# State of the Art in Software Tool Qualification with DO-330: A Survey

Mohamad Ibrahim[1], Umut Durak[2]

**Abstract:** In safety-critical software development, tool qualification is one of the pillars that supports the validity and correctness of the critical software leaving no space for errors that might overthrow the stability of the system. This paper provides a general overview of tool qualification in functional safety standards, then focuses particularly on the avionics standard DO-330 by highlighting the important aspects in the qualification process. In this paper we provide a comprehensive survey on the prime aspects that qualifier should consider when conducting tool qualification, and helpful recommendation on what Qualification Kits should include for tool developers. We first review concepts from inside DO-330 standard, afterwards we move to more practical topics collected from real examples, and finally present Qualification Kits' cases and findings. We conclude with challenges and recommendations.

**Keywords:** Avionics, Safety Standards, Tool Qualification

## 1 Introduction

The automotive domain functional safety standard ISO 26262 [Ro11] defines the software tools to be "a computer program used in the development of an item or element". DO-330 [RT11b], the Software Tool Qualification Considerations supplement to the avionics standards RTCA DO-178C [RT11a] Software Considerations in Airborne Systems and Equipment Certification, defines a tool as "a computer program or a functional part thereof, used to help develop, transform, test, analyze, produce, or modify another program, its data, or its documentation". The tool does not include the final software product code itself, the compiled libraries nor software operating system [Hi17].

CENELEC 50128 [CE20] -The railways software development standard- states that "The qualification of a tool is a process whereby it is demonstrated that a tool is usable for the realization of a software application with a definite SSIL (Software Safety Integrity Level)". The aviation standard's supplement DO-330 defines the tool qualification as "the process necessary to obtain certification credit for a tool". In other words, tool qualification is the documentation of evidences that show the tool is reliable and suitable for the intended purpose in the context of a specific project. Boulanger defines tool qualification in [Bo17]

[1] Technical University of Clausthal, Institute for Informatics, Julius-Albert-Straße 4, 38678 Clausthal-Zellerfeld, Germany, mohamad.ibrahim@tu-clausthal.de

[2] German Aerospace Center (DLR), Institute of Flight Systems, Lilienthalplatz 7, 38108 Braunschweig, Germany, umut.durak@dlr.de

as “a process that allows us to demonstrate that a tool can be used as part of the realization of a software application with a determined safety goal”.

Software tools fall into two categories [Hi17], Software Development Tools and Software Verification Tools. Such categorization was used in the previous RTCA standard DO-178B [RT92] as a basis to determine the Tool Qualification Level (TQL), but this is no more the case with DO-178C [RT11a] (discussed in 2.1). The categories diverge in that the development tools can insert an error in the software but verification tools cannot, however they can fail to detect an error in the software [Hi17]. In that manner, each standard has its own categories’ naming convention. For instance, IEC 61508 [Co10] recognizes three categories of tools: T1 generates no output in the executable code, T2 takes care of the testing, and T3 generates output to the executable code. T1 corresponds to no qualification, while T2 resembles Software Verification Tools and T3 resembles Software Development Tools. Different standards can vary in naming but they all lead to that same concept. Examples of Development Tools are compilers, linkers, modeling tools, code generators, code manipulators, etc. Verification Tools include test cases generators, code static analysis, test automation, structural coverage tools, test results checker, etc.

In software engineering, the utilization of tools introduces risks but it is inevitable [Sl12] for two reasons: first, it automates a lot of the processes which contributes to the reduction of effort and time. Second, it eliminates the human error stemming from the manual and repetitive work. Tool utilization risks are either introducing errors to the software or failing to detect errors. In the light of this and out of the 4-eyes principle [Ma18], the need for tool qualification to prove that the tool functions correctly and as intended becomes evident.

There is a common principle between all functional safety standards: “qualification of the tool is required only when this tool replaces, reduces or automates one of the software life-cycle processes”. However, this qualification is not required in case the output of the target process is verified for integrity. In other words, if the activities or tasks required by a standard rely on the correct functioning of the tool then tool qualification is a must [CSM11].

## 1.1 Tool Qualification Standards

Tool qualification is a process that is conducted in the framework of a functional safety standard which is generally associated with a specific domain. Every domain has its own safety document used in certification acquisition, e.g. RTCA DO-178C for airborne systems [RT11a], ISO 26262 for automotive [Ro11], CENELEC EN-50128 for railways [CE20]. Not to mention, IEC 61508 is however a multi-domain standards for functional safety [Co10]. These standards are descriptive standards. Descriptive standards “focus on defining requirements on appropriate environments, methods and processes to develop safety-critical products” [AEkT12], whereas prescriptive standard “focus on giving an exhaustive list of product features that a safety-critical product should exhibit” [AEkT12]. The first is the most dominant type in functional safety standards.

Some standards uses the term "certified toolsïnstead of "qualified tools"like in IEC 61508. However, what mainly sets the DO-330 from the other domain-specific standards is that it can be used in other domains/industries but only when conjugated with a domain-specific standard [Po13]. However using it in other industries is not a common practise, mainly due to the fact that each domain has its own standard that should be followed to be certified. In addition, the avionic standard is more rigorous than most industries' standards, and that makes it an excluded option. Another difference that has a big impact on the qualification effort is that for relatively low qualification level tools -and sometimes for all levels- it is accepted by many standards (e.g. ISO 26262, EN 50128,IEC 61508) to use pre-qualified tools certified by a domain authority (e.g. TÜV SÜD). For example, Parasoft tool which is a testing solution for safety-critical systems, is TÜV SÜD certified and thus does not need Qualification for all levels, except when used for aerospace [Co]. Sometimes pre-qualified tools that are accepted by certification authorities need additional measures specific to the use of the tool in order to obtain the required certification credit, e.g. Polyspace from Mathworks [CSM11]. In contrast to other standards, TQ in DO-330 follows a methodology that is similar to the certification of a flight software itself developed under DO-178C [Fr11], as its main goal is providing the certification authorities evidence that the tools used on the software lifecycle fit the safety requirements mandated by the standard.

There is no common approach for qualification, each standard has its own [CSM11]. In general, there are four methods/approaches for tool qualification that is accepted by most standards [Ro11]:

1. Increased confidence from use (proven in use argumentation)
2. Evaluation of the development process (process assessment)
3. Validation of the software tool in the operational environment
4. Development in compliance with a safety standard

DO-330 only accepts the last two methods. The 3rd method represents TQL-5 (the lowest Qualification Level of DO-330), and the 4th method consists of presenting evidence of developing the tool according to a safety standard such as DO-178C and thus it involves input from the developer. The validation is the only method that is applicable in every standard [WS14].

Nevertheless, all the standards share the same three phases: Planning, Analysis and performing the Qualification [Ho20]. Planning is to determine what tools are needed, how they are going to be used (their input and output) and what processes they replace, reduce or automate. Following that the qualifier should analyze the impact of using the tool and the error detection-likelihood, and based on that determine the TQL (Tool Qualification Level). TQL term is used in DO-330, other standards have equivalent terms for it, e.g. in ISO 26262 it is TCL (Tool Confidence Level). The standards require that tool users classify the tools by performing risk assessment that shows what impact tool error might have on the final safety-critical system [KRH15]. Finally, performing the Qualification based on one of the methods mentioned above.

**Structure of the paper**. Section 2 is a summarization of the most important aspects of the DO-330. In section 3 listed the most important aspects that are found during the literature scan but not explicitly described in the standard document. Section 4 dedicated to tool qualification kits, their purpose, usage, content and examples. In section 5 we discuss challenges, limitations and important considerations. Finally, we conclude in section 6.

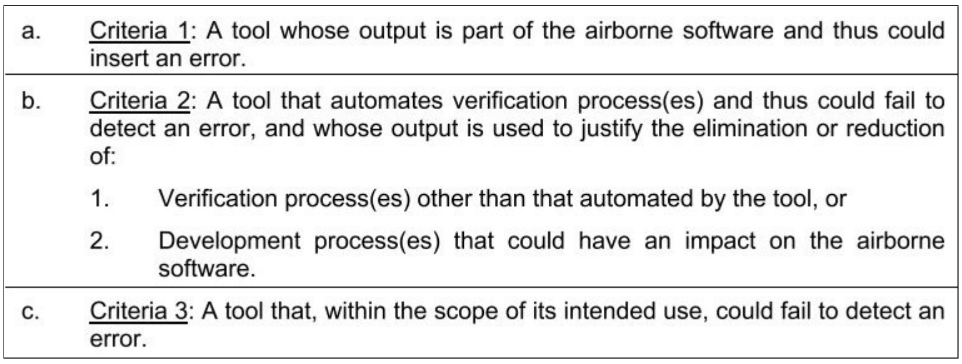
## 2 Theoretical Tool Qualification Aspects

The use of tools to assist software development exist since the early days of assembler programs [BB03]. Nonetheless, there was no mention of tool qualification until the release of DO-178B in 1992. The process of tool qualification in DO-178B was briefly mentioned in the document itself with no further clarification or guidance and no different roles were addressed. Busser et al. described DO-178B in [BB03], "The requirements on tools, and their associated qualification requirements imposed by DO-178B, have not been completely or consistently understood by developers of the software systems that must be qualified". For that reason and due to the new advances in software engineering that triggered a great diversity in tools, a new standard was needed. The purpose of the new standard was "to tackle the complexity of new software development structure and workflows for assuring safety", along with its supplement DO-330 which addresses "Software Tool Qualification Considerations". This section will be dedicated to introducing the most important aspects of tool qualification based on the conducted literature survey.

### 2.1 Tool Classification and Analysis

All the standards adopt the premise "The qualification depends on how the tool will be used and not the tool itself" [Gr]. Thus, all qualification processes starts with evaluating the tool impact on the process workflow of the software lifecycle. DO-178C defines three criteria (Tool Impact) for the Tool under Qualification:

Tab. 1: DO-178C Tool Impact Criteria [RT11b]



| | |
|---|---|
| a. | Criteria 1: A tool whose output is part of the airborne software and thus could insert an error. |
| b. | Criteria 2: A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of:<br>1. Verification process(es) other than that automated by the tool, or<br>2. Development process(es) that could have an impact on the airborne software. |
| c. | Criteria 3: A tool that, within the scope of its intended use, could fail to detect an error. |

Criteria 1 corresponds to Development Tools category from DO-178B, while Criteria 2 and 3 represents verification tools. Criteria 3 tools should be extended to Criteria 2 if they are used for eliminating or reducing a process that is mandated to be used by DO-178C in the Software Lifecycle. [Po13] presents an example that demonstrates this. A proof tool can be used to automate verification of source code, thus criteria 3 is applicable in this case. However, if the tool user claimed that the verification eliminates the need to check for a class of error then criteria 2 becomes applicable. Important aspect worth noting is analysing the tool impact should not be outside the tool context, instead, it should take into account the way the impact of the undetected errors may be leveraged or mitigated by subsequent tools or processes [Ca14].

Based on the criteria of the tool, finding the TQL is a matter of table lookup activity. The following table determines the TQL:

Tab. 2: TOOL QUALIFICATION LEVEL DETERMINATION

| **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 |

The TQ processes benefit from the detailed information provided by the tool analysis and classification phase. The provided information identifies potential errors that should be proved to be absent, contained or avoided by following safety guidelines or by certain measures.

## 2.2 Qualification Lifecycle and Processes

Software quality cannot be added to a product after it is developed [Hi17]. Thus, DO-330 defines TQ lifecycle processes (Fig.1 adopted from [MM17]) to qualify the tools and meet the required quality and assurance:

1. Tool Operational Process
2. Tool Planning Process
3. Tool Development Process
4. Tool Verification Process
5. Integral Processes is done throughout the entire tool qualification lifecycle.

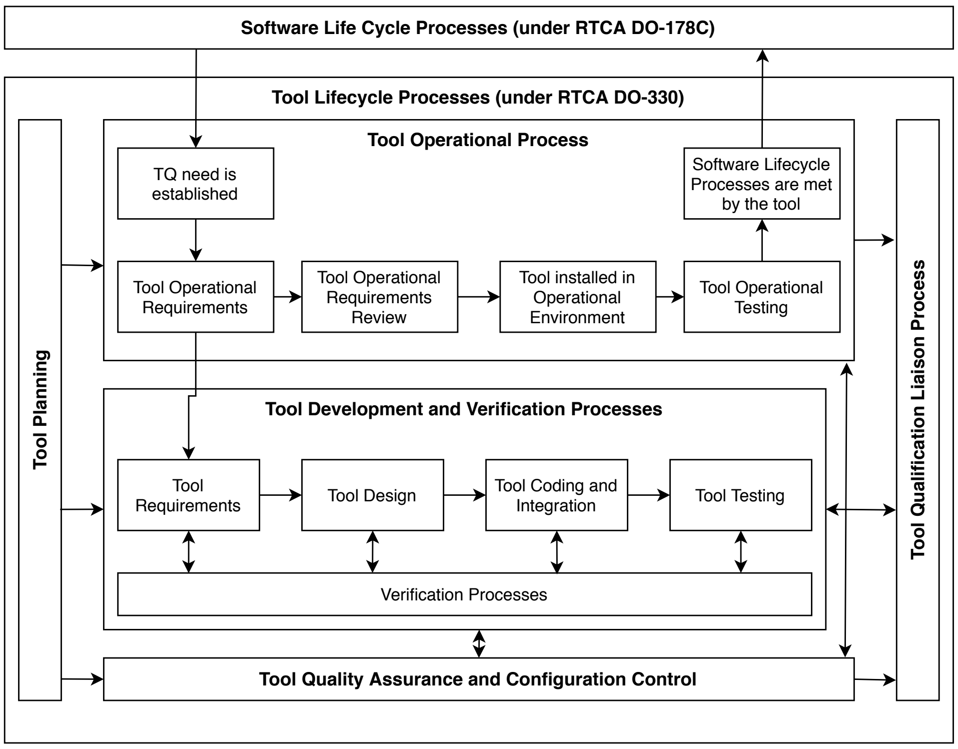

Software Life Cycle Processes (under RTCA DO-178C)
Tool Lifecycle Processes (under RTCA DO-330)
Tool Planning
Tool Operational Process
TQ need is established
Software Lifecycle Processes are met by the tool
Tool Operational Requirements
Tool Operational Requirements Review
Tool installed in Operational Environment
Tool Operational Testing
Tool Development and Verification Processes
Tool Requirements
Tool Design
Tool Coding and Integration
Tool Testing
Verification Processes
Tool Quality Assurance and Configuration Control
Tool Qualification Liaison Process

Fig. 1: Tool Qualification Lifecycle

**Tool Operational Process** is the user responsibility. The main objectives involved in this process include: tool qualification need is established, TOR (Tool Operational Requirements), tool executable code installed and TOR is validated [MM17]. This process main output is TOR, which should be reviewed and tested against Tool Operational Validation and

Verification Case Procedures (TOVVCP) [MM17]. An important aspect of using a tool is the human factor risk, in which the tool user may lose understanding of how to operate the tool/toolchain. This factor is no lesser than others because the user ignorance of the tool may lead him to false evaluation of the error sources. Therefore, the user in this process should familiarize himself enough on the tool/toolchain's functionality, status, operation and needed controls [Ca14].

**Tool Planning Process** defines the tool qualification processes and their interaction/interrelationship for the TQ lifecycle. In the planning process, standards, verification environment and development environment should be defined. Lastly, the qualifier should enumerate the DO-330 objectives and describe how they will be fulfilled. Output data for this phase includes: Plan for Software Aspects of Certification (PSAC), Tool Qualification Plan (TQP), Tool Verification Plan (TVP), Tool Configuration Management Plan (TCMP), Tool Quality Assurance Plan (TQAP) and Tool Standards (requirements, design, code).

**Tool Development Process** is the implementation process of the tool. High Level Requirements (HLR), Low Level Requirements (LLR) and code are developed considering traceability, transition and integration criteria in the process. It is important to distinguish Tool Operational Requirements (TOR) that comes from the tool user and describe the functions and features required by the Tool User, and The tool requirements which is derived from the TOR Fig. 1. Tool (functional) requirements describes the developer prospective of what functions and features the tool will have.

**Tool Verification Processes** is done sequentially in two phases, the first is the verification of the tool functional requirements in which the tool developers' work needs to be verified against the intended functionality (Fig. 1 the "Tool Development and Verification Processes"block). Secondly, is the tool operational verification and validation process in which the tool user's work needs to be verified against the intended usage (Fig. 1 the "Tool Operational Process"block). Verification of the operational requirements means the tool behave as intended, and validation means that the operational requirements are correct.

## 2.3 Qualification Objectives

The objectives of DO-330 are listed in eleven tables in Annex A of the standard. They can be summarized based on the target TQL [Hi17] [He20]:

1. **TQL-5**
   - The software tools to be qualified (PSAC)
   - TOR are defined
   - Qualification strategy (TQP)
   - Tool-specific information (SECI)

- Ensure that TOR covers the processes eliminated or reduced in PSAC (PSAC Coverage)
- Configuration Management: Identify and version control configuration items
- Basic quality assurance is conducted.
- TOR verification and validation

2. **TQL-4**
   - Tool requirements management: includes defining, tracing, verifying and validating the tool requirements.
   - Establishing 5 plans: Tool Qualification Plan, Tool Development Plan, Tool Verification Plan, Tool Configuration Management Plan and Tool Quality Assurance Plan
   - Development of integration of the tool
   - More rigorous configuration management
   - Tool executable code is compatible and robust with the tool requirements
3. **TQL-3**
   - Requirements, design, and code are verified against standards for adherence
   - Transition criteria, interrelationships among processes of the tool is defined
   - Tool development environment is selected and defined.
   - LLR are developed and defined
   - Requirements-based testing is analysed on the level of statement coverage
   - Analysis of the tool data and control coupling
4. **TQL-2**
   - Tool verification process activities should be done with independence
   - External components analysis and definition
   - Tool source code verification
   - LLR verification
   - Requirements-based testing is analysed on the level of decision coverage
5. **TQL-1**
   - Increased independence
   - Verification of external components
   - Modified Condition / Decision Coverage

This numeration is an abstraction of the objectives from the standards DO-330, and additive, meaning that each TQL includes its own objectives plus all the previous levels' objectives.

## 2.4 COTS Tool Qualification

What is significance about COTS tools is that the tool is not developed from the TOR that comes from the user [Po13]. The approach to handle this situation is described in the DO-330 section 11.3. The approach can be summarized by separating the TOR into developer-TOR that was used to develop the Tool, and the user-TOR which provides additional constraints on how the tool should behave in the operational environment. In case of qualifying to TQL-4 or above the developer should provide Qualification Kit/Package to assist the user in the qualification process. Section 4 gives deeper look into Qualification Kits. Nevertheless, It is important to consider that the burden of tool qualification is not on the organization developing the tool, but on the organization qualifying the tool, because tool usage reliability depends on the user's use of the tool rather than the generic capabilities of the tool [BB03].

In the standard DO-330 there is no clear distinction between Tool User and Tool Developer, but certain keywords can identify the role for the task, e.g. öperational"can indicate the task belongs to the Tool User [Po13]. The Tool user is the responsible for initiating the Qualification process by arguing the tool's need and level of qualification. The developer on the other hand provides all the necessary plans that guides the tool development and verification processes [Po13].

The tool is developed according to the plans by the tool developer, afterwards, the tool user integrates the tool in the operational environment [Po13]. In addition to the verification of the output of the planning, requirements, design, coding and integration processes, the developer should conduct tests in the tool verification environment including verification of test results and structural coverage analysis. On the user side verification and validation activities should be conducted in the tool operational environment [Po13].

Software Quality assurance and Software Configuration Management processes is done based on an arrangement shared between the developer and the user [Po13]. Finally, Qualification Liaison Process is the user duty and it is based in gathering the data from the Tool User and Developer processes [Po13]. Marques et al. [MM17] organized the user and developer responsibilities nicely in his work (Table 3 and Table 4).

Tab. 3: Tool User Objectives [MM17]

| DO-330 Table | Number of Objectives |
|---|---|
| T-0 | All objectives: Applicable |
| T-1 | Objectives 3 and 5: Not applicable<br>Others: Applicable |
| T-2 to T-7 | All objectives: Not Applicable |
| T-8 to T-10 | All objectives: Applicable |

Tab. 4: Tool Developer Objectives [MM17]

| DO-330 Table | Number of Objectives |
|---|---|
| T-0 | Objectives 2, 4 and 5: Applicable<br>Others: Not applicable |
| T-1 to T-9 | All objectives: Applicable |
| T-10 | All objectives: Not Applicable |

## 2.5 Qualification Alternative Methods

Some tools are hard to qualify based on the DO-178C/DO-330 standard, therefore, qualifiers usually make workarounds. The DO-330 [RT11b] mentions the possibility to use alternative methods to qualify a tool. Potential alternatives methods according to the standard include but not limited to: exhaustive input testing, formal methods (e.g. [Wa17] [Fr11] and [So09]) and dissimilar tools. Formal methods is the most used among the three with no literature found that uses the other two.

Looking at other popular approaches like [BPS12], in which the authors suggest an approach to avoid qualifying complex components in their model-based testing tools. The approach is based on replaying the test execution against the model to identify erroneous test case executions, and instead of qualifying the test tool they qualify the replaying mechanism and its interface with the system under test. Other approaches depend on the use of qualified tools to verify the output of unqualified tools as discussed theoretically in FAQ D.7 DO-330 [RT11b], and practically in [Sl12] and [Ei19]. It is suggested in [Sl12] chaining the tools in a way that increases the safety of the overall toolchain where some tools can act as error sources and others as error sinks. Eisemann -the author of [Ei19]- considered qualifying TargetLink code generator to be difficult task. Instead, the team qualified tools EmbeddedTester and EmbeddedValidator and used them to apply verification steps required by the DO-178C/DO-331 on the output of TargetLink.

### 2.6 Other Qualification Aspects

**Protection** as defined by DO-330 [RT11b] is "The use of a mechanism to ensure that a tool function cannot adversely impact another tool function". It is used when a tool has multiple functions qualified at different levels. It is noteworthy that protection is used when a tool is used to verify an unqualified tool [Po13] and in toolchains (toolchains are discussed further in subsection 3.3).

**External components** used by the tool should be described with their interfaces based on the tool design process objectives. Then they should be verified for correctness and tested. Examples of external tools are primitive functions provided by the operating system or compiler runtime library, or functions provided by a COTS or open source library [RT11b].

**The use of previously qualified tools** is discussed in section 11.2 of DO-330. This use falls into three aspects [Po13]:

1. Reuse of previously qualified tools that are unchanged: if TQL, lifecycle data, tool operational environment, TOR and the tool version are still the same then the tool needs no re-qualification.
2. Changes to the tool operational environment: in this case a re-qualification is not needed only if the user can prove that the tool verification environment is representative of the new tool operational environment. Additionally, the analysis shown that TOR are still applicable in the new operational environment.
3. Changes to the tool itself: in this case the impact analysis should identify the needed re-verification activities, e.g. “traceability analysis, data coupling and control coupling analysis, regression testing, requirements review, etc.” [RT11b]

## 3 Practical Tool Qualification Aspects

This section surveys TQ literature for prominent and noteworthy aspects that come from practical application of the standard DO-330, and not directly discussed in the standard document itself.

### 3.1 Tool Qualification Level-5 vs 4+

TQL-4 and above require the certification applicant to describe all the functionalities of the tool and do a verification and validation against the requirements [Po13]. Pothon adds, if the applicant cannot have the required data from the developer then he is up against three options:

1. Qualify the tool to TQL-5
2. In case of COTS, section 11 in DO-330 allows the user to augment the data in order to satisfy the applicable TQL objectives.
3. A tool can be qualified to TQL-5 and then use the Service History to justify qualifying it to TQL-4.

What makes TQL-5 an attractive option for many projects is that it does not need data from the Tool Development and Verification Processes which is provided by the developer. This makes the tool qualification process user-dependent activity. In practice, The number of software tools that meet TQL-4 and above is small, "the reason for this is that the effort required for such a tool qualification is certainly a factor of 3-10 greater than a qualification according to TQL-5" [He20]. Further analysis of the methods shows what each TQL's equivalence is [He20]:

Tab. 5: TOOL QUALIFICATION TQL's EQUIVALENCE

| TQL | Equivalent DAL |
|---|---|
| TQL-1 | DAL A |
| TQL-2 | DAL B |
| TQL-3 | DAL C |
| TQL-4 | DAL D |

From our experience in research projects we recommend that qualifier aims for TQL-5, since it produces fast results suitable to meet early milestones. Then at later stages, moving to TQL-5 can be easily be built upon previous TQL-5 data plus new documents and proofs about tool design, tool verification and validation and tool and lifecycle environment configuration. In addition, plans should be provided: TQP, TDP, TVP, TCMP and TQAP.

## 3.2 Qualification Common Cases

Tools which typically reside in categories like: requirements management, configuration management/data management, and quality management can be excluded from the tool qualification process [Hi17]. Hilderman et al. adds on the other hand, compilers, assemblers, and linkers are typically Criteria 1 tools, but their output is often checked by another verification activity (i.e. review or testing). Thus, in most cases they do not require qualification.

Autocode Generator (ACG) is an example of Criteria 1 Tool that require tool qualification. The rationale behind qualifying such a tool is to eliminate the code verification process performed for every version of the software, and instead qualify the ACG once by claiming certification credit for the repetitive verification process that was eliminated [Po13]. Other types that is qualified in practice are: implementation tools that produce code representations, simulation tools and Binary translation tools such as cross-compiler or format generator [Hi17].

Examples of Criteria 2 and 3 tools which may require qualification include [Hi17]:

1. Tools that automate code reviews and design reviews against standards
2. Tools that generate test cases and/or procedures from requirements
3. Tools that determine pass/fail status
4. Tools that track and report structural coverage results
5. Tools which determine requirements coverage results.

We explore practical qualification example that consider different operational scenarios which require different TQLs [KRH15]. Krauss et al. [KRH15] present a qualification example of a safety analysis tool, specifically System-Theoretic Process Analysis (STPA). The tool identifies the system potential errors and hazards based on the Systems-Theoretic Accident Model and Processes (STAMP) accident model [Le04].

The authors distinguished four operational scenarios of the tool. The first scenario, the tool's output is manually verified for completeness and consistency, and according to DO-330 this case does not need any further qualification's effort. For the second scenario the tool is used as a modeling tool without manual verification of the output. Because the verification process is skipped criteria 2 is selected and the corresponding level is TQL-3 for level A and TQL-4 for level B to D. In the third scenario the tool output models are used for verification (e.g. formal model checker). In this case the tool is used as a verification tool, which might fail to detect an error, therefore criteria 2 still applies in scenario 3. The last operational scenario suggested in [KRH15] is to use the tool as Autocode generator as a safety control loop to identify new errors stemming from the generated code. In this case the tool might introduce an error in the final software if there is an error with STPA data, therefore criteria 1 is applied.

## 3.3 Toolchain

The standards discussed in 1.1 define how to classify each tool individually. However, dealing with a toolchain implies extra effort to ensure that the tool integration does not introduce new errors [CBT14]. Toolchain is the orderings of the engineering tools which the software product will transit through [AEkT12]. Tool integration can be defined as "what supports the development process in correctly moving from one engineering tool to another in a toolchain" [AEkT12]. Examples of tool integration include data transformation and event-based scripts in react to user or process action.

Asplund [As15] identifies certain safety-related characteristics of toolchains that should be addressed in the qualification process:

1. Lack of traceability for completeness and consistency
2. Lack of formally defined data semantics
3. Lack of data integrity
4. Lack of automated transformations of data
5. Lack of possibility to automate tool usage
6. Lack of data mining
7. Lack of process notifications and process control
8. Lack of possibility to create customized GUIs

The authors of [Sl12] discuss how to model a toolchain, determine tool impact, identify potential error sources and identify measures to detect and prevent them. Despite the fact that this model was used to express tool qualification based on ISO26262, we claim that it can be applied when qualifying using DO-330 after changing some terms. The model (Fig. 2) represents two aspects [Sl12]:

1. The toolchain structure (tools, use cases and artifacts)
2. The tool confidence (potential errors and assigned checks or restrictions)

When modeling toolchains, there are generally two approaches. A bottom-up approach, in which tools are handled as separate units which may result in ignoring risk introduced by integration. The second approach is a top-down approach for toolchain modelling and qualification, this approach starts with tasks rather than technology [As15] and takes into account risks that stems from the interaction of the toolchain.

A top-down approach is presented in [AEkT12]. The proposed approach organizes development efforts in tool qualification and handle different types of safety constraints from top to bottom. This approach splits the the process to four layers that start from the more general to more specific development efforts. Top level is the management of development efforts (organisation level) that imposes constraints on the next level. Below is the operator level that represents the processes of the software lifecycle. The third level is the toolchain level that specifies the order of the tools, it imposes constraints on the tool level, at this level safety is ensured by addressing the previously mentioned characteristics of toolchains

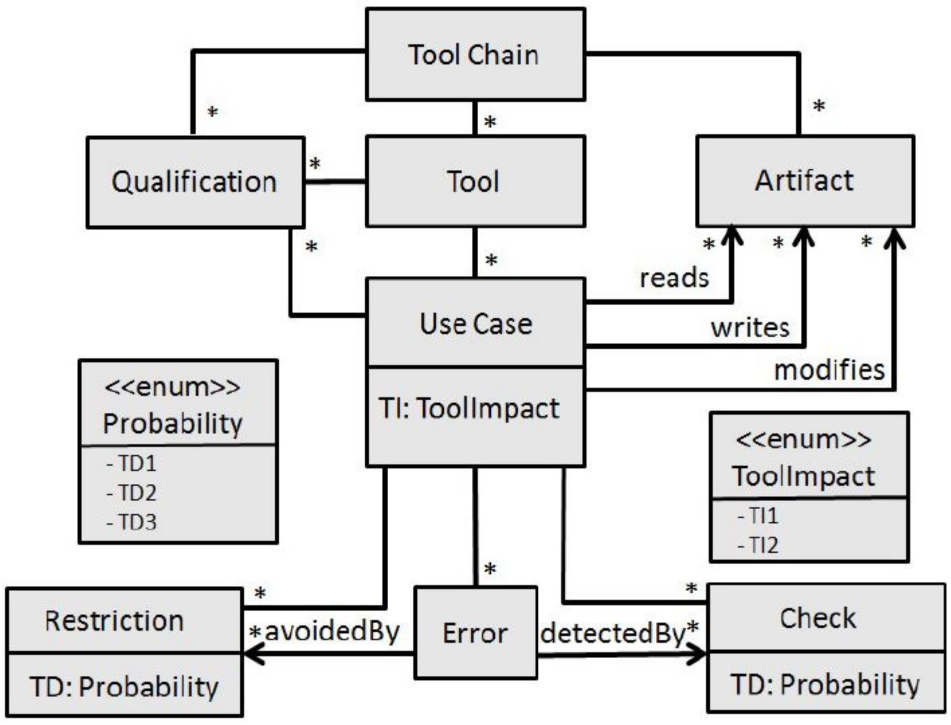


Fig. 2: Domain Model for Toolchain [Sl12]

3.3. Tool level is the basic block and most specific level, here the safety is ensured through standard software qualification taking into account constraints from higher levels.

## 3.4 Model-Based Tool Qualification

Model-based Tool Qualification is the utilization of models to represent, guide and evaluate the Tool Qualification Process. Some Tool Qualification Kits are designed with the purpose of aiding the qualification Process using models. To give an idea we show a case in point, the tool qualification model from [SB13] is split into three models. The first model is intended to capture the complete toolchain, tool's features and use cases. The main focus of the model is tracking the data flow within the toolchain -called SStructural Model". This model is then enriched with certain measures associated with features to mitigate potential error types -called Änalysis Model". Finally, a model that represents the test cases and their assigned potential errors. The TQP for Testwell CTC++ [WS14] uses this qualification model in their approach. A big advantage of the model-based qualification kits/methods is that it is simple to extend the model with new test cases designed by the user. The construction and the usage of the tool qualification model is depicted in Fig. 3 [SB13].

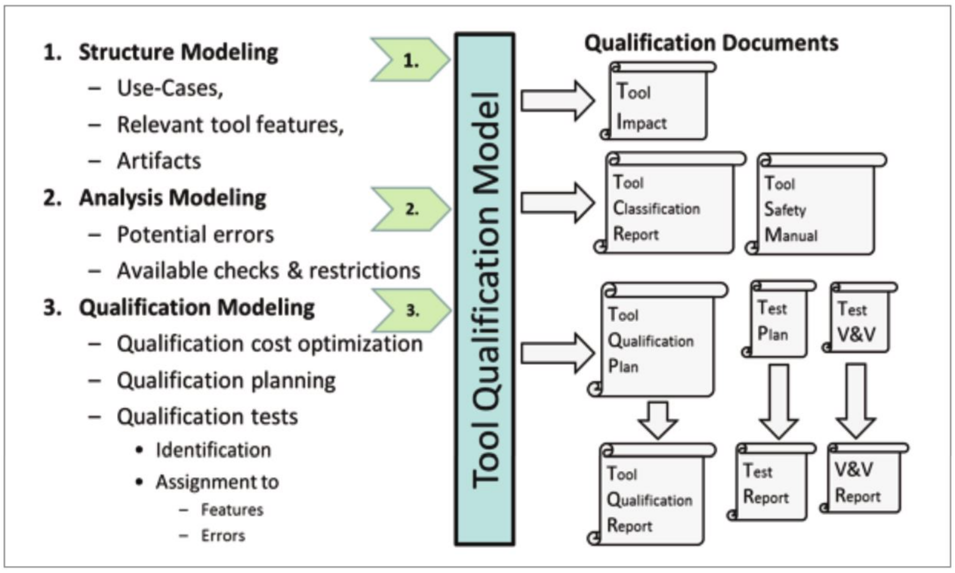


Fig. 3: Tool Qualification Model [SB13]

## 4 Tool Qualification Kit/Package

Approving the tool for use when developing according to a functional safety standard is usually a documentation-heavy formal process. It requires time-consuming manual completion which is prone to human error [Co]. To address this, Tool developers normally create a Tool Qualification Kit/Package to aid the tool user in this endeavour. Slotosch et al. [SB13] define the goal of the Qualification Kit as "the ability to qualify as many features as possible with the fewest error reduction measures that could restrict the tool application and functionalities". Tool Qualification Kit aims to provide the following support:

1. Supports the tool user during the determination of the critical errors and supplies test cases that provide confidence in the absence of the critical errors [SB13].
2. Automates the creation of the documentation required for tool qualification, e.g. Tool Qualification Plan (TQP), Tool Classification Report (TCR), Tool Qualification Report (TQR) [Co].
3. Reduce the amount of work depending on the selected tool's functionalities needed for the project [Co].
4. Provides a guide that reduces the manual testing efforts and executing automated tests for specific capabilities and use cases [Co].

A scan of COTS Tool Qualification Kits that has a publicly accessible data, has revealed that the most common content normally includes:

1. User manual and a starting guide
2. Tool Safety Manual (TSM)
3. Tool Qualification Plan/Plan template (TQP)
4. The default Tool Operational Requirements (TOR)
5. Tool Verification Plan (TVP)
6. Tool Configuration Index (TCI)
7. Tool Accomplishment Summary (TAS)
8. Tool Qualification Test cases/suits
9. Test automation mechanism
10. Standards data/document templates or means to generate them
11. Tool artifacts (Source code, models, etc.) to backup verification and validation

One of the main challenges of creating a Tool Qualification Kit is designing test cases that cover the features and identifies relevant errors for a specific use case. When choosing what tool features to be qualified using a Tool Qualification Kit we will be in one of three cases depending on the test cases provided by the Qualification Kit [SB13]:

1. The feature cannot be used, since there are no tests that covers errors of the feature (Red).
2. The feature can be used with the restriction that a certain set of safety guidelines is applied to eliminate the potential errors of the feature (Yellow).
3. The feature can be used without constraints, since there are tests available that provide evidence for the absence of the potential errors (Green).

A perfect Qualification Kit is the one that leaves the user in the third case whatever tool's features he chooses to use.

### 4.1 COTS Qualification Kits

Here we present some real Qualification Kit examples:

**BTC tools** TQK [Gr] can be used in two methods. The first is with the evaluation of the development process which does not require any additional work from the Tool User side.

BTC tools are pre-certified from TÜV Süd which covers qualification method 2 from 1.1 “Evaluation of the development process”. For the 3rd method "Validation of the software tool", the package provides a test suit that covers 600 software unit (e.g. Simulink models). For that reason, BTC Tool can be qualified based on DO-330 using the Qualification Kit only to TQL-5.

**Parasoft** [Co] is a testing solution for C/C++ embedded application targeting safety-critical software development. Parasoft TQK automates the process of creating the required documentation that supports the qualification process. It includes static analysis, unit testing and coverage of the requirements. Parasoft supports DO-178B/C and DO-330 compatible qualification process for all of its software levels. In addition, it supports pre-qualification as discussed in 1.1 based in the TÜV SÜD certification.

**Polarion** [Sib] is an application Lifecycle Management for requirements, test and quality, change control and configuration management. The Polarion Tool Qualification Kit provides: work items, work item linkage, workflow structure, reports templates and set of standard use cases.

**Axivion** [AG] is static checker of code and design. Its Qualification Kit contains: test suits to verify the use of standards, test automation and repetition mechanism and a set of violation examples of the standards MISRA, CERT and AUTOSAR C++14, which allows tool user to validate the tool.

**VectorCAST** [DO] is a platform addresses the automation of software verification process of the DO-178C. In addition to test cases and templates the kit includes the developer-TOR and Tool Qualification Document (TQD). VectorCAST has a different qualification process approach than the previously mentioned Kits. It starts with submitting to Vector the necessary information about the intended use and the configuration file of the tool operational environment. Then Vector provides a customized qualification package for those specific operational environment.

In the last example we will shed the light on is **Texas Instruments C/C++ ARM compiler** TQK. This Kit allows the user to create specific use case(s) by selecting the features the user intends to use [SB13]. The Qualification Kit includes; User manual, document templates, TQP, TQR, TSM, Tests, Test Automation Unit, An instrumented version of the compiler to measure the coverage and scripts to install and collect code coverage measurement.

## 5 Tool Qualification Challenges, Limitation and Recommendations

The COTS tools are developed by teams other than those using the software tools. Those teams often have no background in the development of safety-related software according to DO-178C or comparable standards [He20], which limits the qualification feasibility. Franca et al. [Fr11] showcase an example where the development team had to use a compiler that they had no idea how it works internally, they had no choice but to verify the output of the compiler. However, even that was expensive and slow to perform. Finally, they decided on making the compiler produce constant set of code patterns for each symbol. This was possible by limiting the compiler optimizations and thus getting away with verifying small set of patterns.

This leads to an important a consequential limitation. Mostly, tools come loaded with many features, extensions and customization options. This load should be minimized to reduce the qualification effort and complexity [ESD13]. Even after striping down the tool's features, it is still hard to verify the code against requirements manually, because it is error-prone and time-consuming and involves line-by-line code review to satisfy objectives [ESD13]. A solution to this is the use of automated code inspection tools like SSimulink Code Inspector" [Sia].

However, in reality the tools used for development are generally not qualified because they cost too much and affect the ROI (return on investment) [ESD13]. An alternative is to create an automated mechanism that verify the output of the development tools using a qualified verification tools which discussed previously in 2.5. This fact is confirmed by [BF10], which states that it is a common practice today to use non-qualified and non-certified development tools in the aerospace industry, especially for compilers, linkers and code generators. Along with the trend of not qualifying development tools goes another trend that stresses qualifying verification tools [ESD13]. Those tools will automate the verification and validation of the output of the non-qualified development tools. That looks to be is the meta/norm in the industry, particularly when model-based design approach is applied.

Nonetheless, a huge downside of TQ remains that the qualified tool cannot use updates and patches from the developers without the need of a re-qualification.

## 6 Conclusion

This paper has surveyed the tool qualification domain, particularly under the avionics functional safety standard DO-178C and its supplement DO-330. It made a comparison with other domain-specific standards and summarizes its most prominent concepts and aspects both from inside and outside the standard document. In addition, practical qualification cases were presented, discussed and analysed for limitations and challenges. Several Tool Qualification Kits have been explored and a typical Qualification Kit contents and functionalities has been induced and presented.

No doubt that tool qualification-wise the standard DO-178C/DO-330 is a substantial improvement over DO-178B. It is better in defining clear and straightforward qualification processes, addressing COTS tools with different roles, suggesting alternative methods despite their rareness and sometimes unusability.

Due to the increased complexity of modern software the qualification process mandated by DO-178C/DO-330 fall behind in two main areas. First, it does not cope with the increased costs of qualification. That may force the organizations to cut efforts on safety and quality to achieve reasonable profit or drop avionics development. The second is the absence of the acceptance of the pre-certified software tools. This can have a valid argument which says that the reuse of software is dangerous. That is to some extent true, however one can argue that when using the software tool in the same use case (e.g. static code checking) the proofs of the validity of the tool does not change, especially when the tool does not affect the behaviour of the avionic software. In addition, the current avionics standards fail on providing the qualifier with enough flexibility in the way they prove their tools correctness and validity.

Out of this survey, authors claim that the area of tool qualification and its methods still posses improvement potential. The endeavour of making the tool qualification process less vague and more time- and effort-efficient by developing and enhancing software engineering techniques will occupy the industry and researchers for years to come.

## Acknowledgment

This work is supported by the project 'Qualifizierte Software-Parallelisierung für Multicore-Avioniksysteme' QSMA funded by the German Federal Ministry of Economics and Energy (BMWi)'s LuFo VI program (Grant No. 20Q1961A).

## Nomenclature

| | |
|---|---|
| *ACG* | Autocode Generator |
| *CENELEC* | European Committee for Electrotechnical Standardization |
| *DAL* | Development Assurance Level |
| *IEC* | International Electrotechnical Commission |
| *ISO* | International Organization for Standardization |
| *PSAC* | Plan for Software Aspect of Certification |
| *RTCA* | Radio Technical Commission for Aeronautics |
| *SECI* | Software Life Cycle Environment Configuration Index |
| *TAS* | Tool Accomplishment Summary |
| *TCI* | Tool Configuration Index |

| | |
|---|---|
| *TCL* | Tool Confidence Level |
| *TCR* | Tool Classification Report |
| *TOR* | Tool Operational Requirements |
| *TOVVCP* | Tool Operational Validation and Verification Case Procedures |
| *TQ* | Tool Qualification |
| *TQK* | Tool Qualification Kit |
| *TQL* | Tool Qualification Level |
| *TQP* | Tool Qualification Plan |
| *TQR* | Tool Qualification Report |
| *TSM* | Tool Safety Manual |
| *TVP* | Tool verification plan |
| *TV* | Technical Inspection Association |

## Bibliography

[AEkT12] Asplund, Fredrik; El-khoury, Jad; Törngren, Martin: Qualifying Software Tools, a Systems Approach. In (Ortmeier, Frank; Daniel, Peter, eds): Computer Safety, Reliability, and Security. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 340–351, 2012.

[AG] Axivion-GmbH: , Tool Qualification Kit - Axivion.

[As15] Asplund, Fredrik: The future of software tool chain safety qualification. Safety Science, 74:37–43, April 2015.

[BB03] Blackburn, Mark; Busser, Robert D: Guidelines for Software Tool Qualification. 2003.

[BF10] Beine, Michael; Fleischer, Dirk: A Model-Based Reference Workflow for the Development of Safety-Related Software. pp. 2010–01–2338, October 2010.

[Bo17] Boulanger, Jean-Louis: Certifiable software applications. 2, 2,. 2017. OCLC: 964698597.

[BPS12] Brauer, Jörg; Peleska, Jan; Schulze, Uwe: Efficient and Trustworthy Tool Qualification for Model-Based Testing Tools. In (Nielsen, Brian; Weise, Carsten, eds): Testing Software and Systems. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 8–23, 2012.

[Ca14] Camus, Jean Louis; Pothon, Frédéric; Dewalt, Michael; Ladier, Gérard; Boulanger, J.-L; Blanquart, Jean-Paul; Quere, Philippe; Ricque, Bertrand; Gassino, Jean: Tool Qualification in Multiple Domains: Status and Perspectives. 02 2014.

[CBT14] Cecile Braunstein, Jan Peleska, Stefan Rieger; Torre, Izaskun De La: Toolchain Qualification Process Description. openETCS ,ITEA2 Project, jan October 2014.

[CE20] CENELEC: EN 50128 - Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems. jun 2020.

[Co] Corporation, Parasoft: , Parasoft Tool Qualification Kits for Certification.

[Co10] Commission, International Electrotechnical: IEC 61508 - Functional Safety of Electrical/-Electronic/Programmable Electronic Safety-related Systems. jun 2010.

[CSM11] Conrad, Mirko; Sandmann, Guido; Munier, Patrick: Software Tool Qualification According to ISO 26262. In: SAE 2011 World Congress & Exhibition. SAE International, apr 2011.

[DO] DO-178/ED-12 Tool Qualification - Vector.

[Ei19] Eisemann, Ulrich: Applying Model-Based Techniques for Aerospace Projects in Accordance with DO-178 C , DO-331 , and DO-333. 2019.

[ESD13] Estrada, Raymond G.; Sasaki, Gen; Dillaber, Eric: Best practices for developing DO-178 compliant software using Model-Based Design. In: AIAA Infotech@Aerospace (I@A) Conference. American Institute of Aeronautics and Astronautics, Boston, MA, August 2013.

[Fr11] França, Ricardo Bedin; Favre-Felix, Denis; Leroy, Xavier; Pantel, Marc; Souyris, Jean: Towards Formally Verified Optimizing Compilation in Flight Control Software. p. 10 pages, 2011.

[Gr] Gros, Markus: , When and how to qualify tools according to ISO 26262 · BTC ES.

[He20] Heininger, Martin: , RTCA DO330 - The standard for tool qualification, July 2020.

[Hi17] Hilderman, Vance: DO-330: TOOL QUALIFICATION OVERVIEWFOR AVIONICS ENGINEERS AND MANAGERS. www.afuzion.com, 2017.

[Ho20] Horvath, Kristof: , Confidence in Software Tools: Tool Qualification in Safety-critical Development, January 2020.

[KRH15] Krauss, Sven Stefan; Rejzek, Martin; Hilbes, Christian: Tool Qualification Considerations for Tools Supporting STPA. Procedia Engineering, 128:15 – 24, 2015. Proceedings of the 3rd European STAMP Workshop 5-6 October 2015, Amsterdam.

[Le04] Leveson, Nancy: A new accident model for engineering safer systems. Safety Science, 42(4):237 – 270, 2004.

[Ma18] Martin Heininger: , IEC 61508 – Tool qualification – When? Why? How?, May 2018.

[MM17] Marques, J.; Marques da Cunha, A.: COTS tool qualification using RTCA DO-330: Common pitfalls. In: 2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC). pp. 1–6, 2017.

[Po13] Pothon, Frédéric; Pomies, Laurent; Comar, Cyrille; Delseny, Hervé; Ben Brosgol, Ben: DO-330/ED-215 Benefits of the New Tool Qualification Document. ACG Solutions, jan 2013.

[Ro11] Road vehicles – Functional safety, jan 2011.

[RT92] RTCA: DO-178B Software Considerations in Airborne Systems and Equipment Certification. Radio Technical Commission for Aeronautics, 1992.

[RT11a] RTCA: DO-178C Software Considerations in Airborne Systems and Equipment Certification. Radio Technical Commission for Aeronautics, 2011.

[RT11b] RTCA: DO-330 Software Tool Qualification and Considerations. Radio Technical Commission for Aeronautics, 2011.

[SB13] Slotosch, Oscar; Beemster, Marcel: Model-Based Tool Qualification of the TI C/C++ ARM® Compiler. 2013.

[Sia] Simulink Code Inspector.

[Sib] Siemens: , Polarion, Tool Qualificaiton Kit.

[Sl12] Slotosch, Oscar; Wildmoser, Martin; Philipps, Jan; Jeschull, Reinhard; Zalman, Rafael: ISO 26262 - Tool chain analysis reduces tool qualification costs. In (Plödereder, Erhard; Dencker, Peter; Klenk, Herbert; Keller, Hubert B.; Spitzer, Silke, eds): Automotive - Safety & Security 2012. Gesellschaft für Informatik e.V., Bonn, pp. 27–38, 2012.

[So09] Souyris, Jean; Wiels, Virginie; Delmas, David; Delseny, Hervé: Formal Verification of Avionics Software Products. In (Cavalcanti, Ana; Dams, Dennis R., eds): FM 2009: Formal Methods. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 532–546, 2009.

[Wa17] Wagner, Lucas G.; Cofer, Darren; Slind, Konrad; Tinelli, Cesare; Mebsout, Alain: Formal Methods Tool Qualification. NASA STI Program, NASA/CR–2017-219371, feb 2017.

[WS14] Wildmoser, Martin; Slotosch, Oscar: , Tool Qualification Plan for Testwell CTC++, nov 2014.