# Guidelines for Software Tool Qualification

**Robert D. Busser**
Software Productivity Consortium
2214 Rock Hill Road, Herndon, VA 22070
busser@software.org
(703) 742-7280

**Mark Blackburn**
Software Productivity Consortium
2214 Rock Hill Road, Herndon, VA 22070
blackburn@software.org
(703) 742-7136

## Abstract

*Software Productivity Consortium (Consortium) members applying the tools for both the requirement and design model-driven verification and validation are often required to certify their software with various certification authorities such as the Federal Aviation Administration (FAA) and Food and Drug Administration (FDA). These certifications require methods and supporting artifacts for qualifying the tools used for software development and verification. This report provides guidelines for how to use tool qualification information to support certification processes.*

## Contents



## Introduction

The technology transfer of the Software Productivity Consortium's (Consortium) Test Automation Framework (TAF) to Consortium members is an integral part of the Consortium's Verification and Validation (V&V) product line. A primary attribute of TAF is its capability to directly automate many of the V&V processes and automatically produce the deliverables required by federal agencies, such as the Federal Aviation Administration (FAA) and Food and Drug Administration (FDA), in support of certification of software-based applications in their respective domains of authority. In addition, under the Global Aviation Traffic Management (GATM), all commercial airborne systems, in addition to all airborne military and space systems (U.S. Air Force 2001), have to comply with FAA regulations for avionics that require DO-178B certification. Several Consortium members have been, or will be, using TAF for this purpose (Busser, Blackburn, and Nauman 2001; Kelly et al. 2001; Statenzi 2000, 2001).

One possible roadblock to greater adoption of TAF by Consortium members is that any software tools used to automate aspects of software development or verification that will be applied for certification credit, without formal review, must be qualified. The tool qualification task can be nontrivial and may be seen as a hindrance when choosing to use TAF on programs governed by this constraint. However, a major survey conducted as part of the FAA's independently commissioned Streamlining Software Aspects of Certification (SSAC) program within the civil avionics development community included a section on tool qualification. The results of this survey indicated that 60% of the respondents considered the cost attributed to tool qualification to be small or negligible, 36% considered the cost to be substantial, and only 4% considered the cost to be prohibitive (Hayhurst et al. 1999). Fortunately for most users of

Bullseye is a copyright of Bullseye Testing Technology.
MATRIXx is a trademark of National Instruments, Inc.
Simulink and Stateflow are registered trademarks of the MathWorks, Inc.

Copyright © 2003, Software Productivity Consortium NFP, Inc. and T-VEC Technologies, Inc. All rights reserved. This document is proprietary property of the Software Productivity Consortium NFP, Inc. The contents of this document shall be kept confidential pursuant to the terms of the Membership Rules, as amended from time to time, of the Software Productivity Consortium NFP, Inc. This document shall only be disseminated in accordance with the terms and conditions of those Rules. All complete or partial copies of this document must contain a copy of this statement.

TAF, the tool qualification cost should be minimal because the tool qualification packages have been developed for several tool components of the TAF (see Tool Qualification Support for TAF and T-VEC for details).

## Scope

The purpose of this paper is to provide Consortium members with information about the tool qualification process, when it is applicable, where to find additional detailed guidance, and the general procedures they will need to follow to qualify their software tools. It also provides guidelines for qualifying use of TAF tools on their specific applications and describes the tool qualification suites, documentation, and support available to assist them with their tool qualification efforts.

The FDA, like the FAA, requires validation of automated process equipment and quality system software that is used to produce FDA-certified products. These guidelines are documented in *General Principles of Software Validation; Final Guidance for Industry and FDA Staff* (U.S. Food and Drug Administration 2002). These guidelines reflect the same general intent as the tool qualification guidelines for FAA qualification (U.S. DOT 2003); however, the FAA guidelines are more specific. While working with members involved in FDA certification and in discussions with FDA representatives that have attended Consortium events, the FDA is aware of the potential use of the FAA guidelines for tool qualification to supplement the current FDA guidelines. Therefore, this document focuses on summarizing these more specific guidelines defined in Order 8110.49, *Software Approval Guidelines* (U.S. DOT 2003), and more specifically Chapter 9, which addresses *Qualification of Software Tools Using RTCA/DO-178B*, while describing their applicability to TAF.

### Audience and Benefits

This paper is applicable to managers, project leads, software developers, quality assurance staff, and test engineers who are responsible for managing, planning, and estimating project effort, cost, and duration. In addition, this paper is applicable to Aircraft Certification Office (ACO) engineers and to Designated Engineering Representatives (DER) as it applies to the application of RTCA/DO-178B, *Software Considerations in Airborne Systems and Equipment Certification* (RTCA 1992), to the qualification of software verification and development tools. The paper assumes that the readers are familiar with TAF and are either using TAF or plan to use TAF in the future. It also assumes that the reader is at least familiar with the main issues of software certification in the context of guidelines such as DO-178B and is interested in learning more about the issues and impacts of applying TAF to assist in the development of software applications governed by similar certification constraints. References to additional information on TAF, DO-178B, and the subject of tool qualification are provided in the section For More Information.

### Organization of This Paper

The section Background Context for Tool Qualification provides context for this paper by introducing one of the primary software certification guidelines, *DO-178B – Software Considerations in Airborne Systems and Equipment Certification*, which first introduced the subject of, and requirements for, tool qualification. It then describes the evolution of clarifications and resulting guidelines for tool qualification since the release of DO-178B. Fundamentals of Tool Qualification present the key aspects of tool qualification. It discusses the two primary categories of tools, development tools and verification tools, and the differences and similarities in how they are treated by DO-178B with respect to qualification requirements. It also discusses how and where tool qualification fits into the overall DO-178B software certification process. Tool Qualification Support for TAF and T-VEC describes the tool qualification

documentation and test suites that have been created by the Consortium and T-VEC Technologies, Inc. Qualifying TAF and T-VEC for a Project provides guidance on qualifying TAF and T-VEC, per DO-178B. The Summary contains concluding remarks.

### Definitions

- ***Modified Condition/Decision Coverage (MC/DC).*** Every point of entry and exit in the program has been invoked at least once; every condition in a decision in the program has taken all possible outcomes at least once; every decision in the program has taken on all possible outcomes at least once; and each condition in a decision has been shown to independently affect that decision's outcome.
- ***Decision Coverage***. Every point of entry and exit in the program has been invoked at least once, and every decision in the program has taken on all possible outcomes at least once.
- ***Statement Coverage.*** Every statement in the program has been invoked at least once.
- ***Software Tool.*** A computer program used to help develop, test, analyze, produce, or modify another program or its documentation. Examples are an automated design tool, a compiler, test tools, and modification tools (RTCA 1992).
- ***Tool Qualification.*** Section 12.2 of RTCA/DO-178B states that qualification of a tool is needed when processes in RTCA/DO-178B "are eliminated, reduced, or automated by the use of a software tool, without its output being verified as specified in section 6" of RTCA/DO-178B. RTCA/DO-178B states, "The objective of the tool qualification process is to ensure that the tool provides confidence at least equivalent to that of the process(es) eliminated, reduced, or automated." (FAA 2003)

## Background Context for Tool Qualification

The use of software tools to assist in developing application software has been common practice since the first assembler program was written to assist in the creation of machine code programs (Salmon 1993). While the correct functionality of a software tool has always been of a concern to the user of the tool, it is not typically a concern of the customer of the application. The responsibility for the functionality of the application is typically left to the application's developer; the customer is not typically interested in how the developer actually produced the application software.

However, in the context of applications that have an impact on health or safety, or other high assurance-related areas such as security, the concerns for correct functionality are much greater. In such domains, verification of correct functionality under possibly all operating conditions often is not only a concern of the customer but also of regulatory agencies such as the FAA and FDA. These concerns are formalized in officially published guidelines such as the FAA's DO-178B (U.S. DOT 1999). These guidelines form the basis for FAA approval of software-based avionics applications by establishing software development *process completion* and *functionality verification* criteria that must be met by the application developer in order to be approved by the FAA for commercial aviation use. These guidelines, DO-178B specifically, include sections on the use of software tools in the application development process and also introduce the concept of tool qualification. Consequently, to fully understand the meaning *tool qualification* and its impact on the use of TAF and T-VEC, a brief history of its derivation is useful.

## History of DO-178B (Paraphrased From Johnson [1998])

In the avionics industry, software was initially viewed as an inexpensive and more flexible way to extend the functionality of mechanical and analog-electrical systems. However, it was quickly realized that the usual statistical approaches to assess the safety and reliability would not work for flight-critical software. An alternative means of assessment, one that addressed design errors rather than component failure rates, was required. From this need, the first version of DO-178 was created.

### DO-178

DO-178 was created to provide a basis for software certification by identifying and documenting software development "best practices," but it was written primarily at a conceptual level. To be compliant, applicants for certification were required to meet the "intent" of DO-178, but there were few details about how to actually do this. Rules of use were developed by trial and error over time. DO-178 was the first to introduce the concept that software verification requirements were dependent on the safety criticality of the software. It divided software applications into three categories: *critical*, *essential*, and *nonessential*.

DO-178 also established the relationship between the software certification process and the other relevant Federal Aviation Regulations (FARs), such as the Type Certification Approval, the Technical Standard Order (TSO) Authorization, and the Supplemental Type Certification.

### DO-178A

While the first version of DO-178 introduced the concept of software certification, the lessons learned from attempting to apply it quickly pointed out the need for revision. RTCA Special Committee 152 was formed to produce this revision, DO-178A, which was published in 1985. DO-178A turned out to be very different from DO-178.

DO-178A described, in systematic and structured detail, software development and verification processes, something that was deemed missing from DO-178. It maintained the safety categories of software applications, *critical*, *essential*, and *nonessential*, but also introduced the associated concept of software certification Levels 1, 2 and 3 corresponding to these criticality categories. This was done to allow for variance between application criticality and software certification level. The idea was that while an entire application would be assigned a criticality level, the level of certification effort for the software making up the application could be reduced according to the system design and implementation techniques. For example, if adequate partitioning of the design could be shown, some of the software of a critical application may only need a Level 2 or 3 certification effort, rather than Level 1, because of its lesser role in the overall functioning of the critical application.

As with DO-178, attempting to apply DO-178A led to many new problems. Interpretation of certification requirements differed from one FAA region to another. Certification deliverables were frequently contended between applicants and the certification authorities and misinterpretation of DO-178A's intent sometimes led to entire software development life cycles being disallowed because they did not follow traditional waterfall-like processes. Lastly, there was an overall lack of understanding and appreciation for the purpose of the certification requirements by the industry in general.

### DO-178B

During the time of DO-178A, the avionics industry was undergoing a major shift from analog to digital systems, and software was being applied in ever larger and more complex applications. Many new

vendors of avionics applications came into existence and were now subject to software certification constraints for the first time. The lack of available DO-178A documentation, training material, development standards, and experienced people became apparent, and the industry again came to the conclusion that a new version of DO-178 was needed to address these issues and incorporate the new lessons learned from application of DO-178A.

To address these concerns, DO-178B was designed to satisfy the following three primary goals.

- Develop objectives for the life-cycle processes
- Provide a description of the activities and design considerations for achieving those objectives
- Provide a description of the evidence indicating the objectives have been satisfied

The intention was to provide enough information and detail as possible in order to lessen the burden and increasing demand on the few experienced software certification people that were available.

DO-178B made two more changes: nomenclature for describing safety criticality was changed from *critical, essential,* and *nonessential* to *catastrophic, hazardous/severe-major, major, minor,* and *no effect;* software Levels *1*, *2*, and *3* were changed to *A* through *E*.

One of the more significant aspects of DO-178B was the change made in software verification requirements. Much greater emphasis was placed on requirements-based testing than in earlier versions. This was done to reduce or eliminate the practice of focusing software testing primarily on structural (i.e., white box) coverage without fully testing the requirements, which would then lead to the need for additional tests. Requirements-based testing, with emphasis on the structure coverage of these functional tests, was seen as a more cost-effective and meaningful way to conduct verification testing.

The emphasis on requirements-based testing resulted in a new and significant focus area in DO-178B to ensure the traceability between requirements, code, and the tests that verify that the code satisfy the requirements. Traceability evidence is required to be thorough and bidirectional, demonstrating that requirements trace forward to code and tests and satisfy requirements, and trace backward from tests to code and to the requirements for which the tests were created. At the higher levels of safety and criticality, DO-178B requires that this traceability evidence show 100% structural code coverage.

DO-178B requires that verification and traceability evidence be made available for audit and analysis by the FAA or their DER to support an assessment of completeness and correctness. While this was also true for prior versions, the enhanced rigor and completeness of verification evidence required by DO-178B has proven to be a significant burden to the avionics industry.

> *Some industry representatives have complained that the certification process requires an inordinate amount of time and expense, with some aspects of that process contributing little or no value. In particular, the FAA has received numerous complaints about software aspects of the certification process.* (Hayhurst et al. 1998)

The human effort involved in producing the degree of systematically rigorous and complete verification and traceability evidence required by DO-178B can be overwhelming. In addition, as the complexity grows, the cost to produce the associated verification evidence grows exponentially with the size of the application. While tools that provide assistance with development, verification, and traceability have been utilized and undergone qualification throughout the time frame of DO-178 and its three versions, the tools designed specifically to meet the demands of DO-178B are being counted on for automated assistance far

beyond those of their predecessors. Consequently, the reliance upon such tools without independent review of their results and artifacts also continues to grow. Thus, the subject of tool qualification takes on additional weight in the context of DO-178B.

### Evolution of Tool Qualification Guidelines

Even though DO-178B was written by a diverse and representative group of industry experts, the application of DO-178B in practice proved to be very difficult, time-consuming, and expensive. To address these concerns, the FAA started the SSAC program (Hayhurst et al. 1998). A detailed survey was widely distributed throughout the portion of the civil aviation community experienced with, and subject to, DO-178B and the issues of software-based product certification. The survey asked respondents detailed and wide-ranging questions about their experiences with DO-178B, what its strong points and weaknesses were, and what the FAA could do to improve how DO-178B was interpreted and applied.

Three workshops were conducted as part of the SSAC program, between January of 1998 and May of 1999, where the survey results were transformed into a set of ten industry recommendations to the FAA. Section J of the final report on the survey results and recommendations (Hayhurst et al. 1999) applied to DO-178B, Section 12.2 on tool qualification. The FAA formally responded to these recommendations in the form of an official letter to the SSAC participants, agreeing to implement the recommendations, and a detailed list of all of the FAA activities that were either ongoing or already completed that addressed each of these ten recommendations. One of these activities was the production of a set of guidelines, FAA Notice N8110.83, specifically addressing tool qualification that were intended to provide clarification to DO-178B, Section 12.2 (FAA 1999). Recently, in June 2003, this FAA Notice was superceded by FAA Order 8110.49, *Software Approval Guidelines*, Chapter 9, *Qualification Of Software Tools Using RTCA/DO-178B* (FAA 2003).

## *Fundamentals of Tool Qualification*

Tool qualification starts by addressing two questions: 1) is a specific tool subject to tool qualification and, if so, 2) what is the tool category? Numerous software tools are used throughout the software development and maintenance life cycle, but the software developer/vendor need only be concerned with a subset of these tools, at least as the tool qualification requirements and evidence of compliance. For that subset of tools, the tool qualification effort involved is determined by the tool's category type. This section focuses on these two fundamental aspects of tool qualification.

### Tool Requiring Qualification

The concept of tool qualification arose in DO-178 as an alternative means of satisfying software verification requirements for certification. The concept was that if a tool was capable of providing sufficient evidence to satisfy DO-178's software verification requirements in an automated manner, and if the software developer was able to negotiate this alternate means of verification compliance with the certification authority, the artifacts produced by the tool would be acceptable in lieu of manual production of the verification evidence being supplanted. The reliability of the tool in question was central to the negotiation of acceptance of this alternative means by the FAA. In order for the artifacts produced by the tool to be relied on and applied for certification credit without further verification review, the tool must be demonstrated to carry out its intended function reliably. DO-178B, Section 12, defines exactly what constitutes tool qualification in this context.

## Tool Categories

The key point in Tool Categories is that only tools whose outputs are to be used directly for certification credit without additional confirmation need to be qualified. During the development of DO178B's Section 12, it was realized that there were two fundamentally different types of software tools that would be governed by tool qualification requirements. These were:

- ***Development Tools.*** Tools whose outputs are part of airborne software and thus can introduce errors.
- ***Verification Tools.*** Tools that cannot introduce errors but may fail to detect them.

The premise was that the safety ramifications of a development tool were very different than those for a verification tool. The development processes and verification evidence required for DO-178B certification directly depends on the level of negative safety impact that software will have on an aircraft's operation. Analogously, the overall effort required to produce the evidence of reliability necessary to qualify a tool also should depend on how directly a tool's incorrect operation can affect the software application being certified. **Table 1** shows the general criteria used for tool qualification as it applies to development and verification tools. Brief explanations of the criteria are provided in **Table 1**. For a more complete description of the criteria and associated tool qualification considerations, the Guidelines for the Qualification of Software Tools Using RTCA/DO-178B are available on the web (http://www2.faa.gov/certification/aircraft/N8110-91.pdf).

**Table 1**. Criteria for Tool Qualification

| # | Criteria | Development | Verification |
|---|---|---|---|
| 1 | Only deterministic tools may be qualified. | Yes | Yes |
| 2 | Qualification should only be for a specific system. | Yes | Yes |
| 3 | Combined tools should be qualified to DO-178B. | Yes | Yes |
| 4 | Software configuration management and software quality assurance process objectives should be applied to tools being qualified. | Yes | Yes |
| 5 | Qualification should satisfy the same objectives as the airborne software. | Yes | No |
| 6 | The software level of the tool may be reduced. | Yes | No |
| 7 | A trial period may be used as a means of qualification. | Yes | Yes |
| 8 | Tool Operational Requirements should be reviewed. | Yes | Yes |
| 9 | Compliance with Tool Operational Requirements under normal operating conditions should be demonstrated. | Yes | Yes |
| 10 | Compliance with Tool Operational Requirements under abnormal operating conditions should be demonstrated. | Yes | No |
| 11 | Requirements-based coverage should be analyzed. | Yes | No |
| 12 | Structural coverage appropriate for the tool's software level should be completed. | Yes | No |
| 13 | Robustness testing appropriate for the tool's software level should be completed. | Yes | No |
| 14 | Potential errors should be analyzed. | Yes | No |

1. Although it is easiest to demonstrate this property if the same output is repeatedly produced from a set of a given set of inputs, the primary concern is for graphical tools that would generate code from a graphical representation in a nondeterministic manner. The ability to establish correctness of the output from the tool is important.

2. Qualification is applied to a particular system certification. Tools cannot be qualified independently of a system certification.

3. Combined development and verification functions, where the output of both the development and the verification functions are being used to eliminate, reduce, or

automate processes of DO-178B, should be qualified irrespective of the other capabilities present in that tool.

4. Defined configuration management (CM) and software quality assurance (SQA) processes should be applied to tools being qualified.

5. For development tools only, the processes and artifacts should be developed to the same level as the airborne software because the resulting artifact will be placed directly into the certified system without manual inspections or reviews.

6. For development tools only, the software level of the tool can be reduced if evidence can be provided to demonstrate that the verification process provides coverage to justify the software level reduction.

7. If the tool has been in use, the historical evidence of the tool can be used as a means for qualification where a verification of the tool output is performed and tool-related problems are analyzed, recorded, and corrected.

8. The Tool Operational Requirements should be reviewed in a manner compliant with the documented quality assurance procedures to ensure correctness, consistency, and completeness.

9. Demonstration of the tool with its Tool Operational Requirements under normal operating conditions is typically shown through the use of documented V&V processes, primarily in the form of test cases or usage scenarios.

10. For development tools only, the tool also must demonstrate compliance with the Tool Operational Requirements under abnormal operating conditions, including external disturbances and selected failures in the environment.

11. For development tools only, requirement-based coverage should be demonstrated and additional tests to complete the coverage of the requirement should be provided, if necessary.

12. For development tools only, structural coverage pertinent to the software level should be demonstrated.

13. For development tools only, robustness testing should be demonstrated for complex data or control flow.

14. For development tools only, analysis of potential errors produced by the tools should be performed to support the demonstration of the validity of the Tool Qualification Plan.

## Tool Qualification Deliverables

The activities to qualify tools used in the product development process need to be planned and documented. The tool qualification process is part of the software certification process, which is part of an overall system certification process, as reflected by **Figure 1**. There are many deliverables required for the overall system, as well as many deliverables required for the DO-178B certification. One of first documents delivered to the certification authorities is a Plan for Software Aspects of Certification (PSAC). The final document is the Software Accomplishment Summary (SAS) that documents compliance with the PSAC. An applicant qualifying a development tool must provide a separate Tool Qualification Plan and Tool Accomplishment Summary referenced by entries in the PSAC and the SAS. An applicant qualifying software verification tools may include information directly in the PSAC and

SAS or may choose to provide a separate Tool Qualification Plan and Tool Accomplishment Summary referenced by entries in the PSAC and the SAS. This is the recommended approach because it provides the added benefit of providing the ability to reference a data package for reuse in subsequent certifications or in different certifications where the usage of the tool can be shown to be identical. Entries are required in the PSAC and SAS for each tool to be qualified.

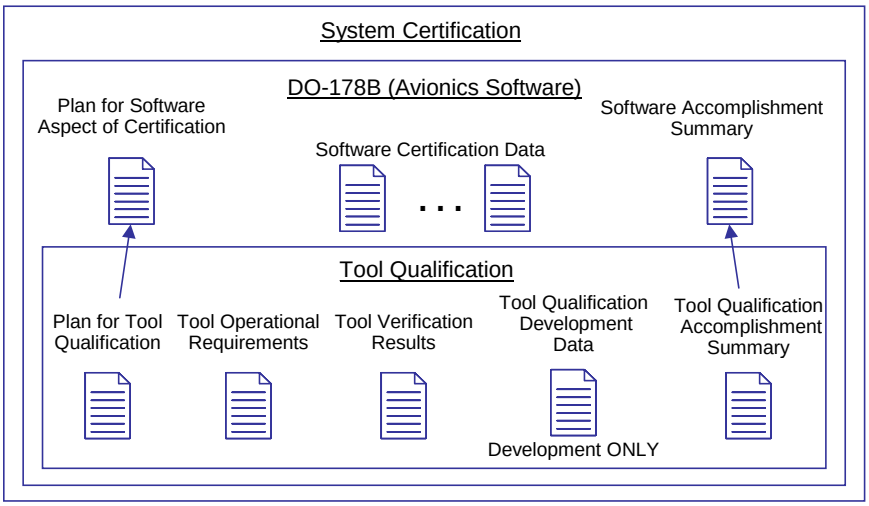


**Figure 1.** Tool Qualification Context

There is no definitive guidance as to the amount of data to be submitted to the FAA for tool qualification. The data submittals vary according to the type of tool being developed. Even though there are some similar requirements for the two tool types, the data requirements for each tool type are different. **Table 2** summarizes the required tool qualification data and indicates whether the data is required to be submitted or available for development or verification tool qualification. This and the related qualification data can supplement the documentation required to support tool qualification. Topic Tool Requiring Qualification through Tool Qualification Accomplishment Summary define and delineate elements of tool qualification.

**Table 2.** Tool Qualification Data

| Tool Qualification Data | Development | Verification |
|---|---|---|
| Plan for Software Aspects of Certification | Submit | Submit |
| Tool qualification plan | Submit | Recommended |
| Tool operational requirements | Available | Available |
| Verification test results | Available | Available |
| Software accomplishment summary | Submit | Submit |
| Tool qualification accomplishment summary | Submit | Submit |
| Tool qualification development data (e.g., design, code, test cases and procedures) | Available | Not required |

## Tool Qualification Plan

The Tool Qualification Plan describes the tool being qualified, provides references to applicable documentation, and identifies the activities and tasks necessary to complete the qualification.

## Tool Operational Requirements

This document provides the usage context, related system component, and requirements that define the set of capabilities and functionalities the tools must meet or provide for its intended use. The verification strategy is defined with each category of requirement. It is often common to include references to user documentation that is delivered with the tools. User documentation tells the user how to interface with the tool and should include instructions and commands for using the software, options available, error messages and resolutions, and limitations of the tool.

### Verification Test Results

The verification test results may reference tool qualification test cases and procedures that define how the tool operational requirements will be verified. Each test case should be traceable to the associated requirements. Each test case should include expected results for each test. The test cases should be executed per the defined verification procedure, and the results should be reported. All tests must be successfully executed and passed for the tool to be validated, or there must be a written justification for why any failure is acceptable. The report of the verification results should be signed (or digitally signed) and dated by the person responsible for performing the tests.

### Tool Qualification Accomplishment Summary

This data item summarizes the results of the tool qualification process, identifies the tool versions and associated operating environment, and describes and references the relevant tool qualification data.

## *Tool Qualification Support for TAF and T-VEC*

TAF integrates various model development and test generation tools to support defect prevention and automated testing of systems and software. Although other modeling tools have been developed for TAF, only those component elements that have tool qualification support are shown in **Figure 2**. TAF supports model analysis and test generation for requirement-based tools like the T-VEC Tabular Modeler (TTM), which is a functional (table-based) modeling tool based on the Software Cost Reduction (SCR) method (Alspaugh et al 1992). TAF also supports model analysis and test generation for design-based modeling, simulation, and code-generation tools such as MATRIXx and MathWorks' Simulink and Stateflow tools. Each of these tools integrates with T-VEC through a translator, which transforms each respective model into a form suitable for processing by T-VEC. Once a model is translated, users can generate tests using T-VEC through a graphical user interface (GUI) or command-line interface. Therefore, for a specific usage of TAF, tool qualification is required for the model tool translator (i.e., Simulink, MATRIXx, or TTM) and for T-VEC. The qualification is categorized as a verification type qualification because the output of the process generates tests that are used to verify the code produced by the autocode generation systems associated with Simulink or MATRIXx, or manually produced code.

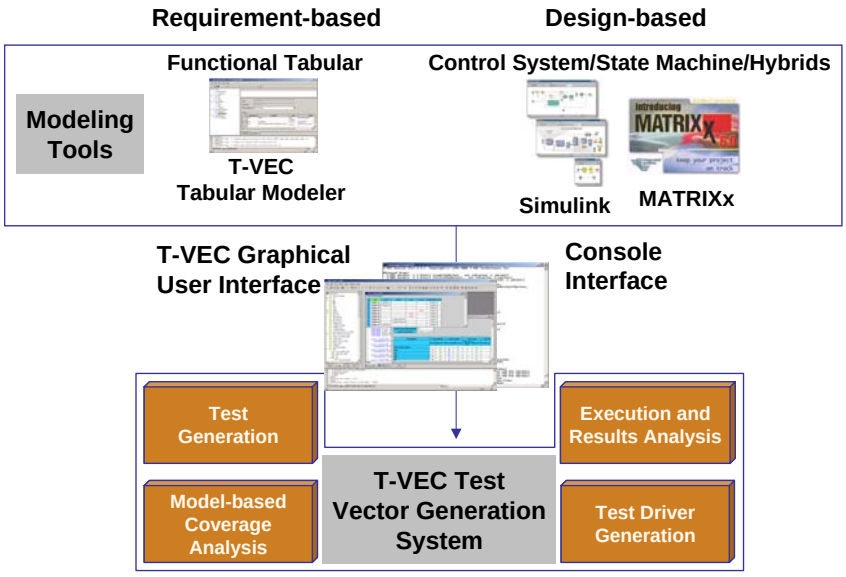


**Figure 2.** TAF Integrated Components

### Process Roles and Flows

This section provides an overview of the process roles and flows for using T-VEC in conjunction with the modeling tool. It describes how the tools are used to develop requirement models in TTM, translate the models into a form suitable for test vector generation, generate test vectors, generate test drivers, and perform execution and results analysis.

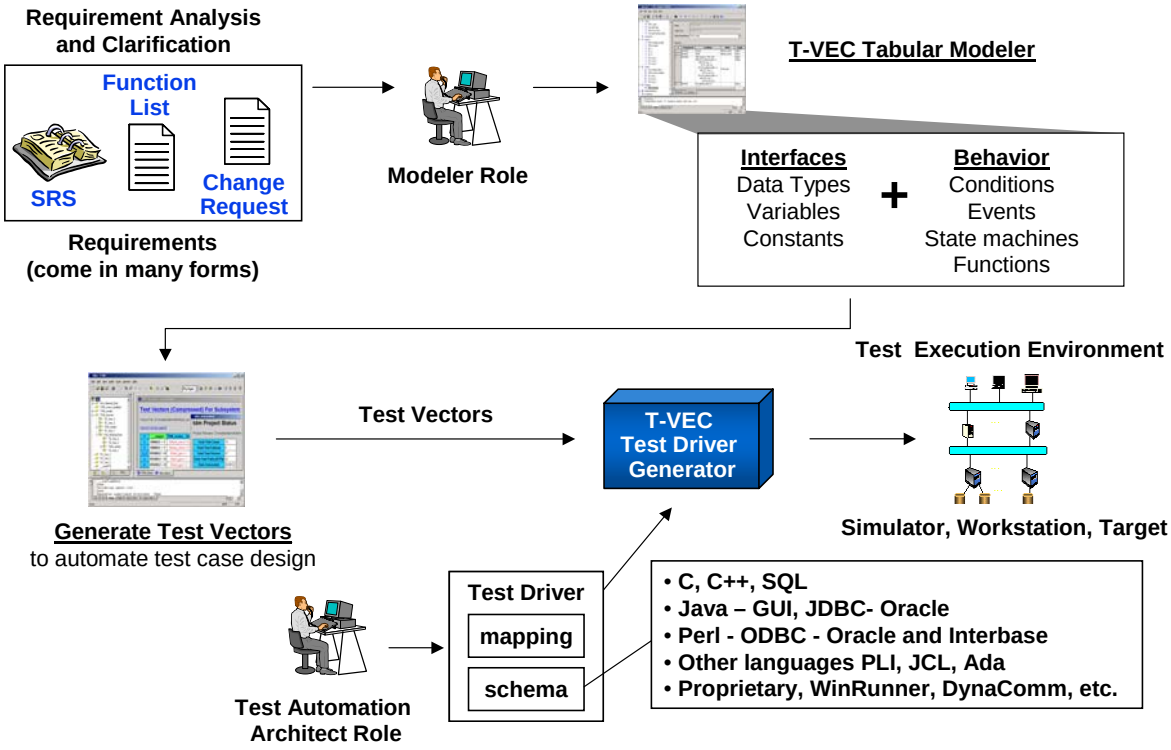


**Figure 3**. Process Roles and Flows

**Figure 3** illustrates a typical process for using T-VEC with TTM. The process identifies the typical organizational roles of a requirements engineer (sometimes referred to as a system engineer who performs requirement analysis and document requirements), designer/implementer (system/software architecture who performs design and implementation), and test engineer (performs verification and some validation). Team members can perform one or more roles.

In this example, TTM is used to develop the verification models. Verification modeling typically exposes requirements weaknesses or problems very early. The modeling process and tools support defect prevention very early in the development process when it is least expensive to resolve. Testing the integrity of the requirements through modeling and model analysis becomes continuous and helps ensure the best possible starting point for downstream development. As the system design and implementation become available, the verification models are used as the basis for automated test case and test driver generation. This approach helps ensure that requirements are well formed through modeling and model analysis and then leverages the requirements model in automatic test generation.

For design-based modeling approaches, the process tends to resemble the illustration in **Figure 4**. Simulink/Stateflow and MATRIXx are hybrid, control system modeling and code generation tools. In this scenario, models undergo translation and static analysis to verify their integrity. Model problems are reported to the engineer responsible for constructing the model for immediate correction. Once modeling is complete, the model is used as the basis for developing tests. Through dynamic analysis (i.e., execution) of the system, anomalies in the model and implementation can be identified and corrected.

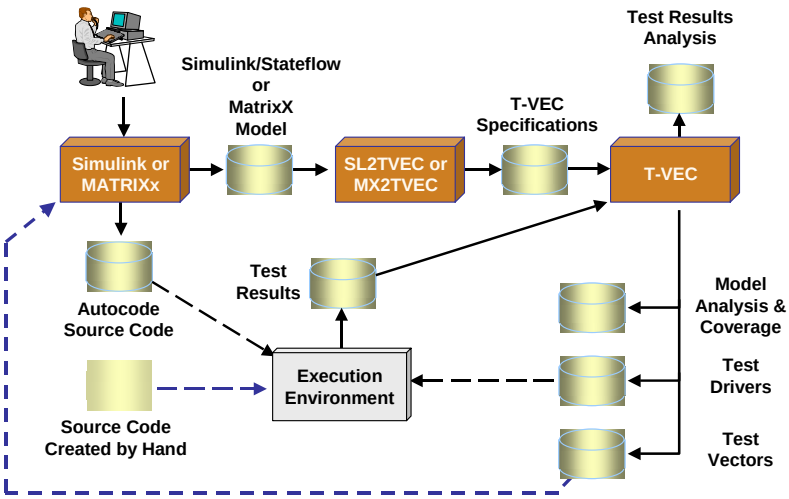


**Figure 4.** Simulink/Stateflow and MATRIXx Modeling Process Flow

## Application of TAF and T-VEC

Depending on the software level of the system being considered for certification, the decision flow shown in **Figure 5** may be required to provide evidence that the model is defect free and that the generated tests provide the required level of code coverage. Details associated with several of these steps are provided below. The process is a follows:

- Construct a model in Simulink, MATRIXx, or TTM.
- Check the model for defects and iteratively correct the model if there are defects (see Process Roles and Flows for details).
- Construct the code. This can be a manual process or can be supported using autocode generation capabilities supported by tools like Simulink and MATRIXx.
- Generate the tests.
- Execute the tests through instrumented code. This may be an optional step (see Code Coverage for details).
- Check to ensure that the tests provide adequate coverage (e.g., MC/DC coverage); if adequate coverage is not achieved, additional tests must be generated.
- Check to ensure that all tests pass.
- Execute tests against target code.
- Check to ensure that all tests pass.
- If tests do not pass, perform test failure analysis, and correct the code or model.

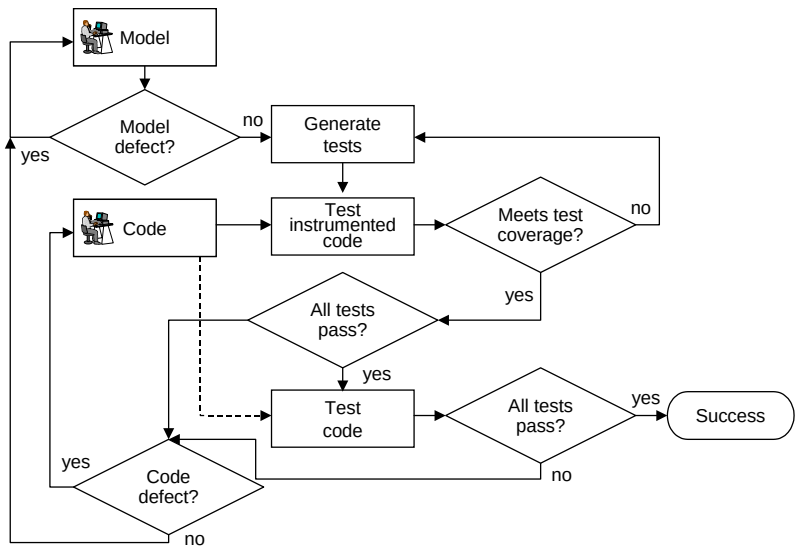


**Figure 5.** Verification Decision Flow

### Model Defect Analysis

One distinguishing characteristic of the T-VEC system is its ability to identify model defects. Consider **Figure 6**, which shows a simple Simulink model, with the two highlighted paths. The model analysis of T-VEC ensures all paths (two in this case) are valid, which means that code

generated from the model is reachable. Without this capability, models can be used to generate code automatically, but the results of executing that code under certain conditions are undefined. This particular capability provides certification authorities with increased confidence as to the integrity of the model.

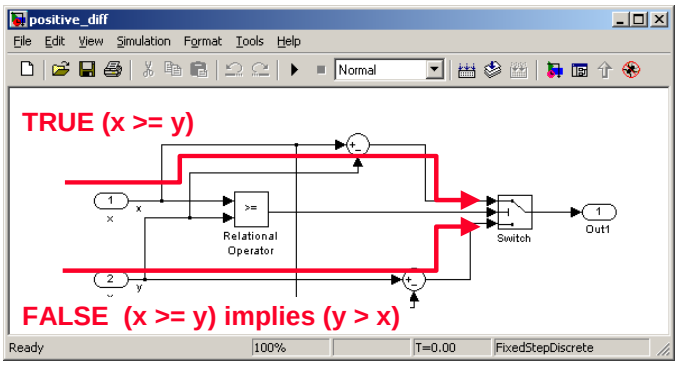


**Figure 6**. Model Paths

## Code Coverage

Another important criterion for critical software is to provide assurance that the tests provide the required level of code coverage. There are various ways to provide this assurance, and there are some qualified tools that provide code coverage measurements. The typical process is to instrument the code that is produced manually or through code generation, then execute the tests against the instrumented code and assess the code coverage. Consider the model shown in **Figure 7** and the associated code coverage information shown in **Figure 8**, which indicates that 12 out of the 12 paths were covered by the generated tests. If all tests pass, and there is complete code coverage, then there is a strong argument that the code fully satisfies the specified functionality of the model. The final step for testing the code is to run the same tests through target code that will be the final certified code. If all tests pass, there is a strong argument that the code is suitable for certification.

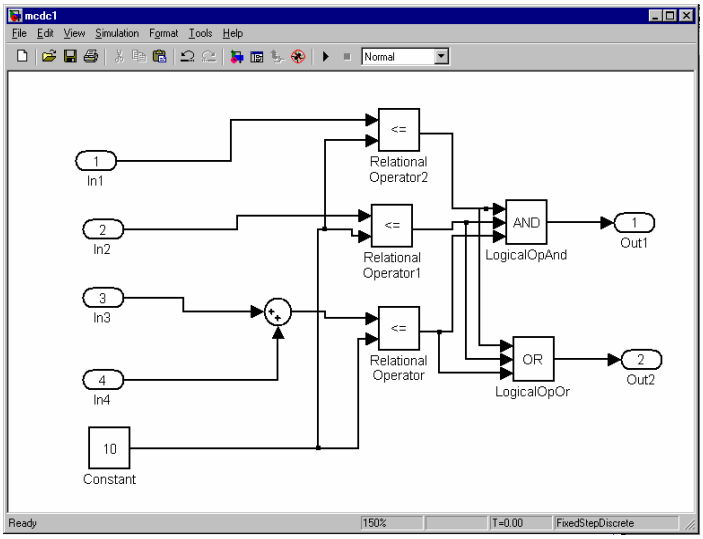


**Figure 7.** Simple Model for Code Coverage Example

## Qualification of T-VEC

T-VEC has been used to support verification and validation of flight-critical, real-time embedded systems since 1989. T-VEC was applied to a portion of a Traffic and Collision Avoidance System (TCAS), which was first FAA-certified in March of 1990. T-VEC was applied to the entire MD90 (McDonnell Douglas) Electrical Power System Variable Speed Constant Frequency (VSCF) system that was FAA-certified in January of 1995. T-VEC also was used in developing component libraries for a family of avionics display systems. T-VEC also has been used in FDA certifications. T-VEC has an up-to-date qualification suite that was applied in all prior FAA and FDA certifications. The qualification suite is compliant with FAA *Software Approval Guidelines*, 8110.49, Chapter 9, Qualification Of Software Tools Using RTCA/DO-178B.

The T-VEC qualification package includes a set of operational requirements that trace to formal specifications and associated tests to verify each component of the T-VEC system, including the compiler, test generator, coverage analysis, and test driver components. In addition, tests are performed to demonstrate that the various operations and parameters that control these components operate properly.

Analysis has been performed to ensure that the supported data types, language constructs, and a comprehensive set of subsystem integration tests have been verified. The outputs are verified manually, and all source specifications, tests, and test results are controlled within a CM system.

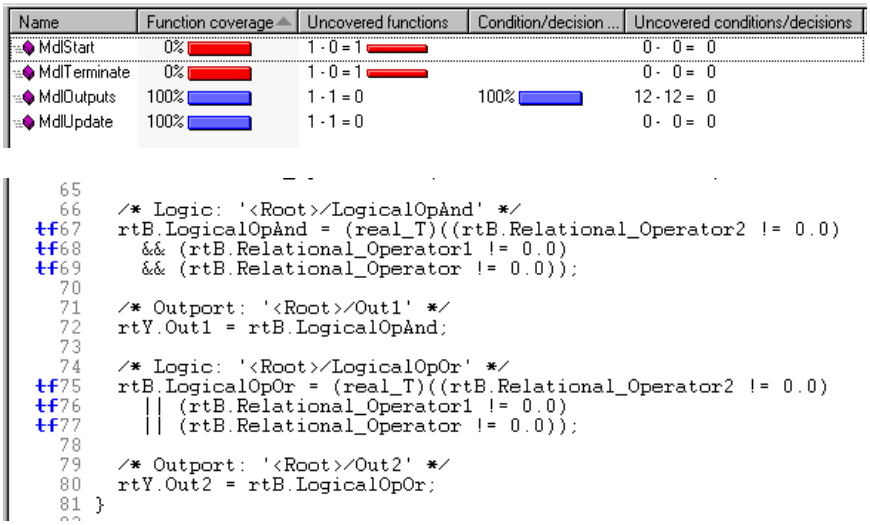


**Figure 8.** Coverage Analysis Screenshot[1]

Screenshots from Bullseye Coverage

## Qualification of TTM

The key capabilities required for normal use of the TTM modeling system are the primary focus of the TTM qualification. Functionality of the tools that contribute to the TTM modeling and its related integration to support test vector generation, requirement-based coverage analysis, and test driver generation are in scope of the qualification. All other user functionality that is not directly related to the production of test vectors and test drivers is not in the scope of the TTM qualification (e.g., model searching features).

The primary focus for the TTM qualification is to ensure that the model translator provides a well-formed representation of each type of modeling construct. The TTM model translator converts a model, which is composed of tables of functions, conditions, events, and modes, into a test specification model. The TTM modeling language and constructs rely on a subset of the capabilities of the T-VEC system because the TTM language is currently not as expressive as the T-VEC specification language.

In a manner similar to the T-VEC qualification, the specification and associated tests cover the modeling language and data types that are mapped to the constructs of T-VEC specification language. The verification of these constructs has been verified in the T-VEC qualification. The purpose of the tests associated with these requirements is to demonstrate that each construct for each data type is properly translated into a valid T-VEC representation. Requirement traceability has been verified to ensure that each specified TTM requirement is translated into a T-VEC specification that will result in test vectors to cover that requirement. The basic approach to the specification supports automated testing, with automated results comparison. The results have been validated manually, and all artifacts are configuration controlled.

## Qualification of Simulink and Stateflow Translator

The tool qualification for the Simulink and Stateflow translator is under development at the time of writing this document, but the philosophy of this translator is the same as that of the TTM translator. The objective is to ensure that all constructs that can be represented in the Simulink and Stateflow are completely and consistently translated to a T-VEC specification. In addition, there are a number of

[1] Use of the Bullseye coverage tool shown in Figure 8 does not imply endorsement of this tool.

translation features that can be specified in the translation interface to enable various types of structural and decision coverage over various modeling constructs of the Simulink and Stateflow language. The following summarizes some verification categories for the tests of the qualification suite:

- Primitive blocks (modeling constructs) tests exercised with coverage options
- Subsystem integration coverage tests
- Signal range propagation tests
- Inline tests
- Test driver generation tests
- Test sequence tests

### Qualification of MATRIXx Translator

The tool qualification for MATRIXx was developed several years ago, and tool development was abandoned. However, in 2003, National Instruments purchased the tool. As the MATRIXx development is resurrected by National Instruments, the associated qualification suite for MATRIXx will have the same type and level of completeness as that of the Simulink and Stateflow qualification package.

## Qualifying TAF and T-VEC for a Project

This section summarizes the steps for planning, executing, and documenting the key steps to support a tool qualification on a project. The process for qualifying TAF and T-VEC on a project involves the following:

- Reference a Tool Qualification Plan for each tool to be used in the certification process in the PSAC. Indicate that the tool type is verification.
- Ensure that the intended use of each tool is compliant with the Tool Qualification Requirements for that tool used to support the certification.
- Demonstrate and document that the verification evidence for the software involved in the certification has been properly verified by qualified tools that have applicable tool qualification accomplishment summaries.
- Ensure that the configuration index includes references to the versions of the tools that have been supplied in the tool qualification accomplishment summary.
- Provide a tool qualification accomplishment summary for each tool with the Software Accomplishment Summary at the completion of the certification.

## Summary

These guidelines describe the history and evolution of the certification process. The increased complexity of systems has led to increased complexity and cost of the verification process. This has increased the need and associated use of automated tools to support the development and verification for certification of software-intensive systems. 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. This document provides a summary of these tool qualification requirements and explains how TAF and T-VEC tools have been developed with tool qualification suites that should aid applicants in the certification process. Finally, it is important to remember that the burden of tool qualification is on the organization applying for certification, rather than on the vendor of the

software tool being qualified, because a tool's application and reliability is directly related to the product developer's use of the tool rather than the generic capabilities of the tool. However, the existing tool qualification suites should reduce the cost of developing these tool qualification suites significantly.

## For More Information

Members with general questions or comments on any of the topics in this paper or related topics, or members interested in applying TAF with Consortium assistance, should contact the author or their member account director (see http://www.software.org/pub/keycontacts.asp).

For more on TAF, see the Consortium's TAF website at http://www.software.org/pub/taf/testing.html, or contact the authors.

## References


| | |
|---|---|
| Alspaugh, T.A., S.R. Faulk, K.H. Britton, R.A. Parker, D.L. Parnas, and J.E. Shore 1992 | "Software requirements for the A-7E aircraft," Tech. Rep. NRL/FR/5546-92-9194, Naval Research Lab., Washington DC. |
| Busser, R.D., M.R. Blackburn, A.M. Nauman 2001 | *Automated Model Analysis and Test Generation for Flight Guidance Mode Logic*, Digital Avionics System Conference. |
| Hayhurst, Kelly J., C. Michael Holloway, Cheryl A. Dorsey, John C. Knight, Nancy G. Leveson, G. Frank McCormick, and Jeffery C. Yang 1998 | *Streamlining Software Aspects of Certification: Technical Team Report on the First Industry Workshop*. NASA/TM-1998-207648, April. |
| Hayhurst, Kelly J., Cheryl A. Dorsey, John C. Knight, Nancy G. Leveson and G. Frank McCormick 1999 | *Streamlining Software Aspects of Certification: Report on the SSAC Survey*, NASA/TM-1999-209519, August. |
| Johnson, Leslie A. (Schad) 1998 | *DO-178B,"Software Considerations in Airborne Systems and Equipment Certification"* STSC Crosstalk, October. http://www.stsc.hill.af.mil/crosstalk/1998/10/ |

| | |
|---|---|
| Kelly, V., E.L. Safford, M. Siok, M. Blackburn 2001 | *Requirements Testability and Test Automation*, Lockheed Martin Joint Symposium, June. |
| Radio Technical Corporation for Aeronautics Special Committee 167 (RTCA) 1992 | DO-178B/ED-12B - *Software Considerations in Airborne Systems and Equipment Certification*, December. |
| Salmon, David 1993 | *Assemblers And Loaders*, Ellis Horwood Ltd (Ellis Horwood Series in Computers and Their Applications) Market Cross House, Cooper Street, Chichester, PO19 1EB, West Sussex, UK. ISBN 0130525642. |
| Statezni, David 2000 | Test Automation Framework, State-based and Signal Flow Examples, *Twelfth Annual Software Technology Conference.* |
| 2001 | T-VEC's Test Vector Generation System, *Software Testing & Quality Engineering*, May/June. |
| U.S. Air Force 2001 | Global Air Traffic Management and Navigation Safety Certification For USAF Aircraft, Air Force Policy Directive 63-13, March. |
| U.S. DOT (FAA) 1999 | Federal Aviation Administration, Order 8110.83 -*Guidelines For The Qualification Of Software Tools Using RTCA/DO-178B*, April. |
| 2003 | Federal Aviation Administration, Order 8110.49 - *Software Approval Guidelines*, June. |
| U.S. Food and Drug Administration (FDA) 2002 | *General Principles of Software Validation; Final Guidance for Industry and FDA Staff*, January. http://www.fda.gov/cdrh/comp/guidance/938.html |