

RTCA, Inc.
1150 18th Street, NW, Suite 910
Washington, D.C. 20036

Software Tool Qualification Considerations

RTCA DO-330
December 13, 2011

Prepared by: SC-205

© 2011 RTCA, Inc.



Documents

Document Sharing Hub

Copies of this document may be obtained from

RTCA, Inc.

Telephone: 202-833-9339

Facsimile: 202-833-9434

Internet: www.rtca.org

Please visit the RTCA Online Store for document pricing and ordering information.

FOREWORD

This report was prepared by RTCA Special Committee 205 (SC-205) and EUROCAE Working Group 71 (WG-71) and approved by the RTCA Program Management Committee (PMC) on December 13, 2011.

RTCA, Incorporated is a not-for-profit corporation formed to advance the art and science of aviation and aviation electronic systems for the benefit of the public. The organization functions as a Federal Advisory Committee and develops consensus-based recommendations on contemporary aviation issues. RTCA's objectives include but are not limited to:

- coalescing aviation system user and provider technical requirements in a manner that helps government and industry meet their mutual objectives and responsibilities;
- analyzing and recommending solutions to the system technical issues that aviation faces as it continues to pursue increased safety, system capacity, and efficiency;
- developing consensus on the application of pertinent technology to fulfill user and provider requirements, including development of minimum operational performance standards for electronic systems and equipment that support aviation; and
- assisting in developing the appropriate technical material upon which positions for the International Civil Aviation Organization and the International Telecommunication Union and other appropriate international organizations can be based.

The organization's recommendations are often used as the basis for government and private sector decisions as well as the foundation for many Federal Aviation Administration Technical Standard Orders.

Since the RTCA is not an official agency of the United States Government, its recommendations may not be regarded as statements of official government policy unless so enunciated by the U.S. government organization or agency having statutory jurisdiction over any matters to which the recommendations relate.

This Page Intentionally Left Blank

TABLE OF CONTENTS

1.0 INTRODUCTION.....	1
1.1 PURPOSE	1
1.2 SCOPE	1
1.3 HOW TO USE THIS DOCUMENT.....	1
1.4 DOCUMENT OVERVIEW.....	2
2.0 PURPOSE OF TOOL QUALIFICATION.....	5
3.0 CHARACTERISTICS OF TOOL QUALIFICATION.....	7
3.1 QUALIFICATION LEVELS	7
3.2 QUALIFICATION STAKEHOLDERS	7
4.0 TOOL QUALIFICATION PLANNING PROCESS	9
4.1 DETERMINING THE TOOL QUALIFICATION NEEDS	9
4.2 TOOL LIFE CYCLE DEFINITION.....	9
4.3 TOOL PLANNING PROCESS OBJECTIVES	10
4.4 TOOL PLANNING PROCESS ACTIVITIES.....	10
5.0 TOOL DEVELOPMENT LIFE CYCLE AND PROCESSES.....	13
5.1 TOOL OPERATIONAL REQUIREMENTS DEFINITION PROCESS	14
5.1.1 <i>Tool Operational Requirements Objectives.....</i>	<i>14</i>
5.1.2 <i>Tool Operational Requirements Activities.....</i>	<i>14</i>
5.2 TOOL DEVELOPMENT PROCESSES	15
5.2.1 <i>Tool Requirements Process</i>	<i>15</i>
5.2.2 <i>Tool Design Process.....</i>	<i>16</i>
5.2.3 <i>Tool Coding Process</i>	<i>17</i>
5.2.4 <i>Tool Integration Process</i>	<i>18</i>
5.2.5 <i>Tool Development Process Traceability.....</i>	<i>18</i>
5.3 TOOL OPERATIONAL INTEGRATION PROCESS.....	19
5.3.1 <i>Tool Operational Integration Process Objectives</i>	<i>19</i>
5.3.2 <i>Tool Operational Integration Activities.....</i>	<i>19</i>
6.0 TOOL VERIFICATION PROCESS.....	21
6.1 TOOL VERIFICATION PROCESS	21
6.1.1 <i>Purpose of Tool Verification Process.....</i>	<i>21</i>
6.1.2 <i>Overview of Tool Verification Activities.....</i>	<i>21</i>
6.1.3 <i>Reviews and Analyses</i>	<i>22</i>
6.1.4 <i>Tool Testing.....</i>	<i>25</i>
6.1.5 <i>Tool Verification Process Traceability.....</i>	<i>28</i>
6.2 TOOL OPERATIONAL VERIFICATION AND VALIDATION PROCESS	29
6.2.1 <i>Tool Operational Verification and Validation Process Objectives</i>	<i>29</i>
6.2.2 <i>Tool Operational Verification and Validation Process Activities</i>	<i>29</i>
7.0 TOOL CONFIGURATION MANAGEMENT PROCESS.....	31
7.1 TOOL CONFIGURATION MANAGEMENT PROCESS OBJECTIVES	31
7.2 TOOL CONFIGURATION MANAGEMENT PROCESS ACTIVITIES	32
7.2.1 <i>Configuration Identification</i>	<i>32</i>
7.2.2 <i>Baselines and Traceability</i>	<i>32</i>
7.2.3 <i>Problem Reporting, Tracking, and Corrective Action.....</i>	<i>33</i>
7.2.4 <i>Change Control</i>	<i>33</i>
7.2.5 <i>Change Review</i>	<i>34</i>
7.2.6 <i>Configuration Status Accounting.....</i>	<i>34</i>
7.2.7 <i>Archive, Retrieval, and Release.....</i>	<i>34</i>
7.3 DATA CONTROL CATEGORIES	35
7.4 TOOL LIFE CYCLE ENVIRONMENT CONTROL	35

8.0	TOOL QUALITY ASSURANCE PROCESS.....	37
8.1	TOOL QUALITY ASSURANCE PROCESS OBJECTIVES	37
8.2	TOOL QUALITY ASSURANCE PROCESS ACTIVITIES	37
8.3	TOOL CONFORMITY REVIEW	38
9.0	TOOL QUALIFICATION LIAISON PROCESS.....	41
10.0	TOOL QUALIFICATION DATA.....	43
10.1	TOOL QUALIFICATION LIAISON PROCESS AND OTHER INTEGRAL PROCESSES DATA.....	43
10.1.1	Tool-Specific Information in PSAC.....	43
10.1.2	Tool Qualification Plan (TQP)	44
10.1.3	Tool Development Plan.....	44
10.1.4	Tool Verification Plan.....	45
10.1.5	Tool Configuration Management Plan	46
10.1.6	Tool Quality Assurance Plan	47
10.1.7	Tool Requirements Standards	47
10.1.8	Tool Design Standards.....	47
10.1.9	Tool Code Standards	48
10.1.10	Tool Life Cycle Environment Configuration Index.....	48
10.1.11	Tool Configuration Index.....	49
10.1.12	Tool Problem Reports	49
10.1.13	Tool Configuration Management Records.....	50
10.1.14	Tool Quality Assurance Records	50
10.1.15	Tool Accomplishment Summary.....	50
10.1.16	Tool-Specific Information in Software Accomplishment Summary	51
10.1.17	Tool-Specific Information in Software Life Cycle Environment Configuration Index (SECI)	51
10.2	TOOL QUALIFICATION DATA PRODUCED DURING THE TOOL DEVELOPMENT PROCESSES AND CORRESPONDING VERIFICATION ACTIVITIES	52
10.2.1	Tool Requirements	52
10.2.2	Tool Design Description.....	52
10.2.3	Tool Source Code.....	53
10.2.4	Tool Executable Object Code	53
10.2.5	Tool Verification Cases and Procedures	53
10.2.6	Tool Verification Results.....	53
10.2.7	Trace Data	54
10.3	TOOL QUALIFICATION DATA PRODUCED DURING THE TOOL OPERATIONAL REQUIREMENTS PROCESS, THE TOOL OPERATIONAL INTEGRATION PROCESS, AND CORRESPONDING VERIFICATION AND VALIDATION ACTIVITIES...	54
10.3.1	Tool Operational Requirements.....	54
10.3.2	Tool Installation Report.....	55
10.3.3	Tool Operational Verification and Validation Cases and Procedures.....	55
10.3.4	Tool Operational Verification and Validation Results	55
11.0	ADDITIONAL CONSIDERATIONS FOR TOOL QUALIFICATION.....	57
11.1	MULTI-FUNCTION TOOLS	57
11.2	PREVIOUSLY QUALIFIED TOOLS	58
11.2.1	Reuse of Previously Qualified Tools.....	59
11.2.2	Changes to the Tool Operational Environment	60
11.2.3	Changes to Previously Qualified Tool	60
11.3	QUALIFYING COTS TOOLS	61
11.3.1	Approach for COTS Tool Qualification.....	61
11.3.2	Activities for the Tool Developer	61
11.3.3	Tool Qualification Activities for the Tool User.....	63
11.3.4	Coordination of Developer and User Activities.....	66
11.4	SERVICE HISTORY	66
11.4.1	Aspects of Tool Service History	66
11.4.2	Tool Service History Activities.....	67

11.5	ALTERNATIVE METHODS FOR TOOL QUALIFICATION.....	68
ANNEX A	TOOL QUALIFICATION OBJECTIVES.....	69
ANNEX B	ACRONYMS AND GLOSSARY OF TERMS	81
APPENDIX A	MEMBERSHIP LIST.....	A-1
APPENDIX B	EXAMPLE OF DETERMINATION OF APPLICABLE TOOL QUALIFICATION LEVELS	B-1
APPENDIX C	FREQUENTLY ASKED QUESTIONS RELATED TO TOOL QUALIFICATION FOR ALL DOMAINS	C-1
1.0	APPENDIX C INTRODUCTION	C-1
1.1	FAQ C.1: WHAT DOES “PROTECTION” MEAN FOR TOOLS AND WHAT ARE SOME MEANS TO ACHIEVE IT?C-1	
1.2	FAQ C.2: WHAT ARE EXTERNAL COMPONENTS AND HOW DOES ONE ASSESS THEIR CORRECTNESS?	C-2
1.2.1	WHAT ARE THE EXTERNAL COMPONENTS OF A TOOL?	C-2
1.2.2	APPLICATION OF STRUCTURAL COVERAGE ANALYSIS FOR TOOL EXTERNAL COMPONENTS	C-2
1.3	FAQ C.3: HOW CAN ONE MAXIMIZE REUSABILITY OF TOOL QUALIFICATION DATA?	C-3
APPENDIX D	FREQUENTLY ASKED QUESTIONS RELATED TO TOOL QUALIFICATION FOR AIRBORNE SOFTWARE AND CNS/ATM SOFTWARE DOMAINS.....	D-1
1.0	APPENDIX D INTRODUCTION	D-1
1.1	FAQ D.1: WHY ARE THE TERMS “VERIFICATION TOOL” AND “DEVELOPMENT TOOL” NOT USED TO DESCRIBE TOOLS THAT MAY BE QUALIFIED?	D-1
1.2	FAQ D.2: CAN TQL BE REDUCED?	D-2
1.3	FAQ D.3: WHEN DO TARGET COMPUTER EMULATORS OR SIMULATORS NEED TO BE QUALIFIED?	D-2
1.3.1	DO-178C/DO-278A BACKGROUND AND GUIDANCE	D-2
1.3.2	USE OF AN EMULATOR OR SIMULATOR	D-3
1.3.3	DEMONSTRATION OF EQUIVALENCE.....	D-3
1.3.4	APPROACH FOR QUALIFYING EMULATOR/SIMULATOR.....	D-4
1.4	FAQ D.4: WHAT CREDIT CAN BE GRANTED FOR TOOLS PREVIOUSLY QUALIFIED USING DO-178B/DO-278?	D-4
1.5	FAQ D.5: WHAT IS THE RATIONALE FOR TOOL QUALIFICATION CRITERIA DEFINITION?	D-5
1.5.1	BACKGROUND.....	D-5
1.5.2	APPROACH TO DEFINE THE APPLICABLE TQL	D-5
1.5.3	TOOL CRITERIA DEFINITION.....	D-6
1.5.3.1	COMPARISON BETWEEN DO-178B/DO-278 TOOL CATEGORIES AND DO-178C/DO-278A TOOL QUALIFICATION CRITERIA	D-6
1.5.3.2	RATIONALE FOR INTRODUCING CRITERIA 2	D-6
1.5.3.3	CONSIDERATIONS ABOUT CRITERIA SELECTION AND EXAMPLES.....	D-7
1.5.3.3.1	APPLICATION OF CRITERIA 1, 2, AND 3	D-7
1.5.3.3.2	EXAMPLES OF CRITERIA 2 OR 3 DETERMINATION	D-8
1.5.3.3.3	MULTILAYERED SAFETY APPROACH AND TOOL QUALIFICATION CRITERIA	D-8
1.5.3.3.4	TQLs.....	D-9
1.6	FAQ D.6: WHAT “VERIFICATION TOOL” QUALIFICATION IMPROVEMENTS WERE MADE IN DO-178C/DO-278A?	D-10
1.6.1	<i>Qualification Criteria for Verification Tools in DO-178B.....</i>	<i>D-10</i>
1.6.2	<i>Clarifications and Improvements for TQL-5.....</i>	<i>D-10</i>
1.6.2.1	<i>Scope of the qualification.....</i>	<i>D-10</i>
1.6.2.2	<i>Determinism.....</i>	<i>D-11</i>
1.6.2.3	<i>TOR Clarification</i>	<i>D-11</i>
1.6.2.4	<i>Tool Integration in the Operational Environment</i>	<i>D-11</i>
1.6.2.5	<i>Tool Verification</i>	<i>D-11</i>
1.6.2.6	<i>Tool Validation</i>	<i>D-11</i>
1.6.2.7	<i>Tool Configuration Management.....</i>	<i>D-12</i>

1.6.2.8	Tool Quality Assurance.....	D-12
1.6.2.9	Certification/Approval Liaison Process for Tool Qualification	D-12
1.6.3	Conclusion.....	D-12
1.7	FAQ D.7: HOW MIGHT ONE USE A QUALIFIED TOOL TO VERIFY THE OUTPUTS OF AN UNQUALIFIED TOOL? ...	D-12
1.7.1	SCOPE AND LIMITATIONS	D-13
1.7.2	DISCUSSION.....	D-14
1.7.2.1	COVERAGE OF UNQUALIFIED TOOL OUTPUT VERIFICATION OBJECTIVES.....	D-15
1.7.2.2	OPERATING CONDITIONS OF THE QUALIFIED TOOL	D-15
1.7.2.3	COMMON CAUSE AVOIDANCE	D-15
1.7.2.4	PROTECTION BETWEEN TOOLS.....	D-16
1.8	FAQ D.8: HOW MIGHT ONE USE A QUALIFIED AUTOCODE GENERATOR?	D-16
1.8.1	TERMINOLOGY	D-16
1.8.2	PURPOSE AND LIMITATIONS OF THIS FAQ	D-16
1.8.3	DISCUSSION.....	D-17
1.8.3.1	SOURCE CODE VERIFICATION OBJECTIVES.....	D-17
1.8.3.2	EXECUTABLE OBJECT CODE VERIFICATION OBJECTIVES	D-18
1.8.3.2.1	SCENARIOS USING THE ACG	D-18
1.8.3.3	COVERAGE OF SOFTWARE STRUCTURE OBJECTIVES.....	D-23
1.8.3.3.1	VERIFICATION OF AIRBORNE (OR CNS/ATM) CODE STRUCTURE.....	D-23
1.8.3.3.2	ABSENCE OF UNINTENDED FUNCTIONS	D-23
1.8.3.3.3	THOROUGHNESS OF REQUIREMENTS-BASED TESTING	D-24
1.9	FAQ D.9: IS QUALIFICATION OF A MODEL SIMULATOR NEEDED?	D-24

LIST OF FIGURES

FIGURE 1-1	DOCUMENT OVERVIEW	3
FIGURE 5-1	OVERALL TOOL DEVELOPMENT LIFE CYCLE.....	9
FIGURE 11-1	PREVIOUSLY QUALIFIED TOOLS EVALUATION	55
FIGURE D-1	TOOL RELATIONSHIP EXAMPLE	D-14
FIGURE D-2	DEVELOPMENT PROCESS WITH AUTOCODE GENERATOR	D-17

LIST OF TABLES

TABLE 11-1	TYPICAL TOOL DEVELOPER OBJECTIVES	62
TABLE 11-2	TYPICAL TOOL USER OBJECTIVES.....	64
TABLE T-0	TOOL OPERATIONAL PROCESSES	66
TABLE T-1	TOOL PLANNING PROCESS	67
TABLE T-2	TOOL DEVELOPMENT PROCESSES	68
TABLE T-3	VERIFICATION OF OUTPUTS OF TOOL REQUIREMENTS PROCESSES	69
TABLE T-4	VERIFICATION OF OUTPUTS OF TOOL DESIGN PROCESS.....	70
TABLE T-5	VERIFICATION OF OUTPUTS OF TOOL CODING & INTEGRATION PROCESSES	71
TABLE T-1	TOOL PLANNING PROCESS	71
TABLE T-2	TOOL DEVELOPMENT PROCESSES	72
TABLE T-3	VERIFICATION OF OUTPUTS OF TOOL REQUIREMENTS PROCESSES	73
TABLE T-4	VERIFICATION OF OUTPUTS OF TOOL DESIGN PROCESS.....	74
TABLE T-5	VERIFICATION OF OUTPUTS OF TOOL CODING & INTEGRATION PROCESS.....	75
TABLE T-6	TESTING OF OUTPUTS OF INTEGRATION PROCESS	76
TABLE T-7	VERIFICATION OF OUTPUTS OF TOOL TESTING	77
TABLE T-8	TOOL CONFIGURATION MANAGEMENT PROCESS.....	78
TABLE T-9	TOOL QUALITY ASSURANCE PROCESS	79
TABLE T-10	TOOL QUALIFICATION LIAISON PROCESS.....	80
TABLE D-1	DO-178B/DO-278 AND DO-178C/DO-278A TQL CORRELATION.....	D-4
TABLE D-2	DO-178B/DO-278 TOOL CATEGORIES AND DO-178C/DO-278A TOOL QUALIFICATION CRITERIA COMPARISON	D-6
TABLE D-3	TQL SUMMARY.....	D-9
TABLE D-4	OVERVIEW OF SOME AUTOCODE GENRATOR USE SCENARIOS	D-19

1.0 INTRODUCTION

1.1 Purpose

The purpose of this document is to provide tool qualification guidance. Additionally, clarification material is provided in the form of Frequently Asked Questions (FAQs).

1.2 Scope

Software tools are widely used in multiple domains, to assist in developing, verifying, and controlling other software. In the context of this document a tool is 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. Examples are automated code generators, compilers, test tools, and modification management tools. This document explains the process and objectives for qualifying tools.

This document was developed for the following reasons:

- a. Tools are different from the software using the tools and form a unique domain; therefore, tool-specific guidance for both tool developers and tool users is needed.
- b. Tools are often developed by teams other than those who use the tools to develop software. These tool development teams frequently do not have software guidance background (examples of guidance include DO-178C or DO-278A). This tool-specific document benefits tool development teams and helps them avoid confusion and misinterpretation.
- c. This document provides guidance for airborne and ground-based software. It may also be used by other domains, such as automotive, space, systems, electronic hardware, aeronautical databases, and safety assessment processes.

1.3 How to Use This Document

When a domain or a project desires to use this document, the following should be considered.

- a. To offer maximum applicability and flexibility, five tool qualification levels (TQLs) are defined; however, not all levels may be applicable for every domain. Each domain should determine the applicability of the TQLs for their particular needs. Please reference the appropriate domain-specific document (such as an RTCA, SAE, or EUROCAE document) to determine if, and to what level, a specific tool needs to be qualified. As an example, Appendix B shows how the DO-278A domain defines the tool qualification criteria and the applicable TQL for ground-based communication, navigation, surveillance, and air traffic management (CNS/ATM) software. However, each domain may use different criteria. For example, they may not use all TQLs; they may have different tool criteria which map to the used TQLs; or they may adapt some of the objectives of this document for their particular applicability.
- b. Throughout this document terms such as “software life cycle”, “software processes”, “software plans”, and “software” are used to refer to the product life cycle, processes, plans, and domain where the tool will be used (that is, a software domain is used instead of a generic domain). For other domains the word “software” may be replaced

by the appropriate domain, such as, “electronic hardware”, “system”, “aeronautical database”, “aviation software”, etc.

- c. The term “certification authority” is used as a generic term to identify the entity in charge of the approval of the tool qualification data. Likewise, the term “certification” is used generically to refer to the approval of the product using the tool by the applicable authorities. The term “applicant” is also used to identify the entity seeking software approval and/or tool qualification.
- d. Possible intentions, interpretations, and adaptations of this document should be included in the appropriate plans in order to obtain approval by the appropriate certification authority.
- e. Coordination between the tool life cycle and the software life cycle should be executed and documented in the following software life cycle data:
 - A plan identifying the tools and their need for qualification. This data is named “Plan for Software Aspects of Certification” (PSAC) throughout this document.
 - A summary and status of all activities performed, including the status of the tools to be qualified. This data is named “Software Accomplishment Summary” (SAS) throughout this document.
 - A configuration index of the software environment, including the identification of the tools and the tool qualification data. This data is named “Software Life Cycle Environment Configuration Index” (SECI) throughout this document.

Regardless of the domain, equivalent data (which may have different titles) are expected to be produced.

- f. In this document, it is assumed that the tool is used to eliminate, reduce, or automate a software life cycle process without the tool’s output being verified, and that these processes are defined in the form of a set of objectives. Therefore, the certification credit of the tool is associated with one or several objectives. It is possible that the guidance for a domain is not defined in the form of “processes” and “objectives”. In this case, the definition of certification credit should be adapted to the domain terminology.
- g. For a domain where no guidance exists to define the tool qualification criteria and applicable TQL, a project in this domain may propose to use this document. The applicable TQL and its rationale should be defined in the project’s PSAC or equivalent document.
- h. Users of this document should reference Annex A in order to determine applicability of objectives for each TQL.

1.4 Document Overview

Figure 1-1 provides an overview of this document. It should be noted that sections 1, 2, and 3, as well as the appendices contain explanatory information to aid the reader in understanding this document’s purpose (these sections are not considered to be guidance).

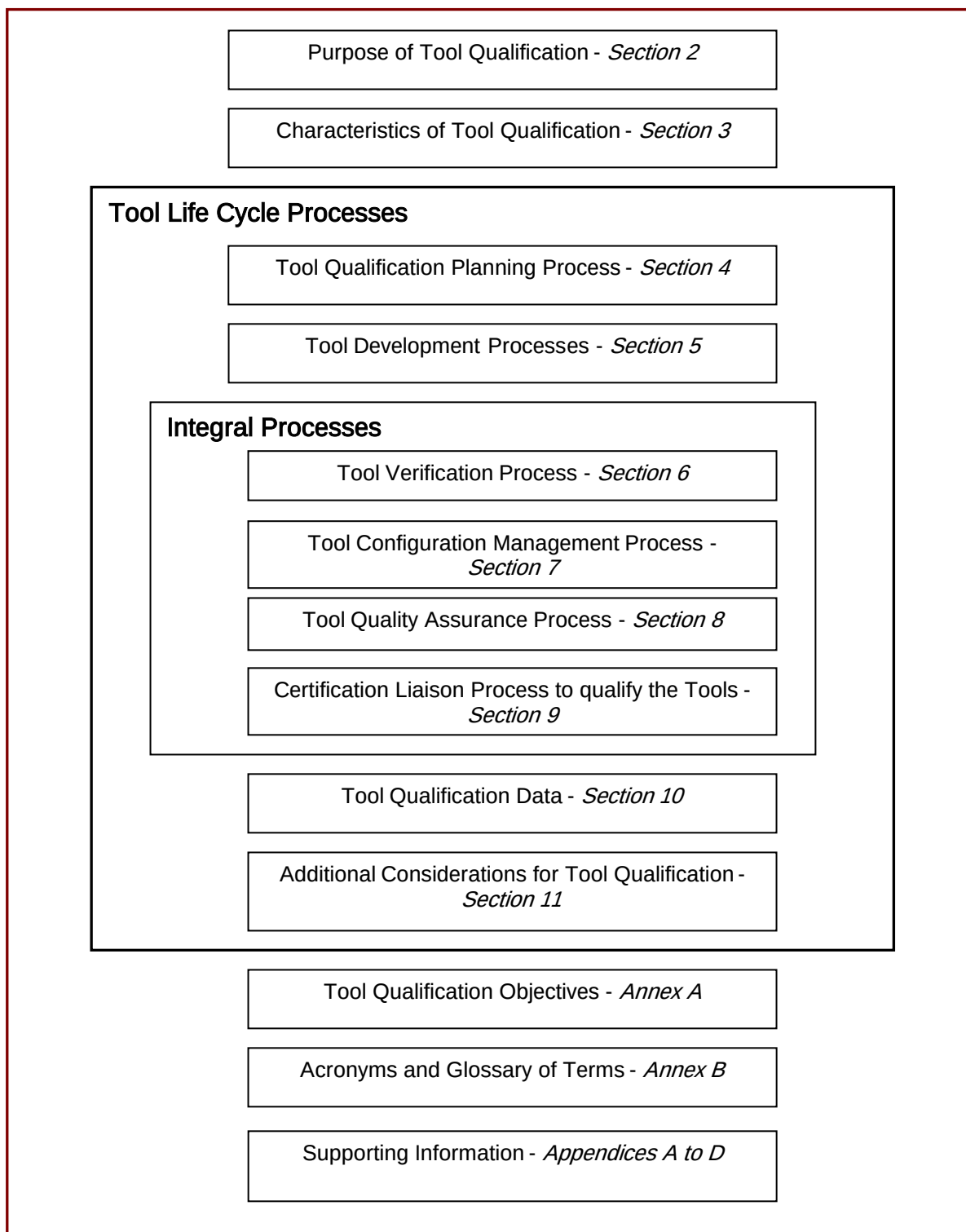


Figure 1-1 Document Overview

This Page Intentionally Left Blank

2.0 PURPOSE OF TOOL QUALIFICATION

Tools can assist in software development to analyze and potentially improve system safety by the automation of the activities performed and by predictably performing functions that may be prone to human error. However, an error in the tool may have a negative impact on software functionality if the tool inadequately performs its intended functions. In order to avoid this risk and to ensure the integrity of the tool functionality, the tool should be developed and verified using adequate processes. Tool qualification serves this purpose.

Tool qualification is the process necessary to obtain certification credit for a tool. This credit may only be granted within the context of a project requiring approval.

The purpose of the tool qualification process is to obtain confidence in the tool functionality. The tool qualification effort varies based upon the potential impact that a tool error could have on the system safety and upon the overall use of the tool in the software life cycle process. The higher the risk of a tool error adversely affecting system safety, the higher the rigor required for tool qualification.

The tool qualification process may be applied to one or more functions in a tool, a single tool, or a collection of tools (see 11.1 for additional information on multi-function tools).

During the qualification effort, the output of all qualified tool functions should be shown to be correct using the objectives of this document. For a tool whose output may vary within expectations, it should be shown that the variation does not adversely affect the intended use of the output and that the correctness of the output can be established. However, for a tool whose output is part of the software and thus could insert an error, it should be demonstrated that qualified tool functions produce the same output for the same input data when operating in the same environment.

This Page Intentionally Left Blank

3.0 CHARACTERISTICS OF TOOL QUALIFICATION

3.1 Qualification Levels

The required qualification level of a tool is based on the tool use and its potential impact in the software life cycle process. This document identifies five Tool Qualification Levels (TQL-1 to TQL-5). As the potential impact of the tool on the software life cycle process increases, the TQL rigor increases due to the potential impact on system safety. The determination of which level is assigned to the tool qualification is determined by an assessment of the tool use in the software life cycle process as defined in the domain related guidance (for example, section 12.2 of DO-178C).

The level of qualification rigor varies according to qualification level. TQL-1 is the most rigorous level and requires well defined and executed tool development, verification, and integral processes with the highest degree of verification independence. The remaining levels require decreasing amounts of rigor. TQL-5 is the least rigorous qualification level. Annex A defines the applicable objectives for each TQL.

3.2 Qualification Stakeholders

Tool qualification typically involves multiple stakeholders. In most projects there will be both a tool user and a tool developer. The tool user typically identifies the tool to be used, assesses its impact on the software processes, addresses the use of the tool in the scope of the software process in which the tool is used, and performs the tool qualification within the context of the software approval. The tool developer typically describes the processes of the tool development, verification, and integral processes, and addresses the development of the tool in compliance with the user needs expressed in the Tool Operational Requirements (TOR).

The typical stakeholders and roles identified above may vary. Likewise, additional stakeholders may be identified (for example, a tool integrator or a tool verifier). The planning documents, PSAC, and/or Tool Qualification Plan (TQP) (as appropriate) should identify the stakeholders and the roles and responsibilities of each stakeholder for the tool qualification effort. The applicant is ultimately responsible for ensuring that all of the tool qualification objectives are satisfied.

This Page Intentionally Left Blank

4.0 TOOL QUALIFICATION PLANNING PROCESS

This section discusses the objectives and activities of the tool qualification planning process. During the tool qualification planning process, the tool is identified; its impact in the software life cycle process is assessed; and the tool development, verification, and integral processes of the complete tool life cycle are described.

The tool qualification planning process includes: (1) the tool qualification assessment as part of the software planning process (see “Planning Process” portion of Annex A [Table T-0](#)), and (2) the tool planning process (see all of Annex A all of [Table T-1](#)). Annex A [Table T-0](#) and [Table T-1](#) summarize the objectives and outputs of the tool qualification planning process for each TQL.

4.1 Determining the Tool Qualification Needs

During the software planning process:

- the tools used in the scope of the software life cycle process are identified;
- each tool’s intended use is described;
- the need for tool qualification is defined;
- the TQLs are determined;
- the tool qualification stakeholders and their assigned roles and responsibilities are identified; and
- the tool operational environment is described.

If there are multiple tool operational environments, then all such environments are described. All these items should be discussed in the additional considerations section of the PSAC. The PSAC explains the use of tools in the software life cycle processes and the need for tool qualification. See section 10.1.1 of this document for the tool aspects that should be addressed in the PSAC.

4.2 Tool Life Cycle Definition

During the tool planning process, a project defines one or more life cycle(s) by choosing the activities for each process, specifying a sequence for the activities, and assigning responsibilities for the activities.

The tool life cycle processes are:

- a. The tool planning process that defines and coordinates the activities of the tool development and integral processes.
- b. The tool development processes that produce the tool.
- c. The integral processes that ensure the correctness, control, and confidence of the tool life cycle processes and their outputs. The integral processes are the tool verification process, the tool configuration management process, the tool quality assurance process, and the qualification liaison process. It is important to understand that the

integral processes are performed concurrently with the tool development processes throughout the tool life cycle.

This document does not prescribe preferred tool life cycles and interactions between them. The separation of the processes is not intended to imply a structure for the organization(s) that performs them. For each tool, the tool life cycle(s) includes these processes.

The tool life cycle definition includes the transition criteria, which are used to determine whether a process may be entered or re-entered. Each process performs activities on inputs to produce outputs. A process may produce feedback to other processes and receive feedback from others. The definition of feedback includes how information is recognized, controlled, and resolved by the receiving process. An example of feedback is problem reporting.

The transition criteria will depend on the planned sequence of tool development processes and integral processes, and may be affected by the TQL. If the transition criteria established for the process are satisfied, every input to a process need not be complete before that process can be initiated.

If a process acts on partial inputs, the inputs to the process should be examined to ensure that they meet the transition criteria. Also, subsequent inputs to the process should be examined to determine that the previous outputs of the tool development and tool verification processes are still valid.

4.3 Tool Planning Process Objectives

The objectives of the tool planning process are:

- a. Define the tool life cycle process(es) in compliance with its intended use as defined in the PSAC, including the items specified in section 4.1 above.
- b. Determine the tool's life cycle, including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria.
- c. Identify the tool development environment, including the methods and tools to be used for the activities of each tool life cycle process.
- d. If necessary, address additional considerations, such as those discussed in section 11 of this document and the need to qualify any tool(s) used in the framework of the tool life cycle processes.
- e. Define tool development standards.
- f. Ensure that tool plans comply with section 10 of this document.
- g. Coordinate the development and revision of the tool plans.

4.4 Tool Planning Process Activities

In order to satisfy the objectives of this document, effective planning is needed. Activities for this process include:

- a. A TQP, Tool Development Plan, Tool Verification Plan, Tool Quality Assurance Plan, and Tool Configuration Management Plan that cover the complete tool life cycle should be developed. These plans may be separate or combined, as appropriate for the specific project needs. As an example, a tool supplier may propose a qualifiable tool and develop plans that cover only the tool development process (see section 11.3 for commercial-off-the-shelf (COTS) tool qualification), but the description of the Tool Operational Requirements definition and tool operational integration processes may be addressed in a separate set of tool plans. Each plan and the supporting standards are briefly described below and detailed in section 10.
 1. The TQP (see 10.1.2) serves as the primary means for communicating the proposed tool life cycle and development methods to the certification authority for agreement. The TQP defines the means of compliance with this document and any other applicable guidance (for example, a domain or project-specific document).
 2. The Tool Development Plan (see 10.1.3) defines the tool life cycle, tool development environment and processes, and the means by which the tool development process objectives will be satisfied.
 3. The Tool Verification Plan (see 10.1.4) defines the means by which the tool verification process objectives will be satisfied.
 4. The Tool Configuration Management Plan (see 10.1.5) defines the means by which the tool configuration management process objectives will be satisfied.
 5. The Tool Quality Assurance Plan (see 10.1.6) defines the means by which the tool quality assurance process objectives will be satisfied.
- b. Tool development standards should be developed. The purpose of these standards is to define the rules and constraints for the tool development processes. The tool development standards include the Tool Requirements Standards (see 10.1.7), the Tool Design Standards (see 10.1.8), and the Tool Code Standards (see 10.1.9). The tool verification process uses these standards as a basis for evaluating the compliance of actual outputs of a process with intended outputs.
- c. The tool verification environment(s) should be defined. The verification environment(s) should be shown to be representative of the tool operational environment. Care should be taken to identify the operating system, additional hardware, additional software components, etc. It should be noted that it may be necessary to have several tool verification environments if there are potentially multiple tool operational environments (for example, a tool may be used on multiple platforms with different versions of operating systems, depending on the specific operational needs).
- d. Reviews of the tool planning process should be conducted to ensure that the plans and standards comply with the guidance of this document and the means to execute them exist.
- e. An assessment on all the tools used in the framework of the tool life cycle processes should be conducted in order to identify the need for qualification of these tools. Qualification of these tools is needed when processes of this document are eliminated, reduced, or automated by the use of a tool without its output verified as

specified in section 6. For a tool that can introduce an error in the outputs of a tool, the applicable TQL is the same as the tool being developed. For a tool that cannot introduce an error in the output of the tool, but may fail to detect an error in the tool life cycle data, the applicable TQL is TQL-5.

5.0

TOOL DEVELOPMENT LIFE CYCLE AND PROCESSES

This section discusses the objectives and activities of the tool development life cycle. The tool development processes are applied as defined by the planning process (see 4) and the Tool Development Plan (see 10.1.3). The tool development life cycle for a specific tool may be tailored based on the TQL. The primary purpose of the tool qualification process is to confirm that a tool meets its requirements and is compliant with the software life cycle process needs. Therefore, the needs from the software life cycle process perspective are defined in the Tool Operational Requirements and the description of all the functionality and characteristics necessary to develop the tool is defined in the Tool Requirements. The overall tool development life cycle includes three major processes, as illustrated in Figure 5-1 and described below. Although these processes are described in the context of a linear waterfall life cycle model, it should be noted that other life cycle models may be used.

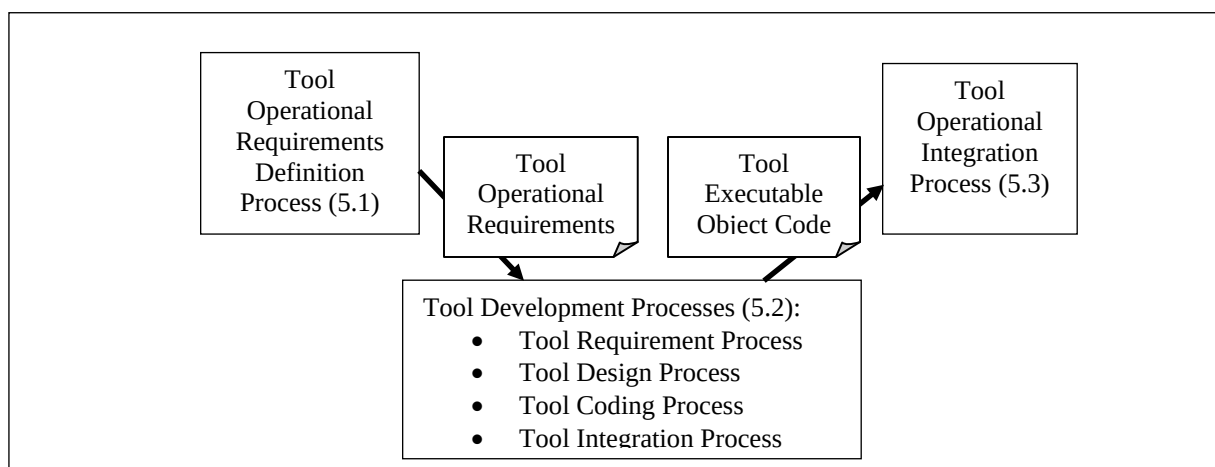


FIGURE 5-1
OVERALL TOOL DEVELOPMENT LIFE CYCLE

Each process is briefly described here and detailed later in sections 5.1 to 5.3.

- a. Tool Operational Requirements Definition Process (see 5.1): The Tool Operational Requirements should identify how the tool is to be used within the software life cycle process. It may not provide all requirements necessary to develop the tool. The amount of detail of the Tool Operational Requirements is dependent on the tool and its intended use. Validation and verification of the Tool Operational Requirements are necessary to confirm that the tool satisfies the needs of the software life cycle process. The Tool Operational Requirements definition process is described in the Tool Development Plan. The output of this process is the Tool Operational Requirements.
- b. Tool Development Processes (see 5.2): The tool development processes are applied as defined by the TQP and the Tool Development Plan. The tool development processes include:
 - Tool requirements process.

- Tool design process.
- Tool coding process.
- Tool integration process.

The tool requirements process produces the Tool Requirements. Tool Requirements are the requirements used to develop the tool. The Tool Requirements should be produced from analysis of the Tool Operational Requirements. However, some tools will be developed for a wide variety of applications and may not be based on any specific Tool Operational Requirements. See section 11.3 for more information about these tools.

The Tool Design Description includes the tool architecture and may generate one or more levels of low-level tool requirements. However, if the Tool Source Code can be generated directly from Tool Requirements then the Tool Requirements are also considered as low-level requirements, and the guidance for low-level requirements also apply to the Tool Requirements. The Source Code is implemented from the tool architecture and the low-level tool requirements.

The Tool Executable Object Code is generated in the tool development environment.

Table T-2 in Annex A is a summary of the objectives and outputs of the tool development processes by TQL.

- c. Tool Operational Integration Process (see 5.3): The tool operational integration process is applied as defined by the tool plans. Inputs to the tool operational integration process are the Tool Executable Object Code, the user instructions, and the installation instructions. They are provided by the tool development process for the tool use and installation into the tool operational environment. The output of the tool operational integration process is the Tool Executable Object Code installed in the tool operational environment.

5.1 Tool Operational Requirements Definition Process

The Tool Operational Requirements identify how the tool is to be used within the software life cycle process. The software plans and the tool plans are the inputs to this process. The Tool Operational Requirements are the primary outputs of this process (see 10.3.1). The “Tool Operational Requirements Process” portion of Table T-0 in Annex A summarizes the objective and output of this process by TQL.

5.1.1 Tool Operational Requirements Objectives

The objective of the tool operational requirements process is:

- a. Tool Operational Requirements are defined.

5.1.2 Tool Operational Requirements Activities

Activities for this process include:

- a. Tool Operational Requirements should be developed in order to include, as a minimum, all items defined in section 10.3.1.

- b. Tool Operational Requirements should be verifiable and consistent. They should include enough detail to demonstrate that the functionality and the outputs of the tool correspond to the identified software life cycle activities to be performed by the tool.
- c. Tool Operational Requirements should provide enough detail to support the verification of the tool's capability to justify taking credit for satisfying the process(es) automated, eliminated, or reduced.

5.2 Tool Development Processes

The purpose of the tool development processes is to define the Tool Requirements and construct the software necessary for their implementation. The typical tool development processes are:

- Tool requirements process.
- Tool design process.
- Tool coding process.
- Tool integration process.

Table T-2 of Annex A summarizes the objectives and outputs of the tool development processes by TQL.

5.2.1 Tool Requirements Process

The Tool Requirements are the requirements used to develop and verify the tool. Tool Operational Requirements, Tool Development Plan, and Tool Requirements Standards are the inputs to this process. The Tool Requirements are the primary outputs of this process (see 10.2.1).

5.2.1.1 Tool Requirements Objectives

The objectives of the tool requirements process are:

- a. Tool Requirements are developed.
- b. Derived tool requirements are defined, if needed.

5.2.1.2 Tool Requirements Activities

Activities for this process include:

- a. The Tool Requirements should be developed and documented in a manner that provides a determination if the tool satisfies the Tool Operational Requirements.
- b. The Tool Requirements should include all functional and interface-related requirements that will be implemented in the tool.
- c. The Tool Requirements should be developed following the processes described in the Tool Development Plan.
- d. The Tool Requirements should be developed using the Tool Requirements Standards.

- e. The Tool Requirements should be verifiable and consistent.
- f. The Tool Requirements should be defined such that abnormal behavior is detected and that invalid output is prevented. For example, it is better for the tool to produce no output than to produce wrong results. Therefore, the Tool Requirements should address failure modes and describe responses to failure conditions.
- g. Each Tool Requirement should trace to one or more Tool Operational Requirements, with the exception of derived tool requirements.
- h. Derived tool requirements are those not traceable to Tool Operational Requirements. The existence of derived tool requirements should be justified, and they should be evaluated to ensure that they do not negatively impact the expected functionality and outputs defined in the Tool Operational Requirements.
- i. The Tool Requirements should provide user instructions, list of error messages, and constraints.
- j. The Tool Requirements should identify requirements related to unused tool functions that do not trace to Tool Operational Requirements. See section 11.1 for guidance on multi-function tools.
- k. The Tool Requirements should be defined to a level of detail appropriate to ensure proper implementation and to assess correctness of the tool (for example, defining underlying models or mathematical theories).

5.2.2 Tool Design Process

Tool Requirements are refined through one or more iterations in the tool design process to develop the tool architecture and low-level tool requirements until it can be used to implement Tool Source Code. The tool design process inputs are the Tool Requirements, the Tool Development Plan, and the Tool Design Standards. When the planned transition criteria have been satisfied, the Tool Requirements are used in the tool design process to develop the tool architecture and low-level tool requirements. The primary output of the tool design process is the Tool Design Description (see 10.2.2).

5.2.2.1 Tool Design Objectives

The Tool Design Description is produced during the tool design process. The objectives are:

- a. The tool architecture is developed.
- b. Low-level tool requirements are developed.
- c. Derived low-level tool requirements are defined, if needed.

5.2.2.2 Tool Design Activities

Activities for this process include:

- a. The tool design process should define the tool architecture.

- b. The tool design should address all tool architecture features, such as protection, if applicable. Protection may be used in order to isolate parts of the tool or the collection of tools in order to apply a different approach or different level of qualification. See section 11.1 for multi-function tools.
- c. The Tool Requirements should be refined into low-level tool requirements that are traceable to Tool Requirements or identified as derived low-level tool requirements.
- d. Derived low-level tool requirements are not traceable to Tool Requirements. The existence of derived requirements should be justified, and they should be evaluated to ensure that they do not impact the expected functionality and outputs defined in the Tool Operational Requirements.
- e. The Tool Design Description, including tool architecture and low-level tool requirements, should conform to the Tool Design Standards.
- f. The tool design should be verifiable and consistent.
- g. Where external components are utilized by the tool, which are not under control of the developer of the tool (such as operating system functions or an external software library), the interfaces to these external components utilized by the tool should be identified in the Tool Design Description. The description of the interface should identify all the external components, such as file management routines, primitives, memory allocation calls, and routines supporting the user interface management (for example, command line or display message).

5.2.3 Tool Coding Process

In the tool coding process, the Tool Source Code is implemented from the Tool Design Description, including tool architecture and low-level tool requirements. The tool coding process inputs are the Tool Design Description, the Tool Development Plan, and the Tool Code Standards. The tool coding process may be entered or re-entered when the planned transition criteria are satisfied. The primary output of this process is the Tool Source Code (see 10.2.3).

5.2.3.1 Tool Coding Objectives

The objective of the tool coding process is:

- a. Tool Source Code is developed from the low-level tool requirements.

5.2.3.2 Tool Coding Activities

Activities for this process include:

- a. The Tool Source Code should implement the low-level tool requirements and conform to the tool architecture.
- b. The Tool Source Code should conform to the Tool Code Standards.
- c. The Tool Source Code should be traceable to the low-level tool requirements.

- d. Inadequate or incorrect inputs detected during the tool coding process should be provided to the process(es) that produced the incorrect input(s) for clarification or correction.

5.2.4 Tool Integration Process

The tool integration process consists of producing the tool in its executable format. This is classically achieved by compiling and linking the Tool Source Code in the tool development environment. If the tool development environment is different than the tool verification environment, then the Tool Executable Object Code should then be installed in the tool verification environment. The tool integration process may be entered or re-entered when the planned transition criteria have been satisfied. The tool integration process input is the Tool Source Code and the Tool Development Plan. The primary outputs of the tool integration process are the Tool Executable Object Code (see 10.2.4) and the compiler and linking data.

5.2.4.1 Tool Integration Objectives

The objectives of the tool integration process are:

- a. Tool Executable Object Code is generated in the tool development environment.
- b. Tool Executable Object Code is installed into the tool verification environment(s).

5.2.4.2 Tool Integration Activities

Activities for this process include:

- a. The Tool Executable Object Code should be generated from the Tool Source Code and the compiler and linking instructions. It should be noted that the Tool Executable Object Code sometimes includes other files, such as configuration files.
- b. Inadequate or incorrect inputs detected during the tool integration process should be provided to the process(es) that produced the inadequate or incorrect input(s).
- c. If the tool verification environment is different than the tool development environment, or if there are multiple tool verification environments, then the Tool Executable Object Code should be installed in the tool verification environment(s).

5.2.5 Tool Development Process Traceability

Tool development process traceability activities include:

- a. Trace Data, showing the bi-directional association between Tool Operational Requirements and Tool Requirements, is developed to:
 - 1. Enable verification of the complete implementation of the Tool Operational Requirements.
 - 2. Give visibility to those derived requirements that are not directly traceable to Tool Operational Requirements.
- b. Trace Data, showing the bi-directional association between the Tool Requirements and low-level tool requirements, is developed to:

1. Enable verification of the complete implementation of the Tool Requirements.
 2. Give visibility to those derived low-level tool requirements that are not directly traceable to Tool Requirements and to the architectural design decisions made during the tool design process.
- c. Trace Data, showing the bi-directional association between low-level tool requirements and Tool Source Code, is developed to:
1. Enable verification that no Tool Source Code implements an undocumented function.
 2. Enable verification of the complete implementation of the low-level tool requirements.

5.3 Tool Operational Integration Process

The purpose of the tool operational integration process is to install the Tool Executable Object Code in the tool operational environment identified in the Tool Operational Requirements and any additional instructions (such as a user's manual), and to gain confidence that the tool is operating as expected by the tool user in the tool operational environment.

The tool operational integration process may be entered or re-entered when the planned transition criteria have been satisfied. The tool operational integration process inputs are the Tool Executable Object Code (including all associated files, if necessary) and the user manual (which may be part of the Tool Requirements).

The primary outputs of the tool operational integration process are the Tool Executable Object Code (see 10.2.4) installed into the tool operational environment and a Tool Installation Report (see 10.3.2). The "Tool Operational Integration Process" portion of Table T-0 in Annex A contains a summary of the objective and output of this process by TQL.

5.3.1 Tool Operational Integration Process Objectives

The objective of the tool operational integration process is:

- a. Tool Executable Object Code is installed into the tool operational environment.

5.3.2 Tool Operational Integration Activities

Activities for this process include:

- a. The Tool Executable Object Code (including all associated files, such as, configuration files) should be installed into the tool operational environment.
- b. Inadequate or incorrect inputs detected during the tool operational integration process should be provided to the process(es) that produced the inadequate or incorrect input(s).
- c. A Tool Installation Report should be generated (see 10.3.2).

This Page Intentionally Left Blank

6.0 TOOL VERIFICATION PROCESS

This section discusses the objectives and describes activities for the overall tool verification process. This process involves two parts:

- a. The tool verification process (see 6.1).
- b. The tool operational verification and validation process (see 6.2).

Note: The verification of the outputs of the planning process is addressed in section 4.4.d.

6.1 Tool Verification Process

Verification is the technical assessment of the outputs of both the tool development processes and the tool verification process. The tool verification process is applied as defined by the tool planning process (section 4) and the Tool Verification Plan (see 10.1.4). Verification includes reviews, analyses, and tests.

Tables T-3 through T-7 of Annex A contain a summary of the objectives and outputs of the tool verification process by TQL.

6.1.1 Purpose of Tool Verification Process

The purpose of the tool verification process is to detect and report errors that may have been introduced during the tool development processes. Removal of the errors is an activity of the tool development processes. The tool verification process verifies that:

- a. The Tool Operational Requirements have been implemented into the Tool Requirements and that the Tool Requirements satisfy and trace to the Tool Operational Requirements.
- b. The Tool Requirements have been implemented into the Tool Design Description (tool architecture and low-level tool requirements) which satisfies the Tool Requirements. Low-level tool requirements should satisfy and trace to the Tool Requirements.
- c. The tool architecture and low-level tool requirements have been developed into the Tool Source Code which satisfies the Tool Design Description. Tool Source Code should satisfy and trace to the tool low-level requirements.
- d. The Tool Executable Object Code satisfies the tool's requirements (that is, intended function) and provides confidence in the absence of unintended functionality.

6.1.2 Overview of Tool Verification Activities

Tool verification objectives are satisfied through a combination of reviews, analyses, the development of test cases and procedures, and the subsequent execution of those test procedures. Reviews and analyses provide an assessment of the accuracy, completeness, and verifiability of the Tool Requirements, Tool Design Description, and Tool Source Code. Reviews also assess compliance to the applicable development standards. The development of test cases and procedures may provide further assessment of the internal consistency and completeness of the Tool Requirements and low-level tool requirements.

The execution of the test procedures provides a demonstration of compliance of the Tool Executable Object Code with the Tool Requirements and the low-level tool requirements.

The inputs to the tool verification process include the Tool Operational Requirements, Tool Requirements, Tool Design Description, Trace Data, Tool Source Code, Tool Executable Object Code, and Tool Verification Plan.

The outputs of the tool verification process are recorded in Tool Verification Cases and Procedures (see 10.2.5), Tool Verification Results (see 10.2.6), and the associated Trace Data (see 10.2.7).

Considerations for this process include:

- a. In order to determine that the higher level of requirements is satisfied by the lower levels of requirements (or source code) and to identify derived requirements, traceability should exist as follows:
 1. Tool Requirements should be traceable to the Tool Operational Requirements.
 2. The low-level tool requirements should be traceable to the Tool Requirements.
 3. The Tool Source Code should be traceable to the low-level tool requirements.
- b. In order to identify the requirements not covered by tests cases, the test cases should be traceable to the requirements (Tool Requirements and low-level tool requirements).
- c. In order to verify all the tests cases are implemented in test procedures, test procedures should be traceable to tests cases.
- d. When it is not possible to verify specific Tool Requirements, other means should be provided and their justification for satisfying the tool verification process objectives defined in the Tool Verification Plan and/or Tool Verification Results.
- e. Deficiencies and errors discovered during the tool verification process should be reported to the appropriate tool development processes for clarification and correction.

6.1.3 Reviews and Analyses

Reviews and analyses should be performed to detect and report errors that may have been introduced during the tool development processes.

6.1.3.1 Reviews and Analyses of the Tool Requirements

The purpose of these reviews and analyses is to detect and report requirements errors that may have been introduced during the tool requirements process. These reviews and analyses activities confirm that the Tool Requirements satisfy these objectives:

- a. Compliance with Tool Operational Requirements: The objective is to ensure that all the Tool Operational Requirements are implemented into the Tool Requirements and that derived tool requirements, and the reason for their existence, are correctly defined.

- b. Accuracy and consistency: The objective is to ensure that each Tool Requirement is accurate, unambiguous, and sufficiently detailed and that the requirements do not conflict with each other. This objective includes verification that the level of detail of the Tool Requirements is appropriate to ensure proper implementation and to assess correctness of the tool (see 5.2.1.2.k).
- c. Compatibility with the tool operational environment: The objective is to ensure that if constraints exist to address the compatibility with the tool operational environment, then compatibility requirements are defined.
- d. Failure modes and errors: The objective is to ensure that the Tool Requirements define the behavior of the tool in response to error conditions and specific requirements addressing failure modes and incorrect inputs are identified.
- e. User information: The objective is to ensure that the Tool Requirements provide user instructions, list of error messages, and constraints.
- f. Verifiability: The objective is to ensure that the Tool Requirements can be verified.
- g. Conformance to standards: The objective is to ensure that the Tool Requirements Standards were followed during the tool requirements process and that deviations from the standards are justified.
- h. Traceability: The objective is to ensure that the Tool Operational Requirements were developed into the Tool Requirements.
- i. Algorithm aspects: The objective is to ensure the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities.

6.1.3.2 **Reviews and Analyses of the Low-Level Tool Requirements**

The purpose of these reviews and analyses is to detect and report requirements errors that may have been introduced during the tool design process. These reviews and analyses activities confirm that the low-level tool requirements satisfy these objectives:

- a. Compliance with the Tool Requirements: The objective is to ensure that the low-level tool requirements satisfy the Tool Requirements and that derived low-level tool requirements and the design basis for their existence are correctly defined.
- b. Accuracy and consistency: The objective is to ensure that each low-level tool requirement is accurate and unambiguous and that the low-level tool requirements do not conflict with each other.
- c. Verifiability: The objective is to ensure that each low-level tool requirement can be verified.
- d. Conformance to standards: The objective is to ensure that the applicable standards (Tool Design Standards and/or Tool Requirements Standards as defined in the TQP and the Tool Development Plan) were followed during the tool design process, and that deviations from the standards are justified.
- e. Traceability: The objective is to ensure that the Tool Requirements and derived tool requirements were developed into the low-level tool requirements.

- f. Algorithm aspects: The objective is to ensure the accuracy and behavior of the proposed algorithms, especially in the area of discontinuities.

6.1.3.3 Reviews and Analyses of the Tool Architecture

The purpose of these reviews and analyses is to detect and report errors that may have been introduced during the development of the tool architecture. These reviews and analyses activities confirm that the tool architecture satisfies these objectives:

- a. Compatibility with the Tool Requirements: The objective is to ensure that the tool architecture does not conflict with the Tool Requirements.
- b. Consistency: The objective is to ensure that a correct relationship exists between the components of the tool architecture. This relationship exists via data flow and control flow.
- c. Conformance to standards: The objective is to ensure that the Tool Design Standards were followed during the tool design process and that deviations from the standards are justified.
- d. Protection: The objective is to ensure that if a protection mechanism is used inside the tool architecture, protection breaches are prevented or isolated. See section 11.1 for multi-function tools.
- e. External components interface: The objective is to ensure that the interface between the tool and the external components is correctly and completely defined (see 5.2.2.2.g).

6.1.3.4 Reviews and Analyses of the Tool Source Code

The purpose of these reviews and analyses is to detect and report errors that may have been introduced during the tool coding process. These reviews and analyses confirm that the outputs of the tool coding process are accurate, complete, and verifiable. Primary concerns include correctness of the code with respect to the low-level tool requirements and the tool architecture, and conformance to the Tool Code Standards. These reviews and analyses are limited to the Tool Source Code; external component verification is addressed in section 6.1.4.3. These reviews and analyses activities confirm that the Tool Source Code satisfies these objectives:

- a. Compliance with the low-level tool requirements: The objective is to ensure that the Tool Source Code is accurate and complete with respect to the low-level tool requirements and that Source Code does not implement any undocumented functions.
- b. Compliance with the tool architecture: The objective is to ensure that the Tool Source Code matches the data defined in the tool architecture.
- c. Verifiability: The objective is to ensure the Tool Source Code does not contain statements and structures that cannot be verified.
- d. Conformance to standards: The objective is to ensure that the Tool Code Standards were followed during the development of the code and that deviations from the standards are justified.

- e. Traceability: The objective is to ensure that the low-level tool requirements were developed into the Tool Source Code.
- f. Accuracy and Consistency: The objective is to determine the correctness and consistency of the Tool Source Code, including fixed point arithmetic overflow and resolution, floating-point arithmetic, resource contention and limitations, exception handling, use of uninitialized variables, unused variables or constants, and data corruption due to task conflicts.

6.1.3.5 Reviews and Analyses of the Tool Integration Process Output

The purpose of these reviews and analyses is to detect and report errors that may have been introduced during the tool integration process. The objective is to:

- a. Ensure that the outputs of the tool integration process are complete and correct.

Activities include conducting a detailed examination of the compiling and linking data. Typical examples of potential errors include:

- Compiler warnings.
- Missing components.
- Incorrect external components interface.

6.1.4 Tool Testing

The tool testing process is used to demonstrate with a high degree of confidence that the tool satisfies its requirements and that errors that could have an impact on the functionality of the tool have been removed.

6.1.4.1 Tool Testing Objectives

The objectives of tool testing are to execute the Tool Executable Object Code in order to confirm that:

- a. The Tool Executable Object Code complies with the Tool Requirements.
- b. The Tool Executable Object Code is robust with the Tool Requirements.
- c. The Tool Executable Object Code complies with the low-level tool requirements.
- d. The Tool Executable Object Code is robust with the low-level tool requirements.

6.1.4.2 Tool Testing Activities

Requirements-based testing is emphasized because this strategy has been found to be the most effective at revealing errors. Activities for requirements-based test case selection include:

- a. Test cases and procedures should be developed to demonstrate that the tool satisfies its requirements.

- Each test case is developed from the requirements (either Tool Requirements or low-level tool requirements) and identifies the set of inputs, the conditions, the expected results, and the pass/fail criteria.
 - Test procedures are generated from the test cases.
 - Trace Data is generated between the test cases and the test procedures.
- b. Normal range tests should be performed, including the following:
- Real and integer input data should be exercised using valid equivalence classes and boundaries values.
 - For state transitions, tests cases should be developed to exercise the transitions possible during normal operation.
 - For requirements expressed by logic equations or for requirements that include input logic combination, the normal test cases should verify the variable usage and the Boolean operators. In particular, for TQL-1, the tests cases should show that each requirement condition independently affects the required outcome by varying just that requirement condition while holding fixed all other possible requirement conditions.
- c. Robustness tests should be performed to address all failure modes (for example, abnormal activation modes, inconsistency inputs, etc.) identified in Tool Requirements.
- d. If necessary, additional robustness tests should also be developed to complete the demonstration of the following:
- The ability of the tool to respond to abnormal inputs or conditions.
 - The detection of abnormal behavior.
 - The prevention of invalid output.

Note: Robustness test cases are requirements-based. The robustness testing criteria cannot be fully satisfied if the tool requirements do not specify the correct response to abnormal conditions and inputs. The test cases may reveal inadequacies in the tool requirements, in which case the tool requirements should be modified. Conversely, if a complete set of requirements exists that covers all abnormal conditions and inputs, the robustness test cases will follow from those tool requirements.

- e. Requirements-based test coverage analysis should be performed to demonstrate that all requirements (Tool Requirements and low-level tool requirements) have been tested. This analysis ensures that each requirement has at least one test case and that appropriate normal and robustness testing has been performed.

6.1.4.3 Analysis of Requirements-Based Testing

6.1.4.3.1 Objectives

The objective of this analysis with the associated resolution is to provide evidence that the tool's code structure was verified to the degree required for the applicable TQL, and to provide confidence, commensurate with the TQL, in the absence of unintended functionality.

The typical means of accomplishing this is using structural coverage analysis. This analysis may reveal code structure that was not exercised during testing.

Some components of a tool may not be under direct control of the tool developer, such as operating system functions or associated middleware (interface components between the operating system and the application). Typically, this type of component supports the operability of the tool within its operational environment. For functions of external components utilized by the tool, tests should be included that address the interface and functionality of each of those functions, as defined in the Tool Design Description (see 5.2.2.2.g).

However, the following components are typically under control of the tool developer and require structural coverage analysis:

- Components developed for limited use or specifically to support the tool.
- Libraries of functions that are used in tool operation (for example, code that implements basic symbols of a model).

Libraries used or incorporated in the tool's output that become part of the resultant software should be verified as a separate activity from the tool qualification (for example, as part of the software life cycle process).

6.1.4.3.2 Analysis

Activities for analysis of requirements-based testing include:

- a. For components under control of the developer of the tool, the analysis should determine which code structure, including interfaces between components, was not exercised by the requirements-based test procedures. For TQL-3, the required level is statement coverage, for TQL-2, the required level is decision coverage, and for TQL-1, the required level is Modified Condition/Decision Coverage (MC/DC). As for any objective, the applicant may propose an alternative means (see 11.5).
- b. For components not directly under control of the tool developer (for example, operating system function or external software library), an additional analysis is needed to ensure completion of requirements-based testing. This analysis should ensure that tests exercise the interface and the functionality of each function of the external components utilized by the tool.
- c. The analysis should confirm that the requirements-based testing has exercised the data coupling and control coupling between the tool code components.

6.1.4.3.3 Resolution

Structural coverage analysis may reveal code structure including interfaces that were not exercised during testing. Resolution would require additional software verification process activity. Possible causes of unexecuted code structure, including interfaces and associated activities to resolve them, include:

- Shortcomings in requirements-based test cases or procedures: The test cases should be supplemented or test procedures changed to provide the missing coverage. The method(s) used to perform the requirements-based coverage analysis may need to be reviewed.
- Inadequacies in requirements: The requirements should be modified, additional test cases developed, and test procedures executed.
- Dead code: The code should either be removed or justified by an impact analysis of the dead code on the tool functionality.
- Deactivated code and unused function: In this case the unexercised Tool Source Code should be either removed or justified. The purpose of this justification is to provide evidence that the unexercised Tool Source Code has no impact on the outputs of the tool in any operational use.

6.1.4.4 Reviews and Analyses of the Test Cases, Procedures, and Results

The purpose of these reviews and analyses is to ensure that the testing of the code was developed and performed accurately and completely. These reviews and analyses activities confirm that the tool testing satisfies these objectives:

- a. Test cases: The objective is to ensure that test coverage of all requirements is achieved (see 6.1.4.2.e for associated activities).
- b. Test procedures: The objective is to verify that the test cases, including expected results, were correctly developed into test procedures.
- c. Test results: The objective is to ensure that the test results are correct and that discrepancies between actual and expected results are explained.

6.1.5 Tool Verification Process Traceability

Tool verification process traceability activities include:

- a. Trace Data is developed to show the bi-directional association between Tool Requirements and their associated test cases, as well as between low-level tool requirements and their associated test cases. The purpose of this Trace Data is to support the requirements-based test coverage analysis.
- b. Trace Data is developed to show the bi-directional association between test cases and test procedures. The purpose of this Trace Data is to enable verification that the complete set of test cases has been developed into test procedures.
- c. Trace Data is developed to show the bi-directional association between test procedures and test results. The purpose of this Trace Data is to enable verification that the complete set of test procedures has been executed.

6.2 Tool Operational Verification and Validation Process

The tool operational verification and validation process is applied as defined by the tool planning process. The “Tool Operational Verification and Validation” portion of Table T-0 in Annex A summarizes the objectives and outputs of the tool operational verification and validation process by TQL.

6.2.1 Tool Operational Verification and Validation Process Objectives

The purpose of the tool operational verification and validation process is to provide confidence that the outputs and the functionality of the tool comply with the software life cycle process needs. This activity includes both verification and validation.

The verification objectives of the tool operational verification and validation process consist of the detection and reporting of errors that may have been introduced during the tool development processes. Removal of the errors is an activity of the tool development processes. Verification objectives are:

- a. Tool Operational Requirements are complete, accurate, verifiable, and consistent.
- b. Functionality and the outputs of the tool installed in the tool operational environment comply with the Tool Operational Requirements.

The validation objectives of the tool operational verification and validation process consist of the analysis of the functionality and the outputs of the tools for correctness and completeness with respect to the software life cycle activities performed. Validation objectives are:

- aa. Ensure that the Tool Operational Requirements are sufficient and correct to eliminate, reduce, or automate the process(es) identified in the PSAC.
- bb. Ensure that the tool meets the needs of the software life cycle process in the tool operational environment.

6.2.2 Tool Operational Verification and Validation Process Activities

The tool operational verification and validation objectives are satisfied through a combination of reviews, analyses, and tests. Reviews, analyses, and development of test cases provide an assessment of the internal consistency and completeness of the Tool Operational Requirements. The execution of the test procedures provides a demonstration of compliance of the Tool Executable Object Code with the Tool Operational Requirements, and its compatibility with the tool operational environment.

The integration of the tool into the tool operational environment should provide demonstration of the correctness and the completeness of the Tool Operational Requirements.

The inputs to the tool operational verification and validation process include the Tool Operational Requirements, the Tool Executable Object Code, and the Tool Verification Plan.

The outputs of the tool operational verification and validation process are Tool Operational Verification Cases and Procedures (see 10.3.3) and the Tool Operational Verification and Validation Results (see 10.3.4).

Activities for the tool operational verification and validation activities include:

- a. Tool Operational Requirements verification: Reviews and analyses should be performed on the Tool Operational Requirements to ensure that:
 1. Each requirement is accurate, unambiguous, complete, and consistent.
 2. Tool operational environment compatibility requirements are identified.
 3. The expected responses of the tool under abnormal operating conditions are defined, such as abnormal activation modes and inconsistent inputs.
- b. Identification and demonstration of relevant process(es): The Tool Operational Requirements should demonstrate the coverage of the process(es) intended to be eliminated, reduced, or automated by the use of the tool.
- c. Verification and validation in tool operational environment: The tool installed in the tool operational environment complies with the need of the software life cycle processes. This activity is both the verification of the compliance of the Tool Executable Object Code to the Tool Operational Requirements and the validation of the Tool Operational Requirements. The activities are:
 1. Test cases and procedures should be developed to ensure that the tool satisfies the Tool Operational Requirements. These test cases and procedures should be requirements-based.
 2. Test procedures should be executed after the installation of the Tool Executable Object Code in the tool operational environment.
 3. Test results should be reviewed to ensure that test results are correct and that discrepancies between actual and expected results are explained.
 4. In order to validate the Tool Operational Requirements the tool should be exercised. The set of inputs selected should represent the actual tool use and its interfaces in the tool operational environment. This validation should confirm that the Tool Operational Requirements are correct and complete.

7.0 TOOL CONFIGURATION MANAGEMENT PROCESS

This section discusses the objectives and activities of the tool configuration management (TCM) process. The TCM process is applied as defined by the tool planning process (see 4) and the Tool Configuration Management Plan (see 10.1.5). Outputs of the TCM process are recorded in Tool Configuration Management Records (see 10.1.13) or in other tool life cycle data.

The TCM process, working in cooperation with the other tool life cycle processes, assists in:

- a. Providing a defined and controlled configuration of the tool throughout the tool life cycle.
- b. Providing the ability to consistently replicate the Tool Executable Object Code or to regenerate it in case of a need for investigation or modification.
- c. Providing control of process inputs and outputs during the tool life cycle that ensures consistency and repeatability of process activities.
- d. Providing a known point for review, assessing status, and change control by control of configuration items and the establishment of baselines.
- e. Providing controls that ensure problems receive attention and changes are recorded, approved, and implemented.
- f. Providing evidence of approval of the tool by control of the outputs of the tool life cycle processes.
- g. Assessing the tool product compliance with requirements.
- h. Ensuring that secure physical archiving, recovery, and control are maintained for the configuration items.

7.1 Tool Configuration Management Process Objectives

The TCM process objectives are:

- a. Each configuration item and its successive versions are labeled unambiguously so that a basis is established for the control and reference of configuration items.
- b. Baselines are defined for further tool life cycle process activity and allow reference to, control of, and traceability between configuration items.
- c. The problem reporting process records process non-compliance with tool plans and standards, records deficiencies of outputs of tool life cycle processes, records anomalous behavior of tool products, and ensures resolution of these problems.
- d. Change control provides for recording, evaluation, resolution, and approval of changes throughout the tool life cycle.
- e. Change review ensures problems and changes are assessed, problems and changes are approved or disapproved, approved changes are implemented, and feedback is provided to affected processes through problem reporting and change control methods defined during the tool planning process.

- f. Status accounting provides data for the configuration management of tool life cycle processes with respect to configuration identification, baselines, problem reports, and change control.
- g. Archival and retrieval ensures that the tool life cycle data associated with the tool product can be retrieved in case of a need to duplicate, regenerate, retest, or modify the tool product. The objective of the release activity is to ensure that only the authorized tool is used in the software life cycle processes, in addition to being archived and retrievable.
- h. Tool life cycle environment control ensures that other tools used to produce the tool itself are identified, controlled, and retrievable.

The objectives for TCM are independent of TQL. However, two categories of tool life cycle data may exist based on the TCM controls applied to the data (see 7.3).

Table T-8 of Annex A is a summary of the objectives and outputs of the TCM process by TQL.

7.2 Tool Configuration Management Process Activities

The TCM process includes the activities of configuration identification, change control, baseline establishment, and archiving of the tool product, including the related tool life cycle data. The TCM process does not stop when the tool product is accepted by the certification authority, but continues throughout the service life of the system or equipment of the resultant software. If tool life cycle activities will be performed by a supplier, then configuration management activities should be applied to the supplier.

7.2.1 Configuration Identification

Activities include:

- a. Configuration identification should be established for the tool life cycle data.
- b. Configuration identification should be established for each configuration item, for each separately controlled component of a configuration item, and for combinations of configuration items that comprise a tool product.
- c. Configuration items should be configuration identified prior to the implementation of change control and traceability analysis.
- d. A configuration item should be configuration identified before that item is used by other tool life cycle processes or referenced by other life cycle data.

7.2.2 Baselines and Traceability

Activities include:

- a. Baselines should be established for configuration items used for certification credit. (Intermediate baselines may be established to aid in controlling tool life cycle process activities.)
- b. A tool product baseline should be established for the tool product and defined in the Tool Configuration Index (see 10.1.11).

- c. Baselines should be established in controlled software libraries (physical, electronic, or other) to ensure their integrity. Once a baseline is established, it should be protected from change.
- d. Change control activities should be followed to develop a derivative baseline from an established baseline.
- e. A baseline should be traceable to the baseline from which it was derived, if a credit is sought for tool life cycle process activities or data associated with the development of the previous baseline.
- f. A configuration item should be traceable to the configuration item from which it was derived, if a credit is sought for tool life cycle process activities or data associated with the development of the previous configuration item.
- g. A baseline or configuration item should be traceable either to the output it identifies or to the process with which it is associated.

7.2.3 Problem Reporting, Tracking, and Corrective Action

Activities include:

- a. A Tool Problem Report should be prepared that describes the process non-compliance with plans, output deficiency, or tool anomalous behavior, and the corrective action taken, as defined in section 10.1.12.

Note: Separate problem reporting systems may be used. For example, errors detected during the tool operational verification and validation process may be recorded in another problem reporting system than errors detected during the tool verification process.

- b. Problem reporting should provide for configuration identification of affected configuration item(s) or definition of affected process activities, status reporting of Tool Problem Reports, and approval and closure of Tool Problem Reports.
- c. Tool Problem Reports that require corrective action of the tool product or outputs of tool life cycle processes should invoke the change control activity.

7.2.4 Change Control

Activities include:

- a. Change control should preserve the integrity of the configuration items and baselines by providing protection against their change.
- b. Change control should ensure that any change to a configuration item requires a change to its configuration identification.
- c. Changes to baselines and to configuration items under change control to produce derivative baselines should be recorded, approved, and tracked. Problem reporting is related to change control, since resolution of a reported problem may result in changes to configuration items or baselines.

Note: It is generally recognized that early implementation of change control assists the control and management of tool life cycle process activities.



- d. Tool changes should be traced to their origin and the tool life cycle processes repeated from the point at which the change affects their outputs. For example, an error that is shown to result from an incorrect design should result in design correction, code correction, and repetition of the associated integral process activities.
- e. Throughout the change activity, tool life cycle data affected by the change should be updated and records should be maintained for the change control activity.

The change control activity is aided by the change review activity.

7.2.5 Change Review

Activities include:

- a. Confirming that all affected configuration items are identified.
- b. Assessing the impact of the change on Tool Operational Requirements.
- c. Assessing the problem or change, with decisions for action to be taken.
- d. Providing feedback of Tool Problem Report or change impact and decisions to affected processes.
- e. Assessing the software process feedback on tool problems and proposed changes identified by the tool process.

7.2.6 Configuration Status Accounting

Activities include:

- a. Reporting on configuration item identification, baseline identification, Tool Problem Report status, change history, and release status.
- b. Defining the data to be maintained and the means to record and report the data's status.

7.2.7 Archive, Retrieval, and Release

Activities include:

- a. Tool life cycle data associated with the tool product should be retrievable from the approved source (for example, an archive at the developing organization or company).
- b. Procedures should be established to ensure the integrity of the stored data (regardless of medium of storage) by:
 - 1. Ensuring that no unauthorized changes can be made.
 - 2. Selecting storage media that minimize regeneration errors or deterioration.
 - 3. Preventing loss or corruption of data over time. Depending on the storage media used, this may include periodically exercising the media or refreshing the archived data.