happens, and the process ends. Finally, *Application.ScreenUpdating* is set to true again, which makes all changes in the target spreadsheet visible.

```
         Next
    Next
Next
Else
'do nothing if user clicked "no" in the message promt
End If
End With
```

```
38. Application.ScreenUpdating = True
39. End Sub ' End action
```

The complete code for an original mapping process (Automotive SPICE 3.0 to Automotive SPICE 2.5) can be found in appendix A.1.

In addition to the already explained problems, which could be solved by comma separation of the IDs, other weak points have been noticed during the tool testing. On the one hand, the question has arisen as to what happens with new processes respectively base practices that have no reference to Automotive SPICE 2.5. Since these could not be assigned to the Automotive SPICE 2.5 pendant, the new features were not visible on the reference sheet. To address this issue, a separate line for each process was added on the Automotive SPICE 2.5 reference sheet, on which all newly added base practices could be mapped. This new row can be addressed in the mapping process with *ProcessName.NEW*. In consultation with the industrial partner, two new lines were also added for each process in Automotive SPICE 2.5 and on the spreadsheet for Automotive SPICE 3.0, describing the purpose and outcomes of the respective process. This information is used to quickly obtain an overview of the process in the spreadsheets of the two VDA standards. The definitions are directly drawn from the standards. The purpose describes the aim and task of each process, and the outcomes reflect what results the process is supposed to deliver. Most base practices also refer to an outcome of the process. These outcomes are then recorded in the individual work products, which can be process-independent. **Figure 6.10** shows the described additions on the reference spreadsheet.

Automotive SPICE 2.5 REFERENCE			
P ID	**BP ID**	**Base Practice**	**Base Practice Full Text Description**
ENG.6	**HIS-Scope**	**Software construction**	
ENG.6		ENG.6 - PURPOSE	The purpose of the Software construction process is to produce verified software units that properly reflect the software design.
ENG.6		ENG.6 - OUTCOMES	1) a unit verification strategy is developed for software units consistent with the software design; 2) software units defined by the software design are analyzed for correctness and testability; 3) software units defined by the software design are produced; 4) software units are verified according to the unit verification strategy; 5) results of unit verification are recorded; and 6) consistency and bilateral traceability are established between software detailed design and software units; *NOTE 1: Analysis of software units will include prioritization and categorization of software units.* *NOTE 2: Unit verification will include unit testing and may include static analysis, code inspection/reviews, checks against coding standards and guidelines, and other techniques.*
ENG.6	**ENG.6.NEW**		
ENG.6	**ENG.6.BP1**	Define a unit verification strategy	Develop a strategy for verification and re-verifying the software units. The strategy should define how to achieve the desired quality with the available and suitable techniques over the complete range of allowed application parameter combinations. [Outcome 1] *NOTE 1: Possible techniques are static/dynamic analysis, code inspection/review, white/black box testing, code coverage, etc..* *NOTE 2: The unit verification strategy must include a unit test strategy if unit testing is stipulated by contract.*

Figure 6.10 Separate rows for process purpose, outcomes and new BPs [38]

The individual base practices and requirements have also been grouped by process in the spreadsheets. This simplifies the handling of more than 400 rows per sheet. For each source, the VBA code described above has been modified to suit typical characteristics. In particular, the target columns on the reference sheet had to be adjusted. The scripts were then linked to buttons that enable mapping per single source. Further scripts have been developed to empty the cells, disable filters, expand or collapse row groupings and customize the view.

Another topic was traceability and documentation for future users. For consistency reasons, the documentation and the manual are stored directly in an extra spreadsheet within the entire file. This means that the documentation cannot be lost, and any user who uses the file for analysis or reference will have the information ready to hand.

A further spreadsheet for diagrams and figures has also been added. This serves as a collective workbook for all process diagrams from or to the Automotive SPICE standard. In particular, the traceability and consistency diagrams as well as dependencies between the individual processes are provided there for reference. This also supports the allocation and analysis of individual base practices and requirements. In a further spreadsheet, the various terms and their definitions from the individual standards and requirements are compared. **Figure 6.11** below shows a screenshot of the finished tool concept for data handling and analysis in the reference sheet Automotive SPICE 2.5. For this example, the grouping for process ENG.5 is expanded, the mapping results of the other sources are not shown here.

	A	B	C	D	E	K	P	U	Z	AE
1	Automotive SPICE 2.5 REFERENCE			reset view / show all / Terms / How-To	Automotive SPICE 3.0	SPICE 3.0 Guidelines	OEM 1	OEM 2	OEM 3	MPT Processes
2	P ID	BP ID	Base Practice	Base Practice Full Text Description						
78	ACQ.15		Supplier qualification							
86	SPL.1		Supplier tendering							
97	SPL.2		Product release							
113	ENG.1		Requirements elicitation							
122	ENG.2	HIS-Scope	System requirements analysis							
132	ENG.3	HIS-Scope	Software architectural design							
142	ENG.4	HIS-Scope	Software requirements analysis							
153	ENG.5	HIS-Scope	Software design							
154	ENG.5		ENG.5 - PURPOSE	The purpose of the Software design process is to provide a design for the software that implements and can be verified against the software requirements.						
155	ENG.5		ENG.5 - OUTCOMES	1) a software architectural design is defined that identifies the components of the software and meets the defined software requirements; 2) the software requirements are allocated to the elements of the software; 3) internal and external interfaces of each software component are defined; 4) the dynamic behaviour and resource consumption objectives of the software components are defined; 5) a detailed design is developed that describes software units that can be implemented and tested; 6) consistency and bilateral traceability are established between software requirements and software architectural design; and 7) consistency and bilateral traceability are established between software architectural design and software detailed design. *NOTE 1: The software design process should take into account all software components such as customer supplied software, third party software and sub-contractor software.* *NOTE 2: Definition of software architectural design and detailed design includes development of verification criteria.*						
156		ENG.5.NEW								
	ENG.5	ENG.5.BP1	Develop software architectural design	Use the functional and non-functional software requirements to develop a software architecture that describes the top-level structure and all the						

ASPICE 2.5 | ASPICE 3.0 | SPICE Guidelines | OEM 1 | OEM 2 | OEM 3 | Internal Processes | Diagrams | Terms | Manual | Template

Figure 6.11 Screenshot of the tool in the reference sheet (ASPICE 2.5 [38])

As a further feature, the code for each source has been modified so that the contents of the reference sheet (target sheet) could be mapped back to the source sheet. Thus, it is now possible to display the individual Automotive SPICE 3.0 base practices or customer requirements as well as the corresponding Automotive SPICE 2.5 counterparts. This enables a targeted individual comparison and verification of the correct allocation already on the source sheet. **Figure 6.12** shows a screenshot of the source sheet of Automotive SPICE 3.0, showing the group for the MAN.3 process (project management). In the colored columns on the right-hand side, both the referenced base practices of Automotive SPICE 2.5 (analogous to the mapping in column A) and the corresponding Automotive SPICE guidelines can be compared.

In addition, extra documents were developed as agreed with the industrial partner to enable mapping of outcomes and work products between Automotive SPICE 2.5 and Automotive SPICE 3.0. This also helps in the deviation analysis between the two norms and facilitates the allocation of base practices.

The result of the concept development is a working concept environment of the tool for data storage and analysis, which can now be filled with the relevant data of the standards and customer requirements. This tool environment provides the possibility to create a comprehensive process model, which allows the analysis of deviations between standards and customer requirements.

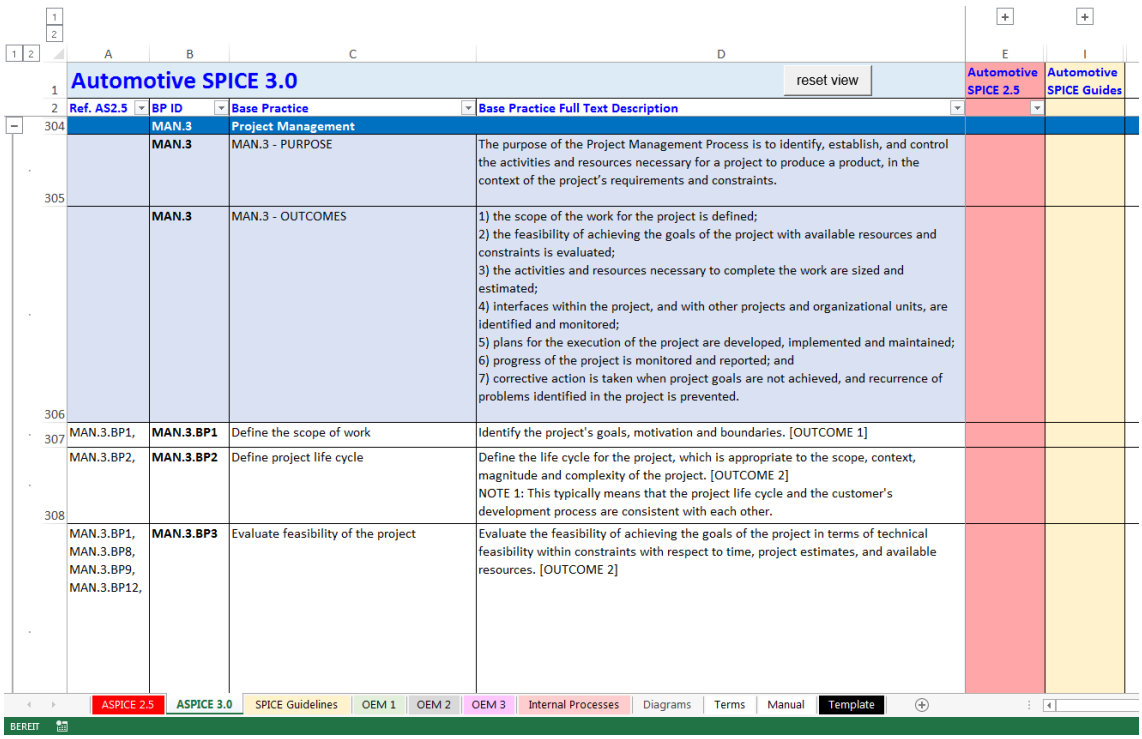


Figure 6.12 Screenshot of the tool in a source sheet (ASPICE 3.0 [34])

7 Optimization of Development Processes

This chapter describes the implementation of the tool concept described in detail in the previous chapter. This chapter also makes a comparison between the two standards Automotive SPICE 2.5 and Automotive SPICE 3.0 using an exemplary process. This comparison is based on the analysis between the allocation of the individual base practices, which was developed with the help of the developed tool concept for data management and analysis. First, the methodology is explained, on the basis of which the completion of the tool is carried out. Afterwards, difficulties that have occurred are identified and the first results and findings that have occurred during the analysis are described. The changes and deviations between two sources are then illustrated using an exemplary process. Subsequently, further potentials will be identified to expand the process optimization with the help of the developed tool.

7.1 Methodology of the Optimization

In order to create the process model and to develop allocations between the base practices and requirements in the tool concept, the tool must first be populated with the relevant data. For this purpose, the individual and relevant base practices and customer requirements were integrated into the tool from all sources with a unique ID. As described in the previous chapter, a separate spreadsheet has been created for each source in addition to Automotive SPICE 2.5. The IDs, names and full text descriptions of the individual base practices and requirements were then included in the tool concept. This basis of all the individual requirements and processes of the different sources allows not only allocation and comparison but also a comprehensive overview of all sources. In this tool, each given source can be searched and filtered by keywords or requirement IDs. This is also facilitated by the same structure of all spreadsheets and thus eliminates the need to search through several standards and customer documents.

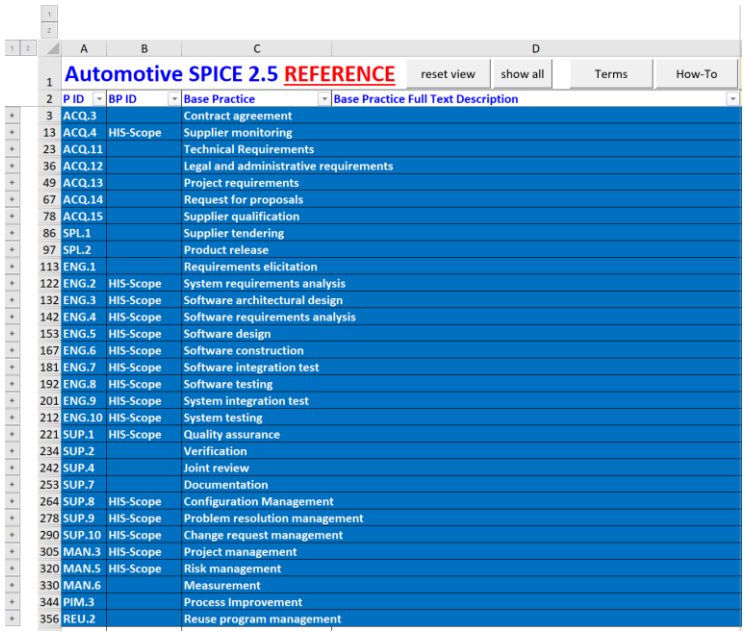
The relevant processes from the Automotive SPICE HIS-Scope were then allocated to the pendants from the relevant sources. As described in the previous chapter, one or more counterparts of the base practices of the standards were assigned to each entry on the source sheets. After these allocations were created, the tool could be automatically filled and mapped on the Automotive SPICE 2.5 reference sheet using all VBA mapping processes. For the mapping to be successful, the new standard Automotive SPICE 3.0 had to be mapped to the reference Automotive SPICE 2.5. After

successfully allocating the individual rules, the Automotive SPICE guidelines were then mapped to the corresponding base practices of Automotive SPICE 3.0. After this mapping process, the relationship between the Automotive SPICE guidelines and the Automotive 2.5 base practices can now be derived from the correlation on the same spreadsheet (Automotive SPICE 3.0). With the help of a further macro, all rules of the Automotive SPICE guidelines from the Automotive SPICE 3.0 spreadsheet could now be mapped to the Automotive SPICE 2.5 reference sheet. After the allocation between requirements and base practices, each source (OEM requirements) can then be mapped directly from the source sheet to the Automotive SPICE 2.5 reference sheet. This sheet now provides an overview and allows a comparison between the requirements and changes of the two standards, the rating rules (guidelines) and the individual customer requirements. For a better overview, another tool similar to the presented tool concept was developed, which allows the mapping and comparison of the individual outcomes and work products between the two versions of the Automotive SPICE standard.
After a successful mapping, the comprehensive process model now provides an overview of all customer requirements, rating rules and guidelines for Automotive SPICE assessors and a comparison of the changes between the Automotive SPICE versions. From this, it is possible to identify specific differences and derive the necessary actions for process optimization. In order to cover a broad spectrum of requirements, it is recommended to use the largest cutting quantity between all included sources. In this context, the tool also makes it possible to allocate a ranking for the effort and the identified deviations. The result of the populated tool is described in the following section.

7.2 Results and Findings

In this section the findings and general results of the analysis are explained in the context of the structure of the process model. An exemplary process is then used to illustrate the comparison and the changes between the two versions of Automotive SPICE and to derive optimization requirements for the development processes. The results described here are based on Automotive SPICE 2.5 and Automotive SPICE 3.0, although some of the Automotive SPICE guidelines are also taken into account. For reasons of confidentiality, the requirements of the individual customer standards cannot be explained. The concept tool for the process model was populated with all relevant data from the standards and sources. In addition to the completeness, the functional representation was also emphasized during the filling of the tool concept. **Figure 7.1**

shows an example of the content for the Automotive SPICE 2.5 base practices, which are grouped according to the processes. For reasons of completeness, it was agreed with the industrial partner that at least for the two versions of the Automotive SPICE standards as well as the Automotive SPICE guidelines all included processes will be transferred into the tool. For the mapping of the other sources, however, the defined HIS scope of the Automotive SIG remains the basis.



	A	B	C	D
1	Automotive SPICE 2.5 REFERENCE	reset view	show all	Terms	How-To
2	P ID	BP ID	Base Practice	Base Practice Full Text Description
3	ACQ.3		Contract agreement	
13	ACQ.4	HIS-Scope	Supplier monitoring	
23	ACQ.11		Technical Requirements	
36	ACQ.12		Legal and administrative requirements	
49	ACQ.13		Project requirements	
67	ACQ.14		Request for proposals	
78	ACQ.15		Supplier qualification	
86	SPL.1		Supplier tendering	
97	SPL.2		Product release	
113	ENG.1		Requirements elicitation	
122	ENG.2	HIS-Scope	System requirements analysis	
132	ENG.3	HIS-Scope	Software architectural design	
142	ENG.4	HIS-Scope	Software requirements analysis	
153	ENG.5	HIS-Scope	Software design	
167	ENG.6	HIS-Scope	Software construction	
181	ENG.7	HIS-Scope	Software integration test	
192	ENG.8	HIS-Scope	Software testing	
201	ENG.9	HIS-Scope	System integration test	
212	ENG.10	HIS-Scope	System testing	
221	SUP.1	HIS-Scope	Quality assurance	
234	SUP.2		Verification	
242	SUP.4		Joint review	
253	SUP.7		Documentation	
264	SUP.8	HIS-Scope	Configuration Management	
278	SUP.9	HIS-Scope	Problem resolution management	
290	SUP.10	HIS-Scope	Change request management	
305	MAN.3	HIS-Scope	Project management	
320	MAN.5	HIS-Scope	Risk management	
330	MAN.6		Measurement	
344	PIM.3		Process Improvement	
356	REU.2		Reuse program management	

Figure 7.1 Populated tool with collapsed process groups (ASPICE 2.5)

The first comparisons and analyses of the two Automotive SPICE versions revealed some differences, which in particular concern the processes of software development. On average, the processes outside the defined HIS scope have not changed as much as the processes in the V-model. It was apparent that comprehensive concepts of the new standard were implemented, such as the uniform integration of traceability and consistency. In Automotive SPICE, these two issues were divided into two separate base practices for bilateral traceability and consistency and were defined per process. These two base practices focus in particular on the relation between requirements and the outcome, the software design. The most prominent feature of the new standard was the division of engineering processes into system engineering and software engineering processes. The resulting plug-in model now allows the integration of suitable processes for the hardware and mechanics development into the entire process model in analogy to the SWE processes. A further differentiation and innovation was the detailing at the

base of the V-model. The Automotive SPICE 2.5 process of unit construction and verification was split into two separate processes. In Automotive SPICE 3.0, these now cover the unit construction and its verification separately.

The differences between the two standards are an integral part of the analysis of the process model, as the VDA Standard Automotive SPICE is the basis of all development processes. All deviations and changes to the new standard must therefore be known and implemented in the company. To achieve a high rating of the process capability level, the contents of the Automotive SPICE guidelines also had to be mapped to the individual base practices. From conversations with the industry partner, the focus was placed on the rating rules of the guidelines, leaving recommendations out of the question. The rules are therefore more important for assessors, since in the case of discrepancies, it is necessary to record in a written form what the differences to the standard are. They also specify directly which results of the individual practices are subject to devaluation or appreciation. Considering this in the process model from the very beginning, one can assume that more mature development processes will be implemented.

Figure 7.2 below shows an extract of the process model within the developed tool concept. The example shows the mapping between the two Automotive SPICE versions, the guidelines and the requirements of an OEM. What is already structurally evident in this sample is largely true for the entire mapping of the individual sources. Normally, when mapping the two Automotive SPICE versions, only one to three base practices are allocated to each other.

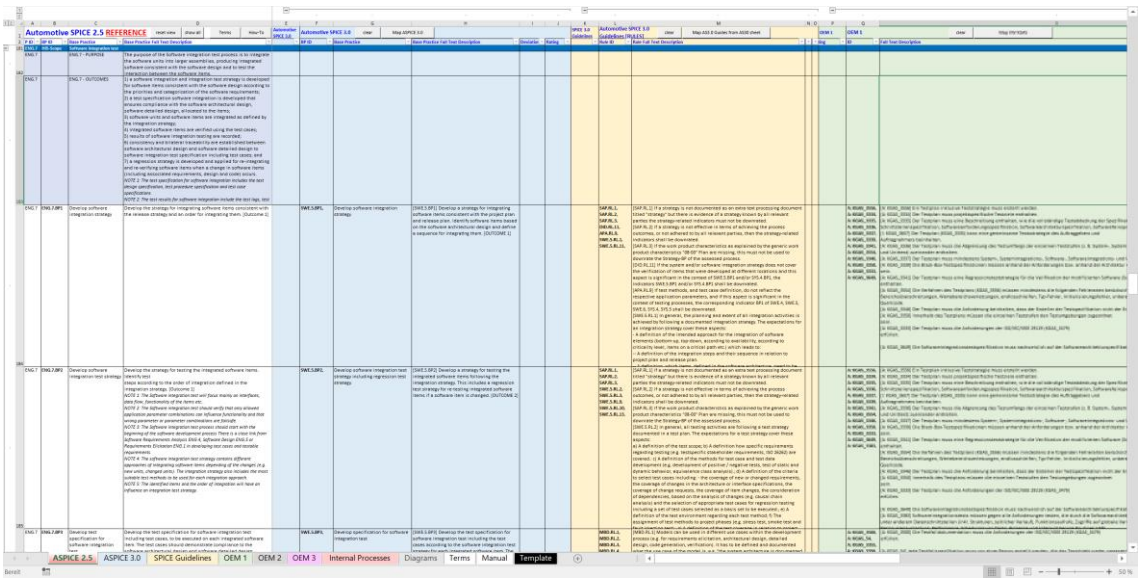


Figure 7.2 Tool screenshot of ASPICE 2.5, 3.0, guidelines and an OEM mapping

The rules of the Automotive SPICE guidelines are already more frequently applied. There are generic rules that apply to several base practices across all processes, as well as process-specific rules that are important for the assessment of individual processes. The OEMs' customer requirements are even more precise. These have divided individual base practices into many independent requirements. The manufacturers have not only subdivided the individual processes and base practices into several unique requirements, but have also in most cases refined and detailed them. As these stringent requirements of the various manufacturers are now known and can be allocated to the base practices of Automotive SPICE processes, this allows a direct comparison of all requirements for the development processes. If the manufacturer-specific requirements are also taken into account at the outset in the comprehensive process model, it is possible to fine-tune the optimization of internal development processes even more precisely to the market. Knowledge of customer requirements and own process capabilities in the development of mechatronic systems for automotive applications ultimately enables the development of more mature products by complying with all requirements. This also helps to save time when analyzing requirements and creating offers for new development projects.

7.3 Exemplary Processes

In the following section, the differences between the two Automotive SPICE versions are explained on the basis of two exemplary process scopes. First, the objectives and purpose of the two processes are explained. The result of the mapping of the individual base practices is then presented. On this basis, differences can now be identified. For reasons of simplicity, only a limited result of the mapping can be presented and explained in the written part of this thesis.

The selection of the two processes was based on the relevance of the previously described topics in the context of development processes. In the first case (MAN.3), this is a one-to-one assignment at process level, although the base practices have been redefined. In the second case (SWE.3), processes ENG.5 and ENG.6 of the Automotive SPICE 2.5 basis were split up into several new processes in Automotive SPICE 3.0. These were specified more precisely and adapted even better to the presented V-model. In addition to the following findings, detailed mappings of the individual base practices are provided in the appendix (starting with appendix A.2).

7.3.1 MAN.3 Project Management

The process MAN.3 (project management) is an important process, which has the objective *to identify, establish, and control the activities and resources necessary for a project to produce a product, in the context of the project's requirements and constraints* (Automotive SPICE 3.0) [34]. This process enables a holistic view of the activities involved in software development projects. MAN.3 is usually the initial process to be addressed in an Automotive SPICE assessment. This process is used by assessors to get a comprehensive overview of the project. The evaluation of the other processes is also heavily dependent on the outcome of the MAN.3 process rating, as the entire development project depends on consistent planning and control.

As a result of the right implementation of this process, the definition of the scope of the project is mentioned in particular. At the same time, there is a further focus on analyzing the feasibility and the necessary resources. Furthermore, the project must identify and monitor further interfaces to other areas of the project. However, the central issue is the planning of the entire development project based on a defined schedule. This project plan must always be monitored and reported. In the event of any deviations, appropriate measures must be taken in order to avoid missing the project goal [34].

Figure 7.3 shows the allocation of both matching process groups between Automotive SPICE 2.5 and Automotive SPICE 3.0. As is evident, the MAN.3 process exists in both versions of Automotive SPICE under the same name. On the basis of this overview, it can be assumed that an allocation and thus the mapping of the base practices works 1:1. However, this is not the case. Within the individual process, some base practices have been changed in the new version of Automotive SPICE. **Table 7.1** compares the respective base practices of the two standards.

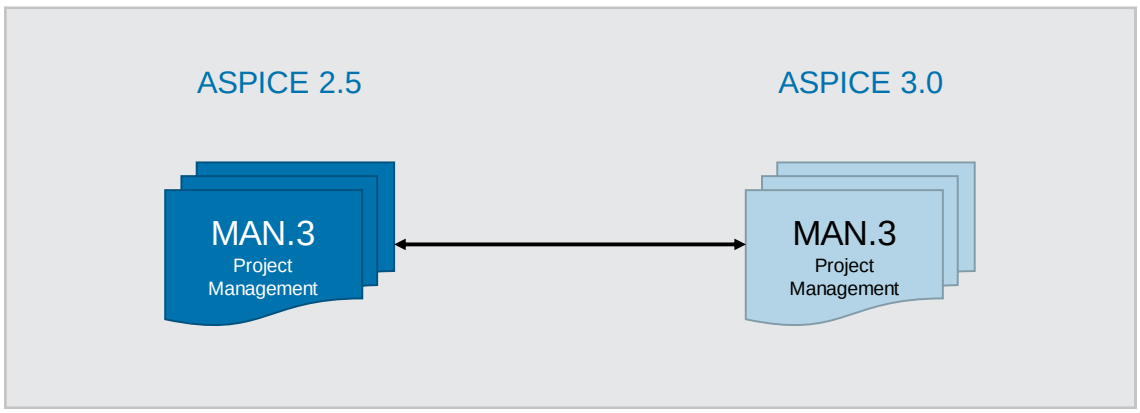


Figure 7.3 Allocation of the MAN.3 process between ASPICE 2.5 and 3.0

Table 7.1 List of individual MAN.3 base practices of ASPICE 2.5 and 3.0 [38, 34]

BP	MAN.3 in Automotive SPICE 2.5	MAN.3 in Automotive SPICE 3.0
1	Define the scope of work	Define the scope of work
2	Define project life cycle	Define project life cycle
3	Determine and maintain estimates for project attributes	Evaluate feasibility of the project
4	Define project activities	Define, monitor and adjust project activities
5	Define skill needs	Determine, monitor and adjust project estimates and resources
6	Define and maintain project schedule	Ensure required skills, knowledge and experience
7	Identify and monitor project interfaces	Identify, monitor and adjust project interfaces and agreed commitments
8	Establish project plan	Define, monitor and adjust project schedule
9	Implement the project plan	Ensure consistency
10	Monitor project attributes	Review and report progress of the project
11	Review and report progress of the project	-
12	Act to correct deviations	-

As can be seen without a detailed analysis, the number of base practices between the two versions has changed. In order to make a more precise analysis of the changes, the individual base practices were compared and mapped together in the tool concept. **Figure 7.4** shows schematically the result of the mapping of individual base practices.

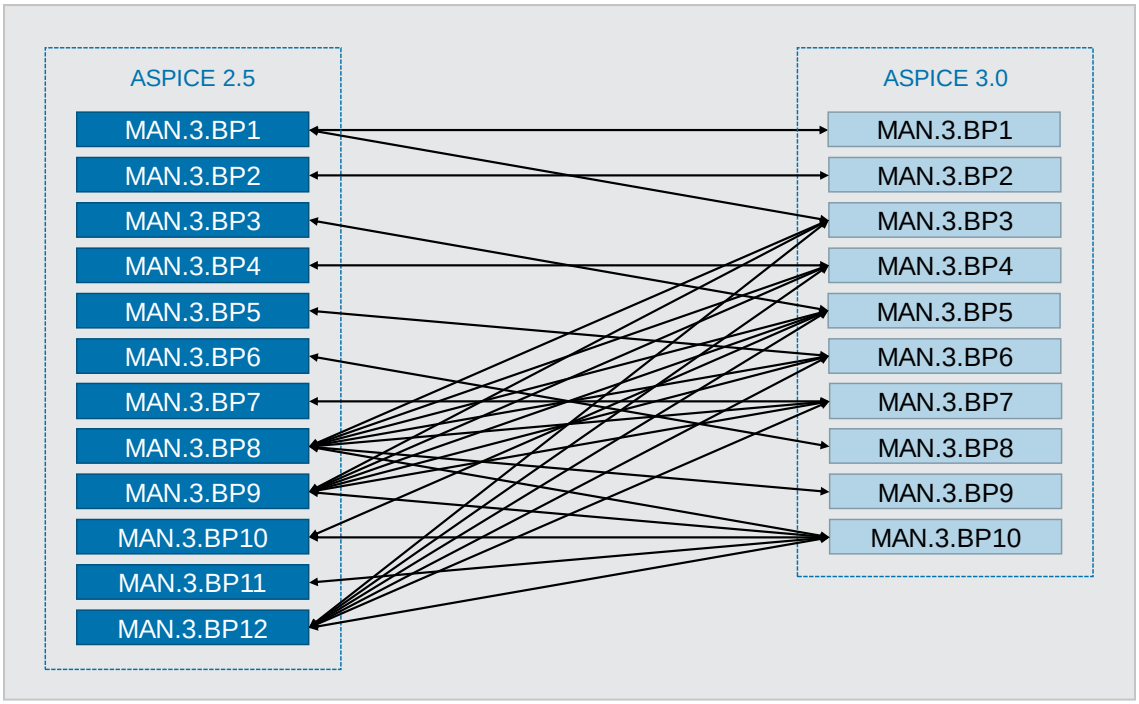


Figure 7.4 Mapping of MAN.3 base practices between ASPICE 2.5 and 3.0

As can be seen, these are complex nested n:m allocations between the individual base practices in a single process. This shows that not only unnecessary base practices have been omitted in the new version, but that stricter requirements have been explicitly defined as new base practices.

As a result of the MAN.3 comparison of both Automotive SPICE versions, the following points can be noted. Individual base practices have been adopted almost identically in the new version of Automotive SPICE (for example, MAN.3.BP2 *define project life cycle*). Existing base practices have also been split into two or more new base practices. This means that content can be defined even more precisely and recorded as separate outcomes. This has been the case, for example, with the MAN.3.BP1 process. The process (define scope of work) was split in the new version into two separate base practices (MAN.3.BP1 *define scope of work* and MAN.3.BP3 *evaluate feasibility of the project*). This does not appear from the individual names of the base practices, but has to be analyzed by means of the individual contents and outcomes. In this case feasibility was previously integrated in MAN.3.BP1. Another change and deviation from the old version of the standard was also discovered in the old base practice 12 (*act to correct deviations*). This covered actions to be taken when the project objectives are no longer achievable. This base practice no longer exists independently in the new version. Instead, this base practice has been integrated into several individual base practices. The addition of the adaptation can now be found, for example, in all base practices dealing with the definition and monitoring of project parameters (*schedule, resources and activities*). Considering that several rating rules of the Automotive SPICE guidelines are mapped to the individual MAN.3 base practices, it becomes evident how complex the entire process model is. In addition, many customer requirements are also mapped to the individual Automotive SPICE 2.5 base practices.

The detailed mapping of the MAN.3 process between the two Automotive SPICE versions in the developed tool concept is shown in appendix A.2.

7.3.2 SWE.3 Software Detailed Design and Unit Construction

The process SWE.3 (*software detailed design and unit construction*) in Automotive SPICE 3.0 mainly deals with the creation of software units using model-based software development. The goal is *to provide an evaluated detailed design for the software units and to produce the software units* (Automotive SPICE 3.0) [34]. In this case, it was more complicated than the previous MAN.3 process, since the SWE.3 process was divided into two individual processes in the old standard. Comparing the processes and their

purpose and outcomes alone, the SWE.3 process of Automotive SPICE 3.0 includes the topics of processes ENG.5 (*software design*) and ENG.6 (*software construction*). This is because the new standard has been detailed at the bottom of the V-model for software development. For the first time, an independent process for verifying the developed software units was created by means of SWE.4 (*software unit verification*). **Figure 7.5** shows the allocation of the individual process groups between Automotive SPICE 2.5 and Automotive SPICE 3.0 in the scope of SWE.3. One challenge is already apparent in this context: mapping the individual base practices beyond process group boundaries.

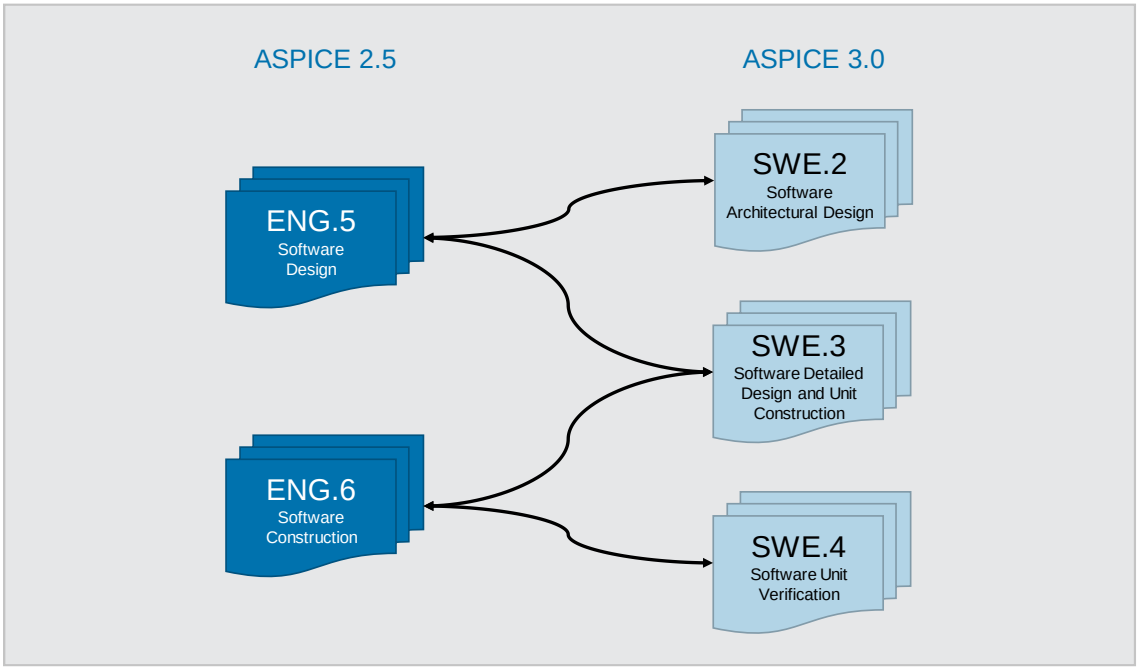


Figure 7.5 Allocation of the SWE.3 process in the context of ASPICE 2.5

The contents and mapping of the two Automotive SPICE versions will now be presented in the following. Because the SWE.3 process from Automotive SPICE 3.0 is considered as an example, the above figure shows a mapping of the base practices with the ENG.5 and ENG.6 process groups from the Automotive SPICE 2.5 reference. **Table 7.2** lists below the individual base practices of the three processes of each Automotive SPICE version. However, these cannot be assigned 1:1 to each other. The individual base practices indicate only sporadic instances of immediate correlation. Based on the names of the individual base practices of all three process groups, it can be seen quickly that the scope and content do not exactly match. SWE.3 has correlating base practices in both ENG.5 and ENG.6 processes.

Table 7.2 Base practices of ASPICE 2.5 and 3.0 (SWE.3 scope) [38, 34]

BP	ENG.5 of Automotive SPICE 2.5	SWE.3 of Automotive SPICE 3.0
1	Develop software architectural design	Develop software detailed design
2	Allocate software requirements	Define interfaces of software units
3	Define interfaces	Describe dynamic behavior
4	Describe dynamic behavior	Evaluate software detailed design
5	Define resource consumption objectives	Establish bidirectional traceability
6	Develop detailed design	Ensure consistency
7	Develop verification criteria	Communicate agreed software detailed design
8	Verify software design	Develop software units
9	Ensure consistency and bilateral traceability of software requirements to software architectural design	-
10	Ensure consistency and bilateral traceability of software architectural design to software detailed design	-
BP	**ENG.6 of Automotive SPICE 2.5**	
1	Define a unit verification strategy	
2	Analyze software units	
3	Prioritize and categorize software units	
4	Develop software units	
5	Develop unit verification criteria	
6	Verify software units	
7	Record the result of unit verification	
8	Ensure consistency and bilateral traceability of software detailed design to software units	
9	Ensure consistency and bilateral traceability of software requirements to software units	
10	Ensure consistency and bilateral traceability of software units to test specification for software units	

After an initial analysis of the two reference processes (ENG.5 and ENG.6) of Automotive SPICE 2.5, it became evident that the individual base practices cannot merely be allocated to the process scope of the SWE.3 process of Automotive SPICE 3.0. Moreover, many base practices of the two processes are mapped to individual or multiple base practices of the process groups SWE.2 and SWE.4 of Automotive SPICE 3.0. In order to map the individual base practices within a complex network across process boundaries, it was not enough to analyze and compare the contents of the individual base practices. In discussion with the industrial partner, a further tool concept was developed, which is capable of mapping and comparing the individual outcomes and work products of the various processes. In particular, this allocation and comparison of outcomes and work products was performed for the process groups SWE.2, SWE.3 and SWE.4 of Automotive SPICE 3.0. The detailed comparison in the new tool concept is provided in appendix A.4.

The now improved overview and allocation of the individual work products and outcomes helped in the following with the allocation and mapping of the actual base practices of the process groups. This is possible because the individual base practices refer to individual outcomes, which are then assigned to the individual work products for each process. This enabled a detailed analysis of the individual base practices and their requirements. **Figure 7.6** shows the result of the mapping of the individual base practices between Automotive SPICE 2.5 and Automotive SPICE 3.0 in the considered SWE.3 scope. In contrast to MAN.3, there are fewer n:m relationships in the mapping of base practices, but the challenge was in the described allocation across process boundaries.

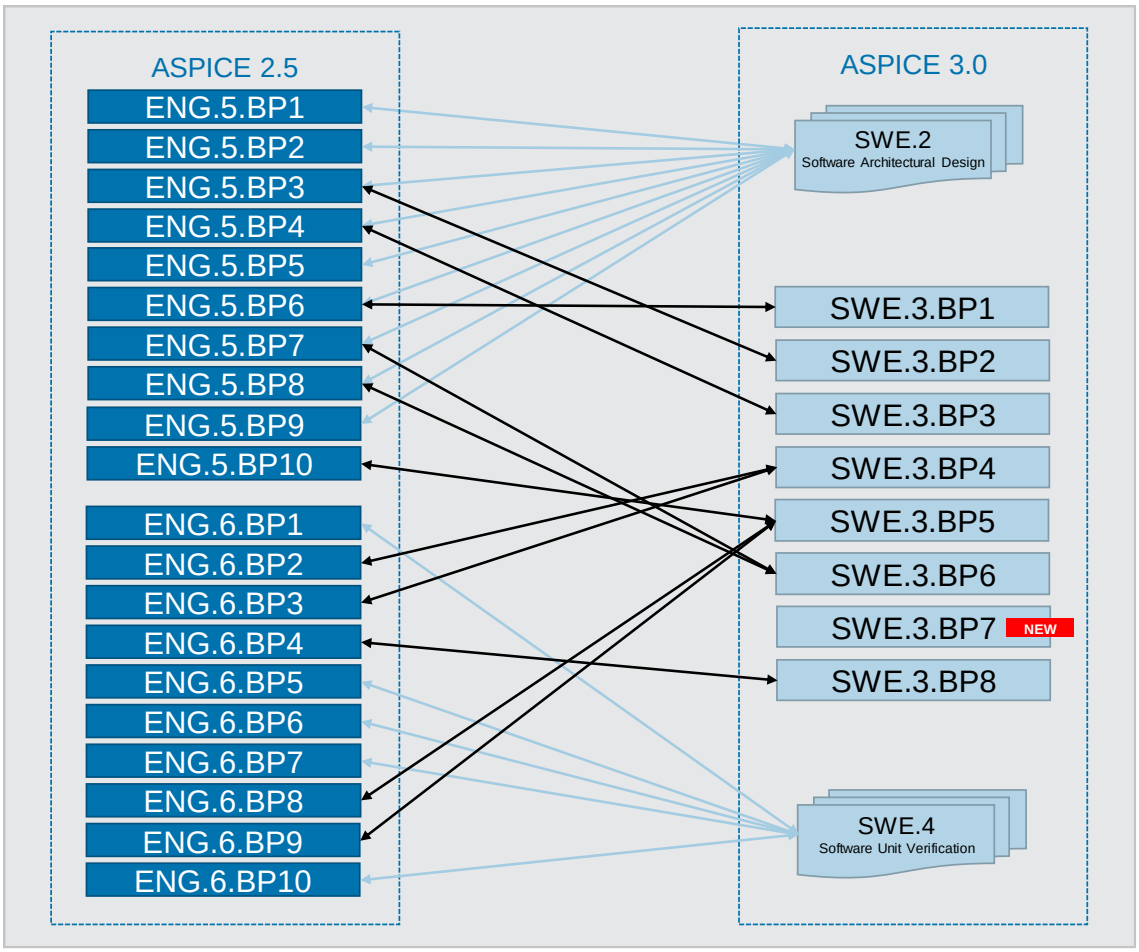


Figure 7.6 Mapping of SWE.3 base practices between ASPICE 2.5 and 3.0

Concepts such as SWE.3.BP1 (*develop software detailed design*) have been adopted directly from the old version of the standard. Other base practices, such as ENG.6.BP2 and BP3, have been combined in a single base practice in SWE.3.BP4 and described in detail. In addition to the remaining mapping and the resulting differences, SWE.3 also

introduced a new base practice, which could not be mapped directly to any existing base practice of Automotive SPICE 2.5. These is SWE.3.BP7 (*communicate agreed software detailed design*). This base practice prescribes that the defined software detailed design as well as any related updates must be communicated to all relevant parties.
A complete overview of the mapping of the base practices (SWE.3 scope) between Automotive SPICE 2.5 and Automotive SPICE 3.0 is provided in the appendix A.3.

7.4 Potentials of Optimization

The potentials for the optimization of internal processes can be derived directly from the deviations of the individual requirements in the process model within the tool concept. It is strongly recommended to implement all requirements of the new standard Automotive SPICE 3.0 and its rating rules from the Automotive SPICE guidelines. This ensures a high rating of the process capability level in Automotive SPICE assessments. The new process model should also incorporate the largest possible cutting quantity of customer requirements. The biggest differences can be found between the Automotive SPICE 2.5 reference and the individual OEM specification requirements. There are many potentials for improving development processes, especially at the bottom of the V-model and its corresponding processes. In particular, customer requirements prescribe stricter requirements for the individual processes and base practices, which even sometimes require explicit solutions and implementations.
Functionally, there are also potentials for further optimization of the resulting process model. In particular, the integration into a database, e.g. in an existing requirements management tool, is promoted by the modular structure of the tool concept.
In addition to the optimization potentials of individual processes, the entire process model can also be optimized by further implementations. For example, the process scope can be extended from the basis of the HIS scope to the complete Automotive SPICE scopes. In the future, the existing process model can also be extended to include all generic practices of the Automotive SPICE versions. This would also cover the assessment of higher process capability levels during Automotive SPICE assessments.
A further potential for optimizing the overall process model is the integration of additional customer requirements. This enables a wider range of customer requirements to be covered, but also allows to determine whether individual customer requirements can be met in advance. This helps to save resources, especially in the requirements and risk analysis of new OEM development projects.

In addition to the expansion of the scope and customer requirements, the process model can also be extended by further standards and other process models. In the following sections, two process models and international standards are presented as examples, which have great potential to further optimize the process model. Potential and commonalities are pointed out, but also deviations and challenges for implementation in the overall process model are highlighted.

7.4.1 Capability Maturity Model Integration

Capability Maturity Model Integration (CMMI) models are a collection of best practices that can help companies improve their processes. These models are developed by product teams from industry, similar to Automotive SPICE but are not industry-specific. CMMI is a model for process improvement of products and services that consists of five maturity levels that are achieved by implementing specific and generic goals of these maturity levels and all previous ones. In order to achieve an objective, generic and specific practices or acceptable alternatives must be fulfilled. Typically, organizations implementing CMMI improve their performance in terms of productivity, predictability and quality of the products. This makes processes more predictable and increases customer satisfaction [39, 40].

CMMI is generally the more extensive process model and is very widespread among companies, especially on the North American market. CMMI and Automotive SPICE have different concepts and approaches, but they are not mutually incompatible. Each of the two models contains aspects that are not present in the other model. Because of structural differences, mapping CMMI to Automotive SPICE is therefore not completely feasible. Nevertheless, CMMI and automotive SPICE can be used together, and an integrated process model makes sense in order to meet both requirements and further increase the own process capability [41].

Both standards cover the four categories of process areas associated with software product development: process management, project management, engineering and support. CMMI on the one hand, covers some disciplines and process areas that ASPICE does not cover. These include, for example, specific process areas such as integrated supplier management, integrated teaming and decision analysis and solution. On the other hand, Automotive SPICE covers some areas that are not fully covered by CMMI. These include supplier monitoring and reuse program management. It is noteworthy, that there are no areas exclusively covered by Automotive SPICE [42].

CMMI Process Areas	Coverage Level in ASPICE
Process Management	
Organizational Process Focus	M
Organizational Process Definition	M
Organizational Training	L
Organizational Process Performance	M
Organizational Innovation and Deployment	M
Project Management	
Project Planning	H
Project Monitoring and Control	H
Supplier Agreement Management	M
Integrated Project Management	H
Risk Management	H
Integrated Teaming	X
Integrated Supplier Management	X
Quantitative Project Management	M

CMMI Process Areas	Coverage Level in ASPICE
Engineering	
Requirements Development	H
Requirements Management	H
Technical Solution	M
Product Integration	H
Verification	H
Validation	L
Support	
Configuration Management	H
Process and Product Quality Assurance	M
Measurement and Analysis	H
Organizational Environment for Integration	X
Decision Analysis and Resolution	X
Causal Analysis and Resolution	M

Figure 7.7 Correspondence of CMMI and ASPICE [42]

Figure 7.7 shows the correspondence between CMMI and Automotive SPICE. The notation according to [42] is defined as follows:

- *H: more than 80% of CMMI-specific practices can be attributed to one or more Automotive SPICE process results.*
- *M: between 50% and 80% of specific practices can be mapped*
- *L: less than 50% of specific practices can be mapped*
- *X: the CMMI process area in the Automotive SPICE process is not covered.*

Since there are indeed some similarities, it would make sense to extend the process model with additional process models such as CMMI in the future. This can result in even more mature processes, taking into account the time and effort involved in implementation.

7.4.2 Relation to ISO 26262 – Functional Safety

The aforementioned standard ISO 26262 requires compliance with specific requirements in the application area of safety-related mechatronic systems. The standard focuses on functional safety assessment through an application model and a framework by proposing an automotive safety lifecycle based on a V-model and adapting the necessary activities during these lifecycle phases. The aim is to prove that all reasonable system safety conditions are met. The standard focuses on functional safety assessment

through an application model and a framework by proposing an automotive safety lifecycle based on a V-model and adapting the necessary activities during these lifecycle phases. The aim is to prove that all reasonable system safety objectives are fulfilled, to validate acceptance of safety on the basis of product-specific product features and to demonstrate the competence for system management by means of targeted verification. Several professional industry users are looking for a mapping between Automotive SPICE and ISO 26262, and indeed the two working groups of the VDA (Automotive SPICE and functional safety) are now working together. The overall finding is that there is a high coverage of the Automotive SPICE scope by the safety standard, but a low level of coverage of ISO 26262 by Automotive SPICE. This is especially due to the fact that ISO 26262 contains special specifications at product level in addition to the demands defined at process level as described in Automotive SPICE. Studies have shown that all processes in the HIS scope are fully supported by ISO 26262, except the processes SUP.8 and SUP.9 (*configuration management and problem resolution management*), which are only partially taken into account. At the same time, the process of ISO 26262 is only partially or not at all covered by Automotive SPICE. This applies in particular to the safety management, hazard analysis and risk assessment, safety concepts, safety validation and safety analysis processes. **Table 7.3** shows the matching concepts between central elements in both standards. There are many direct matches when comparing both standards, but also some deviations [29].

Table 7.3 Comparison and matching concepts (ASPICE and ISO 26262) [29]

ISO 26262 – Functional Safety	**Automotive SPICE**
Safety Lifecycle	Category
Work Product	Work Product
Requirement	Outcome
ASIL	-
-	Base Practice

The relationship between the two models can be summarized as follows: both models have requirements for processes that partly overlap, but partly differ. Automotive SPICE (from process capability level 2) is very beneficial for the implementation of functional safety. This is also a further potential for the partial implementation of ISO 26262 in the developed process model.

8 Summary

This master thesis presents the current state of the art in the development processes of complex mechatronic systems in automotive applications and illustrates the optimization of existing development processes using an example for the integration of Automotive SPICE into a comprehensive process model with customer requirements and internal processes.

At first, the theory of development processes and standardization is examined in detail. Subsequently, the VDA Standard Automotive SPICE is explained at length with a focus on the V-model orientation and assessment criteria. In order to develop mature products, it is now necessary to implement the new Automotive SPICE standards in a process model, taking into account the rating rules from the Automotive SPICE guidelines.

The thesis focuses on the development of the tool concept for data management and analysis of this process model. With the use of the tool, a comprehensive process model can be generated that includes customer requirements and internal processes in addition to the VDA standards and guidelines. Based on the successful mapping of the individual base practices with the new counterparts, rating rules and customer requirements, an extensive process model is developed, which is modularly structured that it can be supplemented by further standards and requirements in the future. In the developed tool concept, it is then possible to compare the different base practices and requirements and to identify differences. The deviations represent the potential for process optimization, whereby the customer requirements should be considered with the largest possible cutting quantity. This master thesis also provides an outlook on the future potential of this tool concept and the integrated process model. In particular, the implementation into a database is mentioned, as well as the later implementation of further standards such as CMMI or ISO 26262 – Functional Safety.

The resulting process model is constantly evolving and should help to assess the requirements and risks of customer projects, save resources and ultimately develop more mature products at an early stage of development.

9 Bibliography

[1] W. Roddeck, Einführung in die Mechatronik, Bochum: Vieweg+Teubner, 2016.

[2] K.-H. Grote and J. Feldhusen, Eds., Dubbel, Springer Berlin Heidelberg, 2007.

[3] Bosch Professional Automotive Information, Automotive Mechatronics, K. Reif, Ed., Friedrichshafen: Springer Vieweg, 2015.

[4] Robert Bosch GmbH, Konventioneller Antriebsstrang und Hybridantriebe, K. Reif, Ed., Wiesbaden: Vieweg+Teubner, 2010.

[5] J. Schäuffele and T. Zurawka, Automotive Software Engineering, Wiesbaden: Vieweg+Teubner, 2013.

[6] AUTOSAR, "Layered Software Architecture," 30 November 2016. [Online]. Available: https://www.autosar.org/fileadmin/files/standards/classic/4-3/software-architecture/general/auxiliary/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf. [Accessed 25 July 2017].

[7] T. Riepl et al., "Elektronik und Mechanik für Motor- und Getriebesteuerung," in *Handbuch Verbrennungsmotor*, Wiesbaden, Springer Fachmedien, 2015.

[8] A. Girard and C. Rommel, "Softwareentwicklung mit optimierten Prozessen," *ATZ elektronik,* no. 1, February 2016.

[9] B. Balasubramanian, "Entwicklungsprozesse für Kraftfahrzeuge unter den Einflüssen von Globalisierung und Lokalisierung," in *Forschung für das Auto von Morgen*, V. Schindler, Ed., Springer Berlin Heidelberg, 2008.

[10] M. Eigner, D. Roubanov and R. Zafirov, Eds., Modellbasierte Virtuelle Produktentwicklung, Kaiserslautern: Springer Vieweg, 2014.

[11] N. G. Leveson, Safeware. System Safety and Computers, Boston, MA: Addison-Wesley, 2001.

[12] R. G. Cooper, Winning At New Products: Accelerating the Process from Idea to Launch, Perseus, Ed., New York: Basic Books, 2001.

[13] R. R. Asche, Embedded Controller, Wiesbaden: Springer Vieweg, 2016.

[14] R. Otterbach and F. Schütte, "Effiziente Funktions- und Software-Entwicklung für mechatronische Systeme im Automobil," Paderborn, 2009.

[15] M. Staron, Automotive Software Architectures, Gothenburg: Springer International Publishing, 2017.

[16] K. Ehrlenspiel and H. Meerkamm, Integrierte Produktentwicklung, Munich: Hanser, 2013.

[17] J. Schlattmann and A. Seibel, Aufbau und Organisation von Entwicklungsprojekten, Hamburg: Springer-Verlag GmbH, 2017.

[18] P. Rodríguez et al., "Analyzing the Drivers of the Combination of Lean and Agile in Software Development Companies," in *Product-Focused Software Process Improvement*, Madrid, 2012.

[19] D. Winkler, R. Mordinyi and S. Biffl, "Research Prototypes versus Products: Lessons Learned from Software Development Processes in Research Projects," in *Systems, Software and Services Process Improvement*, Vienna, 2013.

[20] M. Müller et al., Automotive SPICE® in der Praxis, vol. 2, Heidelberg: dpunkt.verlag, 2016.

[21] K. Blind, A. Jungmittag and A. Mangelsdorf, Der gesamtwirtschaftliche Nutzen der Normung, Berlin: DIN Deutsches Institut für Normung e. V., 2011.

[22] DIN Deutsches Institut für Normung e.V., "DIN EN 45020 — Standardization and related activities," Beuth Verlag, Berlin, 2007.

[23] R. Nevalainen, A. Ruiz and T. Varkoi, "Making Software Safety Assessable and Transparent," in *Systems, Software and Services Process Improvement*, Dundalk, Springer, 2013.

[24] M. Hinsch, "Einführung in zertifizierbare QM-Systeme nach ISO 9001 und EN 9100," in *Qualitätsmanagement in der Luftfahrtindustrie*, Hamburg, Springer Vieweg, 2014.

[25] International Organization for Standardization, "ISO 26262 — Road vehicles — Functional safety," ISO, Geneva, 2011.

[26] P. Anderson, "Mehr Softwaresicherheit Statische Analysetools und die ISO 26262," *ATZ elektronik,* no. 1, February 2017.

[27] P. Liggesmeyer, Software-Qualität, Heidelberg: Spektrum Akademischer Verlag, 2009.

[28] S. Hecht, "Entwicklung des Reifegradmodells," in *Ein Reifegradmodell für die Bewertung und Verbesserung von Fähigkeiten im ERP-Anwendungsmanagement*, Wiesbaden, Springer Gabler, 2014.

[29] M. Adedjouma et al., "Merging the Quality Assessment of Processes and Products in Automotive Domain," in *Product-Focused Software Improvement*, Madrid, Springer Verlag, 2012.

[30] M. Hirz and H. Brunner, "Mechatronics Academy. Development Processes. Automotive SPICE / CMMI," Graz, 2017.

[31] J. Morenzin and B. Vanamali, "Angekommen in der Entwicklung," *QZ Qualität und Zuverlässigkeit,* no. 6, 2009.

[32] F. Bella et al., "Automotive SPICE 3.0," KUGLER MAAG CIE GmbH, Kornwestheim, 2015.

[33] tecmata GmbH, "Automotive SPICE 3.0 – What's new ?," Flörsheim, 2016.

[34] VDA QMC Working Group 13 / Automotive SIG, "Automotive SPICE Process Assessment / Reference Model," 2015.

[35] H. Höhn et al., Software Engineering nach Automotive SPICE, Heidelberg: dpunkt.verlag, 2009.

[36] KUGLER MAAG CIE GmbH, "Be ready for Automotive SPICE ® v3.0 (HIS Scope)," Kornwestheim, 2016.

[37] K. K.-H. Lai, "Statistical Analysis of Automotive SPICE Assessment Results," in *VDA Automotive SYS Conference 2017*, Berlin, 2017.

[38] Automotive SIG, "Automotive SPICE® Process Assessment Model," The SPICE User Group, 2010.

[39] I. L. Margarido et al., "Towards a Framework to Evaluate and Improve the Quality of Implementation of CMMI® Practices," in *Product-Focused Software Process Improvement*, Madrid, Springer-Verlag, 2012.

[40] CMMI Product Team, "CMMI® for Development, Version 1.3," Carnegie Mellon University, Software Engineering Process Management Program, Pittsburgh, 2010.

[41] G. Fessler, "CMMI® und Automotive SPICE© gemeinsam nutzen," June 2012. [Online]. Available: http://www.sbz-pec.de/downloads/detail/cmmi-und-automotive-spice-gemeinsam-nutzen/201202aspicecmmiv6.pdf. [Accessed 11 November 2017].

[42] H. Sassenburg and D. Kitson, "A Comparative Analysis of CMMI and Automotive SPICE," 14 June 2006. [Online]. Available: http://itq.ch/pdf/sepg/CMMI&AutomotiveSPICE_305b.pdf. [Accessed 11 November 2017].

10 List of Figures

11 List of Tables

12 List of Abbreviations

API	Application Programming Interface
ASIL	Automotive Safety Integrity Level
ASPICE	Automotive SPICE[1]
AutoSIG	Automotive Special Interest Group
BP	Base Practice
BUS	Binary Unit System
CAN	Controller Area Network
CASE	Computer-Aided Software Engineering
CMMI	Capability Maturity Model Integration
CPU	Central Processing Unit
DIN	Deutsches Institut für Normung
E/E	Electric / Electronic
ECU	Electronic Control Unit
EEPROM	Electrically Erasable Programmable Read-Only Memory
FuSa	Functional Safety
HIS	Herstellerinitiative Software
I/O	Input / Output
IC	Integrated Circuit
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
OEM	Original Equipment Manufacturer
PAM	Process Assessment Model
PLC	Product Life Cycle
PRM	Process Reference Model
QM	Quality Management
RAM	Random-Access Memory
RTE	Runtime Environment
SoC	System on Chip
SPICE	Software Process Improvement and Capability Determination
VDA	Verband der Automobilindustrie
VDI	Verein Deutscher Ingenieure
WP	Work Product

[1] Automotive SPICE® is a registered trademark of the Verband der Automobilindustrie e.V. (VDA)

A Appendix

A.1 Developed VBA Code for Mapping BPs from One Source

```
Sub MapAS30toAS25()
' Define local variables
Dim value() As String
Dim value_count_from As Integer
Dim value_count_to As Integer
Dim source As String
Dim answer As Integer

' Select source sheet for mapping.
source = "ASPICE 3.0"

' Ensures that the right target sheet is selected.
Worksheets("ASPICE 2.5").Select
' All variables starting with a dot refer to the source sheet
With Worksheets(source)

answer = MsgBox("Map " & source & " to this sheet?", vbYesNo + vbQuestion)
If answer = vbYes Then

' For each base practice ID; Range defined as public constants
For I = row_start To row_count

    ' Splitting the base practice ID (comma separated)
    value = Split(Cells(I, "B"), ", ")
    value_count_from = LBound(value)
    value_count_to = UBound(value)

    ' Clearing the target fields.
    Cells(I, "F") = ""
    Cells(I, "G") = ""
    Cells(I, "H") = ""

    ' For each value found, i.e. value(j)
    For j = value_count_from To value_count_to

        ' Search in source range.
        For k = row_start To row_count

            If (InStr(.Cells(k, "A"), value(j) & ",") > 0) Then
                ' Add/copy BP ID + insert comma.
                Cells(I, "F") = Cells(I, "F") & .Cells(k, "B") & ", "
            End If
            If (InStr(.Cells(k, "A"), value(j) & ",") > 0) Then
                ' Add/copy BP name + insert break.
                Cells(I, "G") = Cells(I, "G") & .Cells(k, "C") & vbCrLf
            End If
            If (InStr(.Cells(k, "A"), value(j) & ",") > 0) Then
                ' Add/copy:[BP Name] + description + break.
                Cells(I, "H") = Cells(I, "H") & "[" & .Cells(k, "B") & "] " &
.Cells(k, "D") & vbCrLf
            End If
        Next
    Next
Next
Else
'do nothing if user clicked "no" in the message promt
End If
End With
End Sub ' End action
```

A.2 Mapping between ASPICE 2.5 and 3.0 (MAN.3 Scope)

The following excerpt shows the mapping of the MAN.3 base practices between the Automotive SPICE 2.5 reference and the new version, Automotive SPICE 3.0.

Automotive SPICE 2.5 REFERENCE				Automotive SPICE 3.0		
P ID	BP ID	Base Practice	Base Practice Full Text Description	BP ID	Base Practice	Base Practice Full Text Description
MAN.3	HIS-Scope	Project management				
MAN.3		MAN.3 - PURPOSE	The purpose of the Project management process is to identify, establish, plan, co-ordinate, and monitor the activities, tasks, and resources necessary for a project to produce a product and/or service, in the context of the project's requirements and constraints.			
MAN.3		MAN.3 - OUTCOMES	1) the scope of the work for the project is defined; 2) the feasibility of achieving the goals of the project with available resources and constraints is evaluated; 3) the tasks and resources necessary to complete the work are sized and estimated; 4) interfaces between elements in the project, and with other project and organizational units, are identified and monitored; 5) plans for the execution of the project are developed, implemented and maintained; 6) progress of the project is monitored and reported; and 7) actions to correct deviations from the plan and to prevent recurrence of problems identified in the project are taken when project goals are not achieved. *NOTE 1: The necessary resources will include - people, development tools, hardware present in the ECU (CPU, RAM, Flash RAM, etc.), test equipment, methodologies.* *NOTE 2: The skills of the people and the technologies used to develop the project will need to be evaluated and if necessary, training courses, tool upgrades, introduction of new technologies, etc. need to be planned.* *NOTE 3: Plans for the execution of the project may contain among other elements, work break down structures, responsibilities, schedules, etc..*			
MAN.3	**MAN.3.BP1**	Define the scope of work	Define the work to be undertaken by the project, and confirm that the goals of the project are feasible with available resources and constraints. [Outcomes 1, 2]	**MAN.3.BP1,** **MAN.3.BP3,**	Define the scope of work Evaluate feasibility of the project	[MAN.3.BP1] Identify the project's goals, motivation and boundaries. [OUTCOME 1] [MAN.3.BP3] Evaluate the feasibility of achieving the goals of the project in terms of technical feasibility within constraints with respect to time, project estimates, and available resources. [OUTCOME 2]
MAN.3	**MAN.3.BP2**	Define project life cycle	Define the life cycle for the project, which is appropriate to the scope, context, magnitude and complexity of the project. [Outcome 2] *NOTE 1: The consistency between the project life cycle and the car development process should be verified.*	**MAN.3.BP2,**	Define project life cycle	[MAN.3.BP2] Define the life cycle for the project, which is appropriate to the scope, context, magnitude and complexity of the project. [OUTCOME 2] NOTE 1: This typically means that the project life cycle and the customer's development process are consistent with each other.
MAN.3	**MAN.3.BP3**	Determine and maintain estimates for project attributes	Define and maintain baselines for project attributes [Outcome 2]. *NOTE 2: Project attributes may include 1) business and quality goals for the project, 2) resources for the project and 3) project effort, schedule and budget.* *NOTE 3 Appropriate estimation methods should be used.* *NOTE 4: A development strategy is determined and resources for the development life cycle to satisfy requirements are estimated.* *NOTE 5: Resources may include required infrastructure and communication mechanisms.* *NOTE 6: Project risks and quality criteria may be considered when estimating project attributes.*	**MAN.3.BP5,**	Determine, monitor und adjust project estimates and resources	[MAN.3.BP5] Define, maintain, and adjust project estimates of effort and resources based on project's goals, project risks, motivation and boundaries. [OUTCOME 2, 3, 7] NOTE 4: Appropriate estimation methods should be used. NOTE 5: Examples of necessary resources are people, infrastructure (such as tools, test equipment, communication mechanisms...) and hardware/materials. NOTE 6: Project risks (using MAN.5) and quality criteria (using SUP.1) may be considered. NOTE 7: Estimations and resources typically include engineering, management and supporting processes.
MAN.3	**MAN.3.BP4**	Define project activities	Plan project activities according to defined project life cycle and estimations, define and monitor dependencies between activities. [Outcome 3, 5] *NOTE 7: The activities and related work packages should be of manageable size to ensure that adequate progress monitoring is possible.*	**MAN.3.BP4,**	Define, monitor and adjust project activities	[MAN.3.BP4] Define, monitor and adjust project activities and their dependencies according to defined project life cycle and estimations. Adjust activities and their dependencies as required. [OUTCOME 3, 5, 7] NOTE 2: A structure and a manageable size of the activities and related work packages support an adequate progress monitoring. NOTE 3: Project activities typically cover engineering, management and supporting processes.
MAN.3	**MAN.3.BP5**	Define skill needs	Identify required skills needed for the project and allocate them to individuals and teams. [Outcome 3]	**MAN.3.BP6,**	Ensure required skills, knowledge, and experience	[MAN.3.BP6] Identify the required skills, knowledge, and experience for the project and make sure the selected individuals and teams either have or acquire these in time. [OUTCOME 3, 7] NOTE 8: In the case of deviations from required skills and knowledge trainings are typically provided.
MAN.3	**MAN.3.BP6**	Define and maintain project schedule	Allocate resources to activities and determine schedule for each activity and for the whole project. [Outcome 3, 5] *NOTE 8: This includes appropriate re-planning.* *NOTE 9: Project time schedule has to keep updated during lifetime of the project continuously.*	**MAN.3.BP8,**	Define, monitor and adjust project schedule	[MAN.3.BP8] Allocate resources to activities, and schedule each activity of the whole project. The schedule has to be kept continuously updated during lifetime of the project. [OUTCOME 3, 5, 7] NOTE 10: This relates to all engineering, management and supporting processes.
MAN.3	**MAN.3.BP7**	Identify and monitor project interfaces	Identify and agree interfaces of the project with other (sub-) projects, organizational units and other stakeholders and monitor agreed commitments. [Outcome 4] *NOTE 10: The project planning and monitoring may include all involved parties like quality assurance, production, car integration, testing and prototype manufacturing.*	**MAN.3.BP7,**	Identify, monitor and adjust project interfaces and agreed commitments	[MAN.3.BP7] Identify and agree interfaces of the project with other (sub-) projects, organizational units and other affected stakeholders and monitor agreed commitments. [OUTCOME 4, 7] NOTE 9: Project interfaces relate to engineering, management and supporting processes.
MAN.3	**MAN.3.BP8**	Establish project plan	Collect and maintain project master plan and other relevant plans to document the project scope and goals, resources, infrastructure, interfaces and communication mechanisms. [Outcome 5]	**MAN.3.BP3,** **MAN.3.BP4,** **MAN.3.BP5,** **MAN.3.BP6,** **MAN.3.BP7,** **MAN.3.BP9,** **MAN.3.BP10,**	Evaluate feasibility of the project Define, monitor and adjust project activities Determine, monitor und adjust project estimates and resources Ensure required skills, knowledge, and experience Identify, monitor and adjust project interfaces and agreed commitments Ensure consistency Review and report progress of the project	[MAN.3.BP3] Evaluate the feasibility of achieving the goals of the project in terms of technical feasibility within constraints with respect to time, project estimates, and available resources. [OUTCOME 2] [MAN.3.BP4] Define, monitor and adjust project activities and their dependencies according to defined project life cycle and estimations. Adjust activities and their dependencies as required. [OUTCOME 3, 5, 7] NOTE 2: A structure and a manageable size of the activities and related work packages support an adequate progress monitoring. NOTE 3: Project activities typically cover engineering, management and supporting processes. [MAN.3.BP5] Define, maintain, and adjust project estimates of effort and resources based on project's goals, project risks, motivation and boundaries. [OUTCOME 2, 3, 7] NOTE 4: Appropriate estimation methods should be used. NOTE 5: Examples of necessary resources are people, infrastructure (such as tools, test equipment, communication mechanisms...) and hardware/materials. NOTE 6: Project risks (using MAN.5) and quality criteria (using SUP.1) may be considered. NOTE 7: Estimations and resources typically include engineering, management and supporting processes. [MAN.3.BP6] Identify the required skills, knowledge, and experience for the project and make sure the selected individuals and teams either have or acquire these in time. [OUTCOME 3, 7] NOTE 8: In the case of deviations from required skills and knowledge trainings are typically provided. [MAN.3.BP7] Identify and agree interfaces of the project with other (sub-) projects, organizational units and other affected stakeholders and monitor agreed commitments. [OUTCOME 4, 7] NOTE 9: Project interfaces relate to engineering, management and supporting processes. [MAN.3.BP9] Ensure that estimates, activities, schedules, plans, interfaces, and commitments for the project are consistent across affected parties. [OUTCOME 3, 4, 5, 7] [MAN.3.BP10] Regularly review and report the status of the project and the fulfillment of activities against estimated effort and duration to all affected parties. Prevent recurrence of problems identified. [OUTCOME 6, 7] NOTE 11: Project reviews may be executed at regular intervals by the management. At the end of a project, a project review contributes to identifying e.g. best practices and lessons learned.

Automotive SPICE 2.5 REFERENCE				Automotive SPICE 3.0		
P ID	BP ID	Base Practice	Base Practice Full Text Description	BP ID	Base Practice	Base Practice Full Text Description
MAN.3	**MAN.3.BP9**	Implement the project plan	Implement planning activities of the project. [Outcome 5]	**MAN.3.BP3, MAN.3.BP4, MAN.3.BP5, MAN.3.BP6, MAN.3.BP7, MAN.3.BP10,**	Evaluate feasibility of the project Define, monitor and adjust project activities Determine, monitor und adjust project estimates and resources Ensure required skills, knowledge, and experience Identify, monitor and adjust project interfaces and agreed commitments Review and report progress of the project	[MAN.3.BP3] Evaluate the feasibility of achieving the goals of the project in terms of technical feasibility within constraints with respect to time, project estimates, and available resources. [OUTCOME 2] [MAN.3.BP4] Define, monitor and adjust project activities and their dependencies according to defined project life cycle and estimations. Adjust activities and their dependencies as required. [OUTCOME 3, 5, 7] NOTE 2: A structure and a manageable size of the activities and related work packages support an adequate progress monitoring. NOTE 3: Project activities typically cover engineering, management and supporting processes. [MAN.3.BP5] Define, maintain, and adjust project estimates of effort and resources based on project's goals, project risks, motivation and boundaries. [OUTCOME 2, 3, 7] NOTE 4: Appropriate estimation methods should be used. NOTE 5: Examples of necessary resources are people, infrastructure (such as tools, test equipment, communication mechanisms…) and hardware/materials. NOTE 6: Project risks (using MAN.5) and quality criteria (using SUP.1) may be considered. NOTE 7: Estimations and resources typically include engineering, management and supporting processes. [MAN.3.BP6] Identify the required skills, knowledge, and experience for the project and make sure the selected individuals and teams either have or acquire these in time. [OUTCOME 3, 7] NOTE 8: In the case of deviations from required skills and knowledge trainings are typically provided. [MAN.3.BP7] Identify and agree interfaces of the project with other (sub-) projects, organizational units and other affected stakeholders and monitor agreed commitments. [OUTCOME 4, 7] NOTE 9: Project interfaces relate to engineering, management and supporting processes. [MAN.3.BP10] Regularly review and report the status of the project and the fulfillment of activities against estimated effort and duration to all affected parties. Prevent recurrence of problems identified. [OUTCOME 6, 7] NOTE 11: Project reviews may be executed at regular intervals by the management. At the end of a project, a project review contributes to identifying e.g. best practices and lessons learned.
MAN.3	**MAN.3.BP10**	Monitor project attributes	Monitor the defined project attributes and document significant deviations of them against the project plan. [Outcome 6] *NOTE 11: At minimum, project attributes of resources, effort and schedule (planned, actual and remaining) should be monitored by the project.*	**MAN.3.BP5, MAN.3.BP10,**	Determine, monitor und adjust project estimates and resources Review and report progress of the project	[MAN.3.BP5] Define, maintain, and adjust project estimates of effort and resources based on project's goals, project risks, motivation and boundaries. [OUTCOME 2, 3, 7] NOTE 4: Appropriate estimation methods should be used. NOTE 5: Examples of necessary resources are people, infrastructure (such as tools, test equipment, communication mechanisms…) and hardware/materials. NOTE 6: Project risks (using MAN.5) and quality criteria (using SUP.1) may be considered. NOTE 7: Estimations and resources typically include engineering, management and supporting processes. [MAN.3.BP10] Regularly review and report the status of the project and the fulfillment of activities against estimated effort and duration to all affected parties. Prevent recurrence of problems identified. [OUTCOME 6, 7] NOTE 11: Project reviews may be executed at regular intervals by the management. At the end of a project, a project review contributes to identifying e.g. best practices and lessons learned.
MAN.3	**MAN.3.BP11**	Review and report progress of the project	Regularly report and review the status of the project against the project plans to all affected parties. This includes reports to the car producer. Regularly evaluate the performance of the project. [Outcome 6] *NOTE 12: Project reviews may be executed at regular intervals by the management. At the end of a project, a project review will normally be held to identify best practices and lessons learned.*	**MAN.3.BP10,**	Review and report progress of the project	[MAN.3.BP10] Regularly review and report the status of the project and the fulfillment of activities against estimated effort and duration to all affected parties. Prevent recurrence of problems identified. [OUTCOME 6, 7] NOTE 11: Project reviews may be executed at regular intervals by the management. At the end of a project, a project review contributes to identifying e.g. best practices and lessons learned.
MAN.3	**MAN.3.BP12**	Act to correct deviations	Take action when project goals are not achieved, correct deviations from plan and prevent recurrence of problems identified in the project. Update project plans accordingly. [Outcome 7]	**MAN.3.BP3, MAN.3.BP4, MAN.3.BP5, MAN.3.BP6, MAN.3.BP7, MAN.3.BP10,**	Evaluate feasibility of the project Define, monitor and adjust project activities Determine, monitor und adjust project estimates and resources Ensure required skills, knowledge, and experience Identify, monitor and adjust project interfaces and agreed commitments Review and report progress of the project	[MAN.3.BP3] Evaluate the feasibility of achieving the goals of the project in terms of technical feasibility within constraints with respect to time, project estimates, and available resources. [OUTCOME 2] [MAN.3.BP4] Define, monitor and adjust project activities and their dependencies according to defined project life cycle and estimations. Adjust activities and their dependencies as required. [OUTCOME 3, 5, 7] NOTE 2: A structure and a manageable size of the activities and related work packages support an adequate progress monitoring. NOTE 3: Project activities typically cover engineering, management and supporting processes. [MAN.3.BP5] Define, maintain, and adjust project estimates of effort and resources based on project's goals, project risks, motivation and boundaries. [OUTCOME 2, 3, 7] NOTE 4: Appropriate estimation methods should be used. NOTE 5: Examples of necessary resources are people, infrastructure (such as tools, test equipment, communication mechanisms…) and hardware/materials. NOTE 6: Project risks (using MAN.5) and quality criteria (using SUP.1) may be considered. NOTE 7: Estimations and resources typically include engineering, management and supporting processes. [MAN.3.BP6] Identify the required skills, knowledge, and experience for the project and make sure the selected individuals and teams either have or acquire these in time. [OUTCOME 3, 7] NOTE 8: In the case of deviations from required skills and knowledge trainings are typically provided. [MAN.3.BP7] Identify and agree interfaces of the project with other (sub-) projects, organizational units and other affected stakeholders and monitor agreed commitments. [OUTCOME 4, 7] NOTE 9: Project interfaces relate to engineering, management and supporting processes. [MAN.3.BP10] Regularly review and report the status of the project and the fulfillment of activities against estimated effort and duration to all affected parties. Prevent recurrence of problems identified. [OUTCOME 6, 7] NOTE 11: Project reviews may be executed at regular intervals by the management. At the end of a project, a project review contributes to identifying e.g. best practices and lessons learned.

Content is directly from Automotive SPICE 2.5 [38] and Automotive SPICE 3.0 [34].

A.3 Mapping between ASPICE 2.5 and 3.0 (SWE.3 Scope)

Automotive SPICE 2.5 REFERENCE				Automotive SPICE 3.0		
P ID	BP ID	Base Practice	Base Practice Full Text Description	BP ID	Base Practice	Base Practice Full Text Description
ENG.5	HIS-Scope	Software design				
ENG.5		ENG.5 - PURPOSE	The purpose of the Software design process is to provide a design for the software that implements and can be verified against the software requirements.			
ENG.5		ENG.5 - OUTCOMES	1) a software architectural design is defined that identifies the components of the software and meets the defined software requirements; 2) the software requirements are allocated to the elements of the software; 3) internal and external interfaces of each software component are defined; 4) the dynamic behaviour and resource consumption objectives of the software components are defined; 5) a detailed design is developed that describes software units that can be implemented and tested; 6) consistency and bilateral traceability are established between software requirements and software architectural design; and 7) consistency and bilateral traceability are established between software architectural design and software detailed design. *NOTE 1: The software design process should take into account all software components such as customer supplied software, third party software and sub-contractor software.* *NOTE 2: Definition of software architectural design and detailed design includes development of verification criteria.*			
ENG.5	**ENG.5.NEW**			**SWE.3.BP7,**	Communicate agreed software detailed design	[SWE.3.BP7] Communicate the agreed software detailed design and updates to the software detailed design to all relevant parties. [OUTCOME 5]
ENG.5	**ENG.5.BP1**	Develop software architectural design	Use the functional and non-functional software requirements to develop a software architecture that describes the top-level structure and all the software components including software components available for reuse. [Outcome 1] *NOTE 1: See also REU.2 – Reuse Program Management.*	**SWE.2.BP1,**	Develop software architectural design	[SWE.2.BP1] Develop and document the software architectural design that specifies the elements of the software with respect to functional and non-functional software requirements. [OUTCOME 1] NOTE 1: The software is decomposed into elements across appropriate hierarchical levels down to the software components (the lowest level elements of the software architectural design) that are described in the detailed design.
ENG.5	**ENG.5.BP2**	Allocate software requirements	Allocate all software requirements to the components of the software architectural design. [Outcome 2]	**SWE.2.BP2,**	Allocate software requirements	[SWE.2.BP2] Allocate the software requirements to the elements of the software architectural design. [OUTCOME 2]
ENG.5	**ENG.5.BP3**	Define interfaces	Identify, develop and document the internal interfaces between the software components and external + interfaces of the software components. [Outcome 3] *NOTE 2: Interfaces include specific interfaces required for application parameter usage.*	**SWE.2.BP3,** **SWE.3.BP2,**	Define interfaces of software elements Define interfaces of software units	[SWE.2.BP3] Identify, develop and document the interfaces of each software element. [OUTCOME 3] [SWE.3.BP2] Identify, specify and document the interfaces of each software unit. [OUTCOME 2]
ENG.5	**ENG.5.BP4**	Describe dynamic behaviour	Evaluate and document the dynamic behaviour of and interaction between software components. [Outcome 4] *NOTE 3: Dynamic behaviour is determined by operating modes (e.g. start-up, shutdown, normal mode, calibration, diagnosis, etc.), processes and process intercommunication, tasks, threads, time slices, interrupts, etc. and shall be evaluated over the complete range of allowed application parameter combinations.* *NOTE 4: Task execution time is highly depended on target and loads on the target which should be considered and documented.*	**SWE.2.BP4,** **SWE.3.BP3,**	Describe dynamic behavior Describe dynamic behavior	[SWE.2.BP4] Evaluate and document the timing and dynamic interaction of software elements to meet the required dynamic behavior of the system. [OUTCOME 4] NOTE 2: Dynamic behavior is determined by operating modes (e.g. start-up, shutdown, normal mode, calibration, diagnosis, etc.), processes and process intercommunication, tasks, threads, time slices, interrupts, etc. NOTE 3: During evaluation of the dynamic behavior the target platform and potential loads on the target should be considered. [SWE.3.BP3] Evaluate and document the dynamic behavior of and the interaction between relevant software units. [OUTCOME 3] NOTE 1: Not all software units have dynamic behavior to be described.
ENG.5	**ENG.5.BP5**	Define resource consumption objectives	Determine and document the resource consumption objectives for all software components. [Outcome 4] *NOTE 5: Resource consumption is typically determined for resources like Memory (ROM, RAM, external/internal EEPROM), CPU load, etc. and can vary over the complete range of allowed application parameter combinations.*	**SWE.2.BP5,**	Define resource consumption objectives	[SWE.2.BP5] Determine and document the resource consumption objectives for all relevant elements of the software architectural design on the appropriate hierarchical level. [OUTCOME 4] NOTE 4: Resource consumption is typically determined for resources like Memory (ROM, RAM, external / internal EEPROM or Data Flash), CPU load, etc.,
ENG.5	**ENG.5.BP6**	Develop detailed design	Decompose the software architectural design into a detailed design for each software component describing all software units and their interfaces. [Outcome 5] *NOTE 6: Task execution time is highly depended on target and loads on the target which should be considered and documented*	**SWE.3.BP1,**	Develop software detailed design	[SWE.3.BP1] Develop a detailed design for each software component defined in the software architectural design that specifies all software units with respect to functional and non-functional software requirements. [OUTCOME 1]
ENG.5	**ENG.5.BP7**	Develop Verification Criteria	Define the verification criteria for each component concerning their dynamic behaviour, interfaces and resource consumption based on the software architectural design. [Outcome 5] *NOTE 7: Verification criteria should be developed over the complete range of allowed application parameter combinations.*	**SWE.2.BP8,** **SWE.3.BP6,**	Ensure consistency Ensure consistency	[SWE.2.BP8] Ensure consistency between software requirements and the software architectural design. [OUTCOME 1, 2, 5, 6] NOTE 8: Consistency is supported by bidirectional traceability and can be demonstrated by review records. NOTE 9: Software requirements include software architectural requirements, refer to BP6. [SWE.3.BP6] Ensure consistency between software requirements an software units. Ensure consistency between the software architectural design, the software detailed design and software units. [OUTCOME 4] NOTE 5: Consistency is supported by bidirectional traceability and can be demonstrated by review records.
ENG.5	**ENG.5.BP8**	Verify Software Design	Ensure that the software design meets all software requirements. [Outcomes 4, 5] *NOTE 8: Software design should be verified over the complete range of allowed application parameter combinations.*	**SWE.2.BP8,** **SWE.3.BP6,**	Ensure consistency Ensure consistency	[SWE.2.BP8] Ensure consistency between software requirements and the software architectural design. [OUTCOME 1, 2, 5, 6] NOTE 8: Consistency is supported by bidirectional traceability and can be demonstrated by review records. NOTE 9: Software requirements include software architectural requirements, refer to BP6. [SWE.3.BP6] Ensure consistency between software requirements an software units. Ensure consistency between the software architectural design, the software detailed design and software units. [OUTCOME 4] NOTE 5: Consistency is supported by bidirectional traceability and can be demonstrated by review records.

Automotive SPICE 2.5 REFERENCE				Automotive SPICE 3.0		
P ID	BP ID	Base Practice	Base Practice Full Text Description	BP ID	Base Practice	Base Practice Full Text Description
ENG.5	**ENG.5.BP9**	Ensure consistency and bilateral traceability of software requirements to software architectural design	Ensure consistency of software requirements including verification criteria to software architectural design including verification criteria. Consistency is supported by establishing and maintaining bilateral traceability between the software requirements including verification criteria and software architectural design including verification criteria. [Outcome 6]	**SWE.2.BP7, SWE.2.BP8,**	Establish bidirectional traceability Ensure consistency	[SWE.2.BP7] Establish bidirectional traceability between software requirements and elements of the software architectural design. [OUTCOME 5] NOTE 6: Bidirectional traceability covers allocation of software requirements to the elements of the software architectural design. NOTE 7: Bidirectional traceability supports coverage, consistency and impact analysis. [SWE.2.BP8] Ensure consistency between software requirements and the software architectural design. [OUTCOME 1, 2, 5, 6] NOTE 8: Consistency is supported by bidirectional traceability and can be demonstrated by review records. NOTE 9: Software requirements include software architectural requirements, refer to BP6.
ENG.5	**ENG.5.BP10**	Ensure consistency and bilateral traceability of software architectural design to software detailed design	Ensure consistency of software architectural design including verification criteria to software detailed design including verification criteria. Consistency is supported by establishing and maintaining bilateral traceability between the software architectural design including verification criteria and software detailed design including verification criteria. [Outcome 7]	**SWE.3.BP5,**	Establish bidirectional traceability	[SWE.3.BP5] Establish bidirectional traceability between software requirements and software units. Establish bidirectional traceability between the software architectural design and the software detailed design. Establish bidirectional traceability between the software detailed design and software units. [OUTCOME 4] NOTE 3: Redundancy should be avoided by establishing a combination of these approaches that covers the project and the organizational needs. NOTE 4: Bidirectional traceability supports coverage, consistency and impact analysis.
ENG.6	**HIS-Scope**	**Software construction**				
ENG.6		ENG.6 - PURPOSE	The purpose of the Software construction process is to produce verified software units that properly reflect the software design.			
ENG.6		ENG.6 - OUTCOMES	1) a unit verification strategy is developed for software units consistent with the software design; 2) software units defined by the software design are analyzed for correctness and testability; 3) software units defined by the software design are produced; 4) software units are verified according to the unit verification strategy; 5) results of unit verification are recorded; and 6) consistency and bilateral traceability are established between software detailed design and software units; *NOTE 1: Analysis of software units will include prioritization and categorization of software units.* *NOTE 2: Unit verification will include unit testing and may include static analysis, code inspection/reviews, checks against coding standards and guidelines, and other techniques.*			
ENG.6	**ENG.6.NEW**			**SWE.4.BP7,**	Summarize and communicate results	[SWE.4.BP7] Summarize the unit test results and static verification results and communicate them to all affected parties. [OUTCOME 5] NOTE 9: Providing all necessary information from the test case execution in a summary enables other parties to judge the consequences.
ENG.6	**ENG.6.BP1**	Define a unit verification strategy	Develop a strategy for verification and re-verifying the software units. The strategy should define how to achieve the desired quality with the available and suitable techniques over the complete range of allowed application parameter combinations. [Outcome 1] *NOTE 1: Possible techniques are static/dynamic analysis, code inspection/review, white/black box testing, code coverage, etc..* *NOTE 2: The unit verification strategy must include a unit test strategy if unit testing is stipulated by contract.*	**SWE.4.BP1,**	Develop software unit verification strategy including regression strategy	[SWE.4.BP1] Develop a strategy for verification of the software units including regression strategy for re-verification if a software unit is changed. The verification strategy shall define how to provide evidence for compliance of the software units with the software detailed design and with the non-functional requirements. [OUTCOME 1] NOTE 1: Possible techniques for unit verification include static/dynamic analysis, code reviews, unit testing etc.
ENG.6	**ENG.6.BP2**	Analyze software units	Analyze the defined software units in terms of interoperability, interaction, criticality, technical complexity, risks and testability. [Outcome 2] *NOTE 3: The results of the analysis may be used for categorization of software units.*	**SWE.3.BP4,**	Evaluate software detailed design	[SWE.3.BP4] Evaluate the software detailed design in terms of interoperability, interaction, criticality, technical complexity, risks and testability. [OUTCOME 1,2,3,4] NOTE 2: The results of the evaluation can be used as input for software unit verification.
ENG.6	**ENG.6.BP3**	Prioritize and categorize software units	Prioritize and categorize the identified and analyzed software units and map them to future releases. [Outcome 2]	**SWE.3.BP4,**	Evaluate software detailed design	[SWE.3.BP4] Evaluate the software detailed design in terms of interoperability, interaction, criticality, technical complexity, risks and testability. [OUTCOME 1,2,3,4] NOTE 2: The results of the evaluation can be used as input for software unit verification.
ENG.6	**ENG.6.BP4**	Develop software units	Develop and document the executable representations of each software unit. [Outcome 3] *NOTE 4: In the development of software units code generation tools can be used to reduce the manual coding effort.*	**SWE.3.BP8,**	Develop software units	[SWE.3.BP8] Develop and document the executable representations of each software unit according to the software detailed design. [OUTCOME 6]
ENG.6	**ENG.6.BP5**	Develop unit verification criteria	Develop and document verification criteria to verify that each software unit satisfies its design, functional and non-functional requirements over the complete range of allowed application parameter combinations. [Outcome 3] *NOTE 5: The verification criteria should include unit test cases, unit test data, coverage goals and coding standards that include the usage of MISRA rules and defined coding guidelines.* *NOTE 6: The verification criteria must include test specifications for software units including test cases if uni testing is stipulated by contract.*	**SWE.4.BP2,**	Develop criteria for unit verification	[SWE.4.BP2] Develop criteria for unit verification that are suitable to provide evidence for compliance of the software units with the software detailed design and with the non-functional requirements according to the verification strategy. For unit testing, criteria shall be defined in a unit test specification. [OUTCOME 2] NOTE 2: Possible criteria for unit verification include unit test cases, unit test data, static verification, coverage goals and coding standards such as the MISRA rules. NOTE 3: The unit test specification may be implemented e.g. as a script in an automated test bench.
ENG.6	**ENG.6.BP6**	Verify software units.	Verify software units against the detailed design according to the verification strategy and the unit verification criteria. [Outcome 4]	**SWE.4.BP3, SWE.4.BP4,**	Perform static verification of software units Test software units	[SWE.4.BP3] Verify software units for correctness using the defined criteria for verification. Record the results of the static verification. [OUTCOME 3] NOTE 4: Static verification may include static analysis, code reviews, checks against coding standards and guidelines, and other techniques. NOTE 5: See SUP.9 for handling of non-conformances. [SWE.4.BP4] Test software units using the unit test specification according to the software unit verification strategy. Record the test results and logs. [OUTCOME 3] NOTE 6: See SUP.9 for handling of non-conformances.

Automotive SPICE 2.5 REFERENCE				Automotive SPICE 3.0		
P ID	BP ID	Base Practice	Base Practice Full Text Description	BP ID	Base Practice	Base Practice Full Text Description
ENG.6	**ENG.6.BP7**	Record the results of unit verification	Document the results of unit verification and communicate to all relevant parties. [Outcome 5]	**SWE.4.BP4, SWE.4.BP7,**	Test software units Summarize and communicate results	[SWE.4.BP4] Test software units using the unit test specification according to the software unit verification strategy. Record the test results and logs. [OUTCOME 3] NOTE 6: See SUP.9 for handling of non-conformances. [SWE.4.BP7] Summarize the unit test results and static verification results and communicate them to all affected parties. [OUTCOME 5] NOTE 9: Providing all necessary information from the test case execution in a summary enables other parties to judge the consequences.
ENG.6	**ENG.6.BP8**	Ensure consistency and bilateral traceability of software detailed design to software units	Ensure consistency of software detailed design including verification criteria to software units including verification criteria. Consistency is supported by establishing and maintaining bilateral traceability between the software detailed design including verification criteria and software units including verification criteria. [Outcome 6]	**SWE.3.BP5,**	Establish bidirectional traceability	[SWE.3.BP5] Establish bidirectional traceability between software requirements and software units. Establish bidirectional traceability between the software architectural design and the software detailed design. Establish bidirectional traceability between the software detailed design and software units. [OUTCOME 4] NOTE 3: Redundancy should be avoided by establishing a combination of these approaches that covers the project and the organizational needs. NOTE 4: Bidirectional traceability supports coverage, consistency and impact analysis.
ENG.6	**ENG.6.BP9**	Ensure consistency and bilateral traceability of software requirements to software units	Ensure consistency of software requirements including verification criteria to software units including verification criteria. Consistency is supported by establishing and maintaining bilateral traceability between the software requirements including verification criteria and software units including verification criteria. [Outcome 6] *NOTE 7: Consistency and bilateral traceability need only be established between software requirements and software units for requirements that cannot be addressed in software detailed design (e.g. non functional requirements, attributes etc.).*	**SWE.3.BP5,**	Establish bidirectional traceability	[SWE.3.BP5] Establish bidirectional traceability between software requirements and software units. Establish bidirectional traceability between the software architectural design and the software detailed design. Establish bidirectional traceability between the software detailed design and software units. [OUTCOME 4] NOTE 3: Redundancy should be avoided by establishing a combination of these approaches that covers the project and the organizational needs. NOTE 4: Bidirectional traceability supports coverage, consistency and impact analysis.
ENG.6	**ENG.6.BP10**	Ensure consistency and bilateral traceability of software units to test specification for software units	Ensure consistency of software units including verification criteria to test specification for software units including test cases for software units. Consistency is supported by establishing and maintaining bilateral traceability between the software units including verification criteria and test specification for software units including test cases for software units. [Outcome 6]	**SWE.4.BP5, SWE.4.BP6,**	Establish bidirectional traceability Ensure consistency	[SWE.4.BP5] Establish bidirectional traceability between software units and static verification results. Establish bidirectional traceability between the software detailed design and the unit test specification. Establish bidirectional traceability between the unit test specification and unit test results. [OUTCOME 4] NOTE 7: Bidirectional traceability supports coverage, consistency and impact analysis. [SWE.4.BP6] Ensure consistency between the software detailed design and the unit test specification. [OUTCOME 4] NOTE 8: Consistency is supported by bidirectional traceability and can be demonstrated by review records.

Content is directly from Automotive SPICE 2.5 [38] and Automotive SPICE 3.0 [34].

A.4 Mapping between Outcomes and WPs (SWE.3 Scope)

Automotive SPICE 2.5 Outcomes (REF.)			Automotive SPICE 2.5 Work Products			Automotive SPICE 3.0 Outcomes		Automotive SPICE 3.0 Work Products		
OC ID	Outcome	Outcome Full Text Description	WP #	BP	WP Name	OC ID	Outcome	WP #	WP Name	WP Full Text
	Software design									
	ENG.5 - PURPOSE	The purpose of the Software design process is to provide a design for the software that implements and can be verified against the software requirements. *NOTE 1: The software design process should take into account all software components such as customer supplied software, third party software and sub-contractor software.* *NOTE 2: Definition of software architectural design and detailed design includes development of verification criteria.*								
ENG.5.NEW	New in Autmotive SPICE 3.0					**SWE.3.OC5,**	the software detailed design and the relationship to the software architectural design is agreed and communicated to all affected parties.	**13-04, 13-25, 13-50,**	Communication record Verification results Test results	[13-04] All forms of interpersonal communication including: - letters - faxes - e-mails - voice recordings - podcast - blog - videos - forum - live chat - wikis - photo protocol - meeting support record [13-25] - Verification check-list - Passed items of Verification - Failed items of Verification - Pending items of Verification - Problems identified during Verification - Risk analysis - Recommendation of actions - Conclusions of Verification - Signature of Verification [13-50] - Level Test Log
ENG.5.OC1	ENG.5 - OUTCOME 1	a software architectural design is defined that identifies the components of the software and meets the defined software requirements.	**04-04,**	Software architectural design	[04-04] – Describes the overall software structure – Describes the operative system including task structure – Identifies inter-task/inter-process communication – Identifies the required software elements – Identifies own developed and supplied code – Identifies the relationship and dependency between software elements – Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption – Describes how variants for different model series or configurations are derived – Describes the dynamic behaviour of the software (Start-up, shutdown, software update, error handling and recovery, etc.) – identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption – Describes which data is persistent and under which conditions – Consideration is given to: – – any required software performance characteristics – – any required software interfaces – – any required security characteristics required – – any database design requirements	**SWE.2.OC1,**	a software architectural design is defined that identifies the elements of the software.	**04-04,**	Software architectural design	[04-04] - Describes the overall software structure - Describes the operative system including task structure - Identifies inter-task/inter-process communication - Identifies the required software elements - Identifies own developed and supplied code - Identifies the relationship and dependency between software elements - Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption - Describes how variants for different model series or configurations are derived - Describes the dynamic behavior of the software (Start-up, shutdown, software update, error handling and recovery, etc.) - Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption - Describes which data is persistent and under which conditions - Consideration is given to: – any required software performance characteristics – any required software interfaces – any required security characteristics required – any database design requirements
ENG.5.OC2	ENG.5 - OUTCOME 2	the software requirements are allocated to the elements of the software.	**04-04, 04-05, 13-22,**	Software architectural design Software detailed design Traceability record	[04-04] – Describes the overall software structure – Describes the operative system including task structure – Identifies inter-task/inter-process communication – Identifies the required software elements – Identifies own developed and supplied code – Identifies the relationship and dependency between software elements – Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption – Describes how variants for different model series or configurations are derived – Describes the dynamic behaviour of the software (Start-up, shutdown, software update, error handling and recovery, etc.) – identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption – Describes which data is persistent and under which conditions – Consideration is given to: – – any required software performance characteristics – – any required software interfaces – – any required security characteristics required – – any database design requirements [04-05] – Provides detailed design (could be represented as a prototype, flow chart, entity relationship diagram, pseudo code, etc.) – Provides format of input/output data – Provides specification of CPU, ROM, RAM, EEPROM and Flash needs – Describes the interrupts with their priorities	**SWE.2.OC2,**	the software requirements are allocated to the elements of the software.	**04-04,**	Software architectural design	[04-04] - Describes the overall software structure - Describes the operative system including task structure - Identifies inter-task/inter-process communication - Identifies the required software elements - Identifies own developed and supplied code - Identifies the relationship and dependency between software elements - Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption - Describes how variants for different model series or configurations are derived - Describes the dynamic behavior of the software (Start-up, shutdown, software update, error handling and recovery, etc.) - Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption - Describes which data is persistent and under which conditions - Consideration is given to: – any required software performance characteristics – any required software interfaces – any required security characteristics required – any database design requirements
ENG.5.OC3	ENG.5 - OUTCOME 3	internal and external interfaces of each software component are defined.	**04-04, 04-05,**	Software architectural design Software detailed design	[04-04] – Describes the overall software structure – Describes the operative system including task structure – Identifies inter-task/inter-process communication – Identifies the required software elements – Identifies own developed and supplied code – Identifies the relationship and dependency between software elements – Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption – Describes how variants for different model series or configurations are derived – Describes the dynamic behaviour of the software (Start-up, shutdown, software update, error handling and recovery, etc.) – identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption – Describes which data is persistent and under which conditions – Consideration is given to: – – any required software performance characteristics – – any required software interfaces – – any required security characteristics required – – any database design requirements [04-05] – Provides detailed design (could be represented as a prototype, flow chart, entity relationship diagram, pseudo code, etc.) – Provides format of input/output data – Provides specification of CPU, ROM, RAM, EEPROM and Flash needs – Describes the interrupts with their priorities	**SWE.2.OC3, SWE.3.OC2,**	the interfaces of each software element are defined. interfaces of each software unit are defined.	**04-04, 17-08, 04-05,**	Software architectural design Interface requirement specification Software detailed design	[04-04] - Describes the overall software structure - Describes the operative system including task structure - Identifies inter-task/inter-process communication - Identifies the required software elements - Identifies own developed and supplied code - Identifies the relationship and dependency between software elements - Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption - Describes how variants for different model series or configurations are derived - Describes the dynamic behavior of the software (Start-up, shutdown, software update, error handling and recovery, etc.) - Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption - Describes which data is persistent and under which conditions - Consideration is given to: – any required software performance characteristics – any required software interfaces – any required security characteristics required – any database design requirements [17-08] - Defines relationships between two products, process or process tasks - Defines criteria and format for what is common to both - Defines critical timing dependencies or sequence ordering - Description of the physical interfaces of each system component like – Bus interfaces (CAN, MOST, LIN, Flexray etc.)
ENG.5.OC4	ENG.5 - OUTCOME 4	the dynamic behaviour and resource consumption objectives of the software components are defined.	**04-04, 04-05, 13-25,**	Software architectural design Software detailed design Verification results	[04-04] – Describes the overall software structure – Describes the operative system including task structure – Identifies inter-task/inter-process communication – Identifies the required software elements – Identifies own developed and supplied code – Identifies the relationship and dependency between software elements – Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption – Describes how variants for different model series or configurations are derived – Describes the dynamic behaviour of the software (Start-up, shutdown, software update, error handling and recovery, etc.) – identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption – Describes which data is persistent and under which conditions – Consideration is given to: – – any required software performance characteristics – – any required software interfaces – – any required security characteristics required – – any database design requirements [04-05] – Provides detailed design (could be represented as a prototype, flow chart, entity relationship diagram, pseudo code, etc.) – Provides format of input/output data – Provides specification of CPU, ROM, RAM, EEPROM and Flash needs – Describes the interrupts with their priorities	**SWE.2.OC4, SWE.3.OC3,**	the dynamic behavior and resource consumption objectives of the software elements are defined. the dynamic behavior of the software units is defined.	**04-04, 04-05, 13-25, 13-50,**	Software architectural design Software detailed design Verification results Test results	[04-04] - Describes the overall software structure - Describes the operative system including task structure - Identifies inter-task/inter-process communication - Identifies the required software elements - Identifies own developed and supplied code - Identifies the relationship and dependency between software elements - Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption - Describes how variants for different model series or configurations are derived - Describes the dynamic behavior of the software (Start-up, shutdown, software update, error handling and recovery, etc.) - Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption - Describes which data is persistent and under which conditions - Consideration is given to: – any required software performance characteristics – any required software interfaces – any required security characteristics required – any database design requirements [04-05] - Provides detailed design (could be represented as a prototype, flow chart, entity relationship diagram, pseudo code, etc.) - Provides format of input/output data - Provides specification of CPU, ROM, RAM, EEPROM and Flash needs

ENG.5.OC5	ENG.5 - OUTCOME 5	a detailed design is developed that describes software units that can be implemented and tested.	**04-05, 13-25, 17-50,**	Software detailed design Verification results Verification criteria	[04-05] – Provides detailed design (could be represented as a prototype, flow chart, entity relationship diagram, pseudo code, etc.) – Provides format of input/output data – Provides specification of CPU, ROM, RAM, EEPROM and Flash needs – Describes the interrupts with their priorities – Describes the tasks with cycle time and priority – Establishes required data naming conventions – Defines the format of required data structures – Defines the data fields and purpose of each required data element – Provides the specifications of the program structure [13-25] – Verification check-list – Passed items of verification – Failed items of verification – Pending items of verification – Problems identified during verification – Risk analysis – Recommendation of actions – Conclusions of verification – Signature of verification [17-50] -Each requirement is verifiable or can be assessed - Verification criteria define the qualitative and quantitative criteria for verification of a requirement. - Verification criteria demonstrate that a requirement can be verified within agreed	**SWE.3.OC1,**	a detailed design is developed that describes software units.	**04-05,**	Software detailed design	[04-05] - Provides detailed design (could be represented as a prototype, flow chart, entity relationship diagram, pseudo code, etc.) - Provides format of input/output data - Provides specification of CPU, ROM, RAM, EEPROM and Flash needs - Describes the interrupts with their priorities - Describes the tasks with cycle time and priority - Establishes required data naming conventions - Defines the format of required data structures - Defines the data fields and purpose of each required data element - Provides the specifications of the program structure
ENG.5.OC6	ENG.5 - OUTCOME 6	consistency and bilateral traceability are established between software requirements and software architectural design.	**04-04, 04-05, 13-22,**	Software architectural design Software detailed design Traceability record	[04-04] – Describes the overall software structure – Describes the operative system including task structure – Identifies inter-task/inter-process communication – Identifies the required software elements – Identifies own developed and supplied code – Identifies the relationship and dependency between software elements – Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption – Describes how variants for different model series or configurations are derived – Describes the dynamic behaviour of the software (Start-up, shutdown, software update, error handling and recovery, etc.) – Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption – Describes which data is persistent and under which conditions – Consideration is given to: – – any required software performance characteristics – – any required software interfaces – – any required security characteristics required – – any database design requirements [04-05] – Provides detailed design (could be represented as a prototype, flow chart, entity relationship diagram, pseudo code, etc.) – Provides format of input/output data – Provides specification of CPU, ROM, RAM, EEPROM and Flash needs – Describes the interrupts with their priorities	**SWE.2.OC5,**	consistency and bidirectional traceability are established between software requirements and software architectural design.	**04-04, 13-19, 13-22,**	Software architectural design Review record Traceability record	[04-04] - Describes the overall software structure - Describes the operative system including task structure - Identifies inter-task/inter-process communication - Identifies the required software elements - Identifies own developed and supplied code - Identifies the relationship and dependency between software elements - Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption - Describes how variants for different model series or configurations are derived - Describes the dynamic behavior of the software (Start-up, shutdown, software update, error handling and recovery, etc.) - Identifies where the data (such as parameters) are stored and which measures (e.g. checksums, redundancy) are taken to prevent data corruption - Describes which data is persistent and under which conditions - Consideration is given to: – any required software performance characteristics – any required software interfaces – any required security characteristics required – any database design requirements [13-19] - Provides the context information about the review: – what was reviewed – lists reviewers who attended – status of the review - Provides information about the coverage of the review:
ENG.5.OC7	ENG.5 - OUTCOME 7	consistency and bilateral traceability are established between software architectural design and software detailed design.	**13-22,**	Traceability record	[13-22] – All requirements (customer and internal) are to be traced – Identifies a mapping of requirement to life cycle work products – Provides the linkage of requirements to work product decomposition (i.e., requirement -> design -> code -> test -> deliverables, etc.) – Provides forward and backwards mapping of requirements to associated work products throughout all phases of the life cycle – NOTE: this may be included as a function ofanother defined work product (example: A CASE tool for design decomposition may have a mapping ability as part of its features)	**SWE.3.OC4,**	consistency and bidirectional traceability are established between software requirements and software units; and consistency and bidirectional traceability are established between software architectural design and software detailed design; and consistency and bidirectional traceability are established between software detailed design and software units.	**13-19, 13-22,**	Review record Traceability record	[13-19] - Provides the context information about the review: – what was reviewed – lists reviewers who attended – status of the review - Provides information about the coverage of the review: – check-lists – review criteria – requirements – compliance to standards - Records information about: – the readiness for the review – preparation time spent for the review – time spent in the review – reviewers, roles and expertise - Review findings: – non-conformances – improvement suggestions - Identifies the required corrective actions: – risk identification – prioritized list of deviations and problems discovered – the actions, tasks to be performed to fix the problem – ownership for corrective action – status and target closure dates for identified problems [13-22] - All requirements (customer and internal) are to be traced
	Software construction								
	ENG.6 - PURPOSE	The purpose of the Software construction process is to produce verified software units that properly reflect the software design. *NOTE 1: Analysis of software units will include prioritization and categorization of software units.* *NOTE 2: Unit verification will include unit testing and may include static analysis, code inspection/reviews, checks against coding standards and guidelines.*								
ENG.6.NEW	New in Autmotive SPICE 3.0					**SWE.4.OC5,**	results of the unit verification are summarized and communicated to all affected parties.	**13-04,**	Communication record	[13-04] All forms of interpersonal communication including: - letters - faxes - e-mails - voice recordings - podcast - blog - videos - forum - live chat - wikis - photo protocol - meeting support record
ENG.6.OC1	ENG.6 - OUTCOME 1	a unit verification strategy is developed for software units consistent with the software design.	**08-52, 08-50,**	Test plan Test Specification	[08-52] - Level Test Plan (according to IEEE definition) - Test strategy (black-box and/or white-boxtesting, boundary class test determination, regression testing strategy, etc.) Additionally where necessary: - Master Test Plan (according to IEEE definition) [08-50] - Level Test Design (according to IEEE definition) - Level Test Case (according to IEEE definition) - Level Test Procedure (according to IEEE definition) - Identification of test cases for regression testing Additionally for system integration: – Identification of required system elements (hardware elements, wiring elements, parameter settings, data bases, etc.) – Necessary sequence or ordering identified for integrating the system elements	**SWE.4.OC1,**	a software unit verification strategy including regression strategy is developed to verify the software units.	**08-52,**	Test plan	[08-52] - Test Plan according to ISO29119-3 - Context – Project/Test sub-process – Test item(s) – Test scope – Assumptions and constraints – Stakeholder – Testing communication – Test strategy – Identifies what needs there are to be satisfied – Establishes the options and approach for satisfying the needs (black-box and/or white-box-testing, boundary class test determination, regression testing strategy, etc.) – Establishes the evaluation criteria against which the strategic options are evaluated – Identifies any constraints/risks and how these will be addressed – Test design techniques – Test completion criteria – Test ending criteria – Test start, abort and re-start criteria – Metrics to be collected – Test data requirements – Retesting and regression testing – Suspension and resumption criteria – Deviations from the Organizational Test Strategy - Test data requirements - Test environment requirements
ENG.6.OC2	ENG.6 - OUTCOME 2	software units defined by the software design are analyzed for correctness and testability.				**SWE.4.OC2,**	criteria for software unit verification are developed according to the software unit verification strategy that are suitable to provide evidence for compliance of the software units with the software detailed design and with the non-functional software requirements.	**08-50,**	Test specification	[08-50] - Test Design Specification - Test Case Specification - Test Procedure Specification - Identification of test cases for regression testing - Additionally for system integration: – Identification of required system elements (hardware elements, wiring elements, parameter settings, data bases, etc.) – Necessary sequence or ordering identified for integrating the system elements

ENG.6.OC3	ENG.6 - OUTCOME 3	software units defined by the software design are produced.	**17-50, 11-05,**	Verification criteria Software unit	[17-50] -Each requirement is verifiable or can be assessed - Verification criteria define the qualitative and quantitative criteria for verification of a requirement. - Verification criteria demonstrate that a requirement can be verified within agreed constraints. (Additional Requirement to 17-00 Requirements specification) [11-05] – Follows established coding standards (as appropriate to the language and application): – – commented – – structured or optimized – – meaningful naming conventions – – parameter information identified – – error codes defined – – error messages descriptive and meaningful – – formatting - indented, levels – Follows data definition standards (as appropriate to the language and application): – – variables defined – – data types defined – – classes and inheritance structures defined – – objects defined – Entity relationships defined – Database layouts are defined – File structures and blocking are defined – Data structures are defined – Algorithms are defined	**SWE.3.OC6,**	software units defined by the software detailed design are produced.	**11-05,**	Software unit	[11-05] - Follows established coding standards (as appropriate to the language and application): – commented – structured or optimized – meaningful naming conventions – parameter information identified – error codes defined – error messages descriptive and meaningful – formatting - indented, levels - Follows data definition standards (as appropriate to the language and application): – variables defined – data types defined – classes and inheritance structures defined – objects defined - Entity relationships defined - Database layouts are defined - File structures and blocking are defined - Data structures are defined - Algorithms are defined - Functional interfaces defined
ENG.6.OC4	ENG.6 - OUTCOME 4	software units are verified according to the unit verification strategy.	**13-25, 08-50, 13-50,**	Verification results Test Specification Test Result	[13-25] – Verification check-list – Passed items of verification – Failed items of verification – Pending items of verification – Problems identified during verification – Risk analysis – Recommendation of actions – Conclusions of verification – Signature of verification [08-50] - Level Test Design (according to IEEE definition) - Level Test Case (according to IEEE definition) - Level Test Procedure (according to IEEE definition) - Identification of test cases for regression testing Additionally for system integration: – Identification of required system elements (hardware elements, wiring elements, parameter settings, data bases, etc.) – Necessary sequence or ordering identified for integrating the system elements [13-50] - Level Test Log (according to IEEE definition) - Anomaly Report (according to IEEE definition) - Level Test Report (according to IEEE definition) Additionally where necessary: - Level Interim Test Status Report (according to IEEE definition) - Master Test Report (according to IEEE definition)	**SWE.4.OC3,**	software units are verified according to the software unit verification strategy and the defined criteria for software unit verification and the results are recorded.	**13-19, 15-01,**	Review record Analysis report	[13-19] - Provides the context information about the review: – what was reviewed – lists reviewers who attended – status of the review - Provides information about the coverage of the review: – check-lists – review criteria – requirements – compliance to standards - Records information about: – the readiness for the review – preparation time spent for the review – time spent in the review – reviewers, roles and expertise - Review findings: – non-conformances – improvement suggestions - Identifies the required corrective actions: – risk identification – prioritized list of deviations and problems discovered – the actions, tasks to be performed to fix the problem – ownership for corrective action – status and target closure dates for identified problems [15-01] - What was analyzed?
ENG.6.OC5	ENG.6 - OUTCOME 5	results of unit verification are recorded.	**13-25, 13-50,**	Verification results Test Result	[13-25] – Verification check-list – Passed items of verification – Failed items of verification – Pending items of verification – Problems identified during verification – Risk analysis – Recommendation of actions – Conclusions of verification – Signature of verification [13-50] - Level Test Log (according to IEEE definition) - Anomaly Report (according to IEEE definition) - Level Test Report (according to IEEE definition) Additionally where necessary: - Level Interim Test Status Report (according to IEEE definition) - Master Test Report (according to IEEE definition)	**SWE.4.OC3,**	software units are verified according to the software unit verification strategy and the defined criteria for software unit verification and the results are recorded.	**13-19, 15-01,**	Review record Analysis report	[13-19] - Provides the context information about the review: – what was reviewed – lists reviewers who attended – status of the review - Provides information about the coverage of the review: – check-lists – review criteria – requirements – compliance to standards - Records information about: – the readiness for the review – preparation time spent for the review – time spent in the review – reviewers, roles and expertise - Review findings: – non-conformances – improvement suggestions - Identifies the required corrective actions: – risk identification – prioritized list of deviations and problems discovered – the actions, tasks to be performed to fix the problem – ownership for corrective action – status and target closure dates for identified problems [15-01] - What was analyzed?
ENG.6.OC6	ENG.6 - OUTCOME 6	consistency and bilateral traceability are established between software detailed design and software units.	**13-22,**	Traceability record	[13-22] – All requirements (customer and internal) are to be traced – Identifies a mapping of requirement to life cycle work products – Provides the linkage of requirements to work product decomposition (i.e., requirement -> design -> code -> test -> deliverables, etc.) – Provides forward and backwards mapping of requirements to associated work products throughout all phases of the life cycle – NOTE: this may be included as a function ofanother defined work product (example: A CASE tool for design decomposition may have a mapping ability as part of its features)	**SWE.3.OC4, SWE.4.OC4,**	consistency and bidirectional traceability are established between software requirements and software units; and consistency and bidirectional traceability are established between software architectural design and software detailed design; and consistency and bidirectional traceability are established between software detailed design and software units. consistency and bidirectional traceability are established between software units, criteria for verification and verification results.	**13-19, 13-22, 13-19, 13-22,**	Review record Traceability record Review record Traceability record	[13-19] - Provides the context information about the review: – what was reviewed – lists reviewers who attended – status of the review - Provides information about the coverage of the review: – check-lists – review criteria – requirements – compliance to standards - Records information about: – the readiness for the review – preparation time spent for the review – time spent in the review – reviewers, roles and expertise - Review findings: – non-conformances – improvement suggestions - Identifies the required corrective actions: – risk identification – prioritized list of deviations and problems discovered – the actions, tasks to be performed to fix the problem – ownership for corrective action – status and target closure dates for identified problems [13-22] - All requirements (customer and internal) are to be traced

Content is directly from Automotive SPICE 2.5 [38] and Automotive SPICE 3.0 [34].