VOLKSWAGEN

AKTIENGESELLSCHAFT

CONFIDENTIAL
VERTRAULICH

Group Basic Software Requirements

Basic demands set by the Volkswagen Group on vehicle software and on software related to the vehicle/software determinated systems and its development processes.

Development, General Project-Independent Performance Specification: LAH.893.909

Author	Volkswagen AG, Audi AG, Porsche AG
Dept./OU	K-PQX-S (Volkswagen), I/GQ-F1 (Audi), EES3 (Porsche)
Phone	-
Cell phone	-
Fax	-
E-mail	software.qualitaet.vwag.r.wob@volkswagen.de

First issue	06.09.2002
Date of revision	25.11.2019
Version	3.3
Baseline	3.26

Freigabe

Wegner, Berend (K-GQX-S)	12.11.2019	
	Datum	Unterschrift

Contents

1 Preamble

1.1 Purpose

[I: KGAS_3041]

According to ISO/IEC 25010:2011 Chapter 4.3.13 "Software Quality" is the degree to which a software product satisfies stated and implied needs when used under specified conditions.

[I: KGAS_3046]

The purpose of the Group Basic Software Requirements (KGAS, ”Konzern Grundanforderungen Software“) as described in this document is to define and specify the requirements of the Volkswagen Group for software quality in vehicles. These requirements shall apply to vehicle integrated and vehicle related software (KGAS_25/KGAS_3666), and among others, they specify requirements for development processes and work products. These are minimum requirements which are applicable for all types of software (e.g. application, drivers, base software or the like).

[I: KGAS_1979]

To ensure a state of the art software development, among others, the KGAS request to use development processes which achieve at least a capability level "level 2" in an Automotive SPICE® Assessment (KGAS_2074). The KGAS requirements do both complement and specify the requirements from Automotive SPICE® but they do not replace them. Partial redundancies may occur especially for important quality aspects.

[I: KGAS_3090]

The methods, basis of the evaluation and consequences of the quality assurance activities of the Volkswagen Group at the suppliers are described in the "Formel Q Capability Software" (KGAS_2834).

[I: KGAS_3633]

The terms "customer" and "supplier" used in this document are equivalent to the terms "customer" and "supplier" used in the Formel Q Capability Software (KGAS_2834).

1.2 Document Owners

[I: KGAS_2085]

Volkswagen AG
Group Quality
Quality Software
Letterbox 1477/0
38436 Wolfsburg
Germany

Contact:
E-mail: software.qualitaet.vwag.r.wob@volkswagen.de

Audi AG
Quality Assurance Vehicle Software
85045 Ingolstadt
Germany

Contact:
E-Mail: software-quality.gq@audi.de

Porsche AG
Electric/Electronics Software
Porschestr. 911
71287 Weissach
Germany

Contact:
E-Mail: software-qualitaet@porsche.de

1.3 Security Incident Management

[A: KGAS_3890]

To communicate security incidents use the incident team of the brand or region which is responsible for the development of the affected product in the Volkswagen group. In case of unclear responsibility contact the incident team of Volkswagen Passenger Cars (KGAS_3876).

[I: KGAS_3876]

Volkswagen Passenger Cars and Volkswagen Commercial Vehicles (light commercial vehicles)

Contact:
Email: CSI-wob@volkswagen.de

[I: KGAS_3891]

Porsche AG

Contact:
Email: csi@porsche.de

[I: KGAS_3892]

MAN SE

Contact:
Email: carsecurity@man.eu

[I: KGAS_3926]

Audi AG

Contact:
E-Mail: vulnerability@audi.de

[I: KGAS_3958]

Contact for products in development: The contact person is the respective project manager of the respective technical development.

[A: KGAS_3959]

Region China

Contact:
E-Mail: csi-cn@volkswagen.com.cn

2 Scope

[A: KGAS_25]

The requirements in this document are valid for software deployable in the vehicle and other software related to the vehicle and its development processes.

[A: KGAS_3666]

Software related to the vehicle includes all software which has an impact on the vehicle and its functionalities.

[A: KGAS_3028]

The requirements in this document are valid for the whole Volkswagen Group.

[A: KGAS_3551]

The requirements in this document are to be recognized as contractual obligations for the supplier and must therefore be performed and fulfilled by the supplier.

[A: KGAS_3546]

If a requirement of the KGAS is inconsistent with a requirement from another applicable document, the supplier must initiate a specific agreement between the customer and the supplier.

[A: KGAS_3899]

An identification of the contradictions in KGAS_3546 is among other things part of the analysis of the requirements. See also KGAS_3266.

[A: KGAS_3610]

If the software is classified safety-relevant by the customer, the software and its development processes are subjects to the Functional Safety Development Guideline (EFS "Entwicklungsrichtlinie Funktionale Sicherheit") (KGAS_3611).

3 Rights of the Customer and Obligations of the Supplier

[A: KGAS_3885]

All development artefacts must be either in English or German language.

[A: KGAS_1806]

On request of the customer, the supplier must provide evidence of the compliance with the KGAS.

[A: KGAS_27]

The supplier must obligate all its sub-suppliers to fulfil the KGAS and muste ensure its execution.

[A: KGAS_2933]

If the supplier or its sub-suppliers cannot fulfil the KGAS completely, the supplier must seek written approval of the deviations from the Volkswagen Group Quality Assurance before the start of the project (see KGAS_2085 for contact).

[I: KGAS_2932]

Unapproved deviations from the KGAS may result in a downgrade of the quality capability rating of the supplier (also see KGAS_2834 "Formula Q Capability Software").

[A: KGAS_3107]

If deviation from KGAS is identified, the supplier must promptly set up an improvement program.

[A: KGAS_42]

The improvement measures of the improvement program (KGAS_3107) must be carried out with defined scope and date, which must be agreed with the customer.

[A: KGAS_2184]

The supplier must assure that the customer is given the possibility to access the system and software development processes and all corresponding work products including the software product.

[I: KGAS_3614]

The access (KGAS_2184) shall confirm to confidentiality obligations of the supplier to third parties in relation to requirements and content of such third parties.

[A: KGAS_1877]

The customer can exercise its insight rights from KGAS (KGAS_2184) entirely or partly through third parties.

[I: KGAS_2948]

The customer will name the third party (KGAS_1877) to the supplier before contracting the third party. Third parties in this case are obligated to uphold confidentiality with regards to content, and must not be direct competitors to the supplier (§15 AktG) weather itself nor its affiliated companies. If the supplier can present objectively verifiable and factual reasons, that the contracting of the named third party is unacceptable for the supplier, the customer will investigate the objections of the supplier; and at his own discretion with consideration of the objections from the supplier, the customer will contract another third party if applicable.

[A: KGAS_3612]

The supplier have to gather metrics befor the strat of the project. The supplier have to gather metrics for each release but at least every four weeks. The minimal extent of metrics is given by the supplier. (see information KGAS 3660)

[A: KGAS_3623]

The extent of the project metrics, the duration of the data collection and the report cycles may be defined by the customer in difference to KGAS_3612 during the begin of the project.

[I: KGAS_3660]

A list of project metrics (KGAS_3612) is available on http://www.vwgroupsupply.com/, in the same folder as the KGAS.

[A: KGAS_3915]

The supplier have to provide the gathered metrics to the customer if he requests it. The exchange format is defined by the customer.

[I: KGAS_3916]

The gathered metrics (KGAS_3915) may be used for approvals by the customer (e.g. build sample approval, sample inspection).

[A: KGAS_3917]

Depending on the typ and the size of the project single subunits (like e.g. function blocks, subcomponents) have to be reported separately beside the overall project if the customer requests it.

[I: KGAS_3628]

The goals of the project metrics (KGAS_3612) contain amongst others a better coordination between the customer and the supplier, a better tracing of the measures of the improvement program and self-control regarding process quality and work product quantity.

[A: KGAS_3624]

Process deviations, identified through the application of the project metrics (KGAS_3612), must be addressed with improvement measures.

4 Terminology

[I: KGAS_1984]

This chapter defines how relevant technical terms in the KGAS are to be interpreted.

4.1 Feature

[I: KGAS_3665]

A feature is a scope of functionalities which is defined by the customer and which is represented by a subset of the customer requirements.

4.2 System

[I: KGAS_2877]

The system is the whole part to be delivered by the supplier.

[I: KGAS_2879]

The system consists of system elements.

4.3 System Element

[I: KGAS_3604]

A System element is a logical part of a system.

[I: KGAS_3639]

System elements are described in the system architecture specification.

[I: KGAS_3606]

Typical system elements are software, hardware (e.g. sensors, actuators, printed circuit boards, parts and connectors) and mechanics (e.g. case).

4.4 Software

[I: KGAS_2876]

Software is the whole software enclosed in the deliverable.

[I: KGAS_3523]

Software consists of one or more software elements.

[I: KGAS_2878]

Typical software parts are application, driver, hardware abstractions, operating system, and implemented algorithms.

[I: KGAS_2880]

Software also includes platform elements, third party software and programmable integrated circuits. A software element is a part of the software with a dedicated specification.

4.5 Software Element

[I: KGAS_3095]

A software element is a part of the software with a dedicated specification.

[I: KGAS_3580]

A software element can consist of more software elements.

[I: KGAS_3602]

A software element encapsulates logically associated functionality and has defined interfaces.

[I: KGAS_3640]

Software elements are described in the software architecture specification.

[I: KGAS_3103]

A software element is, for example, a library (C, C++, Java) or a function on the application level (e.g. ABS-Controller, Diagnostics).

4.6 Component

[I: KGAS_3651]

A component is a software element on the lowest hierarchy level.

[I: KGAS_2991]

A component consists of units, encapsulates logically associated functionality and owns defined interfaces.

[I: KGAS_3641]

Components are represented in the software architecture specification.

[I: KGAS_3642]

Components are described in the software detailed design.

[I: KGAS_3102]

A component, for example, contains one ore multiple source code files in C, or one or multiple classes including inherited properties in object oriented programming languages (e.g. C++, Java).

4.7 Unit

[I: KGAS_2998]

A unit is the smallest separately executable and testable entity of a component.

[I: KGAS_2989]

A unit encapsulates data and statements on the level of a function (C), method (C++, Java) or procedure (Pascal).

[I: KGAS_3643]

Units are described in the software detailed design.

4.8 Unit Element

[I: KGAS_3646]

A unit element is a part of a unit (e.g. calculations, interfaces, function calls or macros).

[I: KGAS_3647]

Unit elements are described in the software detailed design.

4.9 Model Element

[I: KGAS_3527]

A model element is the logical representation of one or more basic objects within a tool for model based source code generation.

[I: KGAS_3528]

A basic object is an atomic object in a tool for model based source code generation, which cannot be divided into sub-objects.

4.10 Other definitions

[I: KGAS_3920]

Own software is software which is developed by or for the customer.

[I: KGAS_3919]

Third-party software is software which isn´t own software.

[I: KGAS_3921]

Auxiliary software is software which is provided by the supplier to the customer within the fulfilment of the contract. This might be third-party or own software.

[I: KGAS_3820]

Free and open source software (FOSS) is any software that is distributed under the terms of use and license terms for free and open source software and subject to sharing or disclosure of the source code of the software under such substantial obligations.

[I: KGAS_3826]

Closed source software is software that is not distributed under license terms for FOSS and whose source code is not freely accessible.

[I: KGAS_3952]

Supplier software is third-party software that has been developed for the supplier or by himself, but not against payment on behalf of the customer.

[I: KGAS_3953]

Supplied Software is the software supplied by the supplier (including supplier software, assistive software, third-party software).

[I: KGAS_3954]

Software of third party is third party software that is not supplier software.

[I: KGAS_3956]

Dead code is in the delivery included code, which can not be executed by the specified program execution (including error handling).

5 System and Software Development

[I: KGAS_3124]

This chapter contains requirements for the organisation, development processes, work products and infrastructure of the supplier.

5.1 Overall Process Requirements

[A: KGAS_2074]

The entire product included in the deliverable must be developed with processes, which achieve at least a capability level "level 2" in an Automotive SPICE® Assessment with VDA Scope according to VDA Automotive SPICE® Guidelines (see KGAS_3813).

[A: KGAS_3792]

The Automotive SPICE® Assessments designated to fulfill the requirement KGAS_2074 must take into account the VDA Automotive SPICE® Guidelines (KGAS_3813).

[A: KGAS_3669]

The requirement KGAS_2074 must also be fulfilled for software already developed. These include, for example, software that was already developed or purchased before nomination.

[A: KGAS_2986]

The bidirectional traceability which is demanded by Automotive SPICE® must be established from the customer requirements through the unit level to the tests.

[A: KGAS_3679]

The supplier must be able to adjust all software-related specifications and the source code, to create the corresponding binary files, and to perform all test steps for at least 15 years after the end of production (EOP).

[A: KGAS_3407]

Each element of a specification must be traceable to its source.

[A: KGAS_2035]

All work products (see KGAS_3483) must be consistent to each other regarding their content at the moment of a customer release.

[A: KGAS_3483]

At least the following work product types muss be available in the project:

- Customer requirements specifications
- System requirements specifications
- System architecture specifications
- Software requirements specifications
- Software architecture specifications
- Software detailed design
- Source code
- Unit test artefacts (specifications, implementation, results)
- Software integration test artefacts (specifications, implementation, results)
- Software test artefacts (specifications, implementation, results)
- System integration test artefacts (specifications, implementation, results)
- System test artefacts (specifications, implementation, results)
- List of baselines
- Strategy documents
- Test plans
- Work breakdown structure

- Schedule
- Work package descriptions
- Entries of problem resolution management
- Entries of the change management
- Artefacts of the quality assurance (e.g. review protocols)
- All work products of the Development Interface Agreements (DIA) according to the Functional Safety Development Guideline (EFS) (KGAS_3611)

[A: KGAS_3552]

The refinement of specification elements (e.g. requirements, architecture elements) from one abstraction or hierarchy level to the abstraction level below must not exceed the ratio 1 to 10. If this ratio is to be exceeded in some isolated cases, the violation must be justified verifiably.

[A: KGAS_3498]

Each change of the development processes and development sites of the supplier, which may have an impact on the compliance with the KGAS, must be notified promptly to those named in KGAS_2085.

[A: KGAS_3794]

System requirements specifying functionalities which do not have a demonstrable relationship with customer requirements must be approved by the customer.

[A: KGAS_3795]

Each software unit contained in the product must provide a demonstrable contribution to the fulfillment of at least one system requirement.

[I: KGAS_3797]

The goal of the requirements KGAS_3794 and KGAS_3795 is to avoid non-commissioned potentially defective software in the product (defective implies, for example, illegal, unsafe, unethical, harmful to the environment, image damaging, privacy violating).

[A: KGAS_3200]

For all documents the following information must be evident:

- Name
- Version
- Purpose
- Validity
- Scope
- Work state (e.g. "new", "in work", "closed")
- Last and current editor
- Any changes (incl. date)
- Auditor
- Review findings or open points (e.g. regarding release)
- Applicable and referenced documents
- Template version und template name
- Confidentiality classification

5.2 Project Management

[A: KGAS_3169]

A project plan and an effort estimation must be available until the start of development.

[A: KGAS_3595]

For all estimations assumptions must be documented.

[A: KGAS_3163]

A work breakdown structure must be available timely before the start of development of a release.

[A: KGAS_3592]

The work breakdown structure must contain a structured list and description of the work packages and activities required for the project.

[A: KGAS_3593]

All work packages which contribute to the development of feature must contain at least the activities specification, implementation, and verification, validation or review.

[A: KGAS_3144]

Effort estimation must be available for each work package and contained activities.

[A: KGAS_3146]

For each work package respective dependencies to other work packages must be evident.

[A: KGAS_3148]

For each work package it must be evident which work products are created or changed by this work package.

[A: KGAS_3167]

For changes and problem resolutions an adequate effort flat-rate must be scheduled.

[A: KGAS_3153]

A feature release plan must be created which contains a breakdown of the features mapped to the customer milestones.

[A: KGAS_3594]

If a feature is implemented over multiple releases, the feature must be further refined in the feature release plan so that an exact and testable scope per release can be implemented.

[A: KGAS_3157]

The features contained in the feature release plan must be assigned to the requirements of the system and software requirements specifications.

[A: KGAS_3158]

All interfaces and responsibilities in the project must be documented (e.g. RACI-matrix, VMI-matrix).

[A: KGAS_3677]

A detailed schedule must be available at least for the next customer release.

[A: KGAS_3177]

The schedule must contain all activities which are derived from the entries of the problem resolution management and change management.

[A: KGAS_3171]

The critical path of the schedule must be systematically identifiable.

[A: KGAS_52]

Skills, experiences and the knowledge which are essential for the project must be identified. It must be ensured that every participating employee has the identified skills, experiences and knowledge or will be trained in time.
Explicit definitions of degrees of fulfillment of work packages and activities must exist and be applied.

[A: KGAS_3170]

The project planning must be regularly updated (at least each 2 weeks) and it must consider both comparisons of actual performance against target goals and updates of degrees of fulfillment.

[A: KGAS_3191]

Project risks must be verifiably identified, evaluated and provided with rectifying measures.
The release level (development stand without use on public road, development stand with use on public road, serial stand) of the delivery must be documented.

5.3 Documentation of Deliverable

[A: KGAS_3214]

The release level (development stand without use on public road, development stand with use on public road, serial stand) of the delivery must be documented.

[A: KGAS_3215]

The implemented changes of the delivery must be documented including the description of possible failure corrections.

[I: KGAS_3938]

The documentation also includes release notes and feature overviews of the whole scope (e.g. modules) of any sub-supplier.

[A: KGAS_3216]

The tests executed for the delivery and their test results must be documented.

[A: KGAS_3218]

The configuration status of the delivery (version of the software, and if applicable electrical and mechanical hardware etc.) must be documented (including version, baseline, state of the underlying requirements, a valid architecture, etc.).

[A: KGAS_3219]

Each hardware version compatible with the software version of the delivery must be documented.

[A: KGAS_3220]

Each software version compatible with the hardware version of the delivery must be documented.

[A: KGAS_3221]

All constraints of the delivery for the commissioning and operation must be documented.

[A: KGAS_3222]

Known failures of the system and their impacts must be documented.

[A: KGAS_3223]

The version of the software used as development basis of the delivered software must be documented.

[A: KGAS_3888]

Build environment, build configuration, definitions, compiler options and compiler optimizations must be documented incl. change history.

5.4 System and Software Requirements Specification

[I: KGAS_3560]

All requirements specified by the customer are to be recognized as customer requirement specification.

[A: KGAS_3406]

All assumptions must be specified as requirements and agreed with the customer.

[A: KGAS_3548]

Own requirements of the supplier (e.g. requirements for production, requirements from platform parts, etc.) must be documented in the system and software requirement specifications.

[A: KGAS_3266]

All requirements must be verifiably created and analyzed considering at least the following aspects:

- Feasibility
- Verifiability
- Self-consistency
- Understandability

- Unambiguousness
- Atomicity

[A: KGAS_3258]

All requirements must be unambiguously identifiable.

[A: KGAS_3535]

All requirements must be assigned to releases or features.

[A: KGAS_3259]

All requirements must be retraceable to their sources.

[A: KGAS_3260]

All requirements must be traceable to the work products derived from them.

[A: KGAS_3256]

For all requirements the verification criteria including a textual description must be documented.

[A: KGAS_3600]

All requirements must be evaluated regarding their impact on the system and software architecture.

[A: KGAS_3558]

All requirements must be prioritized and categorized.

[A: KGAS_3257]

At least the following categories or types have to be assigned:

- safety relevance
- legal relevance
- security relevance
- functional/ non-functional

[A: KGAS_3263]

For each functional requirement next to the positive cases, the negative cases must also be specified in case they are technically existing (e.g. error cases, alternative paths, border cases and worst case scenarios).

[A: KGAS_3262]

It is not permitted to combine requirements from a higher level to lower level requirements if information loss would arise.

[A: KGAS_3264]

Each specification of system and software requirements and software as well as system and software elements and components must follow a defined schema (e.g. [condition] [the system or software or component] [shall or must do] [action or procedure or interface requirement]).

[A: KGAS_3267]

Each specification of system and software requirements and software as well as system and software elements and components must follow a defined schema (e.g. [condition] [the system or software or component] [shall or must do] [action or procedure or interface requirement]).

5.5 System and Software Architecture Specification

[A: KGAS_3272]

All system elements of the system architecture specification must be traceable to the related system requirements.

[A: KGAS_3537]

All software elements of the software architecture specification must be traceable to the related software requirements.

[A: KGAS_3275]

Syntax and semantics of descriptions of system and software elements within system and software architecture specifications must be defined.

[A: KGAS_3278]

All system and software elements must include a textual description containing at least its goal/purpose.

[A: KGAS_3583]

The interfaces of all system elements, software elements, components and units must be described.

[A: KGAS_3276]

Static behavior of all interfaces must be described, at least (if particularly applicable): type, value range, format, structural description, physical and logical data description, resolution, error values, offsets, initial values, logical dependencies.

[A: KGAS_3274]

Dynamic behavior of all interfaces must be described (e.g. execution orders, timing behavior, dependencies, etc.).

[A: KGAS_3661]

For non-deterministic systems, safety mechanisms must be defined for competitive processes.

[I: KGAS_3662]

The requirement KGAS_3661 aims at avoiding unexpected or undefined system states (e.g. due to race conditions).

[A: KGAS_3279]

Shared resources (e.g. global variables) must be considered as interfaces and so they must be fully described regarding read and/or write access.

[A: KGAS_3943]

Error detection mechanisms must be specified on all levels within as well as outside of the vehicle borders (e.g. backend, cloud, app). Example: Communication level, hardware level and software level.

[A: KGAS_3944]

Error handling mechanisms must be specified on all levels within as well as outside of the vehicle borders (e.g. backend, cloud, app). Example: Communication level, hardware level and software level.

[A: KGAS_3294]

Design decisions in the system and software architecture specification must be textually documented and associated to related system and software elements.

[A: KGAS_3281]

Verification criteria for system and software architecture must be textually described.

[A: KGAS_3282]

For all software elements the resource consumptions requirements must be specified. These requirements must at least include maximum CPU time consumption, maximum volatile memory consumption, maximum non-volatile memory consumption.

[A: KGAS_3270]

The system architecture specification must at least be verified by the system integration tester.

[A: KGAS_3536]

The software architecture specification must at least be verified by the software integration tester.

5.6 Software Detailed Design

[A: KGAS_3285]

The software detailed design must include a textual description with goal, purpose and internal structure for each component and each contained unit.

[A: KGAS_3286]

Interfaces and data flow must be described in the software detailed design with in and output values.

[A: KGAS_3288]

If graphical descriptions are used in the software detailed design, syntax and semantics of those descriptions must be defined.

[A: KGAS_3289]

All units and unit elements which are implemented must be described in the software detailed design.

[A: KGAS_3879]

The implementation of any unit must be possible without further assumptions.

[A: KGAS_3901]

The software detailed design have to describe at least the algorithms, calculations, interfaces, function calls, and macros.

[A: KGAS_3291]

Verification criteria must be textually described for all components and units.

[A: KGAS_3298]

The software detailed design must describe the interfaces of all units (e.g. parameters, global and component-global variables, function calls, methods, procedures, etc.).

[A: KGAS_3299]

The software detailed design must include mutual dependencies of all units and dependencies to libraries.

[A: KGAS_3301]

The software detailed design must contain the dynamic behaviour between the units, including timing and scheduling and execution orders.

[A: KGAS_3302]

The software detailed design must include the component-global variables with initial values, value ranges, error values and physical mapping.

[A: KGAS_3455]

The software detailed design must also be created in case of graphical respectively model-based programming.

[A: KGAS_3682]

For each interface a validation check against the interface description (KGAS_3276) must be specified.

[A: KGAS_3683]

In the case of negative validity tests of interfaces, a defined system and software behavior must be specified.

5.7 Software Construction

5.7.1 Programming Languages

[A: KGAS_2050]

The programming language of the software product must be an international standardized (e.g. ISO/IEC) high-level programming language.

[A: KGAS_2837]

The usage of different programming or script languages in the software product is only permitted after justification, verification of suitability and approval by the customer.

5.7.2 Manual Code Construction

[A: KGAS_3948]

This chapter does just apply to software (deliverable) which uses methodes of manually encoded programming.

[A: KGAS_3909]

The contractor must perform an analysis and evaluation of existing coding guidelines with regard to suitability for the project. This analysis must include at least the applicable MISRA guidelines for the project (Guidelines in KGAS_3908).

[A: KGAS_3910]

The contractor has to apply coding guidelines demonstrably appropriate to the project for the entire source code.

[A: KGAS_3911]

The contractor must ensure that the results of the analysis and assessment (KGAS_3909) are taken into account in the selection respectively definition of the applied directives (KGAS_3910).

[A: KGAS_3878]

All deviations from the applied coding guidelines must be justified and documented.

[A: KGAS_3330]

Each unit in the source code has to match to the software detailed design.

[A: KGAS_3561]

Source code must not contain any parts without relation to the software detailed design.

[A: KGAS_3322]

For all units bidirectional traceability between the unit (at least in header-level) and the software detailed design must be available.

[A: KGAS_3323]

For units great in size or with a high complexity (e.g. at least in cases with a cyclomatic complexity greater than 10) the granularity of the traceability must be further refined, so that at least bidirectional traceability between unit elements and the software detailed design is available.

[A: KGAS_3324]

Each unit must be demonstrably verified by source code review.

[I: KGAS_3562]

Possible goals of a source code review (KGAS_3324) could be: to check the non-functional requirements, to check the non-automatically checkable coding guidelines, to check the source code against the software detailed design.

[A: KGAS_3328]

Each unit must be commented with at least a short description of the unit, input and output parameters as well as the return value.

[A: KGAS_3329]

All source code comments must be consistent with the software detailed design.

[A: KGAS_3321]

Naming conventions must be used in the source code (e.g. function's names, macros, variables, type definitions).

[A: KGAS_3325]

The meaning and logical flow of all decision points in the source code (e.g. if-else, for, switch, while) must be commented.

[A: KGAS_3326]

The meaning and logical flow of each implementation of calculations with at least five variables or parameters must be commented.

[A: KGAS_3327]

The meaning and usage of each variables must be commented at the definition place.

5.7.2.1 Source Code Metrics

[A: KGAS_3587]

The source code metrics are valid for the programming language "C". These must be adopted analogously for other programming languages.

[A: KGAS_3570]

If source code metric deviations or boundary value violations are necessary because of project specific reasons, these must be evaluated on unit level and documented including comprehensible justifications.

[A: KGAS_3571]

In case of source code metric deviations or boundary value violations appropriate measures must be applied to ensure the software quality.

[A: KGAS_3462]

The ratio between the number of comments (in- and outside from functions) and the number of statements (Metrics name: COMF) must be greater than 0.2.

[A: KGAS_2769]

The number of non-cyclic execution paths (Metrics name: PATH) must be smaller or equal than 80.

[A: KGAS_3465]

The number of unconditional jumps (Metrics name: GOTO) must be 0.

[A: KGAS_2770]

The cyclomatic complexity (Metrics name: v(G)) must be smaller or equal than 10.

[A: KGAS_2771]

The number of called functions (Metrics name: CALLING) must be smaller or equal than 5.

[A: KGAS_3466]

The number of function calls (Metrics name: CALLS) must be smaller or equal than 7.

[A: KGAS_3467]

The number of function parameters (Metrics name: PARAM) must be smaller or equal to 5.

[A: KGAS_3468]

The number of statements in a function (Metrics name: STMT) must be smaller or equal than 50 but bigger than 0.

[A: KGAS_3469]

The number of nesting levels inside a function (Metrics name: LEVEL) must be smaller or equal than 4.

[A: KGAS_3470]

The number of return statements in a function (Metrics name: RETURN) must be 0 or 1.

[A: KGAS_3472]

The number of recursions (Metrics name: ap_cg_cycle) must be 0.

[A: KGAS_3471]

The vocabular frequency (Metrics name: VOCF) must be smaller or equal than 4.

[A: KGAS_3473]

The total number of MISRA rule violations (Metrics name: NOMV) must be 0. In case of different programming language and coding guidelines the total number of rule violations must be determined according to the coding guidelines defined in KGAS_3908.

[A: KGAS_3474]

The number of MISRA Rule violations per rule (Metrics name: NOMVPR) must be 0. In case of different programming language and coding guidelines the total number of MISRA Rule violations must be determined according to the coding guidelines defined in KGAS_3908.

5.7.3 Graphical Programming and model-based Development

[A: KGAS_3947]

This chapter does just apply for software (deliverable) which uses methodes of grafical programming and/or model based programming.

[A: KGAS_3861]

The supplier must analyze and evaluate existing modeling guidelines regarding their suitability for the project. The analysis must at least consider the project-applicable MISRA guidelines and guidelines of tool manufacturers.

[A: KGAS_3862]

The supplier must verifiably apply one (or more) modeling guideline(s) which are suitable for the project (this might also be an own guideline).

[A: KGAS_3863]

The supplier must ensure that the results of the analysis and evaluation (KGAS_3861) are taken into account for selecting respectively defining the applied guideline(s) (KGAS_3862).

[A: KGAS_3886]

All deviations from the applied coding guideline(s) (KGAS_3862) must be justified and documented.

[I: KGAS_3889]

The hierarchy level in the model which is used for code generation is to be considered as the implementation. This implementation usually consists of basic elements which cannot be further divided and it is the last human-created artifact in the software development chain.

[A: KGAS_3312]

Each model element must be conform to the software detailed design.

[A: KGAS_3893]

The model must not contain any parts without relation to the software detailed design.

[A: KGAS_3894]

For all model elements bidirectional traceability to the software detailed design must be available.

[A: KGAS_3313]

Each model element must be demonstrably verified with review.

[A: KGAS_3314]

Each model element must have a description which contains at least the purpose and goal of the item.

[A: KGAS_3311]

For each defined type of model element, a naming convention must be defined and applied.

[A: KGAS_3456]

The meaning and logical flow of all decision points of the model must be described in the model comments.

5.7.3.1 Metrics for Graphical Programming

[A: KGAS_3865]

The supplier must apply appropriate model metrics with defined boundaries for graphical programming and model-based development.

[A: KGAS_3866]

Selection and appropriateness of the model metrics (KGAS_3865) must be justified (e.g. within a respective strategy document).

[A: KGAS_3902]

If deviations from the model metrics or violations of the defined limit values are necessary for project-specific reasons, these must be evaluated at the unite level and documented with comprehensible justifications.

[A: KGAS_3867]

In case of boundary violations of the model metrics suitable measures must be taken to ensure model quality.

5.7.4 Qualification of Tools

[A: KGAS_3117]

Each software-based tool in the software development toolchain must be qualified. Alternatively, each work product generated by the software-based tool must have been verified by an appropriate verification method (e.g. review, test, validation tool) in order to ensure that it has been properly created according to generation rules and its source work products.

[I: KGAS_3461]

Software-based tools typically include compilers, linkers, assemblers and model-based development tools if they are used for model-based code generation.

[I: KGAS_3675]

The requirement KGAS_3117 is based on the requirements from ISO 26262: 2011 for tool qualification and a software-based tool in the software development toolchain is comparable to a classification of TCL2 or TCL3.

[I: KGAS_3635]

A software-based tool is qualified if at least one of the following requirements is fulfilled: KGAS_3458, KGAS_3459, KGAS_3460.

[I: KGAS_3458]

A software-based tool is qualified if both proven-in-use evidence and a list of known software issues including preventive measures are available.
The proven-in-use evidence (KGAS_3458) must include at least a specification of the tool, a unique version identifier and evidence of successful application in series on the road in at least three comparable and appropriate use cases.

[I: KGAS_3459]

A software-based tool is qualified if the development processes used to develop the tool verifiably achieve a capability level "level 2" in an Automotive SPICE® Assessment (scope acc. to KGAS_2074).

[I: KGAS_3460]

A software-based tool is qualified if a validation result is available which proves fulfillment of the specified requirements of the tool.

[A: KGAS_3481]

Manufacturer information (e.g. manuals, guidelines, erratas) of each software-based tool must be verifiably taken into account in the project.

5.7.5 Compiling

[A: KGAS_2046]

A list of all used macros and definitions used in the build environment must be documented.

[A: KGAS_2954]

Each compiler optimization must be determined at project start. This setup must be used in each test level for the creation of the test objects if binary code is generated for the target platform.

[A: KGAS_3121]

All changes of compiler optimizations during project lifetime must be justified, documented and validated with test measures.

[A: KGAS_2952]

When multiple compiler optimizations are used simultaneously (e.g. runtime and memory optimizations), such optimizations must be validated regarding their mutual compatibility and the validation result must be documented.

5.7.6 Verification of Software Construction Requirements

[A: KGAS_3585]

Before each software delivery to the customer, the supplier must demonstrably verify and document the adherence of the project-relevant coding guidelines, source code metrics, modelling guidelines and model metrics.

[I: KGAS_3586]

For fulfillment of the requirement KGAS_3585, it is recommended to use and integrate state of the art tools for metric and defect analyses within the software development process.

[A: KGAS_51]

The supplier must allow the customer to verify the fulfillment of the KGAS with source code analyses, tool-based analyses and other appropriate methods.

[A: KGAS_2949]

The supplier must support the analyses (KGAS_51) by providing the source code within respective ECU configurations in rooms and presence of the supplier.

[A: KGAS_2782]

With agreement and written approval on part of the supplier, the customer is entitled to provide analysis results to analysis tool manufacturers. Such manufacturers are obligated to uphold confidentiality with regards to content.

5.8 Test

5.8.1 Test Planning

[A: KGAS_3556]

A test plan including a test strategy according to ISO/IEC/IEEE 29119 (KGAS_3479) must be created.

[A: KGAS_3334]

The test plan must include project specific test goals.

[A: KGAS_3335]

The test plan must include a description of how a complete test coverage of all specifications is achieved (e.g. customer requirement specification, interface specification, software requirement specification, software architecture specification, software detailed design).

[I: KGAS_3657]

The test plan (KGAS_3556) can contain a joint test strategy of the customer and supplier.

[A: KGAS_3336]

The test plan must define the test scopes separating the test levels (e.g. system, system integration, software, software integration and unit test).

[A: KGAS_3338]

The test plan must include methods for test case creation, test case selection, creation of test data and test execution.

[A: KGAS_3339]

Black-box test specifications must be created based on requirements respectively based on architecture specifications and software detailed design.

[A: KGAS_3343]

The regression strategy must define methods (e.g. causal chain analysis) to analyze side effects of changes and to select appropriate tests for regression test.

[A: KGAS_3554]

The test plan must at least consider the following software error types: division by zero, overflows, value range violations, infinite loops, type mismatches, initialization errors, unauthorized access, unreachable code.

[A: KGAS_3350]

In the test plan, the dedicated test environment must be assigned to the test levels.

[I: KGAS_3351]

Test cases on all test levels should be automated as far as possible to improve comparability of test results and to reduce test execution efforts.

5.8.2 Test Case Specification

[A: KGAS_3500]

Test case documentation must fulfil the requirements of ISO/IEC/IEEE 29119 (KGAS_3479).

[A: KGAS_54]

Each test case specification must be created by a person who neither implemented nor specified the object to be tested.

[A: KGAS_3367]

Each test case specification must include a textual description in natural language.

[A: KGAS_3355]

Each requirement must be verifiable by at least one dedicated test case on its level (e.g. system, software).

[A: KGAS_3357]

The goal of each test case must be documented.

[A: KGAS_3358]

The expected nominal behavior of each test case must be documented.

[A: KGAS_3538]

The test steps must be documented for each test case.

[A: KGAS_3359]

Any boundary values must be tested for each requirement, interface, parameter, unit and decision point.

[A: KGAS_3362]

Pre and post conditions must be documented for each test case.

[A: KGAS_3363]

Each test case must be independently executable. Mutual dependencies of test cases are not allowed. The formation of test chains to fulfill the test pre-conditions of downstream test cases is permissible, provided that no other influences on the test result arise.

[A: KGAS_3364]

Black-box tests must be specified before white-box tests.

[A: KGAS_3366]

If more than 10 test cases are necessary for verification of a requirement, the quality of the requirement must be assessed (e.g. check if it is atomic). If the requirement cannot be improved, an appropriate structuring of the test cases must be used.

5.8.3 Test Execution in general

[A: KGAS_3360]

All requirements must be fully tested respectively verified regarding the test strategy (e.g. equivalence class partitioning, positive/negative test cases).

[A: KGAS_3369]

Each test result must be documented (e.g. "ok", "not ok").

[A: KGAS_3370]

Each test result must be allocated to an explicit configuration stand of the test object (version of software, hardware, mechanics).

[A: KGAS_3371]

The measured values of the test must be documented.

[A: KGAS_3372]

The used test environment must be documented (e.g. which type of test environment, test bench, software and hardware version).

[A: KGAS_3685]

If the deliverable contains failed test cases, the supplier must analyze the associated risks and communicate them to the customer.

5.8.4 Unit Test

[A: KGAS_3375]

The unit test specification must be verifiably based on the software detailed design.

[A: KGAS_3376]

The unit tests must verify a 100% coverage of the software detailed design.

[A: KGAS_3377]

The unit tests must at least provide 100% branch coverage of the source code (C1).

[A: KGAS_3378]

Deviations of the 100% branch coverage required in KGAS_3377 must be justified.

[A: KGAS_3584]

Source code that cannot be covered by black-box-tests (also see KGAS_3378) must be verified by white-box tests.

5.8.5 Software Integration Test

[A: KGAS_3649]

The software integration test specification must be verifiably based on the software architecture specification.

[A: KGAS_3502]

The interfaces of all software elements and components must be tested regarding static structure, contents and timing behavior.

[A: KGAS_3519]

The hierarchical structure of each software element must be verified.

[A: KGAS_3383]

Software integration tests must test against all requirements which are derived from the software architecture specification. Among others, these requirements include data interfaces (incl. structures and timing), function calls, global variables access, execution orders, resource consumptions, performance, task scheduling, process and interrupt server routines (ISR).

[A: KGAS_3636]

For all tasks, processes and interrupt service routines, the maximum and average net runtimes (see KGAS_3638) on the target hardware must be documented for each release.

[I: KGAS_3638]

The net runtimes are the actual runtimes less the runtime changes caused by runtime measurement.

[A: KGAS_3637]

For each release, the maximum and average resource consumptions (KGAS_3282) of all software elements on the target hardware must be determined, documented and verified against the resource consumption objectives (KGAS_3282).

5.8.6 Software Test

[A: KGAS_3503]

Software tests must verify 100% coverage of the software requirements.

5.8.7 System Integration Test

[A: KGAS_3650]

The system integration test specification must be verifiably based on the system architecture specification.

[A: KGAS_3619]

The interfaces of all system elements must be tested regarding static structure, contents and timing behavior.

5.8.8 System Test

[A: KGAS_3506]

System tests must verify 100% coverage of the system requirements.

[A: KGAS_3507]

For the specification of system tests, the features which are depicted by the system requirements must be considered.

5.9 Quality Assurance and Management

5.9.1 Quality Management

[A: KGAS_53]

The process and product quality assurance of the supplier must be personally and hierarchically independent from the product development.

[A: KGAS_2904]

The goals, evaluation methods, activities and criteria of quality assurance of the supplier must not be influenced by the project lead.

[A: KGAS_3129]

The quality assurance goals must be measurable.

[A: KGAS_3130]

One goal for the quality assurance must be that all work products required for particular processes are created in-time and based on the respective process descriptions and that their quality is assured. This requirement includes at least all work products listed in KGAS_3483.

[A: KGAS_3133]

The quality assurance goals must include a goal that only quality assured products are delivered to the customer.

[A: KGAS_2909]

The supplier quality assurance must regularly report the results of the quality assurance activities within the company structure. This reporting must be independent from the project lead.

[A: KGAS_2911]

The supplier quality assurance must be involved into the approval process of all software deliveries (at least by providing a quality statement).

[A: KGAS_2913]

Employees of the supplier's quality assurance must have the professional qualifications which are necessary to confirm the professional execution of reviews (content-related and formal).

[A: KGAS_2916]

The suppliers quality assurance must regularly verify that quality assurance measures in the project are performed and quality goals are reached.

[A: KGAS_2915]

The supplier quality assurance must be able to make statements on the quality of the processes and work products.

[A: KGAS_3140]

All escalation criteria and escalation channels of the quality assurance must be documented, beginning from the lowest level (clerk Quality Assurance).

5.9.2 Review of Work Products

[A: KGAS_3225]

An analysis must be done which identifies all relevant work products within the project which shall be verified by reviews.

[A: KGAS_3226]

All identified work products (KGAS_3225) but at least the work products listed in KGAS_3483 must be verified by reviews.

[A: KGAS_2942]

For each work product type which is verified by reviews (see KGAS_3225), respective verification criteria must be defined and documented.

[A: KGAS_3508]

Verification criteria (KGAS_2942) must at least include the following:

- Formal requirements
- Content-related requirements
- Consistency
- Plausibility (regarding both within the work product and in relation to parent work products)
- Unambiguousness
- Self-consistency
- Maintainability
- Understandability

[A: KGAS_3234]

Each review method used in the project (e.g. walkthrough, inspection) must be defined and documented.

[A: KGAS_3134]

For each work product that has been identified in KGAS_3225, the state of review must be documented and a respective overview of the review states must be available.

[A: KGAS_3658]

The status information required in KGAS_3134 must be up-to-date, at least on a weekly basis.

[A: KGAS_2941]

The reviews must regularly accompanied by the quality assurance, in order to confirm a professional execution of the reviews.

[A: KGAS_3135]

Each review must be planned regarding review object, verification method, date, effort and participants.

[A: KGAS_3242]

Each review must be documented and for each review at least the following aspects must be evident:

- Date
- Participants
- Roles
- Review object (incl. identifier and version)
- Scope of the review (e.g. chapter, function, scope of changes)
- Verification methods (e.g. inspection, walkthrough)
- Verification criteria
- Criteria fulfilment
- Effort

- Duration
- Deviations (incl. type, reference, severity, violated verification criteria, measures, tracking)

[A: KGAS_3227]

For each review the outcome and further usability of the work product must be evident.

[A: KGAS_3245]

All work products that are to be verified must be verified with a review after initial creation and after modification, latest until customer release.

5.9.3 Verification of Development Processes

[A: KGAS_3477]

Each process must be regularly verified by the supplier quality assurance, at least every two months.

[A: KGAS_2920]

For all identified process deficits, measures must be defined and implemented.

[A: KGAS_2922]

The supplier must inform the customer about all project risks with customer relevance that result from identified process deficits (KGAS_2920).

[I: KGAS_3676]

Examples of customer-relevant project risks are deviations from agreed aspects such as:

- Project objectives
- Delivery dates
- Deliverables
- Quality

5.10 Configuration Management

[A: KGAS_3510]

Configuration elements are single work products, binary files, test environments and development tools which are to be put under configuration management.

[A: KGAS_3386]

The supplier must analyze which work products are to be put under configuration management. At least the work products listed in KGAS_3483 have to be put under configuration management.

[A: KGAS_3482]

For all work products identified in KGAS_3386, the configuration state must be evident and a respective up-to-date (at least weekly) overview of the configuration states must be available.

[A: KGAS_3389]

For each project milestone, quality milestone and release, all configuration elements must be reproducible and recoverable.

[A: KGAS_3390]

For each baseline, the state of all configuration elements must be documented (e.g. "work in progress", "in review", "released").

5.11 Problem Resolution Management

[A: KGAS_3608]

The supplier must communicate all open customer-relevant product problems to the customer at the time of delivery.

[A: KGAS_3410]

The supplier's problem management system must be able to properly map and document the customers problem evaluation system in order to exclude loss of information due to insufficient interfaces.

[A: KGAS_3411]

The supplier's problem evaluation of all product problems must be consistent with the customers problem evaluation.

[A: KGAS_3412]

Product problems found in any test level must be processed by the problem management process.

[A: KGAS_3414]

Work product deviations (e.g. identified by a review) with possible impacts on other work products must be processed by the problem management process.

[A: KGAS_3415]

All process deviations found for both development and supporting processes must be processed by the problem management process.

[A: KGAS_3417]

Problem descriptions must include the particular process step in which the product problem or work product deviation has been identified (e.g. software detailed design review, source code review, unit test, software test, system test).

[A: KGAS_3418]

For all product problems the following information must be evident:

- Hardware version
- Software version
- Initial situation
- Failure severity
- Executed steps
- Expected results
- Observed results
- References to violated specifications
- A statement on the reproducibility of the problem
- Source of the problem

[A: KGAS_3421]

All product problem descriptions must include links to available log files, traces and measurement results necessary for reproducibility.

[I: KGAS_3609]

The problem source is the first faulty work product (e.g. requirement, specification, source code, test specification).

[A: KGAS_3955]

For all work product deviations, the problem analyses must identify the process steps involved in the work product creation.

[A: KGAS_3426]

The change management process of the problem resolution must include all necessary process steps which are to be executed for the correction of the deviation (e.g. revision and review of software requirements specification, revision and review of software detailed design, revision and review of source code, revision and review of unit test specification).

[A: KGAS_3430]

If already delivered or future deliveries with impact on system level are affected by product problems or work product deviations, the supplier must communicate the identified problems and deviations within 2 working days after problem identification to the customer.

[A: KGAS_3431]

Problems in the process must be systematically determined in order to achieve continuous process improvement.

[I: KGAS_3432]

Example for KGAS_3431: A deviation is identified because a requirement in the software architecture is not allocated to a software element. Thus, it needs to be investigated if this is a single case or if the issue is systematic and other requirements are not allocated as well.

[A: KGAS_3434]

Problem descriptions and problem analyses are work products of the project development and must be put under configuration management and must be quality assured with samples (usually 1 out of 5 problems).

5.12 Change Management

[A: KGAS_3518]

All configuration elements (KGAS_3386) must be analyzed in order to identify and define those elements that are to be put under change management. At least the following configuration elements must be put under change management:

customer requirement specifications, system requirement specifications, system architecture specifications, software requirement specifications, software architecture specifications, software detailed design, source code, unit test artefacts (specifications, implementation, reports), software integration test artefacts (specifications, implementation, reports), software test artefacts (specifications, implementation, reports), system integration test artefacts (specifications, implementation, reports), system test artefacts (specifications, implementation, reports), work products of functional safety (hazard and risk analyses, safety analyses, safety case, safety concepts, safety goals, safety requirements).

5.13 Third Party Software

[A: KGAS_3940]

This chapter applies for systems and software (deliverable) which uses third party software.

[I: KGAS_3941]

Free and Open Source Software (see chapter 5.14) is a version of Third Party Software.

[A: KGAS_3438]

The supplier is obliged to encapsulate all third party software within software elements.

[A: KGAS_3883]

The encapsulation (KGAS_3438) must assure that only those functions and interfaces of the encapsulated software can be used which are specified in the software requirements and software architecture.

[A: KGAS_3442]

Systems developed by the supplier must only use complete third party software elements. A partial use (e.g. copy & paste approaches) is not allowed.

[A: KGAS_3142]

Each third party software element must be marked in the software architecture specification.

[A: KGAS_3531]

For each third party software element, origin, author and right holders must be documented.

[A: KGAS_3437]

For all third party software elements, the original requirements on which those third party elements had been developed must be traceable to the software requirements.

[A: KGAS_3440]

The selection of third party software elements (incl. version and patch level) need to be justified and agreed with the customer.

[A: KGAS_3443]

The supplier must ensure that all third party software elements are validated regarding that those components exclusively provide the specified functions and do not provide other, potentially undesired functions.

[A: KGAS_3446]

If third party software elements are used, the supplier must ensure that the usage of all test methods and levels necessary for the development of the whole software is still possible.

[A: KGAS_3923]

The supplier solely bears the responsebility that the usage of the delivered software is admissible in accordance to the contract and any other regulations.

[I: KGAS_3924]

The customer is responsible for the technical condition of the auxiliary software as such.

5.14 Free and Open Source Software

[A: KGAS_3942]

This chapter applies to systems and software (deliverable) which uses Free and Open Source Software (see KGAS_3822).

[A: KGAS_3822]

The use of FOSS is permitted only if the client is informed in writing prior to the use of FOSS by the contractor in accordance with the processes specified by him and confirms that the use of FOSS is in accordance with the license. If Volkswagen AG is the client, the process prior to the use of FOSS provides for the requirement of a written consent on the part of the client.

[I: KGAS_3821]

A copyleft license is a form of usage and licensing disclaimers for open source software, which can lead to the respective software elements integrated or linked with open source software being distributed only under the respective usage and license terms of the copyleft license.

[I: KGAS_3840]

A copyleft effect refers to the use of free and open source software under a copyleft license, and as a result of which any modification ("each derivative work") must also be classified as FOSS licensed under a copyleft license (see also KGAS_3821).

[A: KGAS_3833]

The supplier must confirm that no software element of the delivered software triggers a copyleft effect, which leads to the software product as a whole being classified as FOSS licensed under a copyleft license.

[A: KGAS_3830]

The supplier may only use FOSS in the delivered software which does not restrict the contractual and intended use of its services by the customer and Volkswagen Group companies.

[A: KGAS_3801]

The supplier must provide the customer with information on all free and open source software elements used in the delivered software. For each software element used, the following information must be included:

- Name
- Unique version identifier
- License name with unique license version number
- Complete license text
- Download link of the license text and source code including the last access date
- Source code and copyright notices
- Information on whether source code and copyright notices are to be shared or disclosed
- Any sub-elements required for the use of the software element, including the aforementioned details on licensing
- Information as to whether the license prescribes a mandatory provision of the license information to the end user
- Interface information for the integration of open source software elements with the exclusion of the triggering of copyleft effects
- Any files contained in the Software Element and under a different license, including the aforementioned licensing information.

[A: KGAS_3834]

The supplier must provide the customer with the information required in KGAS_3801 with each version of the software (release, update, version, etc.) as well as at the request of the customer, whereby both a complete overview must be made available as well as a delta overview that marks the changes in comparison with the previous status.

[A: KGAS_3884]

The supplier must assure that the delivered software does not contain any license incompatibilities.

[A: KGAS_3824]

The supplier must analyze the delivered software with a state-of-the-art analysis software before the delivery and must assure that all included FOSS elements including any dependencies and sub element are identified.

[A: KGAS_3828]

At the request of the customer, the supplier must provide the customer with the details, materials, documents and results of the analysis carried out (KGAS_3824).

[A: KGAS_3807]

At the request of the customer, the supplier must allow the customer to perform tool-based analyses in rooms of the supplier and together with the supplier over the entire source code.

[A: KGAS_3810]

If the supplier implements a technical solution patented by the customer, no open source software solutions may be used whose licenses impede the cost liable licensing of the patent.

[A: KGAS_3827]

The supplier solely bears the responsebility that the usage of the delivered software is admissible in accordance to the contract and any other regulations.

[A: KGAS_3925]

If auxiliary software is included, the customer will contribute any necessary license and copyright information for the auxiliary software component.

5.15 Security Relevant Development

5.15.1 General Security Requirements

[A: KGAS_3687]

This chapter applies to systems and software (scope of supply) that have been identified as security-relevant by the client's brand security department.
If the determination of the security relevance has not yet been made for the award of the scope of supply, this must be actively communicated and requested by the contractor. Until the final determination of the security relevance, the contractor must assume that the scope of delivery is security-relevant.

[A: KGAS_3738]

The supplier must conduct and document security risk analyses (Chapter 5.15.5) on system and software level for the entire deliverable.

5.15.2 Security Terminology

[I: KGAS_3703]

Risk analysis
A risk analysis is a methodical approach that identifies security goals and estimates possible damages as well as corresponding threats.

[I: KGAS_3704]

Asset
In the sense of security, assets are entities worth protecting for an institution.

[I: KGAS_3705]

Security goal
A security goal is a specific property of an asset to ensure security of the deliverable.

[I: KGAS_3706]

Threat
A threat is a possible circumstance or event which may violate at least one security goal.

[I: KGAS_3707]

Vulnerability
A vulnerability is a property that allows a successful attack.

[I: KGAS_3868]

Backdoor
A Backdoor is an access to a software or hardware system that bypass the specified access thereby it was implemented intentionally or secretly.

[I: KGAS_3708]

Attack
An attack is an unwanted or unauthorized act that realizes a threat.

[I: KGAS_3709]

Attack vector
An attack vector describes a possibility to perform an attack.

[I: KGAS_3710]

Risk
A risk is a set of threats that are evaluated with respect to the potential damages caused by the violation of security goals as well as the efforts needed for a successful attack or the aggregation of multiple scored threats.

[I: KGAS_3927]

Car-Security-Incident
By a car-security incident is meant a unwanted or unpredicted security event (e.g. focused cyber attack) as well as the reveal of weaknesses or misconfigurations, which threaten the car-security.

[I: KGAS_3811]

Incident response process
An incident response process is a defined process with the goal to adjust series products as soon as possible in case of detected vulnerabilities in order to minimize any risks (possibly accompanied by functional constraints) and to eliminate vulnerabilities with recovery of full functionality.

[I: KGAS_3928]

Penetration test
A penetration test serves to determine the potential for attack on a control unit or on a software. In the penetration test, the security properties of the test object are examined with an explorative and offensive test procedure. Here, the chances of success for an attack are shown in order to derive the necessary security measures. In addition, the effectiveness of the security measures already implemented will be reviewed in order to gain knowledge of how to improve the security measures. The test result lists the identified vulnerabilities incl. their ratings.

[I: KGAS_3711]

Security requirement
Security requirements define requirements for the deliverable specifying properties to prevent or reduce threats.

[I: KGAS_3712]

Security control
A security control describes the (technical) realization of security requirements to reduce risks and logically group security requirements that are needed to successfully implement this security control.

[I: KGAS_3713]

Security concept
The security concept is a work product to document security relevant aspects of the deliverable, which mitigate threats. The security concept comprises especially security controls, considered constraints, architectures as well as made assumptions and conditions.

[I: KGAS_3715]

Data worthy of protection
Data worthy of protection is data which must be secured by means of the security concepts or security measures.

[I: KGAS_3717]

Trustworthy
A system, data source, etc. is trustworthy if a proof exists on which one can rely to a certain extent and there is no compromising.

[I: KGAS_3718]

Trust boundary
A trust boundary describes the transition between different levels of confidence.

[I: KGAS_3720]

OWASP (Open Web Application Security Project)
OWASP is an online community providing inter alia a standard to conduct security verifications on the application layer.
Reference: https://www.owasp.org/

[I: KGAS_3721]

CWE (Common Weakness Enumeration)

CWE is a software community project providing a catalog of software weaknesses and vulnerabilities.
Reference: https://cwe.mitre.org/

[I: KGAS_3904]

CVE (Common Vulnerabilities and Exposures)
CVE® is a list of entries - each containing an identification number, a description, and at least one public reference - for publicly known cybersecurity vulnerabilities. Reference: https://cve.mitre.org/

5.15.3 Security Management

[A: KGAS_3725]

A security plan must be created for the structuring of security relevant projects.

[A: KGAS_3726]

Security activities must be planed and documented (KGAS_3725).

[A: KGAS_3727]

The progress and open points of security activities must be updated at least biweekly (KGAS_3170).

[A: KGAS_3728]

For the development of security relevant systems and software, the supplier must identify needed qualifications and must employ only trained staff.

[A: KGAS_3729]

Aspects of security must be considered and marked in project risk management.

[A: KGAS_3815]

An authorization and role concept must be implemented for configuration management and the actuality must be checked at least biweekly.

[A: KGAS_3851]

After a known unauthorized access to the configuration management system the original state of configuration items must be restored and the client is to be informed.

[A: KGAS_3733]

A responsible person for security activities of the project must be nominated.

[A: KGAS_3735]

The security responsible (KGAS_3733) must assure that the security plan (KGAS_3725) is tracked and satisfied.

5.15.4 Security Project Initiation

[A: KGAS_3736]

The supplier must analyze the security related customer requirements regarding understandability and correctness and clarify discrepancies.

[A: KGAS_3853]

Security requirements must be identifiable and categorized as such.

[I: KGAS_3737]

It is recommended to clarify and document the mutual consent in a cooperative process with the customer. All inconsistencies and uncertainties must be analyzed and resolved accordingly, also referring to existing requirements which until now are not classified as security relevant.

5.15.5 Security Risk Analyses and Security Concept

[A: KGAS_3740]

As part of the security risk analyses, all processed data must be identified and classified according to security goals of IT-security.

[A: KGAS_3741]

As part of the security risk analyses, all interfaces and trust boundaries to and from the commissioned software must be identified and documented.

[A: KGAS_3742]

Threats must be systematically identified and documented for each interface in security risk analyses.

[A: KGAS_3743]

For each threat identified in a security risk analysis, the risk must be classified according to a set of grading criteria specified by the supplier.

[A: KGAS_3744]

The supplier must consider identified and/or specified security requirements in the risk analyses.

[A: KGAS_3749]

For all risks identified in the security risk analyses which have not been accepted, security controls must be defined.

[A: KGAS_3750]

Security controls must verifiably lead to security requirements.

[A: KGAS_3751]

The security concept of the supplier must consider all risks.

[A: KGAS_3745]

In case of changes at system and/or software level, the security risk analyses as well as the security concept must be updated accordingly.

[A: KGAS_3746]

In case of identification of new attack vectors against used technologies during development, all security risk analyses as well as the security concept must be updated accordingly.

5.15.6 Security Architecture and Security Design

[A: KGAS_3753]

System architecture, software architecture and detailed design must verifiably cover all security requirements.

[A: KGAS_3755]

All data sources must be identified and classified either as trustworthy or non-trustworthy.

[I: KGAS_3855]

Data sources that are located outside the specified trust boundaries are not trustworthy and data sources that are located within the specified trust boundaries are trustworthy. The deliverable may not necessarily be a trust boundary. The deliverable can also have more than one trust boundaries, e.g. in case of multiple µC.

[A: KGAS_3756]

All data from non-trustworthy sources must be validated before being processed.

[A: KGAS_3758]

Only specified error messages, log records and diagnostic records must be distributed via specified interfaces.

[I: KGAS_3905]

Objective of the requirement KGAS_3758: The output of unspecified information can be misused to draw conclusions about the system for the purpose of an attack or a compromise.

[A: KGAS_3759]

For all software elements, the used version and patch level (if any) must be documented.

[A: KGAS_3957]

The scope of delivery must not contain any known weak points. Deviations must be considered and justified in the risk analyzes.

[A: KGAS_3929]

For the analysis of the devliverable suitable sources for the identification of weaknesses have to be specified.

[I: KGAS_3930]

Sources for the identification of weaknesses may be beside others publications of CWE (KGAS_3721), CVE (KGAS_3904) or reports from the customer.

[A: KGAS_3761]

It must be ensured that no backdoors exist.

[A: KGAS_3896]

It must be ensured that no unused code (e.g. inaccessible or dead code) exists.

5.15.7 Security Implementation

[A: KGAS_3762]

In addition to applying appropriate coding guidelines (KGAS_3909) or modeling guidelines (KGAS_3861), the contractor must apply security coding guidelines.

[I: KGAS_3869]

MISRA C:2012 security extension (KGAS_3908) may be used as security coding guideline for coding language C.

[A: KGAS_3870]

Security coding guidelines must be identifiable as such if they are specified in the same document as the standard coding guidelines.

[A: KGAS_3772]

The supplier must conduct code analyses, in which the compliance of the KGAS_3762 is checked.

[I: KGAS_3872]

The security code analyses may be conducted manually or by a tool.

[A: KGAS_3764]

All deviations from the security coding guideline (KGAS_3762) must be justified and documented.

[A: KGAS_3765]

In case the contracted deliverable contains web application(s) or similar, the OWASP (KGAS_3720) guidelines must be followed.

5.15.8 Security Integration and Security Verification

[A: KGAS_3769]

The supplier must define and apply security test strategies in parallel to the system and software specification.

[A: KGAS_3770]

The security tests must be derived from the security requirements and linked to them.

[A: KGAS_3771]

A security test report must be available for each release summarizing conducted security test with according results.

[A: KGAS_3773]

Penetration tests must be specified and conducted by project independent instances aiming to compromise the deliverable.

5.15.9 Security Case

[A: KGAS_3775]

A security case must be provided by the supplier with the first delivery of a C-sample.

[A: KGAS_3931]

The security case must be supplemented not later then 0-series by the evidence that the flash process has been secured against unauthorized accesses an manipulations.

[A: KGAS_3818]

In case of any changes, the security case must be continuously updated until delivery of the series product.

[A: KGAS_3776]

The security case must include the results of the security activities (KGAS_3725) planed in the security plan.

[A: KGAS_3817]

The security case must include a summary of the results of all risk analyses.

[I: KGAS_3873]

The risk analyses as well as the detailed results of the risk analyses can be viewed within a Technical Revision at the supplier.

[A: KGAS_3777]

The security case must show that all security requirements were implemented and verified.

[A: KGAS_3819]

The security case must show the compliance with the security coding guidelines.

[A: KGAS_3932]

The security case must show that the reaktion process to handle iditified weaknesses (KGAS_3877) and the active monitoring of the deliverable (KGAS_3784) is established.

5.15.10 Subsequent activities of Security relevant development

[A: KGAS_3874]

The supplier must communicate one central point of contact for security incident management to the according security incident management of the Volkswagen Group (KGAS_3890) until the start of the project.

[A: KGAS_3877]

The supplier must establish an incident response process to handle identified vulnerabilities in agreement with the according security incident management of the Volkswagen Group (KGAS_3890).

[A: KGAS_3784]

The contractor must establish a process for active monitoring of the scope of delivery for vulnerabilities after delivery to the client.

[A: KGAS_3933]

If the supplier needs a sub-supplier chain to deliver the product for the client, he have to ensure the effectiveness and reaktion ability in his sub-supplier chain.

[A: KGAS_3934]

Suspected cases must be reported to the corresponding security incident management of the Volkswagen AG (KGAS_3890) within one working day.

[A: KGAS_3935]

Known vulnerabilities during or after the developement phase must be reported to the customer within 2 working days.

[A: KGAS_3939]

Any communication of the supplier regarding car-security-incidences and suspected cases must base on the need-to-know principle.

[A: KGAS_3936]

Any outward communication which exlusivly affects the customer have to be aligned with the corresponding security incident management of the Volkswagen AG (KGAS_3890).

[A: KGAS_3785]

In case of security incidents, the established incident response process (KGAS_3877) must be performed.

[A: KGAS_3937]

A detailed technical analysis incl. cause, impact and possible measures must be reported to the corresponding security incident management of the Volkswagen AG (KGAS_3890) within 10 working days.

6 References

6.1 Documents of the Volkswagen AG

[I: KGAS_2834]

Formel Q Capability Software: Supplier quality capability evaluation guidelines for software development processes [Volkswagen AG; Software-Quality Assurance] Available on http://www.vwgroupsupply.com/

[I: KGAS_3611]

Entwicklungsrichtlinie Funktionale Sicherheit (EFS) [Volkswagen AG; Software-Quality Assurance] Available on
http://www.vwgroupsupply.com/

[I: KGAS_3908]

List of Coding-/Modeling Guidelines: List of common coding guidelines and modeling guidelines in the automotive context [Volkswagen AG; Software Quality Assurance] Available at http://www.vwgroupsupply.com/

6.2 Documents of the German Association of Automotive Industry (VDA)

[I: KGAS_3813]

Blue-Gold prints of VDA Automotive SPICE® Guidelines. Obligatory after the end of a transitional period defined by the VDA (30.06.2019).

6.3 Documents of the Automotive Special Interest Group

[I: KGAS_2086]

Automotive SPICE®: process reference model (PRM) and related process assessment model (PAM). The process model is available for free from http://www.automotivespice.com.

[I: KGAS_3887]

Automotive SPICE® Process Assessment / Reference Model (PAM/PRM) - RELEASE v3.1 - 1st of November 2017

6.4 Documents of the MISRA

[I: KGAS_2091]

The **M**otor **I**ndustry **S**oftware **R**eliability Association is a work commitee of the (mostly british) automotive industry. The website and publictions are available on: http://www.misra.org.uk.

6.5 International Standards and Norms

[I: KGAS_3043]

ISO/IEC 25010:2011 Systems and software engineering -- Systems and software Quality Requirements and Evaluation ("SQuaRE")

[I: KGAS_3480]

ISO/IEC/IEEE 24765:2010 Systems and software engineering - Vocabulary

[I: KGAS_3512]

IEEE 1016:2009 Standard for Information Technology - Systems Design - Software Design Descriptions

[I: KGAS_3478]

IEEE 828:2012 Standard for Configuration Management in Systems and Software Engineering

[I: KGAS_3479]

ISO/IEC/IEEE 29119:2013 Software and systems engineering - Software testing

[I: KGAS_3895]

ISO 26262:2011 Road vehicles -- Functional safety

[I: KGAS_3790]

ISO/IEC 7498:1994 Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model

Relevant for Security-relevant systems according to KGAS_3687.

7 Confidentiality Disclosure

[A: KGAS_3488]