INTEGRATION TESTING MATURITY ASSESSMENT FOR SAFETY CRITICAL AVIONICS SOFTWARE

GÜVENLİK KRİTİK AVİYONİK YAZILIMLAR İÇİN TÜMLEŞTİRME TEST OLGUNLUĞUNU DEĞERLENDİRME

GÜLSÜM GÜNGÖR

DOÇ. DR. AYÇA KOLUKISA TARHAN

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for he Award of the Degree of Master of Science

in Computer Engineering

2023

ABSTRACT

INTEGRATION TESTING MATURITY ASSESSMENT FOR SAFETY CRITICAL AVIONICS SOFTWARE

Gülsüm GÜNGÖR

Master's Degree, Department of Computer Engineering

Supervisor: Doç. Dr. Ayça KOLUKISA TARHAN

September 2023, 111 pages

Safety-critical software failures lead to serious results such as loss of live or damage to the environment; therefore, safety-critical software verification requires special attention. Avionics system software is one type of safety-critical software. "DO-178C: Software Considerations in Airborne Systems and Equipment Certification" was released in 2011 by RTCA, Inc., (Radio Technical Commission for Aeronautics) which defines processes for aircraft systems software verification and development. On the other hand, there are well-defined guidelines to improve validation and verification processes of software system development, specifically for software testing. TMMI (Test Maturity Model Integration) was produced by TMMI Foundation as a guide for organizations to improve their test processes and product quality. However, avionics system software has own safety-related software

characteristics, and TMMI does not specifically address software testing practices of these characteristics. To fill this gap, in this thesis study, first, avionics software characteristics as the base for software testing are identified. Then, processes and practices in DO-178C and TMMI (Release 1.3) documents are compared with each other bi-directionally. Finally, based on the avionics software characteristics and the results of the comparison, a guidance document approach for integration testing maturity is developed. Considering the critical role of integration testing in preventing safety-critical software defects, it is thought that this approach will be useful for evaluating the integration testing processes of avionics software. A case study was implemented to understand the effectiveness and applicability of this approach. Two groups of test engineers from same team tried to assess test processes applied. The first group applied TMMI model and the second group applied TMMI with guidance approach to assess their processes. At the end, it was observed that the guidance approach provided more improvement actions for avionics integration test processes by referring to domain specific needs of avionics software testing.

Keywords: Safety-critical, avionics software, integration testing, DO-178C, TMMI, test maturity

ÖZET

GÜVENLİK KRİTİK AVİYONİK YAZILIMLAR İÇİN TÜMLEŞTİRME TEST OLGUNLUĞUNU DEĞERLENDİRME

Gülsüm GÜNGÖR

Yüksek Lisans, Bilgisayar Mühendisliği Bölümü

Tez Danışmanı: Doç. Dr. Ayça KOLUKISA TARHAN

Eylül 2023, 111 sayfa

Güvenlik kritik yazılım hataları, can kaybı ve çevresel zararlar gibi birçok ciddi soruna yol açabilmektedir; bu nedenle, güvenlik kritik yazılımların doğrulanması özel bir çaba gerektirmektedir. Güvenlik kritik yazılımların bir türü de aviyonik sistem yazılımlarıdır. 2011 yılında, RTCA (Radio Technical Commission for Aeronautics) tarafından yayınlanmış olan "DO-178C: Software Considerations in Airborne Systems and Equipment Certification" dokümanı, havacılıkta yazılım geliştirme ve doğrulama faaliyetlerine ait süreçlere değinmektedir. Diğer yandan, yazılım geliştirme ve yazılım test faaliyetlerini iyileştirmek için tanımlanmış ve kabul görmüş kılavuzlar bulunmaktadır. Bunlar biri olan TMMI (Test Olgunluk Model Entegrasyon), TMMI Foundation tarafından geliştirilmiştir ve

kurumlarda test süreçlerinin ve ürün kalitesinin iyileştirmesi için kılavuz niteliğindedir. Ne var ki aviyonik sistem yazılımları güvenlik kritik yazılım karakteristiklerine sahiptir ve TMMI modeli, özel olarak bu karakteristiklere değinmemektedir. Bu boşluğu doldurmak amacıyla, bu tez çalışmasında, ilk olarak test aktivitelerine temel olarak aviyonik yazılım karakteristikleri belirlenmiştir. Ardından, DO-178C ve TMMI (Sürüm 1.3) dokümanlarının süreçleri ve pratikleri, birbiriyle çift-yönlü karşılaştırılmıştır. Son olarak aviyonik yazılım karakteristiklerine ve karşılaştırma sonuçlarına dayanarak entegrasyon test olgunluğu için bir kılavuz doküman hazırlama yaklaşımı geliştirilmiştir. Entegrasyon testlerinin güvenlik kritik yazılım hatalarını önlemedeki kritik rolü gözetildiğinde bu yaklaşımın aviyonik yazılımların entegrasyon test süreçlerini değerlendirmek için fayda sağlayacağı düşünülmektedir. Bu yaklaşımın etkisinin ve uygulanabilirliğinin ölçümü için bir durum çalışması gerçekleştirilmiştir. Aynı ekipte yer alan test mühendisleri uyguladıkları test süreçlerini değerlendirmeye çalıştılar. Değerlendirmede birinci grup TMMI modelini, ikinci grup ise TMMI modeline ek olarak kılavuz doküman yaklaşımını kullandılar. Sonuç olarak, kılavuz yaklaşımı aviyonik entegre test süreçleri için daha fazla sayıda iyileştirme önerisi sundu ve bu öneriler aviyonik yazılımlara ait alana özel ihtiyaçlara değinmektedir.

Anahtar Kelimeler: Güvenlik kritik, aviyonik yazılım, entegre test, DO-178C, TMMI, test olgunluğu

ACKNOWLEDGEMENTS

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Abbreviations

CMMI	Capability Maturity Model Integration
ISTQB	International Software Qualifications Board
PTMM	Personal Test Maturity Matrix
RTCA	Radio Technical Commission for Aeronautics
SG	Specific Goal
SP	Specific Practice
TIM	Test Improvement Model
TMMI	Test Maturity Model Integration
TPI	Test Process Improvement
UTMM	Unit Test Maturity Model

1. INTRODUCTION

Safety-critical system failures lead to serious results such as loss of lives or damage to the environment. The software used in avionics systems is classified as safety-critical software in which emerging errors can cause serious consequences. Verification of avionics software is crucial to prevent these undesired results. The first guide to standardize avionics software development was published in 1981 with the name “DO-178: Software Considerations in Airborne Systems and Equipment Certification” [1]. In 2011, DO-178C [2] was released, which addresses software verification processes with different levels of testing (i.e., requirement-based testing, integration testing, hardware and software integration testing).

DO-178C defines integration testing as it aims to guarantee that software components interact correctly and behave as expected, also software requirements are satisfied by components [2]. Defects that can be detected only at the level of integration testing are critical to avoid serious consequences in avionics software. Since DO-178C document heavily focuses on requirement-based integration testing and there is no test maturity approach, obeying to DO-178C alone is not sufficient to evaluate and improve integration testing processes.

On the purpose of testing process and software quality improvements, various models have been developed. Test maturity models such as Test Improvement Model (TIM) [3], Test Process Improvement Model (TPI) [4], Test Maturity Model Integration (TMMI) [5], Unit Test Maturity Model [6] and Personal Test Maturity Matrix (PTMM) [7] are among these models. The presented maturity models can be classified in several groups according to their characteristics. The first group can be defined as tester (or person) skills centered maturity models such as PTMM [7]. This type of models focuses on tester skills to improve testing maturity. The second group includes maturity level-based models that each level has its own goals to be achieved to reach a defined maturity level [5]. The third group of maturity models are testing level-based models that specifically focus on one testing level (such as unit testing) and offer activities for the concerned level [6]. Another group of test maturity models include continuous models that define key performance areas to determine maturity levels [4]. Also, there are some models applicable on automated testing activities [8]. None of the

maturity models expressed above focuses on integration testing level or avionics software (in safety-critical) testing domain. Similarly, the well-recognized software testing standard ISO/IEC 29119 [9] does not focus on software testing maturity or avionics software testing in particular. In order to fill this gap, in this study, it is aimed to offer a maturity model guidance for avionics software considering domain-based requirements.

The content of this thesis is organized as follows: In the first section, general information is provided about the problem defined in this thesis. The second section defines background information about DO-178C and previously defined maturity models and approaches. The third section describes previous works that are related with this thesis. The fourth section explains the method applied in this study in two steps. The first one is comparison of DO-178C and TMMI model, and the second step defines the guidance approach for avionics software testing maturity. Section 5 explains the implementation of the method in detail following these two steps. Guidance approach is applied in a case study and specifically Section 5.3 refers to the details of the case study. And lastly, Section 6 summarizes the conclusions of this thesis study with related discussions.

2. BACKGROUND

Safety systems are classified as critical systems in the event of serious injury, damage to the environment and undesired results. The software used in these systems is considered as safety-critical. The software used in avionics systems, that is aviation electronics system of air vehicles, spacecraft, missiles, satellites etc., is classified as safety-critical software, as its failure can cause undesired results. Some standards have been developed to define specific constraints and structures on software development processes to prevent errors of these systems. The first guide document DO-178, "Software Considerations in Airborne Systems and Equipment Certification" was published in 1982 [1]. The current version, DO-178C handbook, was released by RTCA, Inc., about a decade ago [2]. Verification and testing activities are defined in this guide for defect prevention, revealing errors and ensuring structural coverage [2].

Testing is one of the important issues of safety-critical systems to avoid serious results. Some of the defects can only be discovered within integration testing phase. The quality of safety-critical systems should be continuously assessed and improved. Defined software development standards are not sufficient to guarantee testing quality nor focus on maturity. The ISO/IEC 29119 "Software and systems engineering - Software testing" is a standard that contains "test definitions", "test processes", "test documentation", "test techniques" and "keyword-driven testing" concepts [9]. However, ISO/IEC 29119 guidance focuses on implementation of testing itself, and not specifically on test process improvement goals.

Various approaches and test maturity models focusing on improvement of test processes have been described. Each maturity approach focuses on different aspects of test process improvement. Test Improvement Model (TIM) was described in 1997 by Ericson, which introduces a test improvement approach by focusing on risk management and cost-effectiveness [3]. Test Process Improvement (TPI) model was described in 2004 by Andersin [4], which contains key areas such as test specification techniques and defines test maturity matrix. Test Maturity Model Integration (TMMI) [5] was described by TMMI Foundation, and its structure is similar to CMMI (Capability Maturity Model Integration) [10]. However, CMMI focuses on development processes of software, while TMMI focuses on testing

processes.

Since ISO/IEC 29119 and TMMI are both guidance documents for testing, they are compared in study [11] and it is remarked that ISO/IEC 29119 is not enough to cover all TMMI levels and practices. TMMI offers five maturity levels (Level 1 to 5 respectively; Initial, Managed, Defined, Measured and finally, Optimization) and each of them has own specific goals [5]. Each specific goal has specific practices that are defined to achieve the defined goal [5]. TMMI model and its practices can be applied at all testing levels [5].

On the other hand, testing level-based maturity models are defined, e.g., Unit Test Maturity Model (UTMM) [6]. UTMM defines maturity levels from Level-0: Ignorance to Level-8: Automated Builds and Tasks, and is only applicable within unit testing level.

3. RELATED WORK

This section shares summaries of the related work within the literature. Papers related to test maturity models for integration testing, and test process improvement approaches for safety-critical software are remarked in this section. Studies for safety-critical software process improvement are also involved in this section.

In the study [12] by Duncan et al., test maturity model matrix is defined and the authors focus on safety-critical software in different domains such as medical devices and military. Proposed model has five maturity levels similar to TMMI [5]. The model does not focus on a specific testing level (such as integration testing level).

In the paper "Test Process Improvement with Documentation Driven Integration Testing" by Häser [13] et al., integration testing challenges are determined. The authors present bottom-up testing approach for improving test process maturity at integration testing level.

In the study called "Testing Practices of Software in Safety Critical Systems: Industrial Survey" by Kassab et al. [14], testing methods and techniques as well as testing metrics and defects management and reporting are determined. Non-safety critical and safety-critical system testing activities are compared.

There are several examples of test maturity model applications and TMMI is the most common one as the underlying model. In a paper by Veenendaal et al. [15], researchers present a report for status about TMMI. Also, they define benefits and motivations of using TMMI, and demonstrate its trending results among industries [15].

In the study by Farid et al. [16], improving test processes by comparing TMMI Level-2 process areas and Scrum practices is aimed. The authors reveal that specific practices of TMMI Level-2 are generally covered, and that the organizations using the TMMI model can improve their test processes with the help of Scrum practices [16].

In the study by Garousi et al. [17], a multi-vocal literature review is conducted. A multi-vocal literature review is a type of a Systematic Literature Review which contains both white

papers or blog posts aside from the scientific studies in formal literature. In their study, the researchers introduce 58 different maturity models with various characteristics such as agile models, automated test process-based models and level-based models; however, there is no maturity model reported for integration testing [17].

In the paper by Jang et al. [18], TMMI model is used for automobile control software testing processes by referring to process areas of TMMI and the evaluation of results.

The study called “Defense Software Test Procedure Improvement Measure Reflecting the TMMI” by Park et al. [19] is written in Korean, and the English version of the study cannot not be reached. The abstract of the study in English mentions about TMMI application on defense software, but the results are not involved in the abstract.

4. METHOD

None of the maturity models introduced above focuses on integration testing level in avionics software domain. Similarly, the well-recognized software testing standard ISO/IEC 29119 [9] does not focus on software testing maturity or avionics software testing in particular. In order to fill this gap, this study aims to develop an approach for improving integration testing level test processes of avionics software considering its domain-based requirements.

The DO-178C handbook defines software development life-cycle processes starting from software planning [2]. In the study [20] entitled "Evaluation of accomplishment of DO-178C objectives by CMMI-DEV 1.3", intersection of CMMI-DEV (Capability Maturity Model Integration for Development) practices and DO-178C activities are defined. Some of the CMMI-DEV practices are matched with the DO-178C activities in this study, however, some of them are irrelevant [20]. It is concluded in the study that CMMI-DEV is not sufficient to cover all the software development activities referred in DO-178C and most of the DO-178C verification activities are out of CMMI-DEV's scope [20]. On the other hand, verification and testing activities are in the scope of TMMI since the terminology used in TMMI refers to ISTQB (International Software Qualifications Board) Standard Glossary of Terms used in Software Testing [5]. However, TMMI is a generic testing maturity model and not specific to avionics domain.

In this study, a guidance document approach is developed for improving avionics software verification process, specifically integration testing process. The DO-178C handbook and the TMMI model are analyzed to understand the necessity for a guidance approach that is specific to safety-critical software integration testing. The TMMI model, which can be used complementary to CMMI [10], is found to be convenient to match its processes and practices with the verification activities defined in the DO-178C handbook. Also, TMMI is one of the level-based models that is applicable for all software testing levels, including integration testing, and it covers both manual regression tests and automated tests [5].

In this context, as the first step, processes in DO-178C are inspected and the activities defined in DO-178C processes are mapped with the TMMI (Release 1.3) practices, in order

to understand the similarity between the two as specific to verification and software testing. After this, DO-178C avionics software characteristics that are not specifically indicated in the TMMI model are identified. Since the TMMI model is defined as applicable for all testing levels, some of the TMMI practices are either organizational level practices or more convenient for higher levels of testing (e.g., acceptance testing) [5]. Based on the findings of the mapping, a guidance approach is developed. In this approach, each TMMI practice is reviewed and references to relevant DO-178C handbook sections are provided for practices to implement them considering domain specific characteristics. A reference from a TMMI practice to DO-178C section is called a link. To apply a TMMI practice, links would be helpful to describe and implement given practices within safety-critical avionics software characteristics.

4.1. Comparison between DO-178C and TMMI

Software development life-cycle processes are covered in subtitles of the DO-178C handbook as listed below:

- Software planning process,
- Software development process,
- Software verification process,
- Software configuration management process,
- Software quality assurance process,
- Certification liaison process. [2]

The DO-178C handbook summarizes these software development life-cycle processes within tables in Annex-A [2]. Each process involves objectives and related activities to reach the defined objectives. That is, the tables in Annex-A map activities and objectives for each process [2].

The TMMI model, on the other hand, offers process areas together with their goals and practices to achieve these goals, for each TMMI level [5]. The TMMI model contains five maturity levels, and each level has its own specific practices [5]. Besides, TMMI defines generic practices that are common for all process areas [5].

In the first step of this study, DO-178C process areas are analyzed and each activity defined in DO-178C sections are compared with TMMI (Release 1.3) practices, in order to understand the relation between DO-178C verification activities and TMMI test process maturity practices. Each DO-178C activity is compared with the TMMI's specific practices at all maturity levels and with the generic practices. The analyzed DO-178C activities are grouped as "Covered", "Partially Covered" and "Not Covered" according to the comparison results with the TMMI practices. DO-178 activity that is common for at least one TMMI practice is classified as "Covered". DO-178C activity having scope that is partially matched with any TMMI practice is classified as "Partially Covered". If there is no relevant TMMI practice for the analyzed DO-178C activity, that activity is classified as "Not Covered".

DO-178C activities are analyzed respectively, starting from the first process defined in DO-178C Section-4: Software Planning Process. Table 4.1 shows a snapshot from the comparison between the activities of software planning process of DO-178C and the TMMI process area practices.

Table 4.1. Example Comparison: TMMI Practices & DO-178C "Software Planning Process"

PROCESS AREA	SPECIFIC GOAL	SPECIFIC PRACTICES	Related DO-178C activity
2.1 Test Policy and Strategy	Establish a Test Policy	Define test goals	N/A
2.1 Test Policy and Strategy	Establish a Test Policy	Define test policy	4.2.b
2.1 Test Policy and Strategy	Establish a Test Policy	Distribute the test policy to stakeholders	N/A
...	...	...	...
2.1 Test Policy and Strategy	Establish a Test Strategy	Define test strategy	4.2.b 4.4.2.c 6.4.3 6.4.4.2.a
...	...	...	...
2.2 Test Planning	Establish a Test Approach	Define the test approach	4.4.2.c
2.2 Test Planning	Establish a Test Approach	Define entry criteria	N/A
...	...	...	...
2.2 Test Planning	Develop a Test Plan	Establish the test plan	4.2.b 4.4.2.c
...	...	...	...
2.3 Test Monitoring and Control	Monitor Product Quality against Plan and Expectations	Conduct product quality milestone reviews	4.2.b
...	...	...	...
2.5 Test Environment	Perform Test Environment Implementation	Perform test environment intake test	4.2.b
...	...	...	...
2.5 Test Environment	Manage and Control Test Environments	Report and manage test environment incidents	N/A

DO-178C Activity	TMMI Practice Coverage
4.2.a	NOT COVERED
4.2.b	COVERED
4.2.c	COVERED
4.2.d	NOT COVERED
...	...
4.2.j	NOT COVERED
4.2.k	NOT COVERED
...	...
4.4.2.a	PARTIALLY COVERED
4.4.2.b	PARTIALLY COVERED
4.4.2.c	COVERED
...	...
4.5.d	PARTIALLY COVERED
...	...
...	...
4.6.d	NOT COVERED

Since TMMI focuses on testing and test planning process areas, they are related only with test planning activities within Software Planning Process of DO-178C. Accordingly, the DO-178C software planning activities are classified as “Covered”, “Partially Covered” and “Not Covered” as shown in Table 4.1. The DO-178C software planning activities, which are in the scope of “Test Planning” process area of TMMI, are classified as “Covered”, and it has been observed that the number of activities in “Covered” and “Partially Covered” groups corresponds to only half of the practices in related process area.

For each software development life-cycle process defined in DO-178C sections, a new table (similar to the one in Table 4.1) is created per section (or subsection) considering the structure of Annex-A. It should be reminded that the tables in Annex-A summarize the activities of the processes covered in DO-178C [2]. The handbook’s sections of processes from Section-4 to Section-9 include “Software Planning Process”, “Software Development Process”, “Software Verification Process”, “Software Configuration Management Process”, “Software Quality Assurance Process” and “Certification Liaison Process”, respectively [2].

4.2. Guidance Approach with TMMI model for Avionics Software Integration Testing

In the previous subsection it is stated that the contents of the TMMI model and the DO-178C handbook are compared to understand the requirements of avionics software testing maturity concept. Avionics software characteristics need more specific testing practices to comprise avionics software item verification activities. In this step, DO-178C activities and sections are provided along with relevant TMMI practices. References are defined from TMMI practices to DO-178C activities and sections. It is intended to apply TMMI by considering relevant (referred) DO-178C sections in order to assess avionics software test processes.

5. METHOD IMPLEMENTATION

In the following subsections, firstly, we explain the implementation details of comparison between the TMMI practices and DO-178C activities in order to reveal common threads of them. Also, safety-critical avionics software characteristics are determined during this implementation. Secondly, we explain the details of guidance approach on integration testing level-based maturity for avionics software, by considering findings of the first step. Accordingly, in subsection 5.1, comparison of TMMI and DO-178C documents are described; and in subsection 5.2, guidance approach is explained. Lastly, subsection 5.3 determines the case study method which was implemented to understand the effectiveness and applicability of guidance approach.

5.1. Comparison between DO-178C and TMMI

Each process in DO-178C software development life-cycle has objectives and activities that are defined to achieve related objectives [2]. Activities in subsections are itemized with letters “a”, “b” etc. in the DO-178C handbook. On the other hand, TMMI model has process areas, goals of these process areas, and practices to achieve defined goals [5]. Similarity of structures of these guidance documents helped to compare and determine the needs for avionics software maturity concept. In study [20], a similar structure-based comparison is applied. When comparing TMMI practices and DO-178C activities, all levels of TMMI practices are inspected per related DO-178C activity, so a DO-178C activity can match up with a practice of any TMMI maturity level. Therefore, there is no comparison constraint for TMMI maturity levels of practices.

Mappings of DO-178C processes are discussed in the following separate subsections, and comparison results are summarized in tables for the DO-178C processes. The full version of mapping table of TMMI practices is shared in Appendix-1.

5.1.1. TMMI versus DO-178C Software Planning Process

Software Planning Process is defined in DO-178C section 4. This section includes subsections starting from 4.1 “Software Planning Process Objectives” to 4.6 “Review of the

Software Planning Process". Activities in subsections are itemized with letters, e.g., "4.2.a". The first DO-178C activity related to software planning objectives is about directive plan development of processes for stakeholders [2]. TMMI practices focus on testing and test planning process areas, therefore, its practices are related only with test planning part of DO-178C software planning objectives and activities.

DO-178C activity 4.2.b describes software development standard usage necessity [2]. TMMI refers to various standards in its process areas. In TMMI Level-2 "Test Policy and Strategy" process area, two of its practices called "Define test policy" and "Define test strategy" refer to ISO/IEC 29119-3 standard [9] which is "Software and systems engineering — Software testing — Part 3: Test documentation" [5]. In this practice, test model, risks of the products, test levels and objectives are defined within ISO/IEC 29119-3 standard [5]. TMMI Level-2 "Establish the test plan" practice also explains test plan inputs and outputs with respect to ISO/IEC 29119-3 standard [5]. "Conduct product quality milestone reviews", "Identify and prioritize test conditions, "Identify and prioritize test cases", "Report test incidents" are other TMMI Level-2 practices referring to ISO/IEC 29119-3 standard [5]. Test Environment activities are considered under ISO/IEC 29119-3 standard [5]. Furthermore, master test plan characteristics are defined with respect to this standard in TMMI [5]. Other TMMI levels also refers to various standards. For instance, ISO/IEC 25010 (Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models) standard is represented in the scope of TMMI Level-3 "Analyze non-functional product risks" activity as risk categories [5,21]. TMMI Level-3 refers to ISO/IEC 20246 (ISO/IEC 20246:2017 Software and systems engineering - Work product reviews) standard in the definition of peer review types [5,22]. In TMMI Level-4, product quality characteristics are defined addressing ISO/IEC 25010 standard [5]. One of the generic practices called "Training People" offers topics for different areas and ISO/IEC 25010 standard is addressed for quality characteristics topic [5,21].

DO-178C activity 4.2.c specifies error prevention method or tool selection necessity [2]. TMMI has a major process area for Defect Prevention in Level-5 [5]. This process area contains specific goals such as "Determine Common Causes of Defects and Prioritize" and "Define Actions to Systematically Eliminate Root Causes of Defects" [5]. Firstly, it offers

some practices for defect classification. Defects are analyzed in detail afterward [5]. Pareto Analysis and Histograms methods are offered for defect type analysis [5].

Furthermore, various root cause analysis methods are referred, such as Fault Tree Analysis, FMEA (Failure Mode Effects Analysis), cause and effect diagrams, Ishikawa fishbone diagrams, use of defect classifications, Hardware Software Interaction Analysis and process analysis, which are important methods for complicated safety-critical software systems [5]. Table 5.1 shows mapping of DO-178C activity 4.2.c on TMMI practice list. The complete mapping between the TMMI process area practices and the DO-178C process activities can be reached from [23].

Table 5.1. Mapping of TMMI Practices with DO-178C 4.2.c activity

PROCESS AREA	SPECIFIC GOAL	SPECIFIC PRACTICES	LEVEL	Related DO-178C activity
4.3 Advanced Reviews	Adjust the Test Approach Based on Review Results Early in the Lifecycle	Revise the test approach as appropriate	LEVEL-4	N/A
5.1 Defect Prevention	Determine Common Causes of Defects	Define defect selection parameters and defect classification scheme	LEVEL-5	4.2.c defect detection
5.1 Defect Prevention	Determine Common Causes of Defects	Select defects for analysis	LEVEL-5	4.2.c Pareto Analysis and Histograms
5.1 Defect Prevention	Determine Common Causes of Defects	Analyze causes of selected defects	LEVEL-5	4.2.c
5.1 Defect Prevention	Prioritize and Define Actions to Systematically	Propose solutions to eliminate common causes	LEVEL-5	4.2.c defect detection
5.1 Defect Prevention	Prioritize and Define Actions to Systematically Eliminate Root Causes of Defects	Define action proposals and submit improvement proposals	LEVEL-5	4.2.c

TMMI Level-2 “Test Monitoring and Control” process area offers monitoring test progress against plan and also it is indicated that, when test progress diverges from plan corrective actions can be implemented [5]. These TMMI goals and practices can be considered within DO-178C activity 4.2.e that refers to relation of plan and progress of project [2], however, the scope of DO-178C activity is not limited to test planning.

DO-178C activity 4.2.l specifies that “If software development activities will be performed by a supplier, planning should address supplier oversight” [2]. A generic practice from TMMI Level-2 offers that in test planning process, relevant stakeholders that can be maintainers, developers, testers, customers, end users, suppliers, producers, service personnel, marketers etc. must be determined as planned [5].

DO-178C section 4.3 defines software plans and TMMI can be considered within DO-178C software verification plan (in section 11.3) [2]. Software plan activities are assumed as software verification plan activities and evaluated. TMMI Level-2 test planning indicates that plans need to be updated regarding to changes.

DO-178C section 4.4 describes software lifecycle environment planning [2]. Activity 4.4.1.c of DO-178C handbook indicates that software verification or development standards can be used to reduce related errors rooted in software development environment [2]. Whereas, TMMI focuses on test environment and product risk categories for test planning [5].

DO-178C sub-section 4.4.2 refers to programming language and compiler of software [2]. Changes in compiler may cause to make previous verification process invalid according to 4.4.2.c activity of DO-178C [2]. Some of the TMMI Level-2 practices offers "regression testing" and "re-testing activities" [5] which may be used for handling proposed problem in DO-178C activity. TMMI Level-3 master test plan also offers re-resting and regression testing approaches [5]. However, DO-178C section 4.4.2 defines language and compiler effects on previous verification processes, TMMI does not refer to it as a main reason for re-testing activities [5].

DO-178C section 4.4.3 specifies test environment topic [2]. Test environment specification is proposed in TMMI Level-2 "Test Environment" process area [5]. It is reviewed to ensure its suitability, correctness, feasibility and precise representation of a real-life operational environment [5]. DO-178C defines test environment that could be an emulator, a simulator or target computer [2]. TMMI refers to a practice called "Analyze the test environment requirements" which aims to determine that test environment sufficiently represents the 'real-life' situation [5]. Also, it takes risks related to the test environment requirements into consideration [5].

In TMMI Level-2, the generic practices "Establish an Organizational Policy" and "Monitor and Control the Process" remark that test environment policy refers to a test environment close to real-life environment [5]. This issue is critical for reliability of safety-critical software tests. Since safety critical software failure can cause serious results, testing

activities of safety-critical systems need more attention compared to non-safety critical software systems.

Table 5.2 summarizes comparison results of DO-178C Section 4 activities versus TMMI practices. First column of table shows DO-178C activity number. The analyzed DO-178C activities are grouped as “Covered”, “Partially Covered” and “Not Covered” according to the comparison results with the TMMI practices in the second column. Annex-A summarizes DO-178C activities in tables, and tables are placed in Appendix-2.

DO-178 activity that is common for at least one TMMI practice is classified as “Covered”. DO-178C activity having scope that is partially matched with any TMMI practice is classified as “Partially Covered”. If there is no relevant TMMI practice for the analyzed DO-178C activity, that activity is classified as “Not Covered”.

Table 5.2. Comparison of DO-178C Section 4 activities vs. TMMI practices

4.2.a	NOT COVERED
4.2.b	COVERED
4.2.c	COVERED
4.2.d	NOT COVERED
4.2.e	PARTIALLY COVERED
4.2.f	PARTIALLY COVERED
4.2.g	PARTIALLY COVERED
4.2.h	NOT COVERED
4.2.i	NOT COVERED
4.2.j	NOT COVERED
4.2.k	NOT COVERED
4.2.l	COVERED
4.3.a	NOT COVERED
4.3.b	NOT COVERED
4.3.c	PARTIALLY COVERED
4.4.1.a	NOT COVERED
4.4.1.b	NOT COVERED
4.4.1.c	NOT COVERED
4.4.1.d	NOT COVERED
4.4.1.e	NOT COVERED
4.4.1.f	NOT COVERED
4.4.2.a	PARTIALLY COVERED

4.4.2.b	PARTIALLY COVERED
4.4.2.c	COVERED
4.4.3.a	PARTIALLY COVERED
4.4.3.b	PARTIALLY COVERED
4.5.a	PARTIALLY COVERED
4.5.b	PARTIALLY COVERED
4.5.c	PARTIALLY COVERED
4.5.d	PARTIALLY COVERED
4.6.a	NOT COVERED
4.6.b	NOT COVERED
4.6.c	NOT COVERED
4.6.d	NOT COVERED

5.1.2. TMMI versus DO-178C Software Development Process

Software Development Process is defined in DO-178C section 5. This section includes subsections starting from 5.1 “Software Requirements Process” to 5.5 “Software Development Process Traceability”. Activities in subsections are itemized with letters, e.g., “5.4.1.a”.

DO-178C activity 5.1.2.a defines requirements analysis necessity to avoid ambiguities, inconsistencies and undefined conditions [2]. In the scope of TMMI Level-3 “Peer Reviews” process area, it is noticed that it refers to the practices for performing peer reviews on work products, e.g., reviews implemented by testers [5]. Also, “Establish Peer Review Approach” goal refers to “Identify work products to be reviewed” practice that includes determining work product and peer review type by taking product risks into consideration [5].

Also, TMMI “Peer Review” process area involves “Perform Peer Reviews” specific goal [5]. As the practice of this goal, test basis documents are reviewed by testers for testability, e.g., whether test design techniques which was chosen is applicable. One of the generic practices that relates with this process area proposes peer review policy that refers to peer review attributes in organization [5]. It includes work product to be reviewed, reviewer training issue and role of testers.

DO-178C activity 5.1.2.b defines feedback reporting of software requirements inputs for clarification or correction [2]. In TMMI "Performing Peer Review" process area, two practices propose that peer review results are logged and defects found are reported [5].

Table 5.3 Mapping of TMMI practices with DO-178C activity 5.1.2.a and 5.1.2.b

PROCESS AREA	SPECIFIC GOAL	SPECIFIC PRACTICES	LEVEL	Related DO-178C activity
3.4 Non-functional Testing	Perform Non-functional Test Execution	Write test log	LEVEL-3	N/A
3.5 Peer Reviews	Establish a Peer Review Approach	Identify work products to be reviewed	LEVEL-3	4.2.b [ISO 20246] 5.1.2.a
3.5 Peer Reviews	Establish a Peer Review Approach	Define peer review criteria	LEVEL-3	5.1.2.a
3.5 Peer Reviews	Perform Peer Reviews	Conduct peer reviews	LEVEL-3	5.1.2.b
3.5 Peer Reviews	Perform Peer Reviews	Testers review test basis documents	LEVEL-3	5.1.2.a 5.1.2.b
3.5 Peer Reviews	Perform Peer Reviews	Analyze peer review data	LEVEL-3	6.4.4.1.c
4.1 Test Measurement	Align Test Measurement and Analysis Activities	Establish test measurement objectives	LEVEL-4	N/A

DO-178C section 5.2.2 describes "Software Design Process" activities including low-level requirements and high-level requirements [2]. However, TMMI does not offer specific practices for high-level and low-level requirements. Therefore, section 5.2.2 does not match TMMI practices directly.

TMMI does not specify either user-modifiable or deactivated code that are software characteristics for airborne systems, therefore activities in DO-178C section 5.2.3 and 5.2.4 are unrelated with TMMI approach [2].

Low-level requirements, high-level requirements, their relations and verification processes indicated in DO-178C [2] could be considered in separate activities in test maturity models considering testing levels. Integration testing is the type of requirement-based testing and to propose a test maturity guidance for integration testing level, high-level requirements and their verification should be considered.

DO-178C section 5.3.2 "Software Coding Process Activities" [2] offers source code implementation objectives that is out of TMMI scope, therefore, these DO-178C activities do not match TMMI practices. DO-178C section 5.4.2 "Integration Process Activities" [2] is not relevant to TMMI practices. DO-178C section 5.5 refers to traceability of requirements [2]. Traceability between test conditions and requirements is issue of TMMI [5], however,

traceability of different levels of requirements are not in the scope of TMMI. TMMI also proposes requirements / product risks traceability matrix in level-3. Table 5.4 shows the comparison results of the mapping process for DO-178C section 5 activities.

Table 5.4. Comparison of DO-178C Section 5 activities vs. TMMI practices

5.1.2.a	PARTIALLY COVERED
5.1.2.b	PARTIALLY COVERED
5.1.2.c	NOT COVERED
5.1.2.d	NOT COVERED
5.1.2.e	NOT COVERED
5.1.2.f	NOT COVERED
5.1.2.g	NOT COVERED
5.1.2.h	NOT COVERED
5.1.2.i	NOT COVERED
5.1.2.j	NOT COVERED
5.2.2.a	NOT COVERED
5.2.2.b	NOT COVERED
5.2.2.c	NOT COVERED
5.2.2.d	NOT COVERED
5.2.2.e	NOT COVERED
5.2.2.f	NOT COVERED
5.2.2.g	NOT COVERED
5.2.3	NOT COVERED
5.2.4	NOT COVERED
5.3.2	NOT COVERED
5.4	NOT COVERED
5.5	NOT COVERED

5.1.3. TMMI versus DO-178C Software Verification Process

“Software Verification Process” is defined in DO-178C section 6 [2]. This section includes subsections starting from 6.1 “Purpose of Software” to 6.6 “Verification of Parameter Data Items” [2]. The scope of this section is very large and DO-178C handbook considers sub-sections of section 6 within five tables in Annex-A [2]. Also, some of the objectives are

itemized with letters similar to the activities and placed in the tables of Annex-A [2]. Therefore, in the subsections below, objectives are also discussed in addition to activities considering the tables of Annex-A [2].

5.1.3.1. TMMI versus DO-178C Section 6.3

DO-178C handbook expresses "Review and analyses of high-level requirements" objectives in section 6.3.1 [2], and these objectives are partially covered by TMMI practices. The objectives related with software verification activities are inspected. Both TMMI Level-3 process area "Non-Functional Testing" and Level-4 process area "Product Quality Evaluation" [5] define compatibility characteristics from ISO/IEC 25010 standard that is one of the objectives of DO-178C [2]. Accuracy as a product quality attribute is referred in TMMI [5] while in DO-178C handbook, accuracy and consistency of high-level requirements are addressed [2]. Nevertheless, "Peer Review" [5] practices from TMMI Level-3 are matched with high-level requirement analysis of DO-178C [2]. Traceability is another objective of this section between system and high-level requirements [2] while traceability between requirements and test conditions is issue of TMMI [5]. DO-178C handbook refers to verifiability in section 6.3.1 [2], which is in the scope of TMMI Level-3 "Peer Review" process area as testability [5]. Table 5.5 shows mapped TMMI practices to DO-178C section 6.3.

Table 5.5. Mapping of TMMI Practices with DO-178C Section 6.3

PROCESS AREA	SPECIFIC GOAL	SPECIFIC PRACTICES	LEVEL	Related DO-178C activity
4.2 Product Quality Evaluation	Establish Measurable and Prioritized Project Goals for Product Quality	Identify product quality needs	LEVEL-4	N/A
4.2 Product Quality Evaluation	Establish Measurable and Prioritized Project Goals for Product Quality	Define the project's quantitative product quality goals	LEVEL-4	4.2.b [ISO/IEC 25010] 6.3.1- 6.3.2
4.2 Product Quality Evaluation	Establish Measurable and Prioritized Project Goals for Product Quality	Define the approach for measuring progress toward the project's product quality goals	LEVEL-4	N/A

DO-178C section 6.3.2 defines similar objectives with section 6.3.1, except it refers to low-level requirements which are not considered separately in TMMI model [2].

DO-178C section 6.3.3 and 6.3.4 define objectives for software architecture analysis; review and source code analysis and review objectives, respectively [2]. Scope of these objectives are large and not covered by TMMI practices.

5.1.3.2. TMMI versus DO-178C Section 6.4

DO-178C section 6.4 refers to "Software Testing" [2]. Subsection 6.4.1 refers to "Test Environment" and in activity 6.4.1.a it is expressed that there are some types of errors which can only be detected in the tests of integrated environments, and also DO-178C defines multiple test environment necessity [2]. In TMMI Level-2, the practice called "Define Test Strategy" from "Test Policy and Strategy" process are refers to test environment issue firstly referencing "ISO 29119-3" [5]. However, test environment characteristics are defined in TMMI Level-2 "Test Environment" process area [5]. Test environment and its similarity with target (or real-life environment) are considered in the scope of both DO-178C handbook and TMMI [2] [5].

Emulators or simulators used in verification activities are also expressed in DO-178C [2]. Simulators are placed in one of the test environments needs under specific practice called "Elicit test environment needs" that belongs to "Develop Test Environment Requirements" goal of TMMI [5]. This TMMI practice definition is more detailed than test environment activity defined in DO-178C handbook section 6.4.1 [2]. Test environment requirement documentation and analysis are practices of test environment process area belonged to TMMI [5]. TMMI also refers to "Manage and Control Test Environments" goal that offers various practices and one of them is systems management practice which performs on the test environment that aims efficiently and effectively improving the test execution process [5]. Also, TMMI Level-2 "Report test incidents" practice indicates that incident reports should include test environment information of executed test case [5]. TMMI model's generic practices also take test environment into consideration suggesting that the test environment should be as close as possible to real life [5].

TMMI Level-3 generic goal called "Institutionalize a Defined Process" defines a generic practice that uses number of defects that were not revealed in testing phase because of not sufficient test environment and occurred in production as a measure for improvement of test

processes [5]. Since DO-178C classifies some errors that are only detected during tests executed in integrated target environment [2], the referred TMMI practice becomes crucial and supports safety critical software testing.

DO-178C section 6.4.2.1 defines “Normal Range Test Cases” issue as the subsection of “Requirement-Based Test Selection” [2]. TMMI Level-2 “Test Design and Execution” process area refers to a specific goal called “Perform Test Analysis and Design Using Test Design Techniques” [5]. This goal’s first practice is “Identify and prioritize test conditions” [5]. This practice involves sub-practice for selecting the most appropriate test design techniques among the common ones [5]. “Equivalence Partitioning” and “Boundary Value Analysis” are specified in DO-178C section 6.4.2.1.a as an activity [2] and they are in the scope of this TMMI practice.

State transition testing is another test technique also proposed under “Identify and prioritize test conditions” practice of TMMI [5], which is another activity described in 6.4.2.1.c [2]. White box test techniques, also represented in the previous TMMI practice called “Identify and prioritize test conditions”, can be used to verify Boolean operators and variable usage.

DO-178C section 6.4.3 defines requirement-based testing method with different types of testing such as;

- Requirements-Based Low-Level Testing
- Requirements-Based Hardware/Software Integration Testing,
- Requirements-Based Software Integration Testing [2].

TMMI addresses all test levels, acceptance tests, integration tests and low-level tests [5]. Different test level activities are used to detect different error types and in DO-178C standard they are addressed individually [2].

TMMI Level-2 “Test Policy and Strategy” process area refers to “Define test strategy” practice to identify test levels and the objectives, main tasks, responsibilities and entry or exit criteria are determined for each level [5].

Activities in DO-178C section 6.4.4.1 are related with test coverage analysis and they match with TMMI practices in different maturity levels [2]. Activities expressed in 6.4.4.1.b are not covered by TMMI practices because “Robustness test” defined in DO-178C are not considered in TMMI document [2], [5].

TMMI Level-2 process area “Test Design and Execution” offers a practice for traceability between requirements and test conditions [5]. Also, traceability is described in TMMI Level-3 addressing ISO 29119-3 standard as test design specification [5]. Furthermore, both functional and non-functional requirement traceability are defined in different TMMI levels [5]. Therefore, DO-178C [2] activities related with requirement traceability 6.4.4.1.a and 6.4.4.1.d are covered by TMMI practices.

In TMMI Level-3 “Peer Review” process area, peer review data analysis is defined that involves defect resolution impact analysis [5] and likewise, DO-178C activity 6.4.4.1.c offers defect analysis [2].

Also, “Test Strategy Definition” practice of TMMI Level-2 indicates system requirement coverage, code coverage and user requirement coverage actions for different software test levels addressing ISO 29119-3 standard [5]. Traceability matrix for requirement coverage is represented in the same TMMI level [5]. Coverage analysis tools are defined as generic practices in TMMI Level-2 [5]. TMMI Level-4 “Test Measurement” process area describes structural coverage as a measurement object [5].

In DO-178C section 6.4.4.2 activity “a”, structural coverage analysis for software levels is considered [2]. In TMMI Level-2, coverage levels are defined for different test levels from unit test to acceptance test according to “ISO 29119-3” [5]. Also, coverage analysis is used to determine test exit criteria in the TMMI Level-2 practices [5]. Another TMMI Level-2 sub-practice offers that test coverage as test process exit criteria should be monitored against test plan [5]. “Test Design and Execution” process area offers another generic practice for coverage analysis tools for test processes. TMMI Level-4 “Specify test measures” practice specifies test measures as peer review coverage, structural coverage and requirements

coverage [5]. Table 5.6 shows an example mapping of section 6.4.2 and 6.4.4 activities of DO-178C.

As DO-178C considers requirement levels, requirement coverage and analysis are much more detailed than TMMI practices, and DO-178C section 6.4.4.2 activities could not be considered as fully covered by TMMI practices [2].

Table 5.6. Mapping of TMMI Practices with DO-178C Section 6.4

PROCESS AREA	SPECIFIC GOAL	SPECIFIC PRACTICES	LEVEL	Related DO-178C activity
2.3 Test Monitoring and Control	Monitor Product Quality against Plan and Expectations	Conduct product quality milestone reviews	LEVEL-2	4.2.b (ISO29119-3]
2.3 Test Monitoring and Control	Manage Corrective Actions to Closure	Analyze issues	LEVEL-2	
2.3 Test Monitoring and Control	Manage Corrective Actions to Closure	Take corrective action	LEVEL-2	4.2.e
2.3 Test Monitoring and Control	Manage Corrective Actions to Closure	Manage corrective action	LEVEL-2	4.2.e
2.4 Test Design and Execution	Perform Test Analysis and Design using Test Design Techniques	Identify and prioritize test conditions	LEVEL-2	4.2.b (ISO29119-3] 6.4.2.1.a 6.4.2.1.c 6.4.2.1.d 6.4.4.1.a 6.4.4.1.d 7.2.4.d/e
2.4 Test Design and Execution	Perform Test Analysis and Design using Test Design Techniques	Identify and prioritize test cases	LEVEL-2	4.2.b (ISO29119-3] 6.4.4.1.a 6.4.4.1.d 6.5
2.4 Test Design and Execution	Perform Test Analysis and Design using Test Design Techniques	Identify necessary specific test data	LEVEL-2	N/A
2.4 Test Design and Execution	Perform Test Analysis and Design using Test Design Techniques	Maintain horizontal traceability with requirements	LEVEL-2	6.4.4.1.a 6.4.4.1.d 6.5
2.4 Test Design and Execution	Perform Test Implementation	Develop and prioritize test procedures	LEVEL-2	4.2.b (ISO29119-3] 6.5
2.4 Test Design and Execution	Perform Test Implementation	Create specific test data	LEVEL-2	N/A

5.1.3.3. TMMI versus DO-178C Section 6.5

Relation of TMMI practices and DO-178C Section 6.4.4.1 activities is proposed previously. Both functional and non-functional requirement traceability are expressed in different TMMI levels as horizontal traceability [5]. Horizontal traceability is defined as traceability of requirements between layers of test documentation and it is bi-directional [24]. DO-178C introduces three bi-directional traceability activities in section 6.5 [2].

Other traceability types, except requirement traceability, introduced in DO-178C section 6.5 are traceability between test cases and test procedures and traceability between test results and procedures [2]. TMMI Level-2 and Level-3 practices refer to traceability of test cases [5]. Also, specific practice called "Develop and prioritize test procedures" defines traceability between procedures and test cases in TMMI Level-2 [5].

5.1.3.4. TMMI versus DO-178C Section 6.6

DO-178C section 6.6 defines Parameter Data Item, which is a feature to enable changing behavior of software without modifying its code, is a domain specific characteristic of airborne software. Since TMMI is not a domain specific test maturity model, it does not offer any verification or test maturity practice for parameter data items. Table 5.7 shows the comparison between DO-178C activities in section 6 and TMMI practices.

Table 5.7. Comparison of DO-178C Section 6 activities vs. TMMI practices

DO-178C Activity	TMMI Practice Coverage	DO-178C Activity	TMMI Practice Coverage	DO-178C Activity	TMMI Practice Coverage
6.3.1	PARTIALLY COVERED	6.3.2	PARTIALLY COVERED	6.4.1.a	PARTIALLY COVERED
				6.4.2.1(4)	COVERED
				6.4.2.2(7)	NOT COVERED
				6.4.3(3)	PARTIALLY COVERED
				6.4.4.1.a	COVERED
				6.4.4.1.b	NOT COVERED
				6.4.4.1.c	COVERED
				6.4.4.1.d	COVERED
				6.4.4.2(4)	PARTIALLY COVERED
				6.5(2)	COVERED
				6.5(1)	PARTIALLY COVERED
				6.6	NOT COVERED

5.1.4. TMMI versus DO-178C Software Configuration Management Process

TMMI considers "Configuration Management" in its general practices referring to CMMI configuration management processes [5]. According to TMMI Level-2, configuration management must be a part of determining test strategy [5]. Also, TMMI Level-2 process areas define configuration management objects as test estimation data, test plan, test strategy, product risk assessments, test policy, reports, logs, test case specification etc. [5].

DO-178C section activity 7.2.1.a offers establishing configuration identification [2] and TMMI general practices are related with this activity. Change control is a sub-practice of TMMI "Configuration Management" practice [5], which is also defined as an activity in DO-

178 as 7.2.1.c [2]. However, all of the activities in DO-178C subsection 7.2.1 are not fully covered by TMMI practices [2].

Activities in DO-178C subsection 7.2.2 are related with CMMI [10] practices more than TMMI practices, therefore activities of this subsection are not matched with TMMI practices.

Problem reporting activities are defined in DO-178C that the problem can be software anomalies or defects [2], and TMMI considers problem reporting and defect prevention practices in its various levels of process areas [5]. Defects can occur after execution of test cases. TMMI Level-2 offers reporting and analyzing test incidents [5]. Test incident reports contain "description of the incident (environment, actual results, input, anomalies, expected results, observations, attempts to repeat test procedure steps, and testers), time information, status of test incident and risk" [5]. Test logs are created after test incident reporting phase [5]. Test incident management is the next step to resolve incidents properly [5].

TMMI Level-2 practices define sequential practices for incident management [5]. Configuration (or change) control board (CCB) meetings are arranged to decide how to take action to handle incidents [5]. After CCB decision, incident fixing activities are performed in respective teams and confirmation tests are executed to close incident [5]. Finally, incident status is reported to stakeholders and CBB meetings are arranged to analyze status reports [5]. TMMI "test design and execution" practices cover DO-178C section 7.2.3 activities [5]. Along with functional test incidents, non-functional test incident reporting and analyzing is in the scope of TMMI Level-3 practices [5]. In addition to software incidents, test environment incidents are also reported by applying incident classification scheme in the scope of TMMI-Level 2 test environment practices [5]. Defect prevention activities of TMMI Level-5 are not considered in the scope of DO-178C section 7.2.3 [5] [2].

DO-178C section 7.2.4 and 7.2.5 define change control and change review activities in detail [2]. These sections are partially covered by TMMI configuration management or CCB (configuration control board) practices [5]. Activity "d" of DO-178C section 7.2.4 can be considered in the scope of CCB activities [2]. In TMMI Level-2 it is remarked that, whenever

a requirement change occurs, it may affect test conditions; therefore, the test design specifications and test conditions need to be revised [5].

Corrective action management goal is defined in TMMI Level-2 “Test monitoring and control” process area [5]. Also, TMMI Level-2 “Execute test cases” practice offers that test activities must be repeated by confirmation tests after changes [5]. These TMMI actions are associated with DO-178C activities 7.2.4.d and 7.2.4.e [2].

TMMI Level-2 “Test Planning” process area includes product risk assessment practices [5]. Change related risk is one of the risk categories specified in risk category definition practice of TMMI [5]. Also, requirement changes cause to revise the product risks again [5]. Furthermore, it is pointed that, documentation of the product risks needs to be revised when there are requirement changes or additions that can affect product risks [5]. DO-178C activities “7.2.5.a” and “7.2.5.b” propose requirement or software life-cycle data change impact assessment and system safety assessment issues [2], whereas TMMI level-2 discusses change-related risk assessment [5]. Table 5.8 shows some of the mapped activities from DO-178C Section 7.2.3 on to TMMI practices.

Table 5.8. Mapping of TMMI Practices with DO-178C Section 7.2

PROCESS AREA	SPECIFIC GOAL	SPECIFIC PRACTICES	LEVEL	Related DO-178C activity
2.4 Test Design and Execution	Perform Test Execution	Execute test cases	LEVEL-2	7.2.4.d
2.4 Test Design and Execution	Perform Test Execution	Report test incidents	LEVEL-2	4.2.b (ISO29119-3] 6.4.1.a 7.2.3
2.4 Test Design and Execution	Perform Test Execution	Write test log	LEVEL-2	4.2.b (ISO29119-3] 7.2.3
2.4 Test Design and Execution	Manage Test Incidents to Closure	Decide disposition of test incidents in configuration control board	LEVEL-2	7.2.3. 8.3.d
2.4 Test Design and Execution	Manage Test Incidents to Closure	Perform appropriate action to fix the test incident	LEVEL-2	7.2.3. 8.3.d
2.4 Test Design and Execution	Manage Test Incidents to Closure	Track the status of test incidents	LEVEL-2	7.2.3. 8.3.d

DO-178C subsections 7.2.6., 7.2.7, 7.4 and 7.5 are not relevant to TMMI practices, and activities are considered as “Not Covered” in these sections. Table 5.9 shows mapping result of DO-178C Section 7 activities.

Table 5.9. Comparison of DO-178C Section 7 activities vs. TMMI practices

7.2.1.a	COVERED
7.2.1.b	NOT COVERED
7.2.1.c	COVERED
7.2.1.d	NOT COVERED
7.2.1.e	NOT COVERED
7.2.2	NOT COVERED
7.2.3	COVERED
7.2.4.a	NOT COVERED
7.2.4.b	NOT COVERED
7.2.4.c	NOT COVERED
7.2.4.d	COVERED
7.2.4.e	COVERED
7.2.5.a	COVERED
7.2.5.b	COVERED
7.2.5.c	NOT COVERED
7.2.5.d	NOT COVERED
7.2.6	NOT COVERED
7.2.7	NOT COVERED
7.4	NOT COVERED
7.5	NOT COVERED

5.1.5. TMMI versus DO-178C Software Quality Assurance Process

DO-178C section 8 defines “Software Quality Assurance Process” [2] and TMMI model addresses in its all levels “CMMI Process and Product Quality Assurance” process area practices for its generic practice called “Objectively evaluate adherence” [5]. Also, TMMI Level-3 “Test Organization” process area offers to establish a team of testers that are responsible for determining product quality goals and measuring quality characteristics [5]. This process area offers a practice called “Deploy standard test process and test process assets” and quality assurance is involved in the deployment [5]. TMMI Level-4 “Product Quality Evaluation” process area expresses that quality assurance group must define goals for process and product quality assurance, and evaluate the performance of project or progress in accomplishing these goals [5].

Quality assurance and its relation between testing activities are in the scope of TMMI [5], however, objectives and activities described in DO-178C are not fulfilled by TMMI practices. DO-178C activity 8.2.d.5 that refers to software configuration management plan [2] can be considered as compliant to TMMI configuration management practices [5].

DO-178C section 8.3 “Software conformity review” is almost out of TMMI scope [2]. Nevertheless, DO-178C activity 8.3.d, defined as problem report evaluation and status logging [2], can match up with CCB meeting reporting, test incidents fixing and incident status tracking practices of TMMI Level-2 “Test Design and Execution” process area [5]. Table 5.10 shows comparison result for section 8 activities of DO-178C.

Table 5.10. Comparison of DO-178C Section 8 activities vs. TMMI practices

8.2.	PARTIALLY COVERED
8.3	PARTIALLY COVERED

5.1.6. TMMI versus DO-178C Certification Liaison Process

There is no DO-178C Certification Liaison activity that is related with TMMI practices. Therefore, all of the activities of this section are considered as “Not Covered” by TMMI practices. Table 5.11 shows comparison result for section 9 activities of DO-178C.

Table 5.11. Comparison of DO-178C Section 9 activities vs. TMMI practices

9.2	NOT COVERED
9.3	NOT COVERED

As a result, it has been observed that TMMI practices are not sufficient alone to guide accomplishing DO-178C verification activities. DO-178C has avionics software development characteristics and definitions that are not in the scope of TMMI. DO-178C

defines avionics software specific items and concepts such as verification of parameter data item, user modifiable software, deactivated code, multi-version dissimilar software verification, option selectable software, COTS software and field-loadable software [2] that are not discussed in TMMI. Therefore, some of the process activities in DO-178C (regarding verification and testing of avionics software) do not match with the TMMI practices. For example; low-level requirements, high-level requirements, and their relation to system requirements are defined in detail within DO-178C objectives, but TMMI process area goals do not match this structure which is specific to avionics domain. In addition, some of the change related activities (software change, requirement change, new compiler usage, different loader version, change of development environment or application, etc.) and re-execution needs of tests are defined in DO-178C within safety-critical aspects [2]. Even though TMMI offers change related practices [5], scope of the change should be revised by considering DO-178C safety-critical avionics software development. Moreover, the DO-178C handbook includes a subsection for "Robustness Test Cases" that shows software behavior in abnormal conditions [2]. It is critical to avoid undesired results, but it is not particularly discussed in TMMI practices.

The shortages defined previously should not be considered as weaknesses of the TMMI model since it is a general maturity model that offers many practices to improve testing processes and product quality. Rather, the shortages indicate the need for a testing maturity guidance specific to avionics domain. Finally, on the opposite side of the mapping, some TMMI process areas such as test training programs, incident management and advanced reviews [5] are not discussed in DO-178C processes in detail. Integration testing is one of the critical test levels in the scope of high-level requirements-based testing in DO-178C to avoid undesired results of safety-critical avionics software. It is observed that TMMI process area practices can enrich the activities for integration testing level defined in DO-178C. Therefore, the results of bi-directional comparison between DO-178C activities and TMMI practices have shown that the mutual consideration of these two resources for a maturity guidance approach for integration testing of avionics software is prominent.

5.2. Guidance Document for TMMI Applications on Avionics Software Integration Testing

In the previous section, the contents of the TMMI model and the DO-178C handbook are compared to understand the requirements of avionics software testing maturity concept. Avionics software characteristics need more specific testing practices to comprise avionics software item verification activities. As the next step, a guidance document that employs TMMI practices as complementary to DO-178C activities is developed to effectively improve domain specific software testing processes, more specifically integration testing activities, within avionics software development.

In this guidance approach, domain specific characteristics of safety critical avionics software are defined considering DO-178C handbook sections. Then, each TMMI practice is reviewed to implement it by applying these characteristics and to achieve this, some links are defined between TMMI practices and relevant DO-178C sections by referencing DO-178C sections from within the practices. Another point is that, there are some common terms and concepts (e.g., test strategy, test policy and test goals) placed in multiple TMMI practices. Therefore, when a link (reference from a practice to DO-178C sections) is defined for a practice to implement its common term considering domain specific characteristics, other practices containing the same term will also apply to DO-178C characteristics and all relevant practices will apply to DO-178C characteristics within the links.

Safety critical airborne software has specific terms and definitions, which are gathered from DO-178C handbook as:

1. Failure condition categorization, software level definitions considering failure conditions (from Level-A catastrophic to Level-E no effect) (defined in DO-178C sections 2.3, 2.3.2, 2.3.3)
2. Domain specific software considerations and their verification processes (verification of parameter data items, field-loadable software, multi-version dissimilar software etc.) (defined in DO-178C sections 6.6, 2.5)
3. Traceability definitions and scope (defined in DO-178C section 6.5)

4. Change related re-verification activities, change reviews, configuration management (defined in DO-178C sections 4.4.2, 7.2, 7.2.5)
5. Test environment (defined in DO-178C section 6.4.1)
6. Purpose of software verification (defined in DO-178C section 6.1)
7. Software verification plan and result (defined in DO-178C section 11.3, 11.4)
8. Considerations about testing and scope of testing (defined in DO-178C sections 6.2, 6.4, 6.4.2, 6.4.3)
9. Software quality assurance process (defined in DO-178C section 8.1)
10. Tool qualification (defined in DO-178C section 12.2)

After determining domain specific characteristics from DO-178C handbook, each TMMI practice is reviewed and references to relevant DO-178C handbook sections are provided for practices to implement them considering domain specific characteristics. The reference from a TMMI practice to DO-178C section is called a link. To apply a TMMI practice, links would be helpful to describe and implement given practices within safety-critical avionics software characteristics.

In the document called "TMMi Framework R1 3" [5], a practice can refer to another practice (or a goal) from different process areas when their scopes are related with each other. For example, Level-2 practice called "Perform a generic product risk assessment" refers to "Perform a Product Risk Assessment" goal which is defined in another process area. On the other hand, a TMMI practice can be related with a practice from another level without. Instead of directly referring to the name of relevant practice, the practice can refer to a term that is previously defined in the relevant practice. For example, Level-3 practice called "Define the test organization" offers test organization establishment based on previously described test policy and goals in Level-2.

Therefore, a term or concept can be expressed several times in different TMMI practices and a definition specified in a practice can affect other relevant practices. Considering the whole TMMI practices, the common terms placed in TMMI practices are listed as:

1. Business needs and objectives;
2. Test strategy, test policy and test goals;
3. Test environment;
4. Risk assessment & software level and categorization;
5. Test approach.

Common terms contain references (links) to relevant DO-178C sections and as a result, the items “Business needs and objectives” and “Test strategy, test policy and test goals” appear in 14 different practices. Also, the item “Test environment” is observed in 34 practices of TMMI. “Test approach” is also expressed or revisited in further practices in TMMI. Lastly, “Risk assessment & software level and categorization” is indicated 12 times in TMMI practices. Therefore, links offered for a practice by this guidance approach can affect other practices if it contains common terms or referred by other practices. Nevertheless, it must be known that guidance document offers many links besides common terms containing practices. Common terms are defined to pinpoint that, when a practice with a common term has a link that is offered by guidance document approach, it can affect more than one practice compared to other practices. The TMMI practice list with relevant DO-178C references is provided in Table 5.12. The first column defines TMMI practices and the second columns defines links. The last column called “Relevant Practice(s) Defined in TMMI” refers to practices that are related with previously defined practices or process areas.

Table 5.12. TMMI practice list with DO-178C section references (links)

TMMI PRACTICE	Reference To Relevant DO-178C Section	Relevant Practice(s) Defined in TMMI
Define test goals	1) Section 6.1- Purpose of Software Verification 2) Section 6.4.3.b- Requirement Based Software Integration Testing 3) Section 2.3- System Safety Assessment Process And Software Level	
Define test policy	1) Section 6.4- Software Testing for Definition Of Testing 2) Section 6.2- Overview Of Software Verification Process Activities	
Distribute the test policy to stakeholders	N/A	
Perform a generic product risk assessment	1) Section 2.3.2- Failure Condition Categorization 2) Section 2.3.3- Software Level Definition	

Define test strategy	1) Section 6.4- Software Testing for Test Types 2) Section 6.4.2- Requirement Based Test Selection for Test Case Design Tech. 3) Section 6.4.3.B - Requirement Based Software Integration Testing for Integration Testing 4) Section 6.4.1- Test Environment 5) Section 11.14- Software Verification Results For Test Documentation And Reporting	Test Goals & Test Policy Practices
Distribute the test strategy to stakeholders	N/A	
Define test performance indicators	N/A	Test Goals and Test Policy Practices
Deploy test performance indicators	1) Section 6.4.4 Test Coverage Analysis (6.4.4.1 Test Coverage Analysis And 6.4.4.2 Structural Coverage Analysis)	
Define product risk categories and parameters	1) Section 2.3.2 - Failure Condition Categorization 2) Section 2.3.3-Software Level Definition 3) Section 7.2.5 Change Reviews For Change Related Risks	
Identify product risks	1) Section 2.3.2 - Failure Condition Categorization 2) Section 2.3.3-Software Level Definition	
Analyze product risks	1) Section 2.3.2 - Failure Condition Categorization 2) Section 2.3.3-Software Level Definition	
Identify items and features to be tested	1) Note from Section 6.4 2) Section 6.6- Verification of Parameter Data Items 3) Section 2.5- Software Considerations	
Define the test approach	1) Section 6.4.2.1- Normal Range Test Cases 2) Section 6.4.2.2- Robustness Test Cases 3) Section 6.4.3- Requirement Based Testing Methods 4) Section 6.4.1- Test Environment and Limitations 5) Section 4.4.2.c (for retesting)	
Define entry criteria	N/A	
Define exit criteria	N/A	
Define suspension and resumption criteria	N/A	
Establish a top-level work breakdown structure	1) Section 7.2-Configuration Management Process Activities	
Define test lifecycle	N/A	
Determine estimates for test effort and cost	1) Section 6.4.1- Test Environment 2) Section 2.3.2- Software Level Definition (for Priority Level Of Related Product Risk)	
Establish the test schedule	N/A	
Plan for test staffing	N/A	
Plan stakeholder involvement	N/A	
Identify test project risks	N/A	

Establish the test plan	5) Section 11.3 - Software Verification Plan 6) Section 4.4.2.C (Reverification After Change) 7) Section 12.1.3 (Change of Application Or Development Environment Can Require Reverification) 8) Section 11.14- Software Verification Results (for Test Reporting)	Define Test Approach Practice
Review test plan	1) Section 11.3 - Software Verification Plan (In Level of Integration Testing Activities)	
Reconcile work and resource levels	1) Section 2.5- Software Considerations	
Obtain test plan commitments	N/A	
Monitor test planning parameters	N/A	
Monitor test environment resources provided and used	N/A	
Monitor test commitments	N/A	
Monitor test project risks	N/A	
Monitor stakeholder involvement	N/A	
Conduct test progress reviews	N/A	
Conduct test progress milestone reviews	N/A	
Check against entry criteria	N/A	
Monitor defects	N/A	
Monitor product risks		
Monitor exit criteria	N/A	
Monitor suspension and resumption criteria	N/A	
Conduct product quality reviews	N/A	
Conduct product quality milestone reviews	N/A	
Analyze issues	N/A	
Take corrective action	N/A	
Manage corrective action	N/A	
Identify and prioritize test conditions	1) Section 6.3.1 - Reviews And Analysis Of High-Level Requirement (For Integration Testing) 2) Section 6.4.2- Requirement Based Test Selection (6.4.2.1 Normal Range Test Case & 6.4.2.2 Robustness Test Cases) 3) Section 6.4.3- Requirement Based İntegration Testing 4) Section 6.5-Traceability 5) Section 6.2.b- Overview of Software Verification Process Activities	
Identify and prioritize test cases	1) Section 6.4.4.1 & 6.4.4.2& 6.4.4.3 2) Section 6.4.1 Test Environment	

Identify necessary specific test data	N/A	
Maintain horizontal traceability with requirements	1) Section 6.5- Traceability 2) Section 6.4.4.1- Requirements-Based Test Coverage Analysis	
Develop and prioritize test procedures	1) Note From Section 6.4	
Create specific test data	N/A	
Specify intake test procedure	N/A	
Develop test execution schedule	N/A	
Perform intake test	N/A	
Execute test cases	1) Section 4.4.2.c- Reverification After Change- 2) Section 12.1.3 -Change of Application or Development Environment Can Require Reverification 3) Section 6.2.d For Reverification 4) Section 6.6- Verification Of Parameter Data İtems & Change) 5) Section 11.14- Software Verification Results (for Test Reporting)	
Report test incidents	1) Section 7.2.3- Problem Reporting, Tracking and Corrective Action 2) Section 11.14- Software Verification Results (for Test Documentation and Reporting) 3) Section 11.17- Problem Reports	
Write test log	N/A	
Decide disposition of test incidents in configuration control board	1) Section 7.1.e 2) Section 7.2.5 Change Review	
Perform appropriate action to fix the test incident	1) Section 7.2.3- Problem Reporting, Tracking And Corrective Action	
Track the status of test incidents	1) Section 7.2.3- Problem Reporting, Tracking And Corrective Action	
Elicit test environment needs	1) Section 4.4.3- Test Environment (Planning for Specific Needs) 2) Section 6.4.1 -Test Environment 3) Section 12.3.2- Multi-Version Dissimilar Software Verification Can Require Separate Test Environments 4) Section 6.2.b- Overview of Software Verification Process Activities (for not Testable Requirements)	
Develop the test environment requirements	1) Section 4.4.- Test Environment (Planning) For Specific Needs 2) Section 6.4.1- Test Environment 1) Section 12.3.2 (Multi-Version Dissimilar Software Verification Can Require Separate Teat Environments)	
Analyze the test environment requirements	1) Section 4.4.3- Test Environment (Planning) For Specific Needs 2) Section 6.4.1- Test Environment	

	3) Section 12.3.2- (Multi-Version Dissimilar Software Verification Can Require Separate Teat Environments)	
Implement the test environment	N/A	
Create generic test data	N/A	
Specify test environment intake test procedure	N/A	
Perform test environment intake test	N/A	
Perform systems management	N/A	
Perform test data management	N/A	
Coordinate the availability and usage of the test environments	N/A	
Report and manage test environment incidents	N/A	
Define the test organization	1) Section 6.2.e- Overview of Software Verification Process Activities	Test Goals and Policy, Test Strategy Practices
Obtain commitments for the test organization	N/A	
Implement the test organization	N/A	
Identify test functions	N/A	Test Goals and Policy, Test Strategy Practices
Develop job descriptions	N/A	
Assign staff members to test functions	N/A	
Establish test career paths	N/A	
Develop personal test career development plans	N/A	
Assess the organization's test process	1) Apply To Previous Test Goal/ Policy And Strategy Practices (Referring to DO-178C) - Section 6.4 Software Testing	
Identify the organization's test process improvements	1) Section 6.1- Purpose Of Software Verification 2) 6.4.3.B- Requirement Based Software Integration Testing	Test Goals and Policy, Test Strategy Practices
Plan test process improvements	N/A	
Implement test process improvements	N/A	
Deploy standard test process and test process assets	N/A	
Monitor implementation	N/A	
Incorporate lessons learned into the organizational test process	N/A	Test Goals and Policy, Test Strategy Practices
Identify the strategic test training needs	N/A	Test Goals and Policy, Test Strategy Practices
Align the organizational and project test training needs	N/A	

Establish an organizational test training plan	N/A	
Establish test training capability	N/A	
Deliver test training	N/A	
Establish test training records	N/A	
Assess test training effectiveness	N/A	
Establish standard test processes	1) Section (6.4 To 7.0) Software Testing	
Establish test lifecycle model descriptions addressing all test levels	1) Section (6.4 To 7.0) Software Testing 2) Section 11.3 Software Verification Plan	
Establish tailoring criteria and guidelines	N/A	
Establish the organization's test process database	N/A	
Establish the organization's test process asset library	N/A	Test Goals and Policy, Test Strategy Practices
Establish work environment standards	N/A	
Establish integrated lifecycle models	1) Section 6, Figure 6-1 (Refers To Lifecycle and Processes), (Section 6.4.4.2 & 6.4.4.3) 2) Section 5.1.2 Software Requirement Process Activities	
Review integrated lifecycle models	N/A	
Obtain commitments on the role of testing within the integrated lifecycle models	N/A	
Perform a product risk assessment	N/A	Perform A Product Risk Assessment
Establish the test approach	N/A	Establish A Test Approach
Establish test estimates	N/A	Establish Test Estimates
Define the organization for testing	N/A	
Develop the master test plan	1) Section 6.2 For Software Verification Considerations	Develop A Test Plan Test Environment Establish A Test Approach
Obtain commitment to the master test plan	N/A	
Identify non-functional product risks	N/A	Perform A Product Risk Assessment
Analyze non-functional product risks	N/A	Define Product Risk Categories And Parameters
Identify non-functional features to be tested	N/A	
Define the non-functional test approach	N/A	

Define non-functional exit criteria	N/A	
Identify and prioritize non-functional test conditions	1) Section 6.3.3- Review And Analyses of Software Architectures	
Identify and prioritize non-functional test cases	N/A	
Identify necessary specific test data	N/A	
Maintain horizontal traceability with non-functional requirements	N/A	
Develop and prioritize non-functional test procedures	N/A	
Create specific test data	N/A	
Execute non-functional test cases	N/A	
Report non-functional test incidents	N/A	
Write test log	N/A	
Identify work products to be reviewed	1) Section 6.3.1 to 6.3.5- Software Reviews An Analyses (for Integration Testing Level High-Level Req. Review is Applied)	
Define peer review criteria	1) Section 6.3.1 To 6.3.5-Software Reviews An Analyses (for Integration Testing Level Section: 6.3.1 High-Level Req. Review is Applied)	
Conduct peer reviews	1) Section 6.3.1 To 6.3.5-Software Reviews An Analyses (for Integration Testing Level Section: 6.3.1 High-Level Req. Review İs Applied)	
Testers review test basis documents	1) Section 6.3.1.D - Verifiability Activity Of Reviews And Analyses Of High-Level Requirements (For İntegration Test)	
Analyze peer review data	N/A	
Establish test measurement objectives	N/A	
Specify test measures	1) Section 6.4.4 (6.4.4.1, 6.4.4.2 for Integration Testing)	
Specify data collection and storage procedures	N/A	
Specify analysis procedures	N/A	
Collect test measurement data	N/A	
Analyze test measurement data	N/A	
Communicate results	N/A	
Store data and results	N/A	
Identify product quality needs	1) Section 8.1- Software Quality Assurance Process Objectives (for Review The Organization's Objectives For Product Quality)	

Define the project's quantitative product quality goals	1) Section 8.2 Software Quality Assurance Process Activities	
Define the approach for measuring progress toward the project's product quality goals	N/A	
Measure product quality quantitatively throughout the lifecycle	N/A	
Analyze product quality measurements and compare them to the product's quantitative goals	N/A	
Relate work products to items and features to be tested	1) Section 6.5- Software Verification Process Traceability	Perform A Product Risk Assessment
Define a coordinated test approach	N/A	
Define peer review measurement guidelines	N/A	
Define peer review criteria based on product quality goals	N/A	
Measure work product quality using peer reviews	N/A	
Analyze peer review results	N/A	
Revise the products risks as appropriate	N/A	
Revise the test approach as appropriate	N/A	
Define defect selection parameters and defect classification scheme	N/A	
Select defects for analysis	N/A	
Analyze causes of selected defects	N/A	
Propose solutions to eliminate common causes	1) Section 6.4.3 (for Typical Errors of Integration Tests For Common Cause)	
Define action proposals and submit improvement proposals	1) Section 7.2.5 Change Reviews	
Establish test process performance objectives	N/A	Test Goals and Policy, Test Strategy Practices
Establish test process performance measures	N/A	
Establish test process performance baselines	N/A	
Apply statistical methods to understand variations	N/A	
Monitor performance of the selected test processes	N/A	
Develop operational profiles	N/A	

Generate and execute statistically selected test cases	N/A	
Apply statistical test data to make stop-test decisions	N/A	
Collect and analyze test process improvement proposals	N/A	
Pilot test process improvement proposals	N/A	
Select test process improvement proposals for deployment	N/A	
Identify and analyze new testing technologies	1) Section 12.2- Tool Qualification	
Pilot new testing technologies	1) Section 12.2- Tool Qualification	
Select new testing technologies for deployment	N/A	
Plan the deployment	1) Section 6.4.1 Test Environment	
Manage the deployment	N/A	
Measure improvement effects	N/A	
Identify re-usable test assets	N/A	
Select test assets to be added to the re-use library	N/A	
Deploy re-usable test assets	N/A	
Apply re-usable test assets in projects	N/A	

5.3. Case Study

The effectiveness and validity of the proposed maturity guidance approach for integration testing processes in avionics domain are investigated by applying case study research method [26]. In the following subsections 5.3.1 through 5.3.4; details about research design, research context, data collection and analysis, and research results are presented.

5.3.1 Research Design

This section presents the design of a single embedded case study to address the following the research questions:

- RQ-1: What is the difference between assessment outcomes obtained by guidance approach and TMMI model?
- RQ-2: What are maturity levels assessed by guidance approach and TMMI model?
- RQ-3: What are the challenges and advantages of maturity assessment with guidance approach?
- RQ-4: What is the applicability of improvement actions offered by guidance approach?

The single embedded case study consists of two different assessments of avionics integration testing on the same processes and same projects, as shown in Figure 1. The first one (embedded unit of analysis-1) is the assessment process considering TMMI model. The second one (embedded unit of analysis-2) is the assessment process considering guidance approach. It is chosen to perform an embedded case study, because such a study is helpful to understand the assessment outcomes of guidance approach as different from the ones of TMMI model.

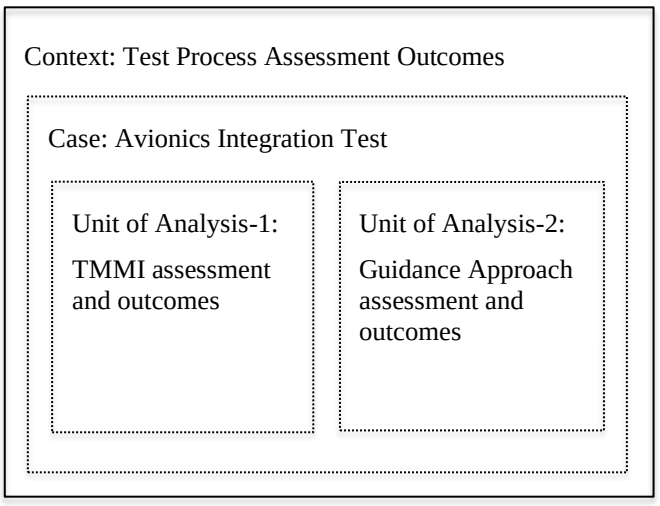

Figure 1. Single Embedded Case Study Design

5.3.2 Research Context (Investigated Company and Project)

The context information of the investigated company, projects and people is given below.

Company: The case study is carried out within the Turkish Aerospace Industries (TAI) which operates in the aviation and space industry in Turkey. TAI has various projects such as design, development and production of utility helicopters, Unmanned Aircraft Vehicles (UAVs), target aircrafts, and air fighters. Also, it offers integration, modernization and modification programs. There are more than 10 thousand employees in TAI working on these projects.

Projects: In the case study, Unmanned Aircraft Vehicle (UAV) projects are considered. There are five major projects in UAV Systems and there is one department which is responsible for integration testing of UAV projects. Since test processes are performed by a same team by using same procedures, all projects are involved in assessment process of this case study.

People: Avionics integration test team members (test engineers) are involved in the case study. The team is divided into two assessment groups to implement two sub-studies simultaneously. Overall, 8 people are involved in this study. Each team consists of 4 test engineers and it is intended to establish balanced groups considering their experiences in testing. Test engineers who have had more than 2.5 years of working experience in integration testing of avionics software are involved in assessment groups. Also, test engineers who are familiar with DO-178C are involved in the second assessment group.

Process: Each assessment group tries to assess test processes of the same projects. In the first sub-study, the first group applies TMMI model for informal (internal) assessment to detect strong and weak points of integration testing processes. On the other hand, the second group applies TMMI model but now also considering guidance approach (with DO-178C references called links proposed in Table 5.12) and implements informal assessment on the same test processes. It should be highlighted that the assessments of the two sub-groups are held simultaneously and the groups are not knowledgeable about findings of each other until the assessments are complete. The obtained results are compared to understand the effect of domain-specific maturity guidance approach.

5.3.3 Data Collection and Analyses

In the case study, since upper levels are not meaningful for integration testing level internal assessment, only level-2 and level-3 practices of TMMI are considered. These practices are used as a checklist to find out strengths and weaknesses of integration test processes. Then, improvement actions are gathered as outcomes. In the first sub-study, strengths and weaknesses are detected and improvement actions from TMMI based assessment are gathered about avionics integration test processes by the first group. In the second sub-study, same TMMI practices with links (defined in guidance approach) are used as a checklist to detect strengths and weaknesses of same test processes and this time more improvement actions are detected by the second group. As a result, improvement actions and outcomes of two sub-studies provided by the two groups are compared and results are reported in this study.

Table 5.13 shows internal assessment results of test processes according to TMMI practices. Since two groups work on the same projects in the same team and try to assess same test processes, strengths and weaknesses of their test processes are the same, as it is shown in Table 5.13. Columns of this table refers to TMMI practices, achievement rate of relative practice, strengths and weaknesses of test processes, respectively. Internal assessment is also called as informal assessment as defined in TMMi Assessment Method Application Requirements (TAMAR) R1.1 [25].

Table 5.13. Assessment of test processes with respect to TMMI practices

Practice	**Achievement Rate** ***F: Fully Achieved*** ***P: Partially Achieved*** ***N: Not Achieved*** ***N/A: Not Applicable***	**Strengths**	**Weaknesses**
Identify items and features to be tested	L	Sub-practices are largely achieved	Documentation is needed for items to be tested and not to be tested.

Define the test approach	L	There is a document defines test approach and test case.	Scope of document need to be revised. Documents are not up-to-date.
Define entry criteria	L	Sub-practices are largely achieved, Release of software, Change request, incident closures etc. are used as entry criteria	
Define exit criteria	L	Sub-practices are largely achieved, Test reports and process diagrams are used	
Define suspension and resumption criteria	L	Sub-practices are largely achieved, There are some unwritten laws	
Establish a top-level work breakdown structure	F	Sub-practices are fully achieved	
Define test lifecycle	F	Sub-practices are fully achieved	
Determine estimates for test effort and cost	L	Sub-practices are fully achieved. Test effort estimated by test executers. Test effort and test environment relation is known.	
Establish the test schedule	F	Sub-practices are fully achieved	
Plan for test staffing	F	Sub-practices are fully achieved	
Plan stakeholder involvement	N		Sub-practices are not achieved
Identify test project risks	N		Sub-practices are not achieved
Establish the test plan	L	Sub-practices are fully achieved	

Review test plan	L	Sub-practices are largely achieved	
Reconcile work and resource levels	L	Sub-practices are largely achieved	
Obtain test plan commitments	N/A	Not applicable	
Monitor test planning parameters	Not achieved	Not achieved	Measure test performance mentioned in TMMI sub-practices
Monitor test environment resources provided and used	Not achieved	Not achieved	Analyze test environment usage and planned (expected).
Monitor test commitments	N/A	Not Applicable	
Monitor test project risks	Not achieved	Not achieved	Test risk assessment and monitoring is needed.
Monitor stakeholder involvement	N/A	Not applicable	
Conduct test progress reviews	N/A	Not applicable	
Conduct test progress milestone reviews	N/A	Not applicable	
Check against entry criteria	N/A	Not applicable	
Monitor defects	Not achieved	Not achieved	No monitoring process for not-critical defects
Monitor product risks	N/A	Not applicable	

Monitor exit criteria	F	Sub-practices are fully achieved. Exit criteria is defined. Verification of change requests, incident closure processes are used as exit criteria.	
Monitor suspension and resumption criteria	F	Sub-practices are fully achieved.	
Conduct product quality reviews	N/A	Not applicable	
Conduct product quality milestone reviews	N/A	Not applicable	
Analyze issues	Not achieved	Not achieved	Differences form plan must be analyzed
Take corrective action	Not achieved	Not achieved	
Manage corrective action	Not achieved	Not achieved	
Identify and prioritize test conditions	L	Sub-practices are largely achieved.	
Identify and prioritize test cases	L	Sub-practices are largely achieved.	
Identify necessary specific test data	N/A	Not applicable	
Maintain horizontal traceability with requirements	F	Sub-practices are fully achieved.	

Develop and prioritize test procedures	L	Sub-practices are largely achieved.	
Create specific test data	F	Sub-practices are fully achieved. Specific test scenarios and layouts were generated and applied in the execution of tests.	
Specify intake test procedure	F	Sub-practices are fully achieved. Intake test executed at the end of processes.	
Develop test execution schedule	F	Sub-practices are fully achieved. Test execution schedule is managed with Excel tables with defining resources and testers.	
Perform intake test	F	Sub-practices are fully achieved. Intake test executed at the end of detailed tests processes.	
Execute test cases	F	Sub-practices are fully achieved. Parameter data item tests are executed (this is not mentioned in TMMI)	
Report test incidents	F	Sub-practices are fully achieved. Incident reporting is managed by a tools.	
Write test log	F	Sub-practices are fully achieved.	
Decide disposition of test incidents in configuration control board	L	Sub-practices are largely achieved. Assessment of test incidents and change effects are discussed in different meetings	.
Perform appropriate action to fix the test incident	F	Sub-practices are fully achieved.	
Track the status of test incidents	L	Sub-practices are largely achieved.	Incident report status is not tracked periodically for no corrective actions taken incidents.

Elicit test environment needs	P	Sub-practices are partially achieved.	Determine expectations and constraints of test environment.
Develop the test environment requirements	P	Sub-practices are partially achieved.	Test environment documents should be revised and there must be documents for other projects.
Analyze the test environment requirements	P	Sub-practices are partially achieved.	Test environment requirements are documented in 2014 for only first project.
Implement the test environment	P	Sub-practices are partially achieved.	Test environment requirements are documented in 2014 just for first project. Documents are not up-to-date. Test environment is implemented firstly within this document but there are no documents/requirements for other projects. Test environment implementation is managed by expertize.
Create generic test data	N	Not achieved	
Specify test environment intake test procedure	N	Not achieved	
Perform test environment intake test	N	Not achieved	
Perform systems management	F	Sub-practices are fully achieved. An organizational test is established for system management.	
Perform test data management	F	Sub-practices are fully achieved. Test procedures and test scripts are stored.	

Coordinate the availability and usage of the test environments	L	Sub-practices are largely achieved.	Test environment usage procedure is needed.
Report and manage test environment incidents	L	Sub-practices are largely achieved. Reporting system can be used for environment incidents.	Environment incident and its solution can be documented.
Define the test organization	F	Sub-practices are fully achieved. Integration test organization was established.	
Obtain commitments for the test organization	F	Sub-practices are fully achieved. Integration test organization was established.	
Implement the test organization	F	Sub-practices are fully achieved. Integration test organization was established.	
Identify test functions	F	Sub-practices are fully achieved. Integration test organization was established. Integration Test, Test Team Leader & team members: Test engineers are the roles.	
Develop job descriptions	F	Sub-practices are fully achieved.	
Assign staff members to test functions	F	Sub-practices are fully achieved.	
Establish test career paths	N/A	Not applicable - organizational level	No specific test career path is defined for test engineers.

Develop personal test career development plans	N/A	Not applicable - organizational level	
Assess the organization's test process	Not achieved	Not achieved	No test process assessment procedure was implemented before.
Identify the organization's test process improvements	Not achieved	Not achieved	No test process assessment procedure was implemented before.
Plan test process improvements	Not achieved	Not achieved	No test process assessment procedure was implemented before.
Implement test process improvements	Not achieved	Not achieved	No test process assessment procedure was implemented before.
Deploy standard test process and test process assets	L	Sub-practices are largely achieved. Documentation is done before. Tools are used actively for test process management.	Documents are not up-to-date.
Monitor implementation	L	Sub-practices are largely achieved.	
Incorporate lessons learned into the organizational test process	P	Sub-practices are partially achieved. Reports are published at the end of test processes.	Test process improvement is not aimed recently.
Identify the strategic test training needs	L	Sub-practices are largely achieved. There are some trainings (mandatory or voluntarily) aiming engineers to gain several competences.	Domain based trainings not determined specifically.

Align the organizational and project test training needs	L	Sub-practices are largely achieved. There are some trainings (mandatory or voluntarily) aiming people to gain several competences.	Domain based trainings not determined specifically
Establish an organizational test training plan	N	Not achieved	Test training plan is not established.
Establish test training capability	N	Not achieved	Test training plan is not established
Deliver test training	N/A	Not applicable-organizational level	Test training plan is not established
Establish test training records	N/A	Not applicable-organizational level	
Assess test training effectiveness	N/A	Not applicable-organizational level	
Establish standard test processes	L	Sub-practices are largely achieved. Documentation is done before for test processes, peer review processes etc. Tools are used actively for test process management. (DOORS, JIRA)	Documents are not up-to-date.
Establish test lifecycle model descriptions addressing all test levels	L	Sub-practices are largely achieved.	
Establish tailoring criteria and guidelines	N/A	Not applicable	
Establish the organization's test process database	N	Not achieved	No database for test processes.

Establish the organization's test process asset library	L	Sub-practices are largely achieved. Documents are placed in a library system.	Documents are not up-to-date.
Establish work environment standards	N/A	Not applicable-organizational level	
Establish integrated lifecycle models	N/A	Not applicable-organizational level	
Review integrated lifecycle models	N/A	Not applicable-organizational level	
Obtain commitments on the role of testing within the integrated lifecycle models	N/A	Not applicable-organizational level	
Perform a product risk assessment	N/A	Not applicable-organizational level	

Firstly, the assessment process has shown that there are some documentation needs. Some of the documents were not up-to-date and they need update. In assessment it is shown that, practices at the top of Table 5.13, which belong to maturity level-2, are largely or fully achieved by test processes. However, some of the level-3 practices are partially achieved. Since it is an informal type assessment, the aim is detecting strengths and weaknesses of own test processes, and there are some practices labeled as not-applicable.

The result of assessment processes is summarized in Appendix-3. Achievement rates of practices are noted. The achievement rate of a process area is determined by the lowest achievement rated practice of it and the level of maturity is defined considering the achievement rate of process areas. Some of the process areas contain practices that are "not achieved" because of the outdated documents and stakeholder involvement practices. Since

this study is an informal assessment and it aims to detect the strong and weak points of the processes, it does not strictly apply ratings and the maturity level can be defined as Level-2.

In the first sub-study, assessment process is managed only by considering TMMI practices. In the second sub-study, on the other hand, assessment process is implemented considering the same practices with their DO-178C links defined for these practices. These studies provided improvement actions. Table 5.14 shows improvement actions for these two sub-studies with relevant TMMI practices.

Table 5.14 Comparison of improvement suggestions by two sub-studies

TMMI Practice	Improvement Actions offered by 1st Sub-study	Improvement Actions offered by 2nd Sub-study
Identify items and features to be tested	1. Define a rule set and provide documentation for items to be teste and not to be tested.	1. Define a rule set and provide documentation for items to be teste and not to be tested. 2. Determine levels of requirements and refer to this level on documents. 3. Refer levels of requirements that are not to be tested to remove duplications on tests. 4. Parameter data items to be tested could be documented.
Define the test approach	1. High level requirement verification is implemented in test processes. Integration tests are generated considering Level-2 requirements. Test approach document should refer the levels and structures.	1. Robustness test should be mentioned in the test approach documents as a test case selection part. 2. Error that can be revealed by integration can be added into test approach documents as a part of error situations. They can be helpful as a checklist for test design phase. 3. Document can refer test case duplication removal considering levels of tests and requirements. 4. High level requirement verification is implemented in test processes. Integration tests are generated considering Level-2 requirements. Test approach document should refer the levels and structures to determine test approach. 5. Parameter data item verification depends on personal knowledge documents should refer them to define standards for.

Establish the test plan	1. High level requirement verification is implemented in test processes. Integration tests are generated considering Level-2 requirements. Test approach document should refer the levels and structures. 2. Test environment limitations should be defined and documented. 3. Some of the tests must be executed on specific test environments. Differences of these environments should be determined and documented.	1. Robustness test should be mentioned in the test approach documents as a test case selection part. 2. Error that can be revealed by integration can be added into test approach documents as a part of error situations. They can be helpful as a checklist for test design phase. 3. Document can refer test case duplication removal considering levels of tests and requirements. 4. High level requirement verification is implemented in test processes. Integration tests are generated considering Level-2 requirements. Test approach document should refer the levels and structures. 5. Parameter data item verification depends on personal knowledge, documents should refer them to define standards for. 6. Test environment limitations should be defined and documented. 7. Reverification conditions should be defined and documented. For example; development environment change effect on previous verification process should be analyzed and the scope of retest needs must be defined. 8. Some of the tests must be executed on specific test environments. Differences of these environments should be determined and documented.
Monitor test planning parameters	1. Measure test performance as mentioned in TMMI sub-practices.	1. Measure test performance as mentioned in TMMI sub-practices.
Monitor test environment resources provided and used	1. Analyze and compare test environment usage (observed) and planned (expected).	1. Analyze and compare test environment usage (observed) and planned (expected).
Monitor test project risks	1. Determine test project risks considering test environment and limitations and document.	1. Determine test project risks considering test environment and limitations. 2. Document risks considering certification for simulator/emulators.
Monitor Defects	1. Monitor defects both critical and not-critical, follow their actions. 2. Meeting arrangements should be done to handle no corrective action taken defects.	1. Monitor defects both critical and not-critical, follow their actions. 2. Meeting arrangements should be done to handle no corrective action taken defects.
Identify and prioritize test conditions	1. High level requirement verification is implemented in test processes. Integration tests are generated considering Level-2 requirements.	1. High-level requirement analysis is done in PR (Peer Review) process as test basis analysis. Updates for documentations should be

	Test approach document should refer the levels and structures. 2. Test environment limitations should be defined and documented.	considered. 2. Robustness test can be added to documents. 3. Test cases generation part of documents refers to common errors and erroneous situations. DO-178C document describes errors that can be revealed by integration test. Document updates should be done considering both definitions and also expert opinions. 4. Documents can define test case duplication conditions to reduce test effort. 5. High level requirement verification is implemented in test processes. Integration tests are generated considering Level-2 requirements. Test approach document should refer the levels and structures. 6. Parameter data item verification depends on personal knowledge, documents should refer them to define standards for. 7. Test environment limitations should be defined and documented. 8. Reverification conditions should be defined and documented. For example; development environment change effect on previous verification process should be analyzed and the scope of retest needs must be defined. 9. Traceability definitions (as mentioned in DO-178C) can be documented and it should be managed by tools.
Identify and prioritize test cases	1. Priority between tests cases can be defined as the part of test process document 2. Test environment needs should be considered before test execution	1. Traceability definitions (as mentioned in DO-178C) can be documented and it should be managed by tools. 2. Priority between tests cases can be defined as the part of test process document. 3. Test environment needs should be considered before test execution. 4. DO-178C test cases can be used as a checklist and it can be helpful to prioritize test cases (robustness test, normal range test, integration test, levels of tests).
Develop and prioritize test procedures	1. Integration test procedure automation should be considered (with tools or scripts), test procedure standard document may be updated	1. Take actions to remove duplication of test procedures. 2. Integration test procedure automation should be considered (with tools or scripts), test procedure standard document may be updated.

Track the status of test incidents	1. Incident status tracking meetings can be arranged for unhandled (no corrective actions taken) incidents	1. Incident status tracking meetings can be arranged for unhandled (no corrective actions taken) incidents.
Elicit test environment needs	1. Determine expectations & constraints of test environment considering closely resemble of target environment.	1. Determine expectations & constraints of test environment considering closely resemble of target environment. 2. Refer if any emulator/ simulator certification.
Develop the test environment requirements	1. Test environment requirements are documented in 2014 for only first project. Document should be updated and there must be documents for other projects.	1. Test environment requirements are documented in 2014 for only first project. Document should be updated and there must be documents for other projects. 2. Documents (requirement) doesn't mention simulators in detail. Details should be provided. 3. Multiple test environment can be defined as a part of documentation.
Coordinate the availability and usage of the test environments	1. Test environment usage procedure is needed. Scheduling with Excel is manually managed by people.	1. Test environment usage procedure is needed. 2. Scheduling with Excel is manually managed by people.
Report and manage test environment incidents	1. Test environment incident reporting meetings, tools or system can be used.	1. Test environment incident reporting meetings, tools or system can be used.
Assess the organization's test process	1. Test process assessment document can be prepared as a checklist.	1. Test process assessment document can be prepared as a checklist. 2. Document can refer normal range test case, robustness test cases and integration test scope
Deploy standard test process and test process assets	1. Document updates can be done. 2. Test process tools could be implemented.	1. Document updates can be done. 2. Test process tools could be implemented.
Incorporate lessons learned into the organizational test process	1. Test process enhancement topic and related actions should be considered.	1. Test process enhancement topic and related actions should be considered.
Establish an organizational test training plan	1. Trainings related with test should be determined and planned for test staff. (Trainings about DO-178C and avionics system tests etc.)	1. Trainings related with test should be determined and planned for test staff. (Trainings about DO-178C and avionics system tests etc.)
Establish standard test processes	1. Document updates can be done. 2. Test process tools could be implemented.	1. Document updates can be done. 2. Test process tools could be implemented.
Establish the organization's test process database	1. Database for test procedures, test reports, test results assessment should be established.	1. Database for test procedures, test reports, test results assessment should be established.

5.3.4 Case Study Results

In this subsection, assessment findings are reported and discussed in relation to the research questions raised for the case study.

RQ-1: What is the difference between assessment outcomes obtained by guidance approach and TMMI model?

It is shown that the number and the depth of improvement actions offered by sub-study 2 is more that the first one. Also, since the sub-study 2 takes domain specific characteristics into account, improvement actions offered by it are more related with avionics test processes. Since some of the TMMI practices contains common terms and are related with each other, some of the improvement actions are proposed more than once for different practices. For example, practices related with business needs and objectives have similar improvement actions. Furthermore, some of the practices like test approach, test plan etc. are referenced by further practices so the improvement actions that will be implemented to enhance test processes will affect further assessments.

Table 5.15 summarizes the number of improvement actions provided by the sub-studies of the 1st and the 2nd assessment groups. The first column represents TMMI process areas which are applied by groups to assess their test processes. The second and the third columns represent the numbers of improvement actions provided as a result of the 1st and the 2nd sub-studies, respectively.

Table 5.15 Number of improvement actions identified in two sub-studies

TMMI Process Area	# of Improvement Actions (1st group) – TMMI only	# of Improvement Actions (2nd group) – Guidance Apr.
2.2 Test Planning	5	17
2.3 Test Monitoring and Control	6	6
2.4 Test Design and Execution	9	19
2.5 Test Environment	6	10
3.1. Test Organization	14	25
3.2 Test Training Program	4	4
3.3 Test Lifecycle and Integration	4	13
TOTAL	*48*	*94*

As a result, it is seen that the number of improvement actions, which are obtained by the 2nd group by following the approach of defining DO-178C links, is 0.95 times more than the number of improvement actions offered by the 1st group by with respect to included TMMI

process areas. Within these, practices that belong to process areas called “2.3 Test Monitoring and Control” and “3.2 Test Training Program” have the same number of identified improvement actions. It should also be noticed that, since TMMI practices contain some of the common terms previously defined, the number of improvement actions are affected by this recurring structure of TMMI and a group of improvement actions are repeated for further practices.

RQ-2: What are maturity levels assessed by guidance approach and TMMI model?

In the sub-studies, the two groups try to assess their test processes which are the same for both groups because their members work in the same department on the same projects. The results of assessment processes are summarized in Appendix-3, and achievement rates of practices are indicated. In the case study, whole TMMI practices are not considered but since this study is an informal assessment and it aims to detect the strengths and weaknesses of the test processes, the maturity level assessed by both groups can be defined as Level-2.

RQ-3: What are the challenges and advantages of maturity assessment with guidance approach?

TMMI model has some organizational level practices and practices relevant to other test levels. They are not included in the case study. Subset of the practices are applied considering the existing test processes and activities of the department. The test team had never implemented any test process assessment methods before. The members of the test team were not familiar with TMMI or test maturity model applications. On the other hand, after the case study assessment, the 2nd group shared that guidance approach helped them to understand expectations of TMMI practices and see where their internal processes are.

Most of the weaknesses was caused because of incompleteness of documents or outdated documents. TMMI assessment application showed the weaknesses or drawbacks in these documents but still, it was not adequately informing about the domain specific needs or characteristics that should be involved in these documents. Guidance approach helped the test team to understand it.

RQ-4: What is the applicability of improvement actions offered by guidance approach?

Most of the improvement actions were about documentation issues and test training plans. Since the documentation updates or test training plans are internal activities for the test team, the most of the improvement actions offered by the guidance approach were remarked applicable by the test team without organizational execution. Some of the improvement actions that needs participation of other teams such as software development teams and system design teams are more comprehensive to execute compared to internal actions but still, other teams were interested in the guidance approach and willing to be the part of the improvement actions.

Improvement actions offered by guidance approach were listed and shared with four members of the integration test team. Two of these members were previously involved in the 1st sub-study group and the other two members were involved in the 2nd sub-study group. These test team members evaluated the improvement actions in Likert Scale [1-5] to understand applicability of these improvement actions. The applicability evaluation scale was as follows: 1: Strongly disagree (not applicable at all), 2: Disagree, 3: Neutral, 4: Agree, and 5: Strongly agree (very applicable). Four people participated in the evaluation process, therefore the median values of their responses were calculated and reported in Table 5.16. Most of the improvement actions (29 of 33, %87) were responded as 5 (very applicable) by the members. Three improvement actions of 33 (%9.09) were evaluated as 4 (mostly applicable). Since the emulator/simulator certification is not easily achievable, team members labeled this improvement action as 3 (neutral).

It should be added that in the early stages of the assessment process, some of the weaknesses were detected such as unpublished documents and the improvement action as *sharing these documents with stakeholders,* and these were executed immediately. However, the test team stated that satisfaction of the improvement actions needs more time and execution plans.

Table 5.16 Applicability Evaluation of Improvement Actions

Improvement Actions	**Median of Scores (in scale 1-5)**
Define a rule set and provide documentation for items to be tested and not to be tested.	5
Determine levels of requirements and refer to this level on documents.	5
Refer levels of requirements that are not to be tested to remove duplications on tests. High-level requirements, low level requirements etc.)	5
Parameter data items to be tested could be documented. Parameter data item verification depends on personal knowledge, documents should refer to them to define standards for.	5
Robustness test should be mentioned in the test approach documents as a test case selection part.	5
Error that can be revealed by integration can be added into test approach documents as a part of error situations. They can be helpful as a checklist for test design phase.	5
Test environment limitations should be defined and documented.	5
Reverification conditions should be defined and documented. For example; development environment change effect on previous verification process should be analyzed and the scope of retest needs must be defined.	5
Some of the tests must be executed on specific test environments. Differences of these environments should be determined and documented.	5
Measure test performance as mentioned in TMMI sub-practices.	5
Analyze and compare test environment usage (observed) and planned (expected).	5
Determine test project risks considering test environment and limitations.	5
Document risks considering certification for simulator/emulators.	5
Monitor defects both critical and not-critical, follow their actions.	5
Incident status tracking is needed. Meeting arrangements should be done to handle no corrective action taken defects	5
Traceability (requirement<->test process, test procedures<->test cases, test result<->test cases) managed by tools.	5
Priority between tests cases can be defined as the part of test process document.	5
Test environment needs should be considered before test execution	5
DO-178C test cases can be used as a checklist and it can be helpful to prioritize test cases (robustness test, normal range test, integration test, levels of tests).	5
Take actions to remove duplication of test procedures (if any).	5
Integration test procedure automation should be considered (with tools or scripts), test procedure standard document can be updated.	4
Determine expectations & constraints of test environment considering closely resemble of target environment.	4
Refer if any emulator/ simulator certification.	3
Test environment requirements are documented in 2014 for only first project. Document should be updated and there must be documents for other projects.	5
Peer review documentation updates can be done.	5
Multiple test environment can be defined as a part of documentation.	5
Test environment usage procedure can be defined.	5
Test planning (scheduling) with Excel should be automated.	5
Test environment incident reporting meetings, tools or system can be used actively.	5
Test process enhancement topic and related actions should be considered.	5
Trainings related with test should be determined and planned for test staff. (Trainings about DO-178C , avionics system, test etc.)	5
Test process tools could be implemented.	4
Database for test procedures, test reports, test results assessment should be established.	5

6. CONCLUSION

This study explains the preliminary steps taken to propose an avionics software testing maturity guidance approach in order to improve integration testing processes of projects obeying to DO-178C requirements. In this context, DO-178C is analyzed to understand avionics software verification activities and needs. TMMI is taken as the base maturity model since its "process area & practice" structure is similar to the process structure in DO-78C. In the first step, TMMI practices and DO-178C activities are analyzed for bi-directional mapping with respect to the needs of avionics software testing.

A guidance approach is proposed by considering avionics software testing characteristics detected previously. In this approach, DO-178C sections called "links" are provided along with relevant TMMI practices. Then, domain specific maturity assessment is aimed. A case study is implemented to understand the effectiveness and usability of guidance approach on avionics integration test processes. Internal assessment is performed in the case study. Case study consisted of two sub-studies and in the first one, an informal TMMI assessment was applied only by considering a subset of TMMI level-2 and level-3 practices by a group of test engineers that have at least 2.5 years of work experience on testing. In the second sub-study, again an assessment process was implemented with the same practices but this time considering DO-178C links defined for these practices by another group of testers. These studies provided improvement actions and action results were compared. Each group consisted of 4 avionics integration test engineers and the total years of experience of groups were similar to each other. Each group tried to assess their test processes and since they work on the same projects in the same team, they tried to assess the same test processes. In the case study, it is aimed to see the difference between the findings obtained by TMMI model and the guidance approach for TMMI model introduced in this thesis.

In the case study it was observed that the second group who used DO-178C links to assess test processes more easily understood assessment and detected their weaknesses. Most of the weaknesses was caused because of incompleteness of documents or outdated documents. TMMI model showed the weaknesses or drawbacks of these documents but still, it was not adequately informing about the domain specific needs or characteristics that should be involved in these documents. At that point, guidance approach and provided links were helpful to improve existing documents considering domain specific needs. Document based

actions were the main test process improvement need detected by TMMI model and are open for enhancement considering the guidance approach. It is also achievable by test teams itself and does not require organizational applicability limitations. Also, test trainings were not planned and implemented in the case study environment as it is defined in TMMI model. Improvement actions related with test trainings are also applicable by the test team.

As a result, it was shown that the second sub-study (guidance approach) provided more improvement actions for avionics test processes. Since, the main problem was deficiency of avionics software domain specific maturity or test process improvement, the improvement actions offered by the second sub-study were domain specific. Therefore, guidance approach is observed as effective to provide improvement actions for avionics integration test teams.

As a future work, improvement actions provided by the guidance approach can be applied to reach higher maturity levels and improve avionics integration testing processes. The maturity assessment can be repeated after implementation of improvement actions. The guidance approach comprises also the upper levels (Level-4 and Level-5) and to reach the highest maturity level, whole guidance document can be applied. Also, the guidance approach points to some “common terms” that are gathered considering the structure of TMMI model, so maturity model studies for another domain can establish a domain based guidance approach with the help of the common terms.

7. REFERENCES

1. DO-178: Software Considerations in Airborne Systems and Equipment Certification, Washington, DC: RTCA Inc., 1981.

2. DO-178C: Software Considerations in Airborne Systems and Equipment Certification, Washington, DC: RTCA Inc., 2011.

3. Ericson, T., Subotic, A. and Ursing, S. (1997), TIM—a test improvement model. Softw. Test. Verif. Reliab., 7: 229-246.

4. J. Andersin, “TPI -a model for Test Process Improvement,” 2004. Accessed: Jun. 01, 2023. [Online]. Available: https://www.cs.helsinki.fi/u/paakki/Andersin.pdf

5. TMMi Foundation, “Test Maturity Model integration (TMMi®) Guidelines for Test Process Improvement Release 1.3 Produced by the TMMi Foundation.” Accessed: Jun. 01, 2023. [Online]. Available: https://www.tmmi.org/tmmi-documents/

6. D. M. Karr, “The Unit Test Maturity Model” Accessed: Jun. 01, 2023. [Online]. Available:http://davidmichaelkarr.blogspot.com/2013/01/the-unit-test-maturity-model.html

7. S. Reid, “Personal Test Maturity Matrix”, Accessed: Jun. 01, 2023. [Online]. Available: https://www.stureid.info/stuart-reid-software-testing/software-testing-white-papers/personal-test-maturity-matrix/

8. V. Garousi, M. Felderer and T. Hacaloğlu, "What We Know about Software Test Maturity and Test Process Improvement," in IEEE Software, vol. 35, no. 1, pp. 84-92, January/February 2018, doi: 10.1109/MS.2017.4541043.

9. ISO/IEC/IEEE 29119-3:2021 Software and systems engineering — Software testing — Part 3: Test documentation, ISO/IEC/IEEE 29119-3:2021, 2021. [Online]. Available: https://www.iso.org/standard/79429.html

10. CMMI Institute, Capability Maturity Model Integration.[Online]. Available:https://cmmiinstitute.com/resource-files/public/cmmi-v2-0-development-model”

11. Veenendaal, Erik. (2016). TMMi and ISO 29119: Friends or Foes? ,White paper TMMi Foundation.

12. Duncan, F.I. & Smeaton, A.G.. (2012). “Assessing and improving software quality in safety critical systems by the application of a SOFTWARE TEST MATURITY MODEL”. 1-4. 10.1049/cp.2012.1509.,

13. Häser, Florian & Felderer, Michael & Breu, Ruth. (2014). Test Process Improvement with Documentation Driven Integration Testing. Proceedings - 2014 9th International Conference on the Quality of Information and Communications Technology, QUATIC 2014. 156-161. 10.1109/QUATIC.2014.29.

14. Kassab, Mohamad. (2018). Testing Practices of Software in Safety Critical Systems: Industrial Survey. 359-367. 10.5220/0006797003590367.

15. Garousi, Vahid & Veenendaal, Erik. (2021). Test Maturity Model integration (TMMi): Trends of worldwide test maturity and certifications. IEEE Software. PP. 10.1109/MS.2021.3061930.

16. Bahaa Farid, Ahmed & Fathy, Enas & Abd, Mahmoud. (2015). Towards Agile Implementation of Test Maturity Model Integration (TMMI) Level 2 using Scrum Practices. International Journal of Advanced Computer Science and Applications. 6. 10.14569/IJACSA.2015.060931.

17. Garousi, Vahid & Felderer, Michael & Hacaloglu, Tuna. (2017). Software test maturity assessment and test process improvement: A multivocal literature review. Information and Software Technology. 85. 10.1016/j.infsof.2017.01.001.

18. Jang, J.-W. (2018). Improvement of the automobile control software testing process using a Test Maturity Model. Journal of Information Processing Systems. 14. 607-620. 10.3745/JIPS.04.0072.

19. Park (2021), Defense software test procedure improvement measure reflecting the TMMI. Journal of the Korea Academia-Industrial cooperation Societyi Volume 22 Issue 6

20. A. Ferreirós and L. A. V. Dias, "Evaluation of Accomplishment of DO-178C Objectives by CMMI-DEV 1.3," 2015 12th International Conference on Information Technology - New Generations, Las Vegas, NV, USA, 2015, pp. 759-760, doi: 10.1109/ITNG.2015.132.

21. ISO/IEC 25010:2011, “Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models”, [Online]. Available:https://www.iso.org/standard/35733.html

22. ISO/IEC 20246:2017 “Software and systems engineering — Work product reviews”, [Online]. Available:https://www.iso.org/standard/67407.html

23. G. Güngör, TMMI&DO-178C Mapping, Accessed: June 2023.[Online]. Available: https://doi.org/10.5281/zenodo.8002215

24. ISTQB Glossary, [Online]. Available: https://glossary.istqb.org/

25. TMMi Assessment Method Application Requirements (TAMAR) R1.1 [Online]. Available: https://www.tmmi.org/tmmi-documents/

26. Yin, Robert K. "Case study methods." (2012)

APPENDIX

Appendix-1. TMMI Practices and DO-178C Activities Mapping

Define test goals	**LEVEL-2**	**N/A**
Define test policy	**LEVEL-2**	**4.2.b**
Distribute the test policy to stakeholders	**LEVEL-2**	**N/A**
Perform a generic product risk assessment	**LEVEL-2**	**N/A**
Define test strategy	**LEVEL-2**	**4.2.b 4.4.2.c 6.4.3 6.4.4.2.a**
Distribute the test strategy to stakeholders	**LEVEL-2**	**N/A**
Define test performance indicators	**LEVEL-2**	**4.2.c defect detection 6.4.4.2.a**
Deploy test performance indicators	**LEVEL-2**	**N/A**
Define product risk categories and parameters	**LEVEL-2**	**4.4.2.c Regression & change related test 7.2.5.b**
Identify product risks	**LEVEL-2**	**N/A**
Analyze product risks	**LEVEL-2**	**7.2.5.a**
Identify items and features to be tested	**LEVEL-2**	**N/A**
Define the test approach	**LEVEL-2**	**4.4.2.c Regression test & re-test**
Define entry criteria	**LEVEL-2**	**N/A**
Define exit criteria	**LEVEL-2**	**6.4.4.2.a**
Define suspension and resumption criteria	**LEVEL-2**	**N/A**
Establish a top-level work breakdown structure	**LEVEL-2**	**N/A**
Define test lifecycle	**LEVEL-2**	**N/A**
Determine estimates for test effort and cost	**LEVEL-2**	**N/A**
Establish the test schedule	**LEVEL-2**	**N/A**
Plan for test staffing	**LEVEL-2**	**N/A**

Plan stakeholder involvement	**LEVEL-2**	**N/A**
Identify test project risks	**LEVEL-2**	**N/A**
Establish the test plan	**LEVEL-2**	**4.2.b (ISO/IEC 29119-3] 4.4.2.c Regression test & re-test**
Review test plan	**LEVEL-2**	**N/A**
Reconcile work and resource levels	**LEVEL-2**	**N/A**
Obtain test plan commitments	**LEVEL-2**	**N/A**
Monitor test planning parameters	**LEVEL-2**	**N/A**
Monitor test environment resources provided and used	**LEVEL-2**	**N/A**
Monitor test commitments	**LEVEL-2**	**N/A**
Monitor test project risks	**LEVEL-2**	**7.2.5.**
Monitor stakeholder involvement	**LEVEL-2**	**N/A**
Conduct test progress reviews	**LEVEL-2**	**N/A**
Conduct test progress milestone reviews	**LEVEL-2**	**N/A**
Check against entry criteria	**LEVEL-2**	**N/A**
Monitor defects	**LEVEL-2**	**N/A**
Monitor product risks	**LEVEL-2**	**7.2.5.a**
Monitor exit criteria	**LEVEL-2**	**6.4.4.2.a**
Monitor suspension and resumption criteria	**LEVEL-2**	**N/A**
Conduct product quality reviews	**LEVEL-2**	**N/A**
Conduct product quality milestone reviews	**LEVEL-2**	**4.2.b (ISO/IEC 29119-3]**
Analyze issues	**LEVEL-2**	
Take corrective action	**LEVEL-2**	**4.2.e**
Manage corrective action	**LEVEL-2**	**4.2.e**

Identify and prioritize test conditions	**LEVEL-2**	**4.2.b (ISO/IEC 29119-3] 6.4.2.1.a 6.4.2.1.c 6.4.2.1.d 6.4.4.1.a 6.4.4.1.d 7.2.4.d/e**
Identify and prioritize test cases	**LEVEL-2**	**4.2.b (ISO/IEC 29119-3] 6.4.4.1.a 6.4.4.1.d 6.5**
Identify necessary specific test data	**LEVEL-2**	**N/A**
Maintain horizontal traceability with requirements	**LEVEL-2**	**6.4.4.1.a 6.4.4.1.d 6.5**
Develop and prioritize test procedures	**LEVEL-2**	**4.2.b (ISO/IEC 29119-3] 6.5**
Create specific test data	**LEVEL-2**	**N/A**
Specify intake test procedure	**LEVEL-2**	**N/A**
Develop test execution schedule	**LEVEL-2**	**N/A**
Perform intake test	**LEVEL-2**	**N/A**
Execute test cases	**LEVEL-2**	**7.2.4.d**
Report test incidents	**LEVEL-2**	**4.2.b (ISO/IEC 29119-3] 6.4.1.a 7.2.3**
Write test log	**LEVEL-2**	**4.2.b (ISO/IEC 29119-3] 7.2.3**
Decide disposition of test incidents in configuration control board	**LEVEL-2**	**7.2.3. 8.3.d**
Perform appropriate action to fix the test incident	**LEVEL-2**	**7.2.3. 8.3.d**
Track the status of test incidents	**LEVEL-2**	**7.2.3. 8.3.d**
Elicit test environment needs	**LEVEL-2**	**4.2.b (ISO/IEC 29119-3] 6.4.1.a**
Develop the test environment requirements	**LEVEL-2**	**6.4.1.a**

Analyze the test environment requirements	**LEVEL-2**	**4.4.3.a real-life test environment 4.4.3.b analyze environment req. Risks 6.4.1.a**
Implement the test environment	**LEVEL-2**	**6.4.1.a**
Create generic test data	**LEVEL-2**	**4.2.b (ISO/IEC 29119-3]**
Specify test environment intake test procedure	**LEVEL-2**	**N/A**
Perform test environment intake test	**LEVEL-2**	**4.2.b (ISO/IEC 29119-3]**
Perform systems management	**LEVEL-2**	**N/A**
Perform test data management	**LEVEL-2**	**N/A**
Coordinate the availability and usage of the test environments	**LEVEL-2**	**N/A**
Report and manage test environment incidents	**LEVEL-2**	**N/A**
Define the test organization	**LEVEL-3**	**N/A**
Obtain commitments for the test organization	**LEVEL-3**	**N/A**
Implement the test organization	**LEVEL-3**	**N/A**
Identify test functions	**LEVEL-3**	**N/A**
Develop job descriptions	**LEVEL-3**	**N/A**
Assign staff members to test functions	**LEVEL-3**	**N/A**
Establish test career paths	**LEVEL-3**	**N/A**
Develop personal test career development plans	**LEVEL-3**	**N/A**
Assess the organization's test process	**LEVEL-3**	**N/A**
Identify the organization's test process improvements	**LEVEL-3**	**N/A**
Plan test process improvements	**LEVEL-3**	**N/A**
Implement test process improvements	**LEVEL-3**	**N/A**

Deploy standard test process and test process assets	**LEVEL-3**	**N/A**
Monitor implementation	**LEVEL-3**	**N/A**
Incorporate lessons learned into the organizational test process	**LEVEL-3**	**4.2.b (ISO/IEC 29119-3]**
Identify the strategic test training needs	**LEVEL-3**	**N/A**
Align the organizational and project test training needs	**LEVEL-3**	**N/A**
Establish an organizational test training plan	**LEVEL-3**	**N/A**
Establish test training capability	**LEVEL-3**	**N/A**
Deliver test training	**LEVEL-3**	**N/A**
Establish test training records	**LEVEL-3**	**N/A**
Assess test training effectiveness	**LEVEL-3**	**N/A**
Establish standard test processes	**LEVEL-3**	**N/A**
Establish test lifecycle model descriptions addressing all test levels	**LEVEL-3**	**N/A**
Establish tailoring criteria and guidelines	**LEVEL-3**	**N/A**
Establish the organization's test process database	**LEVEL-3**	**N/A**
Establish the organization's test process asset library	**LEVEL-3**	**N/A**
Establish work environment standards	**LEVEL-3**	**N/A**
Establish integrated lifecycle models	**LEVEL-3**	**N/A**
Review integrated lifecycle models	**LEVEL-3**	**N/A**
Obtain commitments on the role of testing within the integrated lifecycle models	**LEVEL-3**	**N/A**

Perform a product risk assessment	**LEVEL-3**	**N/A**
Establish the test approach	**LEVEL-3**	**N/A**
Establish test estimates	**LEVEL-3**	**N/A**
Define the organization for testing	**LEVEL-3**	**N/A**
Develop the master test plan	**LEVEL-3**	**4.2.b (ISO/IEC 29119-3]**
Obtain commitment to the master test plan	**LEVEL-3**	**N/A**
Identify non-functional product risks	**LEVEL-3**	**N/A**
Analyze non-functional product risks	**LEVEL-3**	**4.2.b (ISO/IEC 25010)** **6.3.1-6.3.2(ISO/IEC 25010)**
Identify non-functional features to be tested	**LEVEL-3**	**N/A**
Define the non-functional test approach	**LEVEL-3**	**N/A**
Define non-functional exit criteria	**LEVEL-3**	**N/A**
Identify and prioritize non-functional test conditions	**LEVEL-3**	**4.2.b ISO 29119-3** **6.4.4.1.a** **6.4.4.1.d** **6.5**
Identify and prioritize non-functional test cases	**LEVEL-3**	**4.2.b ISO 29119-3** **6.4.4.1.a** **6.4.4.1.d** **6.5**
Identify necessary specific test data	**LEVEL-3**	**N/A**
Maintain horizontal traceability with non-functional requirements	**LEVEL-3**	**6.4.4.1.a** **6.4.4.1.d** **6.5**
Develop and prioritize non-functional test procedures	**LEVEL-3**	**4.2.b ISO 29119-3** **6.5**
Create specific test data	**LEVEL-3**	**N/A**
Execute non-functional test cases	**LEVEL-3**	**N/A**
Report non-functional test incidents	**LEVEL-3**	**4.2.b ISO 29119-3**
Write test log	**LEVEL-3**	**N/A**

Identify work products to be reviewed	**LEVEL-3**	**4.2.b [ISO 20246] 5.1.2.a**
Define peer review criteria	**LEVEL-3**	**5.1.2.a**
Conduct peer reviews	**LEVEL-3**	**5.1.2.b**
Testers review test basis documents	**LEVEL-3**	**5.1.2.a 5.1.2.b**
Analyze peer review data	**LEVEL-3**	**6.4.4.1.c**
Establish test measurement objectives	**LEVEL-4**	**N/A**
Specify test measures	**LEVEL-4**	**6.4.4.2.a**
Specify data collection and storage procedures	**LEVEL-4**	**N/A**
Specify analysis procedures	**LEVEL-4**	**N/A**
Collect test measurement data	**LEVEL-4**	**N/A**
Analyze test measurement data	**LEVEL-4**	**N/A**
Communicate results	**LEVEL-4**	**N/A**
Store data and results	**LEVEL-4**	**N/A**
Identify product quality needs	**LEVEL-4**	**N/A**
Define the project's quantitative product quality goals	**LEVEL-4**	**4.2.b [ISO/IEC 25010] 6.3.1- 6.3.2**
Define the approach for measuring progress toward the project's product quality goals	**LEVEL-4**	**N/A**
Measure product quality quantitatively throughout the lifecycle	**LEVEL-4**	**N/A**
Analyze product quality measurements and compare them to the product's quantitative goals	**LEVEL-4**	**N/A**
Relate work products to items and features to be tested	**LEVEL-4**	**N/A**
Define a coordinated test approach	**LEVEL-4**	**N/A**

Define peer review measurement guidelines	**LEVEL-4**	**N/A**
Define peer review criteria based on product quality goals	**LEVEL-4**	**N/A**
Measure work product quality using peer reviews	**LEVEL-4**	**N/A**
Analyze peer review results	**LEVEL-4**	**N/A**
Revise the products risks as appropriate	**LEVEL-4**	**N/A**
Revise the test approach as appropriate	**LEVEL-4**	**N/A**
Define defect selection parameters and defect classification scheme	**LEVEL-5**	**4.2.c defect detection**
Select defects for analysis	**LEVEL-5**	**4.2.c Pareto Analysis and Histograms**
Analyze causes of selected defects	**LEVEL-5**	**4.2.c**
Propose solutions to eliminate common causes	**LEVEL-5**	**4.2.c defect detection - Potentially appropriate methods, tools and techniques are selected as part of the solutions. Methods, tools and techniques can help the organization define coherent solutions that prevent the defects from occurring again. Methods, tools and techniques can deliver solutions that are not yet used in or known by the organization.**
Define action proposals and submit improvement proposals	**LEVEL-5**	**4.2.c**
Establish test process performance objectives	**LEVEL-5**	**N/A**
Establish test process performance measures	**LEVEL-5**	**N/A**
Establish test process performance baselines	**LEVEL-5**	**N/A**
Apply statistical methods to understand variations	**LEVEL-5**	**N/A**

Monitor performance of the selected test processes	**LEVEL-5**	**N/A**
Develop operational profiles	**LEVEL-5**	**N/A**
Generate and execute statistically selected test cases	**LEVEL-5**	**N/A**
Apply statistical test data to make stop-test decisions	**LEVEL-5**	**N/A**
Collect and analyze test process improvement proposals	**LEVEL-5**	**N/A**
Pilot test process improvement proposals	**LEVEL-5**	**N/A**
Select test process improvement proposals for deployment	**LEVEL-5**	**N/A**
Identify and analyze new testing technologies	**LEVEL-5**	**N/A**
Select new testing technologies for deployment	**LEVEL-5**	**N/A**
Plan the deployment	**LEVEL-5**	**N/A**
Manage the deployment	**LEVEL-5**	**N/A**
Measure improvement effects	**LEVEL-5**	**N/A**
Identify re-usable test assets	**LEVEL-5**	**N/A**
Select test assets to be added to the re-use library	**LEVEL-5**	**N/A**
Deploy re-usable test assets	**LEVEL-5**	**N/A**
Apply re-usable test assets in projects	**LEVEL-5**	**N/A**

Appendix-2. DO-178C Annex-A Tables [2]

Table A-1 Software Planning Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	The activities of the software life cycle processes are defined.	4.1.a	4.2.a 4.2.c 4.2.d 4.2.e 4.2.g 4.2.i 4.2.l 4.3.c	O	O	O	O	PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	①	①	①	①
										①	①	②	②
										①	①	②	②
										①	①	②	②
										①	①	②	②
2	The software life cycle(s), including the inter-relationships between the processes, their sequencing, feedback mechanisms, and transition criteria, is defined.	4.1.b	4.2i 4.3.b	O	O	O		PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	①	①	①	
										①	①	②	
										①	①	②	
										①	①	②	
										①	①	②	
3	Software life cycle environment is selected and defined.	4.1.c	4.4.1 4.4.2.a 4.4.2.b 4.4.2.c 4.4.3	O	O	O		PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	①	①	①	
										①	①	②	
										①	①	②	
										①	①	②	
										①	①	②	
4	Additional considerations are addressed.	4.1.d	4.2.f 4.2.h 4.2.i 4.2.j 4.2.k	O	O	O	O	PSAC SDP SVP SCM Plan SQA Plan	11.1 11.2 11.3 11.4 11.5	①	①	①	①
										①	①	②	②
										①	①	②	②
										①	①	②	②
										①	①	②	②
5	Software development standards are defined.	4.1.e	4.2.b 4.2.g 4.5	O	O	O		SW Requirements Standards SW Design Standards SW Code Standards	11.6 11.7 11.8	①	①	②	
										①	①	②	
										①	①	②	
6	Software plans comply with this document.	4.1.f	4.3.a 4.6	O	O	O		Software Verification Results	11.14	②	②	②	
7	Development and revision of software plans are coordinated.	4.1.g	4.2.g 4.6	O	O	O		Software Verification Results	11.14	②	②	②	

Table A-2 Software Development Processes

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	High-level requirements are developed.	5.1.1.a	5.1.2.a 5.1.2.b 5.1.2.c 5.1.2.d 5.1.2.e 5.1.2.f 5.1.2.g 5.1.2.j 5.5.a	O	O	O	O	Software Requirements Data Trace Data	11.9 11.21	① ①	① ①	① ①	① ①
2	Derived high-level requirements are defined and provided to the system processes, including the system safety assessment process.	5.1.1.b	5.1.2.h 5.1.2.i	O	O	O	O	Software Requirements Data	11.9	①	①	①	①
3	Software architecture is developed.	5.2.1.a	5.2.2.a 5.2.2.d	O	O	O	O	Design Description	11.10	①	①	①	②
4	Low-level requirements are developed.	5.2.1.a	5.2.2.a 5.2.2.e 5.2.2.f 5.2.2.g 5.2.3.a 5.2.3.b 5.2.4.a 5.2.4.b 5.2.4.c 5.5.b	O	O	O		Design Description Trace Data	11.10 11.21	① ①	① ①	① ①	
5	Derived low-level requirements are defined and provided to the system processes, including the system safety assessment process.	5.2.1.b	5.2.2.b 5.2.2.c	O	O	O		Design Description	11.10	①	①	①	
6	Source Code is developed.	5.3.1.a	5.3.2.a 5.3.2.b 5.3.2.c 5.3.2.d 5.5.c	O	O	O		Source Code Trace Data	11.11 11.21	① ①	① ①	① ①	
7	Executable Object Code and Parameter Data Item Files, if any, are produced and loaded in the target computer.	5.4.1.a	5.4.2.a 5.4.2.b 5.4.2.c 5.4.2.d 5.4.2.e 5.4.2.f	O	O	O	O	Executable Object Code Parameter Data Item File	11.12 11.22	① ①	① ①	① ①	① ①

Table A-3 Verification of Outputs of Software Requirements Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	High-level requirements comply with system requirements.	6.3.1.a	6.3.1	●	●	○	○	Software Verification Results	11.14	②	②	②	②
2	High-level requirements are accurate and consistent.	6.3.1.b	6.3.1	●	●	○	○	Software Verification Results	11.14	②	②	②	②
3	High-level requirements are compatible with target computer.	6.3.1.c	6.3.1	○	○			Software Verification Results	11.14	②	②		
4	High-level requirements are verifiable.	6.3.1.d	6.3.1	○	○	○		Software Verification Results	11.14	②	②	②	
5	High-level requirements conform to standards.	6.3.1.e	6.3.1	○	○	○		Software Verification Results	11.14	②	②	②	
6	High-level requirements are traceable to system requirements.	6.3.1.f	6.3.1	○	○	○	○	Software Verification Results	11.14	②	②	②	②
7	Algorithms are accurate.	6.3.1.g	6.3.1	●	●	○		Software Verification Results	11.14	②	②	②	

Table A-4 Verification of Outputs of Software Design Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	Low-level requirements comply with high-level requirements.	6.3.2.a	6.3.2	●	●	○		Software Verification Results	11.14	②	②	②	
2	Low-level requirements are accurate and consistent.	6.3.2.b	6.3.2	●	●	○		Software Verification Results	11.14	②	②	②	
3	Low-level requirements are compatible with target computer.	6.3.2.c	6.3.2	○	○			Software Verification Results	11.14	②	②		
4	Low-level requirements are verifiable.	6.3.2.d	6.3.2	○	○			Software Verification Results	11.14	②	②		
5	Low-level requirements conform to standards.	6.3.2.e	6.3.2	○	○	○		Software Verification Results	11.14	②	②	②	
6	Low-level requirements are traceable to high-level requirements.	6.3.2.f	6.3.2	○	○	○		Software Verification Results	11.14	②	②	②	
7	Algorithms are accurate.	6.3.2.g	6.3.2	●	●	○		Software Verification Results	11.14	②	②	②	
8	Software architecture is compatible with high-level requirements.	6.3.3.a	6.3.3	●	○	○		Software Verification Results	11.14	②	②	②	
9	Software architecture is consistent.	6.3.3.b	6.3.3	●	○	○		Software Verification Results	11.14	②	②	②	
10	Software architecture is compatible with target computer.	6.3.3.c	6.3.3	○	○			Software Verification Results	11.14	②	②		
11	Software architecture is verifiable.	6.3.3.d	6.3.3	○	○			Software Verification Results	11.14	②	②		
12	Software architecture conforms to standards.	6.3.3.e	6.3.3	○	○	○		Software Verification Results	11.14	②	②	②	
13	Software partitioning integrity is confirmed.	6.3.3.f	6.3.3	●	○	○	○	Software Verification Results	11.14	②	②	②	②

Table A-5 Verification of Outputs of Software Coding & Integration Processes

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	Source Code complies with low-level requirements.	6.3.4.a	6.3.4	●	●	○		Software Verification Results	11.14	②	②	②	
2	Source Code complies with software architecture.	6.3.4.b	6.3.4	●	○	○		Software Verification Results	11.14	②	②	②	
3	Source Code is verifiable.	6.3.4.c	6.3.4	○	○			Software Verification Results	11.14	②	②		
4	Source Code conforms to standards.	6.3.4.d	6.3.4	○	○	○		Software Verification Results	11.14	②	②	②	
5	Source Code is traceable to low-level requirements.	6.3.4.e	6.3.4	○	○	○		Software Verification Results	11.14	②	②	②	
6	Source Code is accurate and consistent.	6.3.4.f	6.3.4	●	○	○		Software Verification Results	11.14	②	②	②	
7	Output of software integration process is complete and correct.	6.3.5.a	6.3.5	○	○	○		Software Verification Results	11.14	②	②	②	
8	Parameter Data Item File is correct and complete	6.6.a	6.6	●	●	○	○	Software Verification Cases and Procedures Software Verification Results	11.13 11.14	① ②	① ②	② ②	② ②
9	Verification of Parameter Data Item File is achieved.	6.6.b	6.6	●	●	○		Software Verification Results	11.14	②	②	②	

Table A-6 Testing of Outputs of Integration Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	Executable Object Code complies with high-level requirements.	6.4.a	6.4.2 6.4.2.1 6.4.3 6.5	O	O	O	O	Software Verification Cases and Procedures Software Verification Results Trace Data	11.13 11.14 11.21	① ② ①	① ② ①	② ② ②	② ② ②
2	Executable Object Code is robust with high-level requirements.	6.4.b	6.4.2 6.4.2.2 6.4.3 6.5	O	O	O	O	Software Verification Cases and Procedures Software Verification Results Trace Data	11.13 11.14 11.21	① ② ①	① ② ①	② ② ②	② ② ②
3	Executable Object Code complies with low-level requirements.	6.4.c	6.4.2 6.4.2.1 6.4.3 6.5	●	●	O		Software Verification Cases and Procedures Software Verification Results Trace Data	11.13 11.14 11.21	① ② ①	① ② ①	② ② ②	
4	Executable Object Code is robust with low-level requirements.	6.4.d	6.4.2 6.4.2.2 6.4.3 6.5	●	O	O		Software Verification Cases and Procedures Software Verification Results Trace Data	11.13 11.14 11.21	① ② ①	① ② ①	② ② ②	
5	Executable Object Code is compatible with target computer.	6.4.e	6.4.1.a 6.4.3.a	O	O	O	O	Software Verification Cases and Procedures Software Verification Results	11.13 11.14	① ②	① ②	② ②	② ②

Table A-7 Verification of Verification Process Results

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	Test procedures are correct.	6.4.5.b	6.4.5	●	○	○		Software Verification Results	11.14	②	②	②	
2	Test results are correct and discrepancies explained.	6.4.5.c	6.4.5	●	○	○		Software Verification Results	11.14	②	②	②	
3	Test coverage of high-level requirements is achieved.	6.4.4.a	6.4.4.1	●	○	○	○	Software Verification Results	11.14	②	②	②	②
4	Test coverage of low-level requirements is achieved.	6.4.4.b	6.4.4.1	●	○	○		Software Verification Results	11.14	②	②	②	
5	Test coverage of software structure (modified condition/decision coverage) is achieved.	6.4.4.c	6.4.4.2.a 6.4.4.2.b 6.4.4.2.d 6.4.4.3	●				Software Verification Results	11.14	②			
6	Test coverage of software structure (decision coverage) is achieved.	6.4.4.c	6.4.4.2.a 6.4.4.2.b 6.4.4.2.d 6.4.4.3	●	●			Software Verification Results	11.14	②	②		
7	Test coverage of software structure (statement coverage) is achieved.	6.4.4.c	6.4.4.2.a 6.4.4.2.b 6.4.4.2.d 6.4.4.3	●	●	○		Software Verification Results	11.14	②	②	②	
8	Test coverage of software structure (data coupling and control coupling) is achieved.	6.4.4.d	6.4.4.2.c 6.4.4.2.d 6.4.4.3	●	●	○		Software Verification Results	11.14	②	②	②	
9	Verification of additional code, that cannot be traced to Source Code, is achieved.	6.4.4.c	6.4.4.2.b	●				Software Verification Results	11.14	②			

Table A-8 Software Configuration Management Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	Configuration items are identified.	7.1.a	7.2.1	○	○	○	○	SCM Records	11.18	②	②	②	②
2	Baselines and traceability are established.	7.1.b	7.2.2	○	○	○	○	Software Configuration Index SCM Records	11.16 11.18	① ②	① ②	① ②	① ②
3	Problem reporting, change control, change review, and configuration status accounting are established.	7.1.c 7.1.d 7.1.e 7.1.f	7.2.3 7.2.4 7.2.5 7.2.6	○	○	○	○	Problem Reports SCM Records	11.17 11.18	② ②	② ②	② ②	② ②
4	Archive, retrieval, and release are established.	7.1.g	7.2.7	○	○	○	○	SCM Records	11.18	②	②	②	②
5	Software load control is established.	7.1.h	7.4	○	○	○	○	SCM Records	11.18	②	②	②	②
6	Software life cycle environment control is established.	7.1.i	7.5	○	○	○	○	Software Life Cycle Environment Configuration Index SCM Records	11.15 11.18	① ②	① ②	① ②	② ②

Table A-9 Software Quality Assurance Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	Assurance is obtained that software plans and standards are developed and reviewed for compliance with this document and for consistency.	8.1.a	8.2.b 8.2.h 8.2.i	●	●	●		SQA Records	11.19	②	②	②	
2	Assurance is obtained that software life cycle processes comply with approved software plans.	8.1.b	8.2.a 8.2.c 8.2.d 8.2.f 8.2.h 8.2.i	●	●	●	●	SQA Records	11.19	②	②	②	②
3	Assurance is obtained that software life cycle processes comply with approved software standards.	8.1.b	8.2.a 8.2.c 8.2.d 8.2.f 8.2.h 8.2.i	●	●	●		SQA Records	11.19	②	②	②	
4	Assurance is obtained that transition criteria for the software life cycle processes are satisfied.	8.1.c	8.2.e 8.2.h 8.2.i	●	●	●		SQA Records	11.19	②	②	②	
5	Assurance is obtained that software conformity review is conducted.	8.1.d	8.2.g 8.2.h 8.3	●	●	●	●	SQA Records	11.19	②	②	②	②

Table A-10 Certification Liaison Process

	Objective		Activity	Applicability by Software Level				Output		Control Category by Software Level			
	Description	Ref	Ref	A	B	C	D	Data Item	Ref	A	B	C	D
1	Communication and understanding between the applicant and the certification authority is established.	9.a	9.1.b 9.1.c	O	O	O	O	Plan for Software Aspects of Certification	11.1	①	①	①	①
2	The means of compliance is proposed and agreement with the Plan for Software Aspects of Certification is obtained.	9.b	9.1.a 9.1.b 9.1.c	O	O	O	O	Plan for Software Aspects of Certification	11.1	①	①	①	①
3	Compliance substantiation is provided.	9.c	9.2.a 9.2.b 9.2.c	O	O	O	O	Software Accomplishment Summary Software Configuration Index	11.20 11.16	① ①	① ①	① ①	① ①

Appendix-3. Summary of Assesment (Achievement Rates by Practices)

PRACTICE	ACHIEVEMENT RATE F: Fully Achieved P: Partially Achieved Not Achieved N/A:Not Applicable
Identify items and features to be tested	L
Define the test approach	L
Define entry criteria	L
Define exit criteria	L
Define suspension and resumption criteria	L
Establish a top-level work breakdown structure	F
Define test lifecycle	F
Determine estimates for test effort and cost	L
Establish the test schedule	F
Plan for test staffing	F
Plan stakeholder involvement	Not achieved
Identify test project risks	Not achieved
Establish the test plan	L
Review test plan	L
Reconcile work and resource levels	L
Obtain test plan commitments	N/A
Monitor test planning parameters	Not achieved
Monitor test environment resources provided and used	Not achieved
Monitor test commitments	N/A
Monitor test project risks	Not achieved
Monitor stakeholder involvement	N/A
Conduct test progress reviews	N/A
Conduct test progress milestone reviews	N/A
Check against entry criteria	N/A
Monitor defects	Not achieved
Monitor product risks	N/A
Monitor exit criteria	F
Monitor suspension and resumption criteria	F
Conduct product quality reviews	N/A
Conduct product quality milestone reviews	N/A
Analyze issues	Not achieved
Take corrective action	Not achieved
Manage corrective action	Not achieved

Identify and prioritize test conditions	L
Identify and prioritize test cases	L
Identify necessary specific test data	N/A
Maintain horizontal traceability with requirements	F
Develop and prioritize test procedures	L
Create specific test data	F
Specify intake test procedure	F
Develop test execution schedule	F
Perform intake test	F
Execute test cases	F
Report test incidents	F
Write test log	F
Decide disposition of test incidents in configuration control board	L
Perform appropriate action to fix the test incident	F
Track the status of test incidents	L
Elicit test environment needs	P
Develop the test environment requirements	P
Analyze the test environment requirements	P
Implement the test environment	P
Create generic test data	N
Specify test environment intake test procedure	N
Perform test environment intake test	N
Perform systems management	F
Perform test data management	F
Coordinate the availability and usage of the test environments	L
Report and manage test environment incidents	L
Define the test organization	F
Obtain commitments for the test organization	F
Implement the test organization	F
Identify test functions	F
Develop job descriptions	F
Assign staff members to test functions	F
Establish test career paths	N/A
Develop personal test career development plans	N/A
Assess the organization's test process	Not achieved
Identify the organization's test process improvements	Not achieved
Plan test process improvements	Not achieved

Implement test process improvements	Not achieved
Deploy standard test process and test process assets	L
Monitor implementation	L
Incorporate lessons learned into the organizational test process	P
Identify the strategic test training needs	L
Align the organizational and project test training needs	L
Establish an organizational test training plan	Not achieved
Establish test training capability	Not achieved
Deliver test training	N/A
Establish test training records	N/A
Assess test training effectiveness	N/A
Establish standard test processes	L
Establish test lifecycle model descriptions addressing all test levels	L
Establish tailoring criteria and guidelines	N/A
Establish the organization's test process database	Not achieved
Establish the organization's test process asset library	L
Establish work environment standards	N/A
Establish integrated lifecycle models	N/A
Review integrated lifecycle models	N/A
Obtain commitments on the role of testing within the integrated lifecycle models	N/A
Perform a product risk assessment	N/A

Summary of Assessment (Achievement Rates by Process Areas)

TMMI Process Area	Achievement Result (Lowest achievement rate of a practice in the process area)
2.2 Test Planning	Largely Achieved
2.3 Test Monitoring and Control	Not achieved
2.4 Test Design and Execution	Largely Achieved
2.5 Test Environment	Partially Achieved
3.1. Test Organization	Not achieved
3.2 Test Training Program	Not achieved

Appendix-4 Questionnaire

TMMI PROCESS AREA	**GOAL**	**PRACTICE**	**REFERENCE TO RELEVANT DO-178C SECTION**	**ACHIEVEMENT RATE F: Fully Achieved P: Partially Achieved L: Largely Achieved N: Not Achieved N/A:Not Applicable**	**STRENGHTS**	**WEAKNESSES**
2.1 Test Policy and Strategy	Establish a Test Policy	Define test goals	1) Refer To Section 6.1- 2) Refer To 6.4.3.B- 3) Refer To Section 2.3			
2.1 Test Policy and Strategy	Establish a Test Policy	Define test policy	1) Refer To Section 6.4. 2) Refer To 6.2			
2.1 Test Policy and Strategy	Establish a Test Policy	Distribute the test policy to stakeholders	N/A			
2.1 Test Policy and Strategy	Establish a Test Strategy	Perform a generic product risk assessment	1) Refer To 2.3.2 2) Refer To 2.3.3			

2.1 Test Policy and Strategy	Establish a Test Strategy	Define test strategy	Refer To Test Goals & Policy Practices 1)Refer To Do-178c Section 6.4 2)Refer To Section 6.4.2- 3) Refer To Section 6.4.3.B 4) Refer To 6.4.1 5) Refer To Section 11.14			
2.1 Test Policy and Strategy	Establish a Test Strategy	Distribute the test strategy to stakeholders	N/A			
2.1 Test Policy and Strategy	Establish Test Performance Indicators	Define test performance indicators	1) Refer To Test Goals & Policy Practices			
2.1 Test Policy and Strategy	Establish Test Performance Indicators	Deploy test performance indicators	1) Refer To Section 6.4.4 & 6.4.4.1 & 6.4.4.2			

2.2 Test Planning	Perform a Product Risk Assessment	Define product risk categories and parameters	1) Refer To 2.3.2 2) Refer To 2.3.3 3) Refer To Section 7.2.5			
2.2 Test Planning	Perform a Product Risk Assessment	Identify product risks	1) Refer To 2.3.2 2) Refer To 2.3.3			
2.2 Test Planning	Perform a Product Risk Assessment	Analyze product risks	1) Refer To 2.3.2 2) Refer To 2.3.3			
2.2 Test Planning	Establish a Test Approach	Identify items and features to be tested	1) Refer To Note From Section 6.4 2) Refer To Section 6.6. 3) Refer To Section 2.5			
2.2 Test Planning	Establish a Test Approach	Define the test approach	1) Refer To 6.4.2.1 2) Refer To 6.4.2.2 3) Refer To 6.4.3 & 6.4.3.B 4) Refer To 6.4.1 5) Refer To 4.4.2.C			
2.2 Test Planning	Establish a Test Approach	Define entry criteria	N/A			

2.2 Test Planning	Establish a Test Approach	Define exit criteria	N/A			
2.2 Test Planning	Establish a Test Approach	Define suspension and resumption criteria	N/A			
2.2 Test Planning	Establish Test Estimates	Establish a top-level work breakdown structure	Refer To Section 7.2			
2.2 Test Planning	Establish Test Estimates	Define test lifecycle	N/A			
2.2 Test Planning	Establish Test Estimates	Determine estimates for test effort and cost	1)Refer To 6.4.1 2) Refer To Section 2.3.2			
2.2 Test Planning	Develop a Test Plan	Establish the test schedule	N/A			
2.2 Test Planning	Develop a Test Plan	Plan for test staffing	N/A			
2.2 Test Planning	Develop a Test Plan	Plan stakeholder involvement	N/A			
2.2 Test Planning	Develop a Test Plan	Identify test project risks	N/A			

2.2 Test Planning	Develop a Test Plan	Establish the test plan	Refer To Define Test Approach Practice + 1)Refer To Section 11.3 - 2)Refer To Activity 4.4.2.C (Reverification After Change) Also Refer To 12.1.3 Change Of Application Or Development Environment Can Require Reverification) 3) Refer To Section 11.14			
2.2 Test Planning	Obtain Commitment to the Test Plan	Review test plan	1) Refer To Section 11.3			
2.2 Test Planning	Obtain Commitment to the Test Plan	Reconcile work and resource levels	1) Refer To Section 2.5			
2.2 Test Planning	Obtain Commitment to the Test Plan	Obtain test plan commitments	N/A			

2.3 Test Monitoring and Control	Monitor Test Progress against Plan	Monitor test planning parameters	N/A			
2.3 Test Monitoring and Control	Monitor Test Progress against Plan	Monitor test environment resources provided and used	Test Environment			
2.3 Test Monitoring and Control	Monitor Test Progress against Plan	Monitor test commitments	N/A			
2.3 Test Monitoring and Control	Monitor Test Progress against Plan	Monitor test project risks	Test Environment			
2.3 Test Monitoring and Control	Monitor Test Progress against Plan	Monitor stakeholder involvement	N/A			
2.3 Test Monitoring and Control	Monitor Test Progress against Plan	Conduct test progress reviews	N/A			
2.3 Test Monitoring and Control	Monitor Test Progress against Plan	Conduct test progress milestone reviews	N/A			
2.3 Test Monitoring and Control	Monitor Product Quality against Plan and Expectations	Check against entry criteria	N/A			
2.3 Test Monitoring and Control	Monitor Product Quality against Plan and	Monitor defects	N/A			

	Expectations					
2.3 Test Monitoring and Control	Monitor Product Quality against Plan and Expectations	Monitor product risks	N/A			
2.3 Test Monitoring and Control	Monitor Product Quality against Plan and Expectations	Monitor exit criteria	N/A			
2.3 Test Monitoring and Control	Monitor Product Quality against Plan and Expectations	Monitor suspension and resumption criteria	N/A			
2.3 Test Monitoring and Control	Monitor Product Quality against Plan and Expectations	Conduct product quality reviews	N/A			
2.3 Test Monitoring and Control	Monitor Product Quality against Plan and Expectations	Conduct product quality milestone reviews	N/A			
2.3 Test Monitoring and Control	Manage Corrective Actions to Closure	Analyze issues	N/A			
2.3 Test Monitoring and Control	Manage Corrective Actions to Closure	Take corrective action	N/A			
2.3 Test Monitori	Manage Corrective	Manage corrective action	N/A			

ng and Control	Actions to Closure					
2.4 Test Design and Execution	Perform Test Analysis and Design using Test Design Techniques	Identify and prioritize test conditions	1) Refer To Section 6.3.1 2) Refer To 6.4.2 6.4.2.1 & 6.4.2.2 3) Refer To 6.4.3 4) Refer To Section 6.5 5) Refer To Activity 6.2.B			
2.4 Test Design and Execution	Perform Test Analysis and Design using Test Design Techniques	Identify and prioritize test cases	1) Refer To Section 6.4.4.1 - 6.4.4.2- 6.4.4.3 2) Refer To Section 6.4.1			
2.4 Test Design and Execution	Perform Test Analysis and Design using Test Design Techniques	Identify necessary specific test data	N/A			
2.4 Test Design and Execution	Perform Test Analysis and Design using Test Design Techniques	Maintain horizontal traceability with requirements	1) Refer To Section 6.5 2) Refer To Section 6.4.4.1			

2.4 Test Design and Execution	Perform Test Implementation	Develop and prioritize test procedures	1) Refer To Note From Section 6.4			
2.4 Test Design and Execution	Perform Test Implementation	Create specific test data	N/A			
2.4 Test Design and Execution	Perform Test Implementation	Specify intake test procedure	N/A			
2.4 Test Design and Execution	Perform Test Implementation	Develop test execution schedule	N/A			
2.4 Test Design and Execution	Perform Test Execution	Perform intake test	N/A			
2.4 Test Design and Execution	Perform Test Execution	Execute test cases	1) Refer To Activity 4.4.2.C (Reverification After Change) Also Refer To 12.1.3 2) Refer To Section 6.2.D 3) Refer To Section 6.6 4) Refer To Section 11.14			

2.4 Test Design and Execution	Perform Test Execution	Report test incidents	1) Refer To Section 7.2.3 2) Refer To Section 11.14 3.Refer To Section 11.17			
2.4 Test Design and Execution	Perform Test Execution	Write test log	N/A			
2.4 Test Design and Execution	Manage Test Incidents to Closure	Decide disposition of test incidents in configuration control board	1) Refer To Section 7.1.e 2) Refer To Section 7.2.5			
2.4 Test Design and Execution	Manage Test Incidents to Closure	Perform appropriate action to fix the test incident	1)Refer To Section 7.2.3 2) Refer To Section			
2.4 Test Design and Execution	Manage Test Incidents to Closure	Track the status of test incidents	1) Refer To Section 7.2.3			
2.5 Test Environment	Develop Test Environment Requirements	Elicit test environment needs	1) Refer To Section 4.4.3 2)Refer To Section 6.4.1 3) Refer To Section 12.3.2			

			4) Refer To Section 6.2.B			
2.5 Test Environment	Develop Test Environment Requirements	Develop the test environment requirements	1)Refer To Section 4.4.3 2)Refer To Section 6.4.1 3) Refer To Section 12.3.2			
2.5 Test Environment	Develop Test Environment Requirements	Analyze the test environment requirements	1)Refer To Section 4.4.3 2)Refer To Section 6.4.1 3) Refer To Section 12.3.2			
2.5 Test Environment	Perform Test Environment Implementation	Implement the test environment	N/A			
2.5 Test Environment	Perform Test Environment Implementation	Create generic test data	N/A			
2.5 Test Environment	Perform Test Environment Implementation	Specify test environment intake test	N/A			

		procedure				
2.5 Test Environment	Perform Test Environment Implementation	Perform test environment intake test	N/A			
2.5 Test Environment	Manage and Control Test Environments	Perform systems management	N/A			
2.5 Test Environment	Manage and Control Test Environments	Perform test data management	N/A			
2.5 Test Environment	Manage and Control Test Environments	Coordinate the availability and usage of the test environments	N/A			
2.5 Test Environment	Manage and Control Test Environments	Report and manage test environment incidents	N/A			