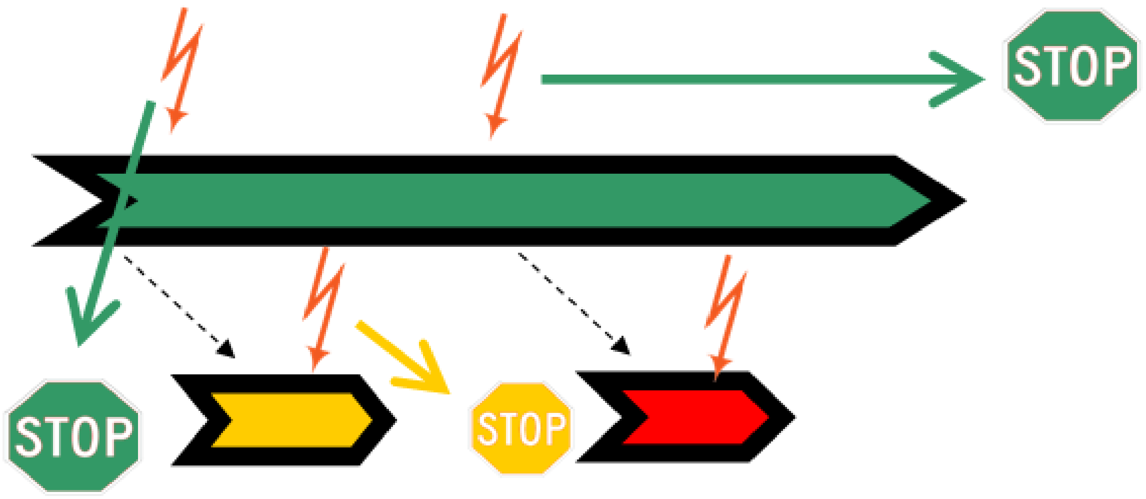
Tool Qualification Plan
for
Testwell CTC++



Version:	0.8
Date:	2014-11-17
Status:	Generic / Adapted / **Presented** / Generated / Reviewed / Final
Author:	Dr. Martin Wildmoser, Dr. Oscar Slotosch
File:	TQP_ManualPart.docx
Size:	40 Pages

Verifysoft
TECHNOLOGY

Revision History:

Version	Date	Status	Author	Change
0.1	2010-02-10	Generic	Slotosch	Template created
0.2	2013-10-10	Adapted	Slotosch	Adapted to Testwell CTC++
0.3	2014-01-23	Presented	R. Bär	Reviewed from Verifysoft
0.4	2014-01-30	Reviewed	Slotosch	Embedded fonts as review feedback
0.8	<generation date>	Generated	Generator Tool	Filled model-dependent parts
0.9	<review date>	Reviewed	<Customer>	Reviewed and updated
1.0	<finalization date>	Final	<Customer>	Finalized document

Contents

1 Scope of this Document

This document describes how the tool Testwell CTC++ is going to be qualified for usage by <Customer>. The tool is a T3 tool according to IEC61508. Therefore the tool needs to be qualified. The applied qualification method is "tool validation".

This document specifies the Testwell CTC++ in detail as it is to be qualified (see Section 5) and the validation goals (see Section 6) that have to be shown for this tool. The validation goals are derived from general IEC61508 requirements and from the potential tool errors identified for the tool's use cases in the Tool Criteria Evaluation Report [TCR], which cannot be detected or prevented with high probability within the development project. The Tool Criteria Evaluation Report is an ISO26262 conformant document that determines the required tool confidence (TCL) by analyzing the potential errors in the used tool features and the possibilities to detect them. This work is integrated into the argumentation as derivation of the validation goals (see Section 6).

The aim of tool validation is to provide sufficient evidence for the absence of these potential errors in the use cases of the tool. For the validation goals a qualification environment (see Section 7) has been created and is applied to the tool in a validation project (see Section 8) resulting in a Tool Qualification Report.

2 Glossary

This section defines technical terms used within this document.

Term	**Definition**
Check	possibility to detect an error
Error	in this document used as "potential error"
Error (model) element	representation of an (potential) error in the model
Feature (model) element	representation of a function in the model.
Function	an elementary or composed function of the tool, that can be required in one or more use-cases, e.g. load, save, "perform" functions
Qualification environment	TAU and tests, a validation suite according to ISO 26262
Restriction	possibility to avoid an error
Safety Guideline	Guideline to mitigate some potential errors of the tool. Modeled as a *Check* or *Restriction*, either in an usual *UseCase* or *Feature* of the *Tool*, or in a separate, virtual *Feature* that can be required (added) by any use case of the same tool. Safety Guidelines are listed in the tool classification report and applied in the tool safety manual.
software off-line support tool (IEC 6108)	According to IEC61508-4-3.2.11: software tool that supports a phase of the software development lifecycle and that cannot directly influence the safety-related system during its run time.
TAU	Test Automation Unit: executes tests for the test suite

TD	Tool Error Detection (TD) probability for a potential error to be detected / avoided in a defined process
	TD1=high detection probability,
	TD2=medium detection probability,
	TD3=low or unknown detection probability

TCL (ISO 26262-8)	Tool Confidence Level (ISO 26262): required confidence in the tool when used in the analyzed tool chain
	TCL1=low confidence required ,
	TCL2=medium confidence required,
	TCL3=high confidence required[1]

TCR	Tool Classification Report, also called tool criteria evaluation report in ISO 26262
Test	Single test with result PASS/FAIL/ABORT
Test Directory	A directory containing one or more test (directories)
Test (model) element	Representation of a test directory in the model including a test description that specifies it
Test Suite	structured set of single tests
Test Plan	list of test (directories) to be executed
Tool	a development tool according to ISO 26262
Tool Chain	a collection of tools, not necessarily forming an input/output chain

[1] Of course once the tool with TCL>1 have been qualified, the TCL can be regarded as existing tool confidence for the qualified ASIL rather than required tool confidence.

Tool classes (IEC 61508-4)	Software off-line support tools are classified into the following tool classes: T1: generates no outputs which can directly or indirectly contribute to the executable code (including data) of the safety related system T2: supports the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software T3: generates outputs which can directly or indirectly contribute to the executable code of the safety related system.
Tool Classification	determination of the required tool confidence level (ISO26262: TCL or IEC 61508: tool classes)
Tool Evaluation	or tool criteria evaluation: see tool classification
TQP	This Tool Qualification Plan
TQR	Tool Qualification Report, extension of this document according to [QKit_UM]
Use-Case	the purpose of using the tool in development process
Use Case (model) element	representation of an use-case in the model
Virtual Feature	A *Feature* is called virtual, if it's virtual attribute is set to true. *Virtual Feature*s are modeled in a *Tool*, but are not implemented in the tool. They are used to model safety guidelines (documents) and can be added flexible as required features to use cases to denote that the use cases follow them. Virtual feature do not have errors.

Note that elements, relations and actions from the model that have a formal semantic in the TCA are written in capital and with italic font, e.g. "*Error* element", or "*Export -> Excel Review*"

3 Qualification Method

The relevant safety standards have comparable approaches to tool qualification. In all standards the goal is to ensure that the tools can not impact the safety of the product, i.e. that all potential errors of the tool are either absent or cannot impact the safety. And all standards do this by a combination of application and installation methods. The application methods are safety guidelines that explain how to use the tool and avoid/detect the potential errors, while the installation methods ensure that the installed tool works as expected, e.g. by testing it to show the absence of the potential errors.

All standards have a classification phase to determine the required confidence into the tool and a qualification phase that provides this confidence or restricts the usage of the tools to confident scenarios. However the classification and qualification methods differ in some details. Nevertheless our qualification approach is suitable for all standards and does not require unnecessary work.

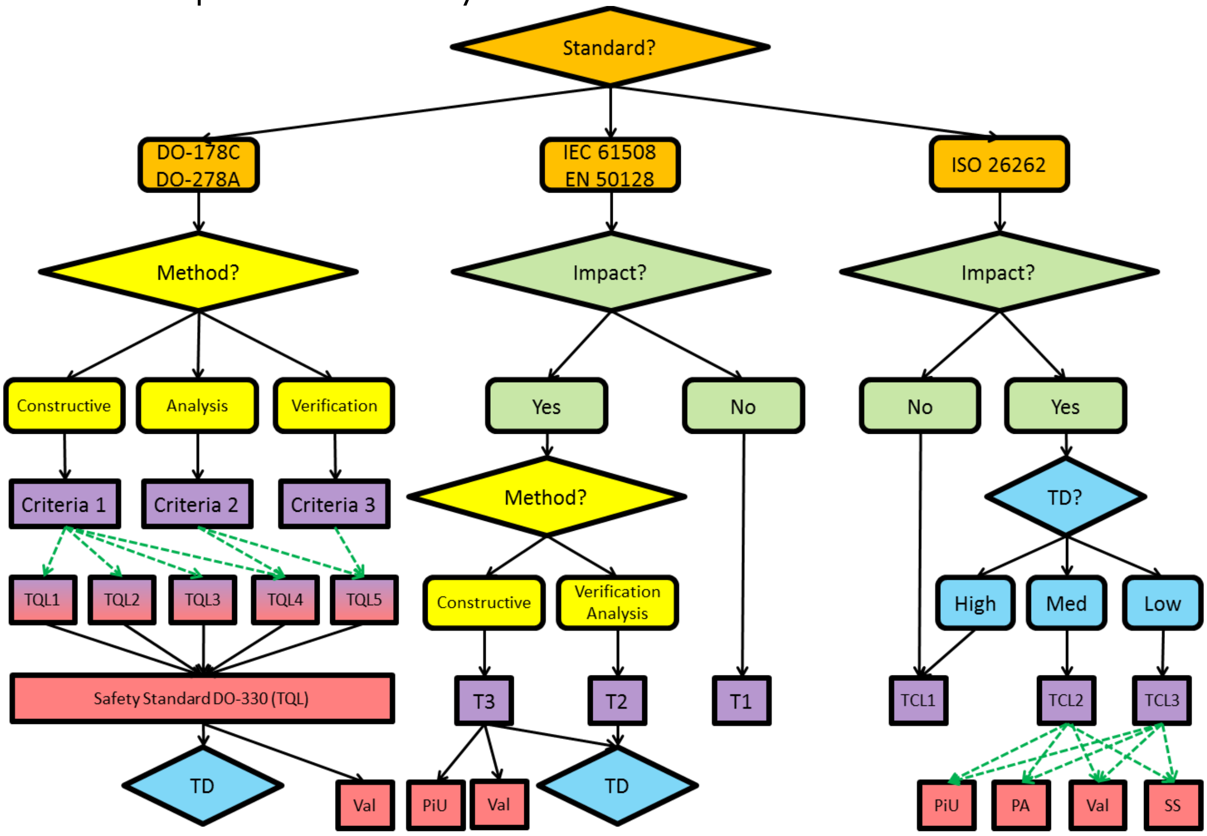


Figure 1: Comparison of Qualification Approaches

Figure 1 gives an overview on the different approaches. The main difference between ISO 26262 and the other standards is that the classification of tools depends on the analysis of the potential errors and their detection, which increases the variability of the classification. The impact and the supported methods/processes are considered in all standards as part of the classification. While the ISO does not differentiate between kinds of the tools the other standards do and classify the tools for constructive methods (e.g. code generators and compilers) as more critical than the other tools verification, automation and analytic tools. The

results of the classification is the confidence needs (represented in pink color in Figure 1). The DO expresses the tool confidence requirement by criteria 1-3 the ISO 26262 as tool confidence levels and IEC 61508 and EN 50128 as tool classes T1-T3. The next step is to derive the qualification methods from the qualification needs of the tool and the criticality of the developed software. ISO 26262 and DO 178C, DO 278A do have tables that map the software criticality to qualification methods, e.g. a validation is required from ISO 26262 for TQL 3 tool in ASIL C and D projects. In DO the qualification methods are determined by the tool qualification level (TQL) that is the interface to the DO-330 and determines the development of the tool, which is a specific qualification method. This criticality dependent selection of qualification methods is depicted in Figure 1 using green dotted lines. The qualification methods differ also. While the DO allows only the development according to the DO-330, a safety standard (SS), the other standards include also a proven in use argumentation (PiU) and a process assessment (PA). Since DO-330 requires also a validation, the validation is the only method that is applicable in every standard. Furthermore the analysis of potential tool errors and their detection (TD) is required in every approach for tools that have impact.
Therefore this classification report contains the determination of the tool confidence need and the analysis of the potential errors and their detection, that belongs to the classification in the ISO 26262.
The qualification method is tool validation by testing the safety relevant parts of the tool. The safety relevant parts are determined by the tool chain analysis together with the determination of the qualification need.

We formalize the tool chain to determine the required confidence using the following model:

- Use case: describes an application scenario of the tool
- Feature: a tool function utilized in use cases
- Potential error: a potential error that could occur during the application of a tool
- Error mitigation: a check or restriction applied during the tool operation phase
- Qualification: a method to show that a tool or a feature satisfies its specified requirements by demonstrating the absence of potential errors.

This tool qualification plan describes the validation of the tool to show the absence of the potential errors.

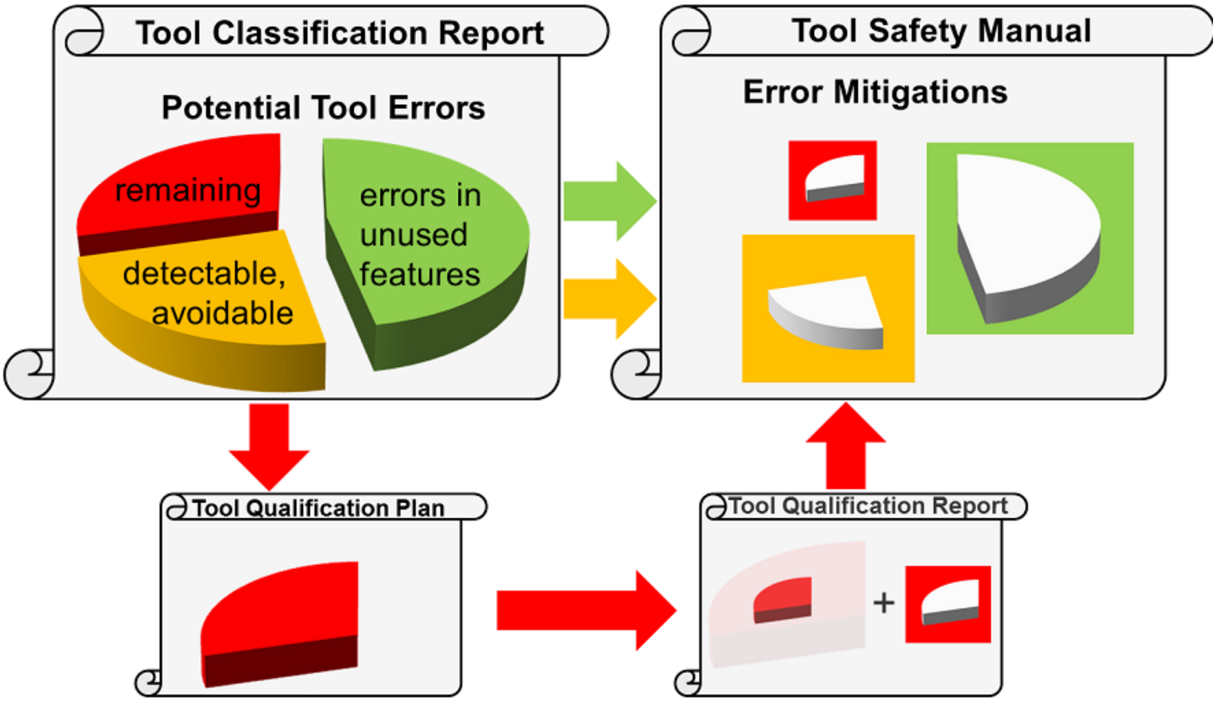


Figure 2: Derivation of Tool Safety Manual Contents

The safety manual for a tool has to contain the mitigations against all potential tool errors that are considered during tool evaluation [TCR]. The errors can be grouped into the three classes (see Figure 2):

- Potential errors in unused features (green in Figure 2)[2]: Using these features is prohibited in the safety manual.
- Potential errors with mitigations: detections and restrictions (yellow in Figure 2): These mechanisms are described in the safety manual, especially if the checks/restrictions have to be triggered by the user of that tool.
- Remaining potential errors (red in Figure 2): Demonstrating their absence has to be the goal of the tool qualification (tool qualification plan). The tool qualification report possibly shows some concrete errors that are instances of the potential error classes. The qualification report contains proposed workarounds for these concrete errors that have to be part of the safety manual (together with the workaround for other already known relevant errors.)

The safety manual / tool application guide therefore has to contain the following information:

- Allowed features and configurations of the tool

[2] Note that the analysis of potential errors in unused functions is not required, but the features need to be identified.

- For potential errors that might occur in required features and that are not excluded by tool qualification: Requirements to apply checks and restrictions to mitigate potential tool errors
- Workarounds for known errors and errors found during qualification
- Other information required by the standards to identify the tool exactly (version, configuration, etc.).

The tool qualification plan has to ensure that the identified potential errors of Testwell CTC++ that are not detectable / avoidable cannot occur. This is done by applying a validation suite in a systematic way that shows the absence of these potential errors.

Since the tool Testwell CTC++ shall be qualified using validation accruing to this qualification plan we have to provide the following documents:

- *Test Plan*: to plan the execution of tests
- *Test Report*: contains the test results
- Test Automation Unit Manual: To execute the planned tests cases correctly
- *Test suite validation and verification documents (plan and report)*: to ensure that the test suite shows the absence of the potential errors if passed successfully

The documents that depend on the model are typed using *cursive font*.

In the case that the model and the validation suite needs to be extended and new test cases need to be produced and validated, the following documents are required, or need to be extended:

- *Test specifications* including a test strategy to show the absence of the absence of the potential errors.
- *Test suite V&V plan & report*

The test specification is part of the model (descriptions). The test suite needs validation against the potential errors of the model and verification against the implementation using a review. This quality process creates the confidence into the effectiveness of the test suite. The V&V documents for the test suite are contained in the qualification kit to demonstrate the confidence to the user. If the test suite is extended these documents shall also be extended.

Figure 3 shows the relation between the documents and their variability, i.e. which are constant and which depend on the use case: It describes how to derive the safety manual by a validation suite that consist of tests that show the absence of the identified critical errors in the tool evaluation report. Depending on the used features of the tool and the applied mitigation measures this set of errors might vary. For every required test (or group of tests) that show the absence of one or more errors there needs to be a test specification (including a test strategy) that explains how the absence of the errors is ensured if the tests pass. The tests in the test suite need to be validated to conform to the test specification. This is planned in a V&V plan of the kit and documented in the V&V report. Having a V&V report is the prerequisite for applying the validation suite to a use case. In Figure 3 the use case specific documents are in a green/inner, dashed box where the contents of the qualification kit are in the outer/blue box. Of course the sequence of creating the documents (indicated by the sequence numbers) starts with the non use-case specific documents in the qualification kit. The tool qualification is planned in the qualification plan and requires executing tests (planned in the test plan) using the test automation unit manual. The test results are documented in a test report which is than analyzed and documented in the qualification report.

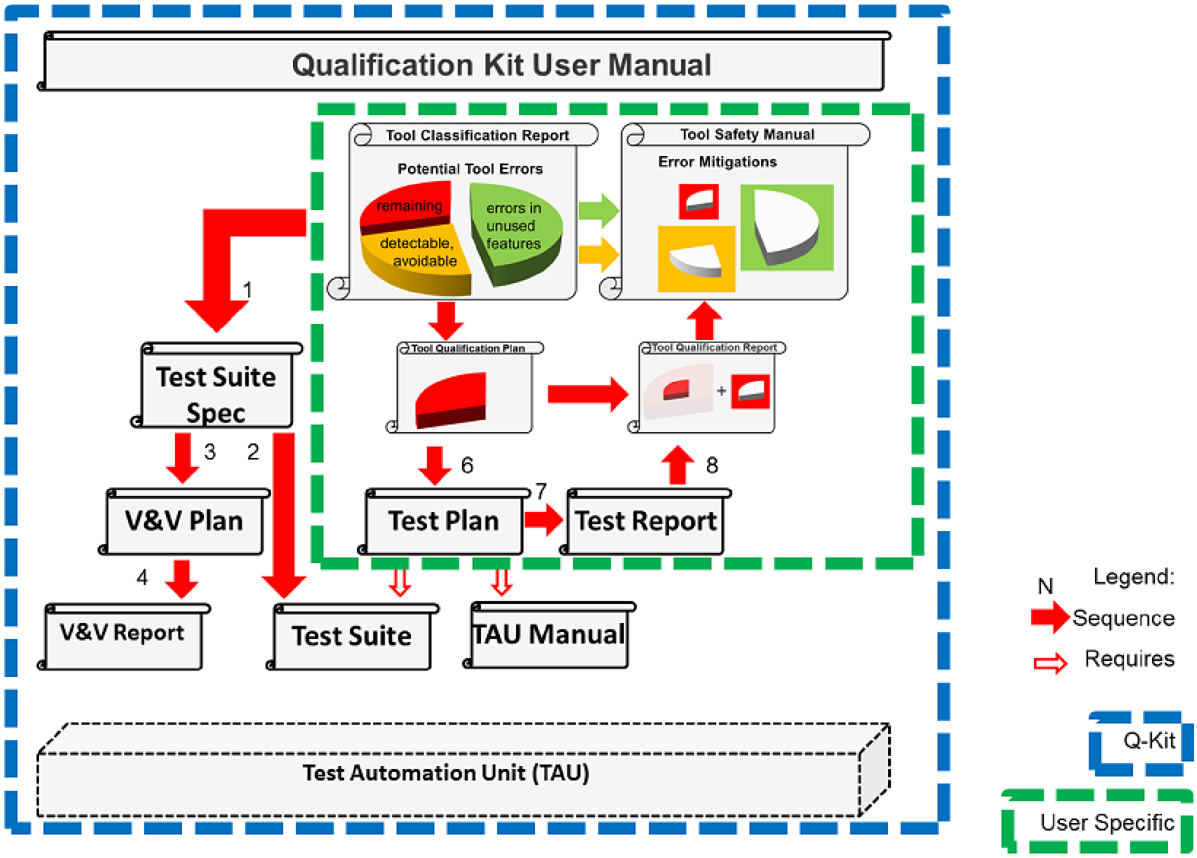


Figure 3: Documentation Plan

There are many documents in Figure 3 that are required and that need to be adapted depending on the user's process captured in the qualification model by selecting the required tool features and the executed mitigations during the process. The use case specific parts in the user specific documents are generated from the qualification model.

4 Validation Process

The validation process consists of the following phases:

1. Update tool qualification plan
2. Validate qualification environment
3. Run validation suite
4. Analyse test results
5. Write qualification report

These phases are explained in detail in the following subsections.

4.1 Update Tool Qualification Plan

The tool qualification plan needs to be updated and reviewed. The following information needs to be updated or checked for updates:

- Tool Definition (see Section 5), including
    - General facts like version and release number
    - Use Cases according to the classification results
    - Features according to the classification results
- Test Requirements, see Section 6.5.2 according to the classification results
- Validation Project, see Section 8.

This section shall remain as it is (even after update) in order to document the process.

Updating the qualification plan can be done by reusing the tool evaluation results [TCR] as described in the qualification kit user manual [QKit_UM] for the Testwell CTC++.

4.2 Validate Qualification Environment

Before the application of the validation suite starts the qualification environment is validated in order to ensure that it fulfills the goals of this qualification plan. The qualification environment consists of the test automation unit (TAU) and the test cases.

The validation is split into two steps:

- The validation that the test cases are suitable for showing the absence of the errors. This is done separately in the test suite V&V activities that are planned in a test suite V&V plan and documented in the test suite V&V report. This has been done during the development of the test suite.
- The validation that the qualification environment is working correctly and that the Testwell CTC++ is used in the right configuration for the qualification. The result of these validation steps shall be documented in the tool qualification report. These steps are:

- Install and configure the tool according to the tool definition. Try out the tool on a few examples to find out if the configuration is correct, e.g. by checking the release number.
- Review the Tool Qualification Plan. The following points should be checked.
    - Are the validation goals adequate and complete?
    - Are the tests available for all qualification goals?
    - Have the tests been verified by review to be suitable for showing the absence of the relevant potential errors?
    - Is the validation project plan realistic? Are all resources available? Are the milestone dates acceptable?

4.3 Run Validation Suite

The validation suite is applied to the tool to be qualified in a so called golden run, which involves the following activities:

a) Create a protocol and document every of the following steps.
b) Get a test computer and check if it provides the environment specified for the tool to be qualified in this Tool Qualification Plan.
c) Install precautions ensuring that the golden run runs without disturbations, e.g. put an information sign in front of the computer.
d) Install the Testwell CTC++ as defined in Section 5.
e) Install the TAU and the test suite according to [TAU_UG].
f) Check the correct installation and configuration for the tool and the TAU as described in [TAU_UG].
g) Generate a test plan from the qualification model with the test cases to execute as described in [QKit_UM]
h) Execute the test plan with the TAU as described in [TAU_UG].
i) Check the correct and complete execution. Does the log file indicate the successful completion of the test run? Do the progress messages indicate the execution of the correct version of the test cases and tools? Have all tests produced a verdict?
j) Archive the files produced for every test case during the test run. This involves the test reports and also intermediate files produced during the run. The latter may be helpful for the analysis of the test results.

End criterion: All tests have been run and archived.

4.4 Analyze Test Results

All test cases with result FAIL or ABORT need to be analyzed. Before analysis starts an analysis work list should be created containing the names of all test cases that need to be analyzed. There shall be columns for the analysis status ("NotAnalyzed", "Analyzed"), the validity ("Valid", "Invalid") and reproducibility ("Reproducible", "NotReproducible") of the test case.

Until not all tests in the list are "Analyzed" and "Vaild": pick any test case with status "NotAnalyzed" and follow this work flow:

a) Check the log and check if the test has been run correctly (correct configuration?, errors?, warnings?). If there are problems due to the test case or the test run, mark the test case as "Invalid" in the analysis work list and continue with e). Otherwise mark it as "Valid".
b) Save the test files and repeat the test run for the test case. Check if the FAIL/ABORT result can be reproduced. If the result is reproducible continue with e), otherwise with c).
c) Mark the test as "NotReproducible" in the analysis work list.
d) Analyse potential causes for the deviation. Typically anomalous operating conditions or non-deterministic behavior in the tool are the cause for such tool failures. Continue with g).
e) Mark the test as "Reproducible" in the analysis work list.
f) Try to find a plausible explanation (root-cause) for the deviation. This may involve variations of the test case.
g) Mark the test case as "Analyzed" in the analysis work list.

The result of this process is an explanation for all invalid tests and a list of findings with explanations.

4.5 Write Tool Qualification Report

Writing the Tool Qualification Report involves the following activities:

a) Discuss all analysis findings. Let clearness and plausibility of the findings be confirmed by tool experts.
b) Suggest, measures to detect or avoid the erroneous tool outputs, eventually together with Verifysoft
c) Write the tool qualification report.

The tool qualification report contains:

a) the protocol for the golden validation run.
b) the verdicts and/or a statistical summary of the tests.
c) the discussed analysis results in an abstracted form.
d) the measures for error detection or avoidance if available.
e) a clear decision on whether the tool is qualified for use or not.

The tool qualification report can be created by updating this document as follows:

- change the title to tool qualification report
- update the Section 1. Scope of this Document.
- the following sections shall be added:

9) Test Results

- Description of validation process results (see Section 4.2)
- Description of the test data and log files
- Description of the test results for all tests

10) Test Analysis

- Analysis of failed and aborted tests
- Mitigations for failed test

11) Qualification Summary

- Recommendation on the usage of the tool Testwell CTC++

And the document shall be reviewed.

5 [generated tool]

6 Validation Goals

This section describes the validation goals derived from [ISO26262], [IEC61508] and [EN50128].

6.1 ISO 26262 Part 8, Section 11.4

In part 8, Section 11 ("Confidence in the use of software tools") of [ISO26262] the following requirements are described. Many of them are covered by this tool qualification and the tool classification in [TCR] and have a VS-ISO-ID. The others are argued to be not applicable for this qualification and shall be covered by the surrounding safety process.

Req ID (if applicable)	Source	Requirement Text
Is covered by the tool chain analysis in [TCR]	11.4.1.1	If the safety lifecycle incorporates the use of a software tool for the development of a system, or its hardware or software elements, such that activities or tasks required by ISO 26262 rely on the correct functioning of a software tool, and where the relevant outputs of that tool are not examined or verified for the applicable process step(s), such software tools shall comply with the requirements of this clause.
NA, since no independent input is used	11.4.2.1	If the confidence level evaluation or qualification of a software tool is performed independently from the development of a particular safety-related item or element, the validity of this predetermined tool confidence level or qualification shall be confirmed, in accordance with ISO 26262-2:2011, Table 1, prior to the software tool being used for the development of a particular safety-related item or element.
Safe tool usage is described in the [TSM]	11.4.3.1	When using a software tool, it shall be ensured that its usage, its determined environmental and functional constraints and its general operating conditions comply with its evaluation criteria or its qualification.
See sub-items	11.4.4.1	The usage of a software tool shall be planned, including the determination of:
See [TSM]	11.4.4.1.a	the identification and version number of the software tool
See [TSM]	11.4.4.1.b	the configuration of the software tool
See [TSM]	11.4.4.1.c	the use cases of the software tool
See [TSM]	11.4.4.1.d	the environment in which the software tool is executed
See [TSM]	11.4.4.1.e	the maximum ASIL of all the safety requirements, allocated to the item or the element that can be violated, if the software tool is malfunctioning and producing corresponding erroneous output
VS-ISO-10	11.4.4.1.f	the methods to qualify the software tool, if required based on the determined level of confidence
See sub-items	11.4.4.2	To ensure the proper evaluation or usage of the software tool, the following information shall be available:
See [TCR]	11.4.4.2.a	a description of the features, functions and technical properties of the software tool
See [TSM]	11.4.4.2.b	the user manual or other usage guides, if applicable,

See [TSM]	11.4.4.2.c	a description of the environment required for its operation
See [TSM]	11.4.4.2.d	a description of the expected behaviour of the software tool under anomalous operating conditions, if applicable
See [TSM]	11.4.4.2.e	a description of known software tool malfunctions and the appropriate safeguards, avoidance or work-around measures, if applicable, and
See [TSM]	11.4.4.2.f	the measures for the detection of malfunctions and the corresponding erroneous output of the software tool identified during the determination of the required level of confidence for this software tool
See sub-items	11.4.5.1	The description of the usage of a software tool shall contain the following information
See [TCR]	11.4.5.1.a	the intended purpose
See [TCR]	11.4.5.1.b	the inputs and expected outputs, and
See [TCR]	11.4.5.1.c	the environmental and functional constraints, if applicable
See sub-items	11.4.5.2	The intended usage of the software tool shall be analysed and evaluated to determine
See sub-items	11.4.5.2.a	the possibility that a malfunction of a particular software tool can introduce or fail to detect errors in a safety-related item or element being developed. This is expressed by the classes of Tool Impact (TI):
See [TCR]	11.4.5.2.a.1	TI1 shall be selected when there is an argument that there is no such possibility
See [TCR]	11.4.5.2.a.2	TI2 shall be selected in all other cases
See sub-items	11.4.5.2.b	the confidence in measures that prevent the software tool from malfunctioning and producing corresponding erroneous output, or in measures that detect that the software tool has malfunctioned and has produced corresponding erroneous output. This is expressed by the classes of Tool error Detection (TD):
See [TCR]	11.4.5.2.b.1	TD1 shall be selected if there is a high degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected;
See [TCR]	11.4.5.2.b.2	TD2 shall be selected if there is a medium degree of confidence that a malfunction and its corresponding erroneous output will be prevented or detected
See [TCR]	11.4.5.2.b.3	TD3 shall be selected in all other cases
See [TCR]	11.4.5.3	If the correct selection of TI or TD is unclear or doubtful, TI and TD should be estimated conservatively
See [TCR]	11.4.5.4	If a software tool is used for the tailoring of the development process in such a way that activities or tasks required by ISO 26262 are omitted, TD2 shall not be selected
See [TCR]	11.4.5.5	Based on the values determined for the classes of TI and TD, the required software tool confidence level shall be determined according to Table 3 of [ISO26262]-8
VS-ISO-33	11.4.6.1	For the qualification of software tools classified at TCL3, the methods listed in Table 4 of [ISO26262]-8 shall be applied. For the qualification of software tools classified at TCL2, the methods listed in Table 5 of [ISO26262]-8 shall be applied. A software tool classified at TCL1 needs no qualification methods
See sub-items	11.4.6.2	The qualification of the software tool shall be documented including the following
VS-ISO-35	11.4.6.2.a	the unique identification and version number of the

		software tool
VS-ISO-36	11.4.6.2.b	the maximum Tool Confidence Level for which the software tool is classified together with a reference to its evaluation analysis
VS-ISO-37	11.4.6.2.c	the pre-determined maximum ASIL, or specific ASIL, of any safety requirement which might be violated if the software tool is malfunctioning and produces corresponding erroneous output
VS-ISO-38	11.4.6.2.d	the configuration and environment for which the software tool is qualified,
VS-ISO-39	11.4.6.2.e	the person or organization who carried out the qualification,
VS-ISO-40	11.4.6.2.f	the methods applied for its qualification in accordance with 11.4.6.1,
VS-ISO-41	11.4.6.2.g	the results of the measures applied to qualify the software tool, and
VS-ISO-42	11.4.6.2.h	the usage constraints and malfunctions identified during the qualification, if applicable
NA, since this method is not applied	11.4.7	Increased confidence from use
NA, since this method is not applied	11.4.8	Evaluation of the tool development process
See sub-items	11.4.9	Validation of the software tool
See 11.4.9.2	11.4.9.1	If the method “Validation of the software tool” according to Table 4 or Table 5 in [ISO26262]-8 is applied for the qualification of a software tool, the requirements of this subclause shall be complied with.
See sub-items	11.4.9.2	The validation of the software tool shall meet the following criteria
VS-ISO-48	11.4.9.2.a	the validation measures shall demonstrate that the software tool complies with its specified requirements
VS-ISO-49	11.4.9.2.b	the malfunctions and their corresponding erroneous outputs of the software tool occurring during validation shall be analysed together with information on their possible consequences and with measures to avoid or detect them, and
VS-ISO-50	11.4.9.2.c	the reaction of the software tool to anomalous operating conditions shall be examined
See sub-items	11.4.10	Confirmation review of qualification of a software tool This subclause applies to ASILs (B), C, D, in accordance with 4.3 in [ISO26262]-8. The confidence in the use of the software tool shall be evaluated in accordance with ISO 26262-2:2011 Table 1 to ensure:
See Reviews of [TCR]	11.4.10.a	the correct evaluation of the required level of confidence in the software tool, and
VS-ISO-53	11.4.10.b	the appropriate qualification of the software tool in accordance with its required level of confidence.

6.2 IEC 61508 Part 3, Section 7.4.4

In part 3, Section 7.4.4 ("Requirements for support tools, including programming languages") of [IEC61508] the following requirements are described. Many of them are covered by this tool qualification and have a VS-IEC-ID. The others are argued to be not applicable for this qualification and shall be covered by the surrounding safety process.

Req ID (if applicable)	Source	Requirement Text
NA: tool is a offline-tool	7.4.4.1	A software on-line support tool shall be considered to be a software element of the safety related system
NA: tool selection is out of scope	7.4.4.2	Software off-line support tools shall be selected as a coherent part of the software development activities.
VS-IEC-3	7.4.4.3	The selection of the off-line support tools shall be justified
VS-IEC-4	7.4.4.4	All off-line support tools in classes T2 and T3 shall have a specification or product documentation which clearly defines the behaviour of the tool and any instructions or constraints on its use.
VS-IEC-5	7.4.4.5	An assessment shall be carried out for offline support tools in classes T2 and T3 to determine the level of reliance placed on the tools, and the potential failure mechanisms of the tools that may affect the executable software. Where such failure mechanisms are identified, appropriate mitigation measures shall be taken.
VS-IEC-6	7.4.4.6	For each tool in class T3, evidence shall be available that the tool conforms to its specification or documentation. Evidence may be based on a suitable combination of history of successful use in similar environments and for similar applications (within the organization or other organizations), and of tool validation as specified in 7.4.4.7.
VS-IEC-7	7.4.4.7	The results of tool validation shall be documented covering the following results: a) a chronological record of the validation activities; b) the version of the tool product manual being used; c) the tool functions being validated; d) tools and equipment used; e) the results of the validation activity; the documented results of validation shall state either that the software has passed the validation or the reasons for its failure; f) test cases and their results for subsequent analysis; g) discrepancies between expected and actual results.
VS-IEC-8	7.4.4.8	Where the conformance evidence of 7.4.4.6 is unavailable, there shall be effective measures to control failures of the executable safety related system that result from faults that are attributable to the tool.
NA: integration is out scope	7.4.4.9	The compatibility of the tools of an integrated toolset shall be verified.
VS-IEC-10	7.4.4.10	To the extent required by the safety integrity level, the

		software or design representation (including a programming language) selected shall: a) have a translator which has been assessed for fitness for purpose including, where appropriate, assessment against the international or national standards; b) use only defined language features; c) match the characteristics of the application; d) contain features that facilitate the detection of design or programming mistakes; e) support features that match the design method.
NA: VS-IEC-10 is fully satisfied	7.4.4.11	Where 7.4.4.10 cannot be fully satisfied, the fitness for purpose of the language, and any additional measures which address any identified shortcomings of the language shall be justified.
VS-IEC-12	7.4.4.12	Programming languages for the development of all safety-related software shall be used according to a suitable programming language coding standard.
NA: out of scope	7.4.4.13	A programming language coding standard shall specify good programming practice, proscribe unsafe language features (for example, undefined language features, unstructured designs, etc.), promote code understandability, facilitate verification and testing, and specify procedures for source code documentation. Where practicable, the following information shall be contained in the source code: a) legal entity (for example company, author(s), etc.); b) description; c) inputs and outputs; d) configuration management history.
VS-IEC-14	7.4.4.14	Where automatic code generation or similar automatic translation takes place, the suitability of the automatic translator for safety-related system development shall be assessed at the point in the development lifecycle where development support tools are selected.
NA: usage constraints on the tool that are out of validation scope	7.4.4.15	Where off-line support tools of classes T2 and T3 generate items in the configuration baseline, configuration management shall ensure that information on the tools is recorded in the configuration baseline. This includes in particular: a) the identification of the tool and its version; b) the identification of the configuration baseline items for which the tool version has been used; c) the way the tool was used (including the tool parameters, options and scripts selected) for each configuration baseline item.
NA: out of scope of tool qualification	7.4.4.16	Configuration management shall ensure that for tools in classes T2 and T3, only qualified versions are used.
NA: out of scope of tool qualification	7.4.4.17	Configuration management shall ensure that only tools compatible with each other and with the safety-related system are used.
NA: tool validation of each new version is	7.4.4.18	Each new version of off-line support tool shall be qualified. This qualification may rely on evidence provided for an earlier version if sufficient evidence is provided that:

done by automated testing and no other arguments are required		a) the functional differences (if any) will not affect tool compatibility with the rest of the toolset; and b) the new version is unlikely to contain significant new, unknown faults.
NA: out of scope of tool qualification	7.4.4.19	Depending on the nature of the software development, responsibility for conformance with 7.4.4 can rest with multiple parties. The division of responsibility shall be documented during safety planning (see Clause 6 of IEC 61508-1).

Table 1 Validation Suite Requirements from IEC 61508 part 3

6.3 EN 50128, Section 6.7.4

In Section 7.4.4 ("Support tools and languages") of [EN50128] the following requirements are described. Many of them are covered by this tool qualification and have a VS-EN-ID. The others are argued to be not applicable for this qualification and shall be covered by the surrounding safety process.

Req ID (if applicable)	Source	Requirement Text
NA, since Software tool planning is out of scope of this report	6.7.4.1	Software tools shall be selected as a coherent part of the software development activities
See [TCR]	6.7.4.2	The selection of the tools in classes T2 and T3 shall be justified (see 7.3.4.12). The justification shall include the identification of potential failures which can be injected into the tools output and the measures to avoid or handle such failures.
See [TSM]	6.7.4.3	All tools in classes T2 and T3 shall have a specification or manual which clearly defines the behaviour of the tool and any instructions or constraints on its use.
See sub-items	6.7.4.4	For each tool in class T3, evidence shall be available that the output of the tool conforms to the specification of the output or failures in the output are detected. Evidence may be based on the same steps necessary for a manual process as a replacement for the tool and an argument presented if these steps are replaced by alternatives (e. g. validation of the tool). Evidence may also be based on
NA, this method is not used in this qualification	6.7.4.4.a	a suitable combination of history of successful use in similar environments and for similar applications (within the organization or other organizations),
See VS-EN-11 to -17	6.7.4.4.b	tool validation as specified in 6.7.4.5
See [TCR] and [TSM]	6.7.4.4.c	diverse redundant code which allows the detection and control of failures resulting in faults introduced by a

		tool,
See [TCR]	6.7.4.4.d	compliance with the safety integrity levels derived from the risk analysis of the process and procedures including the tools
See [TCR]	6.7.4.4.e	other appropriate methods for avoiding or handling failures introduced by tools
See sub-items	6.7.4.5	The results of tool validation shall be documented covering the following results
VS-EN-11	6.7.4.5.a	a record of the validation activities
VS-EN-12	6.7.4.5.b	the version of the tool manual being used
VS-EN-13	6.7.4.5.c	the tool functions being validated
VS-EN-14	6.7.4.5.d	tools and equipment used
VS-EN-15	6.7.4.5.e	the results of the validation activity; the documented results of validation shall state either that the software has passed the validation or the reasons for its failure
VS-EN-16	6.7.4.5.f	test cases and their results for subsequent analysis
VS-EN-17	6.7.4.5.g	discrepancies between expected and actual results
VS-EN-18	6.7.4.6	Where the conformance evidence of 6.7.4.4 is unavailable, there shall be effective measures to control failures of the executable safety related software that result from faults that are attributable to the tool
See sub-items	6.7.4.7	The software or design representation (including a programming language) selected shall
VS-EN-20	6.7.4.7.a	have a translator which has been evaluated for fitness for purpose including, where appropriate, evaluated against the international or national standards
VS-EN-21	6.7.4.7.b	match the characteristics of the application
See [TSM]	6.7.4.7.c	contain features that facilitate the detection of design or programming errors
VS-EN-23	6.7.4.7.d	support features that match the design method.
NA this is just an explanation	6.7.4.7.1	A programming language is one of a class of representations of software or design. A Translator converts a software or design representation (e.g. text or a diagram) from one abstraction level to another level. Examples of Translators include: design refinement tools, compilers, assemblers, linkers, binders, loaders and code generation tools.
VS-EN-25	6.7.4.7.2	The evaluation of a Translator may be performed for a specific application project, or for a class of applications. In the latter case all necessary information on the tool regarding the intended and appropriate use of the tool shall be available to the user of the tool. The evaluation of the tool for a specific project may then be reduced to checking general suitability of the tool for the project and compliance to the “specification or manual” (i.e. proper use of the tool). Proper use might include additional verification activities within the specific project.
VS-EN-26	6.7.4.7.3	A validation suite may be used to evaluate the fitness for purpose of a Translator according to defined criteria, which shall include functional and non-functional requirements. For the functional Translator requirements, dynamic testing may be a main validation technique. If possible an automatic testing suite shall be used.
Software	6.7.4.8	Where 6.7.4.7 cannot be fully satisfied, the fitness for

selection is not in the scope of tool qualification		purpose of the language, and any additional measures which address any identified shortcomings of the language shall be justified and evaluated.
See [TCR]	6.7.4.9	Where automatic code generation or similar automatic translation takes place, the suitability of the automatic Translator for safety-related software development shall be evaluated at the point in the development lifecycle where development support tools are selected
Configuration Management is not in the scope of tool qualification	6.7.4.10	Configuration management shall ensure that for tools in classes T2 and T3, only justified versions are used.
No support for delta qualifications (classification in [TCR] may be reused)	6.7.4.11	Each new version of a tool that is used shall be justified (see Table 1 in [EN50128]). This justification may rely on evidence provided for an earlier version if sufficient evidence is provided that
VS-EN-31	6.7.4.12	The relation between the tool classes and the applicable sub-clauses is defined within Table 1 in [EN50128].

Table 2 Validation Suite Requirements from EN 50128

6.4 IEC 62304 & FDA Validation

While the most important medical standard the [IEC62304] "Medical device software –Software life cycle processes" has no special chapter on tool qualification, the most common interpretation from the US Food and Drug Administration (FDA) set common the interpretation of the standard in [FDA2002]. There are many requirements that require a safety process to satisfy them all. We have made a short selection from the perspective of tool qualification that covers at least one requirement of every section in [FDA2002] and shows that our approach (including the tool code coverage measurement) is a minimum requirement to satisfy the validation requirement.

For a complete medical tool qualification that is accepted by an authority also other evidences might be required from the certification authority that are out of the scope of this qualification kit (e.g. audits of the tool provider to verify the existence of coding guidelines or developer tests).

The following requirements are in [FDA2002]:

- FDA-2.1.3 (APPLICABILITY) "Software used in the production of a device (e.g., programmable logic controllers in manufacturing equipment)"
- FDA-2.1.4 (APPLICABILITY) "Software used in implementation of the device manufacturer's quality system"
- FDA-2-4-3 (REGULATORY REQUIREMENTS FOR SOFTWARE VALIDATION): "This requirement applies to any software used to automate device design, testing, component acceptance,

manufacturing, labeling, packaging, distribution, complaint handling, or to automate any other aspect of the quality system."

- FDA-3-1-2-4 (DEFINITION Validation): "software validation is a matter of developing a "level of confidence" that the device meets all requirements and user expectations for the software"
- FDA-3-2-1 (SOFTWARE DEVELOPMENT AS PART OF SYSTEM DESIGN): "software validation must be considered within the context of the overall design validation for the system"
- FDA-4-1 (REQUIREMENTS): "A documented software requirements specification provides a baseline for both validation and verification"
- FDA-4-2 (DEFECT PREVENTION): "In order to establish that confidence, software developers should use a mixture of methods and techniques to prevent software errors and to detect software errors that do occur."
- FDA-4-3 (TIME AND EFFORT): "The final conclusion that the software is validated should be based on evidence collected from planned efforts conducted throughout the software lifecycle"
- FDA-4-4 (SOFTWARE LIFE CYCLE): "the software life cycle contains specific verification and validation tasks that are appropriate for the intended use of the software"
- FDA-4-5 (PLANS): "The software validation process is defined and controlled through the use of a plan."
- FDA-4-6 (PROCEDURES): "The procedures should identify the specific actions or sequence of actions that must be taken to complete individual validation activities, tasks, and work items"
- FDA-4-7 (SOFTWARE VALIDATION AFTER A CHANGE): "Design controls and appropriate regression testing provide the confidence that the software is validated after a software change."
- FDA-4-8 (VALIDATION-COVERAGE): "The selection of validation activities, tasks, and work items should be commensurate with the complexity of the software design and the risk associated with the use of the software for the specified intended use."
- FDA-4-9 (INDEPENDENCE OF REVIEW):"Validation activities should be conducted using the basic quality assurance precept of "independence of review."
- FDA-4-10 (FLEXIBILITY AND RESPONSIBILITY): "The device manufacturer has flexibility in choosing how to apply these validation principles, but retains ultimate responsibility for demonstrating that the software has been validated"
- FDA-5-1 (SOFTWARE LIFE CYCLE ACTIVITIES): "Activities in a typical software life cycle model include the following":
    - Quality Planning
    - System Requirements Definition
    - Detailed Software Requirements Specification
    - Software Design Specification
    - Construction or Coding
    - Testing
    - Installation

  - Operation and Support
  - Maintenance
  - Retirement
- FDA-5-2-1 (Quality Planning): “The plan should include”:
  - The specific tasks for each life cycle activity;
  - Enumeration of important quality factors (e.g., reliability, maintainability, and usability);
  - Methods and procedures for each task;
  - Task acceptance criteria;
  - Criteria for defining and documenting outputs in terms that will allow evaluation of their conformance to input requirements;
  - Inputs for each task;
  - Outputs from each task;
  - Roles, resources, and responsibilities for each task;
  - Risks and assumptions; and
  - Documentation of user needs
- FDA-5-2-2 (Requirements): „Typical software requirements specify the following:”
  - All software system inputs;
  - All software system outputs;
  - All functions that the software system will perform;
  - All performance requirements that the software will meet, (e.g., data throughput, reliability, and timing);
  - The definition of all external and user interfaces, as well as any internal software-to-system interfaces;
  - How users will interact with the system;
  - What constitutes an error and how errors should be handled;
  - Required response times;
  - The intended operating environment for the software, if this is a design constraint (e.g.,hardware platform, operating system);
  - All ranges, limits, defaults, and specific values that the software will accept; and
  - All safety related requirements, specifications, features, or functions that will be implemented in software.
- FDA-5-2-2-1 (Safety Requirements): “Software safety requirements are derived from a technical risk management process that is closely integrated with the system requirements development process”, “The consequences of software failure should be evaluated, along with means of mitigating such failures”.
- FDA-5-2-3 (Design): “The software design specification should include:
  - Software requirements specification, including predetermined criteria for acceptance of the software;
  - Software risk analysis;
  - Development procedures and coding guidelines (or other programming procedures);

  - Systems documentation (e.g., a narrative or a context diagram) that describes the systems context in which the program is intended to function, including the relationship of hardware, software, and the physical environment;
  - Hardware to be used;
  - Parameters to be measured or recorded;
  - Logical structure (including control logic) and logical processing steps (e.g., algorithms);
  - Data structures and data flow diagrams;
  - Definitions of variables (control and data) and description of where they are used;
  - Error, alarm, and warning messages;
  - Supporting software (e.g., operating systems, drivers, other application software);
  - Communication links (links among internal modules of the software, links with the supporting software, links with the hardware, and links with the user);
  - Security measures (both physical and logical security); and
  - Any additional constraints not identified in the above elements.
- FDA-5-2-3-1 (Design-Traceability): "A traceability analysis should be conducted to verify that the software design implements all of the software requirements. As a technique for identifying where requirements are not sufficient, the traceability analysis should also verify that all aspects of the design are traceable to software requirements."
- FDA-5-2-3-2 (Design Review): "a Formal Design Review should be conducted to verify that the design is correct, consistent, complete, accurate, and testable"
- FDA-5-2-4 (Construction or Coding): "Decisions on the selection of programming languages and software build tools (assemblers, linkers, and compilers) should include consideration of the impact on subsequent quality evaluation tasks".
- FDA-5-2-4-1 (Code Traceability): "A source code traceability analysis is an important tool to verify that all code is linked to established specifications and established test procedures. A source code traceability analysis should be conducted and documented to verify that:
  - Each element of the software design specification has been implemented in code;
  - Modules and functions implemented in code can be traced back to an element in the software design specification and to the risk analysis;
  - Tests for modules and functions can be traced back to an element in the software design specification and to the risk analysis; and
  - Tests for modules and functions can be traced to source code for the same modules and functions."

- FDA-5-2-5 (Testing by the Software Developer): "A software testing process should be based on principles that foster effective examinations of a software product. Applicable software testing tenets include:
    - The expected test outcome is predefined;
    - A good test case has a high probability of exposing an error;
    - A successful test is one that finds an error;
    - There is independence from coding;
    - Both application (user) and software (programming) expertise are employed;
    - Testers use different tools from coders;
    - Examining only the usual case is insufficient;
    - Test documentation permits its reuse and an independent confirmation of the pass/fail status of a test outcome during subsequent review."
- FDA-5-2-6 (User Site Testing): "Testing at the user site is an essential part of software validation."
    - "Documented evidence of all testing procedures, test input data, and test results should be retained."
    - "This testing should take place at a user's site with the actual hardware and software that will be part of the installed system configuration."
    - "There should be evidence that hardware and software are installed and configured as specified. Measures should ensure that all system components are exercised during the testing and that the versions of these components are those specified. The testing plan should specify testing throughout the full range of operating conditions"
    - "Some of the evaluations that have been performed earlier by the software developer at the developer's site should be repeated at the site of actual use."
- FDA-5-2-7 (Maintenance and Software Changes): "When changes are made to a software system, either during initial development or during post release maintenance, sufficient regression analysis and testing should be conducted to demonstrate that portions of the software not involved in the change were not adversely impacted."
- FDA-6 (VALIDATION OF AUTOMATED PROCESS EQUIPMENT AND QUALITY SYSTEM SOFTWARE): "Software tools are frequently used to design, build, and test the software that goes into an automated medical device. Many other commercial software applications, such as word processors, spreadsheets, databases, and flowcharting software are used to implement the quality system. All of these applications are subject to the requirement for software validation".
- FDA-6-1 (HOW MUCH VALIDATION EVIDENCE IS NEEDED?): "The level of validation effort should be commensurate with the risk posed by the automated operation.", "Documented requirements and risk analysis of the automated process help to define the scope

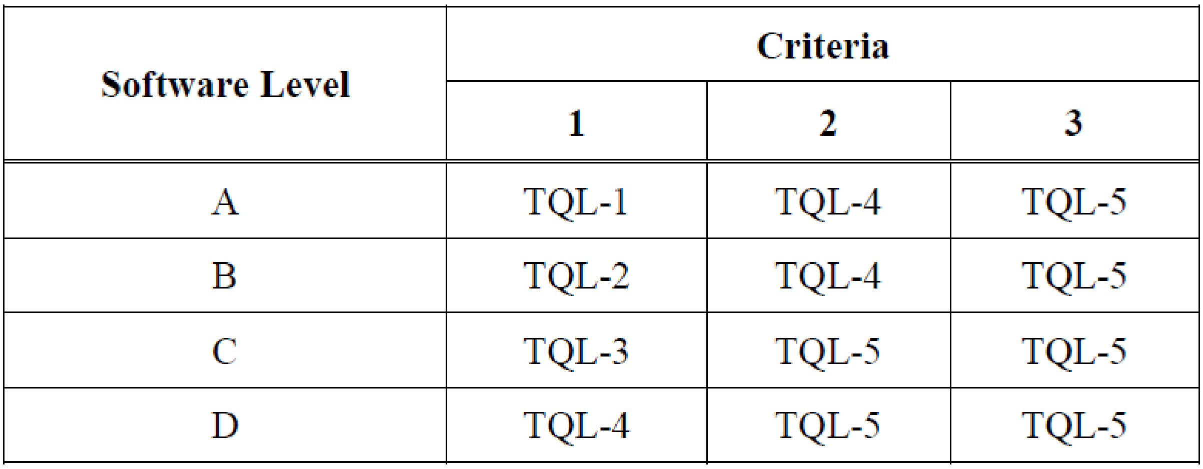
of the evidence needed to show that the software is validated for its intended use.”

- FDA-6-2 (DEFINED USER REQUIREMENTS): “A very important key to software validation is a documented user requirements specification that defines:
  - the “intended use” of the software or automated equipment; and
  - the extent to which the device manufacturer is dependent upon that software or equipment for production of a quality medical device.”
- FDA-6-3 (VALIDATION OF OFF-THE-SHELF SOFTWARE AND AUTOMATED EQUIPMENT): “For some off-the-shelf software development tools, such as software compilers, linkers, editors, and operating systems, exhaustive black-box testing by the device manufacturer may be impractical. Without such testing – a key element of the validation effort – it may not be possible to validate these software tools. However, their proper operation may be satisfactorily inferred by other means. For example, compilers are frequently certified by independent third-party testing”

For “Off-The-Shell” (OTS) software there is a special interpretation of the FDA (and others): [FDA_OTS].

6.5 DO-178C / DO-330

The DO-178C classified tools into the criteria 1, criteria 2 and criteria 3 tools, see [DO178C]. Together with the risk class of the system a so called tool qualification level (TQL) is derived, where TQL-1 is the most rigorous level and TQL-5 the least rigorous, see Figure 4.



Software Level	Criteria		
	1	2	3
A	TQL-1	TQL-4	TQL-5
B	TQL-2	TQL-4	TQL-5
C	TQL-3	TQL-5	TQL-5
D	TQL-4	TQL-5	TQL-5

Figure 4: TQL Determination in DO-178C

The DO-330 has detailed requirements on the development and application of the tool [DO330]. There are about 450 requirements in the standard that are not listed here.

6.6 Satisfying the Validation Requirements

The previous sections listed the tool classification and qualification requirements from important standards and selected those that are relevant for the qualification of the Testwell CTC++ by giving them IDs. These requirements are listed in this section, together with arguments for their fulfillment.

So the main requirements are to show the absence of the potential errors in the used features that have no high detection or prevention probability by specified measures. This analysis has to be done and documented in

- The tool classification report (with the considered features and their potential errors in the use case). This is done in the [TCR].
- Tool qualification reports that extends this plan by the analyzed test results and shows the absence of errors. This is described in this document (Section 6.5.2 and 6.7.3) and the tool qualification report which extends this document by the analyzed test results.
- Tool application guide / tool safety manual that describes the way how the tool is used conforming to the analysis and the qualification results. This is done in [TSM].

Therefore all validation goals of [IEC61508] are satisfied by this approach and all requirements for offline-support tool are satisfied if the not applicable requirements are considered during the product development process.

6.6.1 ISO 26262

The requirements from section 6.1 for the tools with "qualification needs" are covered as follows:

1. VS-ISO-10: Section 3 describes the qualification method, which is validation extended by a safety analysis
2. VS-ISO-33: The applied qualification method (combination between validation and safety analysis) satisfies the requirements for TCL2 and TCL3 in all ASILs including ASIL D
3. VS-ISO-35: The unique identification of the tool is described in Section5.
4. VS-ISO-36: The determination of the TCL is in [TCR] and in any case less than TCL 3 and ASIL D
5. VS-ISO-37: The maximal safety requirements than can be violated are always equal or less then the satisfied ASIL D, hence no further analysis is required
6. VS-ISO-38: configuration and environment are specified in Section 5.
7. VS-ISO-39: the qualification plan in Section 8 contains roles and names of involved persons.
8. VS-ISO-40: Section 3 describes the qualification method, which is validation extended by a safety analysis
9. VS-ISO-41: The tool qualification report, that extends this qualification plan contains the qualification results

10. VS-ISO-42: The tool qualification report, that extends this plan contains the observed constraints and malfunctions
11. VS-ISO-48: The tests described in Section 6.5 are designed to demonstrate the absence of failures (compliance) of the used features (specified requirements).
12. VS-ISO-49: The qualification results in the qualification report imply that the erroneous outputs are analyzed and mitigations are described.
13. VS-ISO-50: The tests for anomalous operating systems are specified in Section 6.7.3
14. VS-ISO-53: This qualification plan is reviewed, together with the tool classification plan.

6.6.2 IEC 61508

The requirements from section 6.2 for the T3 tools are covered as follows:

1. VS-IEC-3: The selection is based on the need of the tool for the product development and justified by this qualification
2. VS-IEC-4: The tool has an extensive tool documentation which is amended by the tool safety manual that respects the validation results of this validation.
3. VS-IEC-5: The assessment is done in [TCR]. For all potential errors there are either measures taken into the tool application guide [TSM], or they are covered by this qualification.
4. VS-IEC-6: The evidence is available as a combination of the successful product development using the tool and by the qualification of the tool that is planned here.
5. VS-IEC-7: The validation is documented in the tool qualification report including
    a. The activities see the process and the project plan in section 4 and section 8
    b. The tool version, see section 5
    c. The tool Features, see Section 5
    d. The used tool automation unit, see [TAU_UG]
    e. The results in Section 9
    f. Test cases and results in test plan and test report
    g. Discrepancies are part of the test report and analysed in tool qualification plan
6. VS-IEC-8: This allows to use also non-testable features of the tool (where the conformance cannot be shown using tests) by applying measures to detect the potential errors identified in this feature in the [TCR], provided that they have a high error detection probability. The measures will be documented in the tool application guides, the list features that need no qualification (i.e. have tool confidence level 1) are part of the classification report [TCR].
7. VS-IEC-10: The criteria are satisfied as follows:
    a. The translator is the compiler and it is accessed within this validation

    b. The constraints for undefined or un-considered features are listed in the tool application guide
    c. The matching of the application characteristics s done by comparing the code coverage within the tool during qualification and application.
    d. Satisfied by the input of the tool (C language is general purpose)
    e. Satisfied by the input of the tool (C language is general purpose)
8. VS-IEC-12: is satisfied by the tool application guide that guides the user to the application of the tool (including the modeling rules)
9. VS-IEC-14: This is satisfied by validating the tool.

6.6.3 EN 50128

The requirements from section 6.3 for T3 tools for are covered as follows:

1. VS-EN-11: The tool qualification report which extends this tool qualification plan documents the performed activities
2. VS-EN-12: The version of the tool is contained in Section 5 and the [TSM].
3. VS-EN-13: The validated tool functions are modeled as Features and are listed in Section 5
4. VS-EN-14: The TAU is described in [TAU_UG] that is referred in this qualification plan and the resulting qualification report.
5. VS-EN-15: The tool qualification report which extends this tool qualification plan documents the result of the validation
6. VS-EN-16: The test report which is generated from the TAU by processing the test plan contains the test results.
7. VS-EN-17: The test report contains also the discrepancies (failed tests)
8. VS-EN-18: [TCR] and [TSM] contain effective measures for features that are not qualified according to this qualification plan.
9. VS-EN-20: This qualification plan for the Testwell CTC++ satisfies international standards
10. VS-EN-21: The matching to the application is achieved by selecting the required features of the tool and verified by comparing the code coverage between the qualification and the application of the Testwell CTC++. This is documented in [TCR] and [TSM].
11. VS-EN-23: The required features have been selected from the user to match is development process and are listed in Section 5.
12. VS-EN-25: This qualification is project specific by selecting the features and comparing the code coverage in the Testwell CTC++ during qualification with the application.
13. VS-EN-26: The selected qualification method is validation of the selected features and contains functional tests (see Section 6.5.2) as well as robustness tests (see Section 6.7.3)

14. VS-EN-31: This qualification plan captures all relevant requirements of the EN50128 for T3 tools. For T1 and T2 tools no qualification is required, but an analysis of potential errors as was done during the creation of the qualification kit. This is configured from the user during application of the kit and documents the relevant analysis in [TCR] and [TSM] that justify the selection of the Testwell CTC++.

6.6.4 IEC 62304 and FDA Validation (Draft)

The selection FDA validation requirements from Section 6.4 is satisfied as follows:

- …

6.6.5 DO-178C / DO-330 (TQL-5)

For the TQL-5 there are only tool operational requirements and requirements for the qualification liaison process in the DO-330, so the tool development process is not touched. Using the TCA model can satisfy most of the requirements for TQL-5 and only a few qualification process requirements remain to be satisfied. A detailed tracing of the requirements to the DO-330 requirements is available in [TCA_Use].

For TQL-5 there are no requirements for the tool provider that are not contained in the TCA qualification model, and only a few requirements for the user of the tool. These arise mainly from the configuration management for the tool operational requirements and the liaison process:

- DO-330-7.2.1: Is Configuration Management System used?
- DO-330-7.2.7: Are elements archived, retrieved and released (e.g. using svn commit, update, tag)?
- DO-330-4.1: Is the qualification model (with the qualification needs) in configuration management stored as a CC1 element?
- DO-330-8.1.b: How is it ensured that the tool is only used as described in the model (e.g. using a conformity review of DO-330-8.1.d)?
- DO-330-8.1.d: Is the tool conformity reviewed?
- DO-330-9: The certification authority should accept the qualification method and the TQP and it should review the resulting TQR, together with the known and found bugs in the TSM.

If the tool user conforms to the listed requirements, the tool qualification is conformant to DO-330, TQL-5.

6.7 Test Strategy

The test strategy for the Testwell CTC++ is based on the selected features of the compiler. From the testing perspective there are three kinds of features in the Testwell CTC++:

- Language features that are part of C language and can be covered by testing the corresponding language inputs to the Testwell CTC++
- Configuration features that are not part of the C language and can be tested by fixing the configuration and combining the features with the execution of the required language feature tests
- Uncritical feature tests are features that do not need specific tests, since all potential errors in the Testwell CTC++ would be detected during the development process.

In this section we will list all potential errors that have been identified in the tool classification report [TCR] and that have are critical, i.e. have no high detection/prevention mechanism assigned together with the test cases that show their absence.

This test strategy is analyzing the features independently of each other and provides tests and argumentations for each of them. There is no combination of the features in the argumentation, even if there might be interactions between the features that would require special test cases. This is justified by analyzing the code coverage in the Testwell CTC++. If the test coverage during the qualification can be compared with the test coverage during the application this gap in the argumentation can be closed by the following steps:

1. Measure the coverage in the Testwell CTC++ during the qualification tests
2. Measure the coverage in the Testwell CTC++ during the application of the tool
3. compare the coverage in the source code of the Testwell CTC++ between the two scenarios and analyze the differences.

If there are code parts (functions) that are not covered in the qualification, but covered during the application, this indicates that there have been some relevant feature interactions in the Testwell CTC++ that need to be covered or that the classification & selection of the features of the Testwell CTC++ has been incomplete. In the case that the code coverage is sufficient this is an indication that the tested features are sufficiently independent.

The remainder of this section is structured as follows:

- Section 6.7.1 shows the critical potential errors that have no mitigations and need to be tested
- Section 6.7.2 contains the tests that demonstrate the absence of the critical errors
- Section 6.7.3 complements the tests by some robustness test cases.

6.7.1 [generated errors]

6.7.2 [generated tests]

6.7.3 Testing Anomalous Operating Conditions

The behavior in anomalous operating conditions should be documented in the tool application guide as far as known. The following table defines a checklist for typical anomalous operating conditions that should be considered for the validation of the tool if they are relevant. In case an anomalous operating condition is known to have no effects on the tool or there is a measure that prohibits it, it is declared as irrelevant with a short comment on the reason. In case of further conditions the table should be extended.

Table 3: Anomalous Operating Conditions

Condition ID	Description	Relevant (Yes / No)?
ENV	Test with modified/destroyed environment variables: Assume the tool depends on the environment variable <X>, which is set on installation of the tool. The user or another program is not aware of this and has redefined <X> to another value.	
INST	Test with modified/destroyed installation: Assume that somebody has accidently deleted some files from the installation directory, or the installation has not completed. Assume the tool depends on some dlls from the operating system or other extensions, e.g. service packs or .NET packages. What happens if these dll files are accidently replaced by other versions? Does the tool recognize this, e.g. by checking if the correct versions of expected dlls are present?	
REG	Test with modified Windows registry or other comparable mechanism: Assume that the tool depends on some entries to the windows registry, which are set on installation of the tool. The user has installed other tools, e.g. other versions of the same tool, or done something else, so that these registry values have changed to other values.	
RES	Test without other resources like disc space, network access,..	
PAR	Test with parallel execution of the tool (e.g. run in background): Assume that two instances of the tool are executed on the same windows session at the same time. Are both instances running completely independently? Is it possible that both instances write/read data, e.g. temporary files, to/from the same resource?	
CPU	Test with high CPU load	

RAM	Test with low RAM availability: Assume the tool is executed in a situation where the available RAM becomes lower than specified in the minimal system requirements. Can this situation cause deviations in the tool’s outputs?	

7 Qualification Environment

The qualification environment (sometimes also called validation suite) consists of a test automation unit (TAU) and a test suite. The TAU is used to automate test case execution of tests for the Testwell CTC++ and the test suite is a structured collection of test cases. The structure of a test case in a directory (configuration files, input files, expected outputs) depends on the TAU and the kind of the use cases to be tested. The contents of the test suite depend on the use cases and the critical errors identified in the previous section.

7.1 Test Automation Unit

The test automation unit (TAU) provides functions to demonstrate that the Testwell CTC++ complies with its specified requirements. It supports the execution of test plans for the Testwell CTC++ and generates a test report with the results PASS / FAIL / ABORT and test data to analyze the test.

Details can be found in the user manual of the TAU in [TAU_UG].

7.2 Test Suite and Test Plan

The test suite for the Testwell CTC++ consists of a set of test directories. Every test directory is associated with a list of tool features and a set of potential errors that can be detected. Within the directories there can be either one test, a set of tests or test directories.
Furthermore, there is a description explaining the rationale that the tests in the directory show the absence of the specified errors in the specified tool features.

The selection of the test to be executed is done as follows:

- for every use case with qualification need listed in Section 5
- for every tool feature with qualification need listed in Section 5
- for every potential error with TD >1 listed in Section 6.5.1 and the robustness tests in Section 6.7.3
- select the available test cases for execution and create a **test plan** that contains the directories of these test cases for execution

This selection is supported by the TCA tool, see the qualification kit user's guide [QKit_UM].

7.3 Test Design Methods

The construction of the test cases employs various test design methods:

1. Equivalence classes on the input domain(s) according to the use case descriptions using boundary and ordinary values of the input domains equivalence classes
2. Combinational testing
   a. 2-wise combinatorial testing based on equivalence classes
   b. 3-wise combinatorial testing based on equivalence classes with default values

    c. Randomized combinatorial testing
3. Equivalence testing: test if equivalent inputs result in equivalent outputs
4. Error guessing: try to create error prone inputs
5. Optimization tests: test if the known optimizations rules result into errors
6. Big inputs: either using generated or existing data

The usage of these (and other) methods is explained in the test description of the test suite where the methods are applied.

7.4 Test End Criteria

The major criterion for testing is a systematic coverage of the input domain with test inputs based on use cases and identified potential errors. The conditions listed here can be applied both to test execution and to test design.

1. Coverage of all test goals.
2. Complete run of all test cases in the test plan.
3. Complete analysis of all executed test cases.

7.5 Robustness Tests TF_Anomalous

The tests in the directory TF_Anomalous[3] are robustness tests of the Testwell CTC++ under the identified anomalous operating conditions (see Section 6.7.3). The tests are manual as they involve complicated changes to the environment, e.g. deleting files from the install directory or starting other programs in parallel. The tests in TF_Anomalous consist of a repetition of a selected choice of test cases from the other test fields under anomalous operating conditions. It is checked if the tool produces the same results than under normal operating conditions.

Every robustness test in TF_Anomalous has the following steps:

a) Normal Run: Run the specified test, e.g. a test from another directory, under normal operating conditions.
b) Anomalous Environment Startup: Perform steps/changes to establish the specified anomalous operating condition.
c) Anomalous Test Run: Repeat the test run under the anomalous operating conditions.
d) Anomalous Environment Cleanup: Perform steps/changes to re-establish the normal operating condition on the test computer.
e) Diff Results: Diff the test results and generated result files with the results from the test run under normal operating conditions.

[3] TF stands for test field, which can be used as a structuring principle of validation suites for test with the same test method.

8 [generated qualification]

9 References

[DO330] RTCA. DO-330: Software Tool Qualification Considerations 1st Edition 2011-12-13.

[DO178C] RTCA. DO-178C: Software Considerations in Airbone Systems and Equipment Certification, 2011-12-13.

[EN50128] BS EN 50128:2011, Railway applications — Communication, signalling and processing systems — Software for railway control and protection systems, BSI Standards Publication

[FDA2002] General Principles of Software Validation; Final Guidance for Industry and FDA Staff, Jan 2002, from http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm085371.pdf

[FDA_OTS] Guidance for Industry, FDA Reviewers and Compliance on Off-The-Shelf Software Use in Medical Devices, Center for Devices and Radiological Health (CDRH), from http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/ucm073779.pdf

[IEC61508]International Electrotechnical Commission, IEC 61508, Functional safety of electrical/electronic/programmable electronic safety-related systems, Edition 2.0, Apr 2010.

[IEC62304] International Electrotechnical Commission, IEC 62304, Medical device software –Software life cycle processes

[ISO26262] International Organization for Standardization. ISO 26262 Road Vehicles –Functional safety–. 1st Edition, 2011-11-15.

[QKit_UM] User Manual: Qualification Kit for Testwell CTC++

[SAFECOMP12] Determining Potential Errors in Tool Chains: Strategies to Reach Tool Confidence According to ISO 26262, SAFECOMP 2012, Wildmoser, Philipps, Slotosch

[TAU_UG] Tool Automation Unit for Testwell CTC++, contained in this qualification kit contained in the documentation of this kit in the file: TAU_UserGuide.docx

[TCA] Tool Chain Analyzer, tool available on www.validas.de/TCA.html Version 1.9.1

[TCA_UM] Tool Chain Analyzer, Version 1.9.1, User Manual, (<TCAHome>/plugins/Documentation/UserManual.pdf)

[TCA_Use] Using the Tool Chain Analyzer for Development of Qualifiable Tools, Validas white paper (draft)

[TCR] Tool Classifciation Report for Testwell CTC++

[TSM] Tool Safety Manual for Testwell CTC++