VOLKSWAGEN

AKTIENGESELLSCHAFT

Group Basic Software Requirements

Basic requirements that the Volkswagen Group demands on vehicle-based and vehicle-related software.

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

First issue	06.09.2002
Date of revision	05.05.2023
Version	4.1

Contents

1 Preamble

1.1 Purpose

[I: KGAS_4110]

KGAS defines the framework specified by Volkswagen AG to ensure that the development of software for the deployment in the vehicle and vehicle environment is in accordance with the state of the art.

[I: KGAS_4111]

The Automotive SPICE process model serves as the fundamental basis for state-of-the-art software development (see KGAS_3887).

[I: KGAS_4112]

KGAS specifies the Automotive SPICE process model by means of process requirements (see KGAS_3887).

[I: KGAS_4113]

KGAS addresses all project members of a software development project.

[I: KGAS_4114]

KGAS thus creates the basis for achieving the software quality in the product through the achievement of stable processes (see KGAS_3043).

[I: KGAS_3090]

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

[I: KGAS_3633]

The terms "contracting authority" and "contractor" 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 81250
38436 Wolfsburg
Germany

Contact: See ONE.Konzern Business Plattform

Audi AG
Quality Assurance Vehicle Software
85045 Ingolstadt
Germany

Contact: See ONE.Konzern Business Plattform

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

Contact: See ONE.Konzern Business Plattform

1.3 Cybersecurity Incident Management

[R: 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: See ONE.Konzern Business Plattform

[I: KGAS_3891]

Porsche AG

Contact: See ONE.Konzern Business Plattform

[I: KGAS_3892]

MAN SE

Contact: See ONE.Konzern Business Plattform

[I: KGAS_3926]

Audi AG

Contact: See ONE.Konzern Business Plattform

[I: KGAS_3958]

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

[I: KGAS_3993]

Škoda Auto a.s.

Contact: See ONE.Konzern Business Plattform

[I: KGAS_4017]

SEAT S.A.

Contact: See ONE.Konzern Business Plattform

[I: KGAS_3959]

Region China

Contact: See ONE.Konzern Business Plattform

2 Scope

[I: KGAS_4057]

This document includes requirements [A], information [I] and frame conditions [R].

[R: KGAS_4058]

Adherence to requirements must be demonstrable.

[R: KGAS_4059]

Adherance to framework conditions must be ensured.

[I: KGAS_4060]

Information is used to understand the requirements.

[R: KGAS_25]

The requirements in this document apply to on-board software/software-defined systems and other vehicle-related software/software-defined systems and their development processes.

[I: KGAS_3666]

Vehicle-related software/software-defined systems includes all software components that have an impact on the vehicle and its functionalities.

[R: KGAS_3028]

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

[R: KGAS_3551]

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

[R: KGAS_3546]

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

[I: 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.

[R: KGAS_3610]

If the software is classified safety-relevant by the contracting authority, the software and its development processes are subjects to the ISO 26262:2018 Road vehicles -- Functional safety (KGAS_3895).

3 Rights of the contracting authority and Obligations of the contractor

[A: KGAS_3885]

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

[A: KGAS_1806]

On request of the contracting authority, the contractor must provide evidence of the compliance with the KGAS.

[A: KGAS_27]

The contractor must obligate all its sub-contractors to fulfil the KGAS and must ensure its execution.

[A: KGAS_2933]

If the contractor or its sub-contractors cannot fulfil the KGAS completely, the contractor must seek written approval of the deviations from the Quality Assurance of the contracting authority before the start of the project. The agreed and approved changes are to be sent to Group Quality (contact please refer KGAS_2085).

[I: KGAS_2932]

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

[A: KGAS_3107]

If deviation from KGAS is identified, the contractor 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 contracting authority.

[A: KGAS_2184]

The contractor must assure that the contracting authority 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 contractor to third parties in relation to requirements and content of such third parties.

[R: KGAS_1877]

The contracting authority may have its rights of inspection (KGAS_2184) arising from the KGAS exercised in fully or in part by third parties.
The contractor may request that an aggregated report be prepared by a third party to protect its confidential information. This report must ensure the evaluation of the quality of the contractor's processes, but the information concerning the contractor's concrete process flow may be omitted.

[I: KGAS_2948]

The contracting authority shall nominate the third party to the contractor before commissioning of the third party (KGAS_1877). Third parties in this sense are obliged to maintain confidentiality and may not be competitors of the contractor either themselves or any of their affiliated companies (§15 AktG). In event that the contractor can provide objectively comprehensible, factual reasons and prove that the commissioning of the aforementioned third party is unreasonable for the contractor, the contracting authority and contractor will collectively - considering the respective economic interests - agree on the further course of action and nominate another third party if necessary. Affiliated companies of the contracting authority are not third parties within the sense of this provision.

[A: KGAS_3612]

The contractor must collect metrics from the beginning of the project. The contractor must collect the metrics for each release and on request, however at a minimum every 4 weeks.

[A: KGAS_3915]

The contractor must provide the collected metrics to the contracting authority upon their request. The exchange format is defined by the contracting authority.

[A: KGAS_3623]

The duration of the data capture and the reporting cycles can be determined by the contracting authority at the beginning of the project in deviation to KGAS_3612.

[A: KGAS_4107]

The minimum set of metrics is defined in KGAS_4093 unless otherwise specified by the contracting authority.

[I: KGAS_3916]

The collected metrics (KGAS_3915) are used for approvals by the contracting authority (e.g. build sample approval, PPF procedure).

[A: KGAS_4002]

As part of the PPF procedure (formerly Sampling), the metrics contained in (KGAS_4093) shall be provided with the "KPIs" identifier as evidence.

[A: KGAS_3624]

Improvement actions with target dates must be defined for process deviations that are identifiable through the utilization of the metrics (KGAS_3612).

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 contracting authority and which is represented by a subset of the requirements.

4.2 System

[I: KGAS_2877]

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

[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, components and connectors) and mechanics (e.g. housing).

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.

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 or more source code files in C, one or more 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's element is a part of a unit (e.g. calculations, interfaces, function calls or macros).

[I: KGAS_3647]

Unit's 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 a software which is developed by or for the contracting authority.

[I: KGAS_3919]

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

[I: KGAS_3921]

Auxiliary software is a software which is provided by the contracting authority to the contractor 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 is subject to sharing or disclosure of the source code of the software under such substantial obligations.

[I: KGAS_3826]

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

[I: KGAS_3952]

Contractor software is a third-party software that has been developed for the contractor or by themself, but not against payment on behalf of the contracting authority.

[I: KGAS_3953]

Supplied Software is the software supplied by the contractor (including contractor software, auxillary software, third-party software).

[I: KGAS_4066]

The contents of the scope of delivery are defined within the scope of an order.

[I: KGAS_3954]

Software of third party is a third party software that is not contractor software.

[I: KGAS_4067]

Whitebox testing is Testing based on an analysis of the internal structure of the component or system.

[I: KGAS_4068]

Blackbox test is a test technique based on an analysis of the specification of a component or system according to ISO/IEC/IEEE 29119 (KGAS_3479).

[I: KGAS_3956]

Dead code is a code that is included in the delivery and that cannot be executed by the program flow (including error handling) and not included in the specification.

[I: KGAS_3962]

Dead code is fixed, when the code is not executed because of these
- is no longer required,
- is not invoked,
- cannot be invoked.

[I: KGAS_3964]

Dead code is not fixed, when
- a requirement is planned, implemented, but not used by the contracting authority,
- the code is not invoked by parameterization (e.g. target data container).

5 System and Software Development

[I: KGAS_3124]

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

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_4095]

If cybersecurity relevant parts are included in the scope of delivery (see KGAS_3687), they must be developed with processes that achieve at least a capability of "Level 2" in an Automotive SPICE for Cybersecurity® Assessment with the assessment scope according to the VDA Automotive SPICE for Cybersecurity® Guidelines (KGAS_4096).

[A: KGAS_3669]

The requirement KGAS_2074 must also be fulfilled for software already developed. These includes, 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 stakeholder requirements until the unit level and up to the tests.

[A: KGAS_3679]

The contractor must be able to implement in the software, all deemed necessary error corrections from the contracting authority, up to 15 years after the end of production (EoP). The contractor must guarantee to provision that the requirements from KGAS for the delivery of software and that all necessary prerequisites for processing and delivery of the software are met.

[A: KGAS_3407]

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

[A: KGAS_3483]

An analysis must be performed and documented, to discover which additional work products, are part of the process description, in addition to the work products required in Automotive SPICE®. This can e.g. be the following:

- strategy documents
- All work products of the Development Interface Agreements (DIA) according to ISO 26262:2018 Road vehicles -- Functional safety (KGAS_3895).

[A: KGAS_2035]

All work products stipulated by the process must be consistent with each other as regards to their content at the time of the release of contracting authority.

[A: KGAS_3552]

The degree of refinement of specification elements (for example, requirements, architectural elements) from one level of abstraction of respective hierarchy level to the next lower level of abstraction must not exceed a ratio of 1 to 10.
If this ratio has to be exceeded in individual cases, the reasons for this must be substantiated.

[R: KGAS_3498]

Each change of the development processes and development sites of the contractor, 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 which specify functionalities that have no traceable link to requirements from contracting authority must be approved by the contracting authority.

[A: KGAS_3795]

Each software unit included in the product must have a traceable link to the implementation of at least one system requirement.

[I: KGAS_3797]

The goal of the requirements KGAS_3794 and KGAS_3795 is to avoid unordered, potentially malicious software components in the product (malicious means e.g.: illegal, safety and security violating, unethical, environmentally damaging, image damaging, data protection violating).

[R: KGAS_3968]

In order to ensure that the use of individual control units in the field, provide data protection conformation , the data protection requirements must be taken into account from the start of development. The "Guideline for the data protection requirements for the (further) development of control units with memory function" is to be adhered (KGAS_3966).

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 estimates underlying assumptions must be documented.

[A: KGAS_3163]

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

[A: KGAS_3593]

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

[A: KGAS_3146]

For all work packages, existing 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.

[A: KGAS_3167]

For change and problem resolution a size appropriate effort must be scheduled.

[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 in 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_3177]

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

[A: KGAS_3178]

Explicit definition for the degree of fulfillment of work packages and activities must exist and be applied.

[A: KGAS_3191]

Project risks must be traceably identified, evaluated and include counter measures.

5.3 Documentation of Deliverable

[A: KGAS_4115]

The documentation is provided in Release Notes KGAS_4116, unless otherwise agreed between the contracting authority and the supplier.

[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 in the scope of delivery must be documented, including a description of any bug fixes.

[I: KGAS_3938]

The documentation also includes the release notes and the feature overviews of all scopes (e.g. modules) from sub-contractors.

[A: KGAS_3216]

The tests carried out for the scope of the delivery and their test results must be documented.

[A: KGAS_3218]

The configuration status for the scope of delivery (version of the software and possibly hardware, mechanics etc.) must be documented (including version, baseline, status of the underlying requirements, the valid architecture, etc.).

[A: KGAS_3219]

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

[A: KGAS_3220]

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

[A: KGAS_3221]

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

[A: KGAS_3222]

Known failures of the system respectively software and their impacts must be documented (see although KGAS_3430).

[A: KGAS_3223]

The version of the underlying software on which the software supplied was developed 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

[A: KGAS_3406]

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

[A: KGAS_3548]

Own requirements of the contractor (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_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

[A: KGAS_3263]

For each functional requirement the positive cases and the negative cases must 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 non-functional requirement must be verifiably considered into the derived requirements and work products.

[A: KGAS_3267]

Each specification of system and software requirements 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 must be defined for the description of the system and software elements within the system and software architecture specifications.

[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_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_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_4061]

The software detailed specification must describe the solution approach (KGAS_4062) for the externally perceivable behavior of a unit.

[I: KGAS_4062]

The solution approach defines algorithms, calculations, interfaces, function calls and macros and the behavior in the event of an error, as applicable in each case.

[A: KGAS_4063]

All necessary information to implement a solution approach (KGAS_4062) shall be described or referenced.

[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_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 for every graphical or model-based program.

[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 contracting authority.

5.7.2 Manual Code Construction

[R: KGAS_3948]

This chapter does just apply to software (deliverable) which uses methods 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 (see KGAS_3909).

[A: KGAS_3911]

The contractor must ensure that the results of the analysis and assessment (KGAS_3909) are taken into account when selecting or defining the respective applied guidelines (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 greater in size or with a higher complexity (e.g. at least 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 source code must be commented for all computations that include at least five variables or parameters with regard to the meaning or logic.

[A: KGAS_3327]

Each variable definition must be commented with its meaning and usage.

5.7.2.1 Source Code Metrics

[A: KGAS_4099]

The source code metrics are used to measure the quality of the source code.
The metrics are indicators according to the ISO 25010 quality criteria (KGAS_3043).

[R: KGAS_3587]

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

[A: KGAS_4100]

In the case of manual source code development, the contractor shall apply appropriate state-of-the-art source code metrics with defined thresholds.

[A: KGAS_3570]

The selection and appropriateness of the source code metrics (KGAS_4100) must be justified (e.g., in an appropriate strategy document).

[A: KGAS_4065]

Deviations from the source code metrics shall be documented with comprehensible justifications.

[A: KGAS_4101]

Threshold violations must be justified in a comprehensible manner at the same level at which they are identified.

[A: KGAS_4102]

Threshold violations must be evaluated in terms of risks and impacts.

[A: KGAS_3571]

Based on risk assessments, appropriate measures must be taken to ensure software quality.

[A: KGAS_4103]

If the contractor does not use appropriate source code metrics (see KGAS_4100), the metrics according to KGAS_4104 apply.

5.7.3 Graphical Programming and model-based Development

[R: KGAS_3947]

This chapter only applies to software (scope of delivery) that uses methods of graphic programming and/or model-based programming.

[A: KGAS_3861]

The contractor 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 (guideline see KGAS_3908).

[A: KGAS_3862]

The contractor 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 contractor 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 modeling 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 objects 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 conform 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 verified by review.

[A: KGAS_3314]

Each model element must have a description that contains at least the aim and purpose.

[A: KGAS_3456]

In the model, comments are must for all decision points regarding the meaning or logic.

5.7.3.1 Metrics for Graphical Programming

[I: KGAS_4105]

The model metrics are used to measure the quality of the graphical programming.
The metrics are indicators according to the quality criteria of ISO 25010 (see KGAS_3043).

[A: KGAS_3865]

The contractor must apply appropriate model metrics with defined boundaries for graphical programming.

[A: KGAS_3866]

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

[A: KGAS_4064]

Deviations from the model metrics shall be documented with comprehensible justifications.

[A: KGAS_3902]

Threshold violations must be justified in a comprehensible manner at the same level at which they are identified.

[A: KGAS_4106]

Threshold violations must be evaluated in terms of risks and impacts.

[A: KGAS_3867]

Based on risk assessments, appropriate measures must be taken to ensure software quality.

5.7.4 Machine Learning (ML)

[R: KGAS_4009]

This chapter only applies to software (scope of delivery) in which machine learning (ML), neural networks (NN) or comparable data-based components are used. It is assumed that components have been trained offline.

5.7.4.1 Terminology

[I: KGAS_4069]

ML-Model
ML model refers to a component, such as a neural network, in which machine learning or comparable data-based approaches are used.

[I: KGAS_4022]

Training data
Training data is the data that is used to train the ML model (e.g. neural network).

[I: KGAS_4023]

Validation data
Validation data is the data that is used to check and control the generalization capability of the ML model (e.g. neural network) in the learning phase.

[I: KGAS_4024]

Test data
Test data is the data from an additional test data set that is used in the test phase to determine the performance and generalization capability of the ML model (e.g. neural network).

[I: KGAS_4025]

Online Training
Online Training corresponds to data point based learning, in which adaptations are still carried out in productive use (the training thus also extends to the period of productive use).

[I: KGAS_4026]

Offline Training
Offline Training corresponds to data record base learning, in which no further adaptations are carried out in productive use (the training is thus completed before productive use).

[I: KGAS_4028]

Learning
Learning refers to any change in a system that allows it to perform the same task or a task of the same type more efficiently or effectively the next time the same process is repeated.

[I: KGAS_4029]

Supervised learning
Supervised learning refers to the learning in which the ML model is specifically exposed to a particular stimulus in a training mode and the response of the ML model leads to the corresponding appropriate changes in a target/actual comparison.

[I: KGAS_4030]

Unsupervised learning
Unsupervised learning describes learning in which the model learns patterns from unlabeled data in a training mode.

[I: KGAS_4070]

Labeled Data
Labeled data are data with annotated labels, where a label represents a target class in the case of classification and a target value in the case of regression.

[I: KGAS_4071]

Baseline Models
Baseline models provide a reference regarding the performance of the ML model. In this context, they can be derived from non-ML approaches in addition to the state of the art. (For example, a baseline of a classification problem between A and B with a distribution of 90% A and 10% B would be a 90% accuracy, since this accuracy would also be achieved if A was always an output).

[I: KGAS_4031]

All terminologies used are based on VDI/VDE guideline 3550, sheet 1.

5.7.4.2 Data acquisition

[A: KGAS_4010]

A data acquisition strategy must exist that includes goals and procedures for generating, procuring, filtering, and preparing data, considering the aspects of completeness, representativeness, and balance.

[I: KGAS_4072]

Completeness describes the goal of generating, procuring, filtering and processing data in accordance with all the requirements placed on it. Completeness thus includes the existence of requirements placed on the data.

[I: KGAS_4073]

Representativeness is achieved when the generated, procured, filtered and processed data represents the specified application area

[I: KGAS_4074]

Balance is achieved when the generated, procured, filtered and processed data are distributed in accordance with the requirements.

[A: KGAS_4075]

The data acquisition strategy must include goals and procedure for labeling the data; even in the case of unsupervised learning, a decision must be made about the need for individual labeled data and, if necessary, an appropriate procedure defined.

[A: KGAS_4076]

The data acquisition strategy must include goals and procedure for partitioning the data into training, testing, and validation data.

[A: KGAS_4077]

The data acquisition strategy must include goals and procedure for archiving the data (training, test, and validation data) and for integrating it into configuration management.

[A: KGAS_4078]

The data acquisition strategy must include goals and procedure for modifying the training data and validation data according to the training strategy.

[A: KGAS_4015]

A data protection strategy must be defined and applied. The data protection strategy must fulfill the GDPR and comparable applicable standards and laws.

[A: KGAS_4011]

Evidence of the appropriate implementation of the data acquisition strategy must be documented and made available to the contracting authority on request, e.g. in the form of a review protocol.

5.7.4.3 Training und Testing

[A: KGAS_4012]

There must be a training strategy that contains input conditions.

[A: KGAS_4081]

The training strategy must contain training end criteria.

[A: KGAS_4082]

The training strategy must contain goals and procedure for the choice of the architecture, including the evaluation against so-called "baseline models".

[A: KGAS_4083]

The training strategy must include goals and procedure for training iterations including validation.

[A: KGAS_4084]

The training strategy must include goals and procedure for modifying the hyper-parameters.

[A: KGAS_4086]

The training strategy must include objectives and a procedure for creating and archiving training baselines including trained weights, sets of hyper-parameters, used metrics, and KPIs.

[A: KGAS_4016]

The test strategy must also include the tests on the target platform and the tests for checking the availability of hardware resources (see also KGAS_3556).

[A: KGAS_4013]

Evidence of the appropriate implementation of the training strategy in accordance with the test strategy must be documented and made available to the contracting authority upon request, e.g. in the form of a review protocol.

[A: KGAS_4014]

It must be made possible to monitor the correct operation of the component before the contracting authority, e.g. through online monitoring or meaningful logging.

5.7.5 Qualification of Tools

[A: KGAS_3117]

Every software-based tool in the software development tool chain must be qualified based on the normative requirements of ISO 26262: 2018 (KGAS_3895) and the requirements of ISO/SAE 21434 (KGAS_4094).
For non functional safety relevant and non safety relevant delivery items, every 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.

[A: KGAS_3481]

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

5.7.6 Verification of Software Construction Requirements

[A: KGAS_3585]

Before each software delivery to the contracting authority, the contractor 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 analysis within the software development process.

[R: KGAS_51]

The contractor must allow the contracting authority to verify the fulfillment of the KGAS with source code analysis, tool-based analysis and other appropriate methods.

[R: KGAS_4000]

The contracting authority may have its analyses (KGAS_51) resulting from the KGAS performed in fully or in partly carried out by third parties.

[R: KGAS_2949]

The contractor must support the analysis (KGAS_51) by providing the source code, corresponding to ECU configurations in the premises and presence of the contractor.

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 contracting authority and contractor.

[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 from respective architecture specifications and software detailed design.

[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 anyone 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]

Possible 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]

It must be possible to execute each test case independently. Mutual dependencies of test cases on each other are not allowed. The setup of test chains to fulfill the test preconditions of lower-level test cases is permitted, provided that this does not otherwise influence the test result.

[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_3361]

If equivalence classes are used to reduce test vectors, at least values from the following types need to be used: lower boundary, upper boundary, a random value between the boundaries.

[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 state of the test object (version of software, hardware, mechanics).

[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 contractor must analyze the associated risks and communicate them to the contracting authority.

5.8.4 Unit Test

[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_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 (ISR), the maximum and average net runtimes (see KGAS_3638) must be determined and documented on the target hardware for each release.

[I: KGAS_3638]

The net runtimes are the runtimes minus the runtime changes caused by the measurements.

[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_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.

5.9 Quality Assurance and Management

5.9.1 Quality Management

[A: KGAS_53]

The process and product quality assurance of the contractor 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 contractor must not be influenced by the project lead.

[A: KGAS_3129]

The quality assurance goals must be measurable.

[A: KGAS_3130]

A goal of quality assurance must be that all work products mandated by the process are created and quality assured in-time and in accordance with the process descriptions.

[A: KGAS_3133]

One goal of quality assurance must be that only quality-assured products are delivered to the contracting authority.

[A: KGAS_2911]

The quality assurance of the contractor must be involved in the release process of the software deliverables (at least by providing a quality statement).

[A: KGAS_2913]

Employees of the contractor's quality assurance department must have the technical qualifications to be able to confirm that the reviews have been carried out properly (content-related and formal).

[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]

All work products mandated by the process must be verified in accordance with the process specifications by reviews.

[A: KGAS_3508]

Verification criteria of a review 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]

In this respect, the reviews must be overseen by the quality assurance department so that a professional conduct of the reviews can be confirmed.

[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 release of the contracting authority.

5.9.3 Verification of Development Processes

[A: KGAS_3477]

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

[A: KGAS_2922]

The contractor must inform the contracting authority of all for contracting authority relevant project risks arising from identified process deficits.

5.10 Configuration Management

[I: 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_3482]

For all work products which defined by process, 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 contractor must communicate all open for contracting authority relevant product problems to the contracting authority at the time of delivery.

[A: KGAS_3410]

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

[A: KGAS_3411]

The contractor's problem evaluation of all product problems must be consistent with the contracting authoritys problem evaluation.

[A: KGAS_3412]

Product problems found in any test level 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 analysis 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 an problem has system effects (A-SPICE Sys level), the contracting authority must be informed within 2 working days.

[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 analysis 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 work products prescribed by the process must be placed under change management according to the process specifications.

5.13 Third Party Software

[R: 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 contractor 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 contractor 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 contracting authority.

[A: KGAS_3443]

The contractor 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.

[R: KGAS_3446]

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

[R: KGAS_3923]

The contractor shall bear sole responsibility for ensuring that the use of the delivered software is permissible in accordance with the contract and the intended use.

[I: KGAS_3924]

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

5.14 Free and Open Source Software

[R: KGAS_3942]

This chapter applies to systems and software (deliverable) which uses Free and Open Source Software (see KGAS_3820).

[A: KGAS_3822]

The use of FOSS is only permitted if the contracting authority is informed in writing by the contractor prior to the use of FOSS in accordance with the processes specified by the contracting authority and the contractor confirms that the use of FOSS is in accordance with the licence. The contractor commits to operate the client's processes and, in particular, to observe possible written consent requirements.

[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 contractor 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.

[R: KGAS_3830]

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

[A: KGAS_4097]

At the time of delivery of the software, the contractor grants the client the sub-licensable and transferable right to modify proprietary software contained in the contractual products for his own use and to carry out reverse engineering for the purpose of troubleshooting (debugging) such processing, insofar as this proprietary software is compatible with the GNU Lesser General Public License v2.1 (LGPL-2.1) licensed software components.

[A: KGAS_4098]

The contractor shall ensure to grant the contracting authority the aforesaid right (KGAS_4097) also with regard to any third party software components.

[I: KGAS_4005]

The FOSS licenses included in the allow list (see KGAS_4003) can usually be used by the contractor without hesitation. The Allow List does not replace any applicable processes and does not exempt from an examination by the client.

[A: KGAS_3801]

The contractor must provide the contracting authority 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 contractor must provide the contracting authority 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 contracting authority, 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 contractor must assure that the delivered software does not contain any license incompatibilities.

[A: KGAS_3824]

The contractor 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 contracting authority, the contractor must provide the contracting authority with the details, materials, documents and results of the analysis carried out (KGAS_3824).

[A: KGAS_3807]

At the request of the contracting authority, the contractor must allow the contracting authority to perform tool-based analysis in premises of the contractor and together with the contractor over the entire source code.

[R: KGAS_3810]

If the contractor implements a technical solution patented or for which a patent has been applied for by the contracting authority, no open source software solutions may be used, whose licenses impede the cost liable licensing of the patent.

[R: KGAS_3827]

The contractor shall bear the sole responsibility for ensuring that the use of the delivered Software is permitted in accordance with the contract and the intended use. The contracting authority fulfills its license obligations towards the end users.

[A: KGAS_3925]

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

5.15 Cybersecurity Relevant Development

5.15.1 General Cybersecurity Requirements

[R: KGAS_3687]

This chapter applies to systems and software (scope of delivery) that have been classified as security relevant by the contracting authority's Brand Security Department.

[A: KGAS_4109]

If the cybersecurity relevance has not yet been identified when the scope of delivery is awarded, this must be actively communicated and requested by the contractor. Until the final identification of the cybersecurity relevance, the contractor must assume that the scope of delivery is cybersecurity relevant (see KGAS_3790).

[A: KGAS_3738]

The contractor must conduct and document cybersecurity risk analysis (Chapter 5.15.5) on system and software level (based on requirements and architecture) for the entire deliverable.

5.15.2 Cybersecurity Terminology

[I: KGAS_3703]

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

[I: KGAS_3704]

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

[I: KGAS_3705]

Cybersecurity goal
A Cybersecurity 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_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_3969]

Cybersecurity Information
All information that is recorded as part of the Monitoring Process and whose cybersecurity relevance (potential weakness) has not yet been classified.

[I: KGAS_3970]

Cybersecurity Event
Cybersecurity information, which is classified as potential weakness (without risk assessment) for the company or its products and requires further steps (CSI Process, Information Assessment, etc.).

[I: KGAS_3971]

Cybersecurity Weakness
Error in the cybersecurity concept (technical, organizational, procedural) of a value.

[I: KGAS_3707]

Cybersecurity Vulnerability
The weakness of an asset, through which one or more threat(s) can be exploited

[I: KGAS_3927]

Cybersecurity Incident
Individual or series of unwanted or unexpected cybersecurity events that prove the exploitation of a vulnerability and could have a significant influence on the security of a component / function (e.g. cause damage to the Value).

[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_3711]

Cybersecurity Requirement
Cybersecurity requirements define requirements for the deliverable specifying properties to prevent or reduce threats.

[I: KGAS_3712]

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

[I: KGAS_3713]

Cybersecurity Concept
The cybersecurity concept is a work product to document cybersecurity relevant aspects of the deliverable, which mitigate threats. The cybersecurity concept comprises especially cybersecurity 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 cybersecurity 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 cybersecurity 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 Cybersecurity Management

[A: KGAS_3725]

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

[A: KGAS_3726]

Cybersecurity activities, resources and work products must be planed and documented (KGAS_3725).

[A: KGAS_3727]

The progress and open points of cybersecurity activities must be updated at least biweekly.

[A: KGAS_3728]

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

[A: KGAS_3729]

Aspects of cybersecurity 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 up-to-dateness 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 contracting authority is to be informed.

[A: KGAS_3733]

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

[A: KGAS_3735]

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

5.15.4 Cybersecurity Risk Assessment and Cybersecurity Concept

[A: KGAS_3736]

The contractor must analyze the cybersecurity related stakeholder requirements regarding understandability and correctness and clarify discrepancies.

[A: KGAS_3853]

Cybersecurity 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 contracting authority. All inconsistencies and uncertainties must be analyzed and resolved accordingly, also referring to existing requirements which until now are not classified as cybersecurity relevant.

[A: KGAS_3973]

As part of the cybersecurity risk analysis all to be protected values have to be identified.

[A: KGAS_3740]

As part of the cybersecurity risk analyses, all processed data must be identified and classified according to cybersecurity protection goals.

[A: KGAS_3741]

As part of the cybersecurity 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 cybersecurity risk analyses.

[A: KGAS_3974]

The contractor must maintain up-to-date threats and measures catalog.

[A: KGAS_3743]

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

[A: KGAS_3744]

The contractor must consider identified and/or specified cybersecurity requirements in the risk analysis.

[A: KGAS_3749]

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

[A: KGAS_3750]

Cybersecurity controls must verifiably lead to cybersecurity requirements.

[A: KGAS_3751]

The cybersecurity concept of the contractor must consider all risks.

5.15.5 Cybersecurity Riskmanagement

[A: KGAS_3745]

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

[A: KGAS_3746]

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

[A: KGAS_3980]

Identified weaknesses must be verifiably managed until the risk has been minimized to an acceptable level.

[A: KGAS_3981]

The acceptance of residual risks must be justified.

5.15.6 Cybersecurity Architectural and Design

[A: KGAS_3753]

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

[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 vulnerabilities. Deviations must be considered and justified in the risk analyzes.

[A: KGAS_3929]

For the analysis of the deliverable 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 contracting authority.

[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 Cybersecurity Implementation

[A: KGAS_3762]

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

[A: KGAS_3772]

The contractor must conduct code analysis, in which the compliance of the KGAS_3762 is checked.

[I: KGAS_3872]

The cybersecurity code analysis may be conducted manually or by a tool.

[A: KGAS_3764]

All deviations from the cybersecurity 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 Cybersecurity Integration and Cybersecurity Verification

[A: KGAS_3769]

The contractor must define and apply cybersecurity test strategies in parallel to the system and software specification.

[A: KGAS_3770]

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

[A: KGAS_3771]

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

5.15.9 Cybersecurity Case

[A: KGAS_3775]

The contractor must provide the cybersecurity case by the "Function Complete" milestone (100% software functionality is implemented).

[A: KGAS_3931]

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

[A: KGAS_3818]

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

[A: KGAS_3776]

The cybersecurity case must include the results of the cybersecurity activities (KGAS_3725) planned in the cybersecurity plan.

[A: KGAS_3817]

The cybersecurity case must include a summary of the results of all risk analysis.

[A: KGAS_3983]

The cybersecurity case must include the adequateness and effectiveness of the cybersecurity measures.

[I: KGAS_3873]

The risk analysis as well as the detailed results of the risk analysis can be viewed within a technical revision at the contractor.

[A: KGAS_3777]

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

[A: KGAS_3819]

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

[A: KGAS_3932]

The cybersecurity case must show that the incident response process to handle identified weaknesses (KGAS_3877) and the active monitoring of the deliverable (KGAS_3784) is established.

[A: KGAS_3984]

The cybersecurity case must be verified by an authority independent of the project.

5.15.10 Cybersecurity activities in post-development (Operations and Maintenance)

[A: KGAS_3874]

The contractor must communicate a central contact point for cybersecurity incident management to the corresponding cybersecurity incident management of the Volkswagen Group (KGAS_3890) at the beginning of the project.

[A: KGAS_3985]

For the exchange of confidential data, the contractor must support the mechanisms and standards used by VW for e-mail encryption.
in accordance with IT Security Guidelines.
(see KGAS_4087 and KGAS_4088)

[A: KGAS_3877]

The contractor must establish a bidirectional response process to deal with identified cybersecurity weaknesses, vulnerabilities and incidents.

[A: KGAS_3784]

For the scope of delivery, the contractor must establish a process for active and continuous monitoring for possible cybersecurity information, events, weaknesses, vulnerabilities and incidents, which includes the points (KGAS_3929) and (KGAS_3930).

[A: KGAS_3785]

In case of cybersecurity information, events, weaknesses, vulnerabilities and incidents the established incident response process (KGAS_3877) must be performed.

[A: KGAS_3933]

If the contractor needs a sub-contractor chain to deliver the product for the contracting authority, contractor has to ensure the effectiveness and incident response capability in their sub-contractor chain.

[A: KGAS_3934]

Cybersecurity weaknesses, vulnerabilities and incidents must be reported to the corresponding cybersecurity incident management of the Volkswagen Group (KGAS_3890) in a reasonable time.

[A: KGAS_3986]

If the notification (KGAS_3934) is made by the contracting authority, the contracting authority can set a deadline.

[A: KGAS_3988]

The contractor must send an acknowledgment of receipt within a reasonable time. The confirmation of receipt must contain a unique reference. The contracting authority and the contractor agree on a unique reference that is used in communication.

[A: KGAS_3939]

Any communication by the contractor regarding cybersecurity weaknesses, vulnerabilities und incidents must be on a need-to-know basis.

[A: KGAS_3936]

External communication that exclusively concerns the contracting authority must be coordinated with the corresponding cybersecurity incident management of the Volkswagen Group (KGAS_3890). This does not apply to communication due to legal requirements. Due to legal requirements, the contracting authority must be informed of the respective cybersecurity incident management through a communication.

[A: KGAS_3937]

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

[A: KGAS_3989]

The risk assessment (KGAS_3937) must contain the following minimum information:
- Description of weakness

- Description of the attack path
- Description of the impact
- Assessment of the probability of occurrence
- Resulting risk related to the scope of delivery of the contractor

[A: KGAS_3990]

In the case there is a request for solution to be provided by the contractor, a detailed documentation must be provided and it should contain:
- Difference between before and after the change in the product (e.g. with software or hardware)
- Description of the tests / scenarios for effectiveness control.
- The test results.

[A: KGAS_3991]

Upon request, the contractor must provide hardware and / or software samples that enable Volkswagen AG to verify the weaknesses in the solution provided.

[A: KGAS_3992]

Identified cybersecurity weaknesses must be taken into account in current developments (see KGAS_3746).

6 References

6.1 Documents of the Volkswagen AG

[I: KGAS_2834]

Formel Q Capability Software: contractor quality capability evaluation guidelines for software development processes [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/

[I: KGAS_4093]

Smart Quality Analytics (SQA): This is the minimum set of project metrics. [Volkswagen AG; Software Quality Assurance]
Available at http://www.vwgroupsupply.com/

[I: KGAS_4116]

ReleaseNotes: Documentation of the scope of delivery and the defined metrics. [Volkswagen AG; Software Quality Assurance]
Available at http://www.vwgroupsupply.com/

[I: KGAS_3966]

Guideline for the data protection requirements for the (further) development of control units with memory function
Available at http://www.vwgroupsupply.com/

[I: KGAS_4003]

Allow List FOSS Licenses KGAS: This list contains FOSS licenses that can usually be used by the contractor without hesitation.
Available at http://www.vwgroupsupply.com/

[I: KGAS_4087]

Guideline Secure Data Exchange
Available at http://www.vwgroupsupply.com/

[I: KGAS_4088]

No. 02.06. Guidelines for contractors
Available at http://www.vwgroupsupply.com/
(for employees of Volkswagen AG see KGAS_4089)

[I: KGAS_4089]

No. 02.02 Guidelines for employees v.5.0 (only valid for Volkswagen AG)
Available at https://volkswagen-net.de/wikis/display/Security/Information+Security+Guidelines

[I: KGAS_4104]

VW SW Source Code Metrics: This is the set of sourcecode metrics. [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]

Current version of Blue-Gold prints of VDA Automotive SPICE® Guidelines.

[I: KGAS_3887]

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

[I: KGAS_4096]

Current version of Blue-Gold prints of VDA Automotive SPICE for Cybersecurity® Guidelines

6.3 Documents of the MISRA

[I: KGAS_2091]

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

6.4 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_3479]

ISO/IEC/IEEE 29119:2022 Software and systems engineering - Software testing

[I: KGAS_3895]

ISO 26262:2018 Road vehicles -- Functional safety

[I: KGAS_4094]

ISO/SAE 21434:2021 Road vehicles – Cybersecurity engineering

[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 Release Notes

[I: KGAS_4055]

The table with the changes to previous version could be found under http://www.vwgroupsupply.com/.

8 Confidentiality Disclosure

[R: KGAS_3488]