WHITE PAPER

5 Process Pitfalls to Avoid in ISO 26262 Compliance

Tips for Practitioners of Model-Based Design

MathWorks®

Introduction

ISO 26262 is rapidly becoming the de facto standard for developing safety-related automotive systems. Today, almost every engineer working on or around a safety-related feature is adopting ISO 26262.

Development organizations can accelerate ISO 26262 compliance using this guidance and adapting it to their individualized process and goals. They should consider:

- The recommended functional safety activities in the ISO 26262 standard
- The application's automotive safety integrity level (ASIL)
- Tools to be used during the development process

This white paper describes common pitfalls that can derail an organization's adherence to ISO 26262 requirements and can lead to failure of its ISO 26262 assessment. The paper also provides recommendations on how to avoid these pitfalls.

ISO 26262 Process Pitfalls

The ISO 26262 standard has 12 sections that range from functional safety requirements management to attributes that are required in a software development process. Most of the requirements related to software development are in Part 6 of the standard: ISO 26262-6:2018. Organizations that plan to use Model-Based Design in their development process must determine where and how they will use Model-Based Design. To facilitate this work, MathWorks IEC Certification Kit gives guidance on how to achieve a complete ISO 26262-6 compliance workflow using MATLAB® and Simulink® products.

IEC Certification Kit should be used as a starting point for the process migration journey. The kit provides an overview of how to achieve ISO 26262-6 compliance with Model-Based Design along with documentation templates and reference workflows.

By understanding the content, organizations can then perform a gap analysis between their existing process and one that's compliant with ISO 26262-6 using the following steps:

1. Determine the ASIL requirement of the electrical systems and components being developed.
2. Decide which process elements of the ISO 26262-6 standard will be followed.
3. Map these process elements to the organization's existing development process.
4. Analyze and understand the gaps and areas of improvement.
5. Define a process improvement strategy and implementation plan.

A gap analysis is a critical step for any process transformation activity. Unfortunately, many organizations underestimate the rigor required when starting an ISO 26262 project. With an "update as you go" mentality, these organizations typically face project delays or discover they missed steps during product certification.

MathWorks Consulting Services has conducted countless Model-Based Design ISO 26262 gap analyses. The goal of the service is to provide an objective view of a development

organization's ISO 26262-6 compliance process and tool usage methods down to the implementation level. The service not only flags possible process compliance issues but also provides tool usage recommendations. These recommendations often go beyond processes and include best practices for modeling and tool application.

The five common pitfalls MathWorks has observed while working with organizations are:

1. Lack of a software architecture strategy for design and implementation
2. No clearly defined and mapped ISO-compliant process
3. No environment for automation infrastructure
4. No artifact archiving strategy
5. Not leveraging supplier-provided tool qualification kits

1. Lack of Software Architecture Design and Implementation Strategy

It is important for the functional safety and architecture teams to have a strategy to meet the product's functional safety goals. ISO 26262 allows for software to be developed in two main ways:

1. All software units and components are developed at the highest ASIL rating required for the system.
2. Development processes are separate for different ASIL and the quality management (QM) ratings.

Some development organizations choose option 1 for the simple reason of having a single process that can be used for all software units. While this might seem like a good choice at first, in practice the amount of work involved greatly overshadows the simplicity in such architectural designs. As a result, many functional safety and architecture teams opt for option 2. Option 2 allows the organization to have a few highly rated units that may be classified as ASIL A–D using safety analysis methodologies such as dependent failure, failure mode and effects analysis (FMEA), and treating the remainder of the software as QM units.

Once the analysis is done, the electrical system and software need to be architected in a way such that different levels of ASIL and QM components are properly partitioned. This concept of partitioning is referred to as freedom from interference and is described in detail in ISO 26262-6:2018 Annex E. Freedom from interference is needed to ensure that one ASIL does not interact with another ASIL to degrade or harm the other ASIL functionalities. There are three main concepts discussed in Annex E that should be considered:

- Timing and execution
- Memory
- Exchange of Information

Timing and execution of a real-time OS-based embedded system needs to be closely analyzed. Memory and exchange of information should be carefully thought through and managed during the design and development of the software units. It is our experience that development organizations often can create a conceptual design but fail in translating the design to the correct implementation inside Simulink. There are modeling constructs and

model configurations, such as model references, memory sections, and custom methods to exchange information, that are necessary to meet the partitioning intent. MathWorks has documented these recommended modeling constructs and modeling configurations in *11 Best Practices for Developing ISO 26262 Applications with Simulink*.

Topics discussed in *11 Best Practices for Developing ISO 26262 Applications with Simulink*.

The best practices presented in the paper can be used and customized as needed based on different application needs, including mixed-criticality applications. Without having such guidance in place, many organizations end up in a state where significant modifications are needed to achieve the concept of freedom from interference. By managing this before development begins, the amount of rework and verification activities will be greatly reduced.

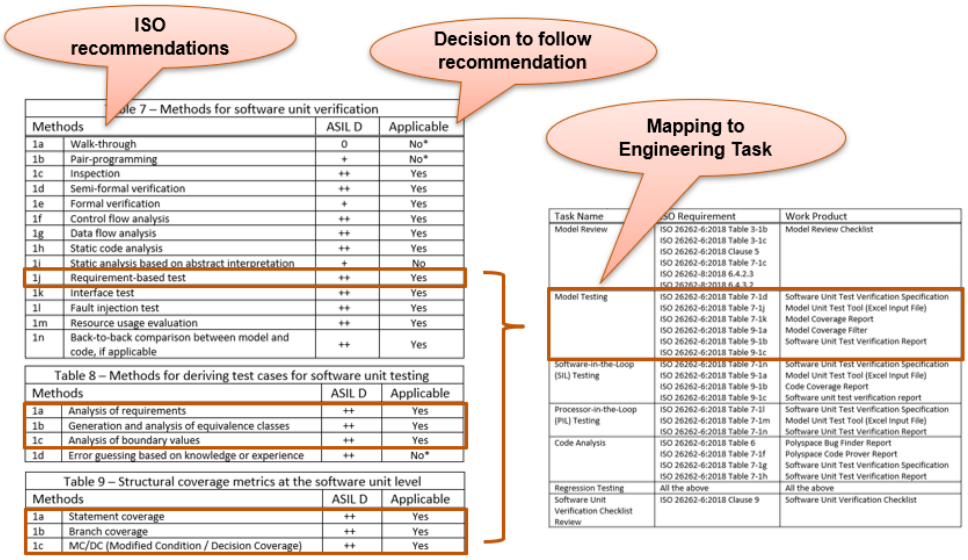
2. No Clearly Defined and Mapped ISO-Compliant Process

In the automotive industry, most tier 1 suppliers and OEMs have experienced engineers who can develop high-quality software. However, when asked, many of these development organizations do not have proper documentation to explain their entire design, development process, or both. The organizations instead rely on their engineers to “do the right thing.” If these development organizations were to be audited for ISO compliance, they would likely fail. This is because engineers typically focus on the activities and less so on the evidence of their design decision and outcome. Completeness and due diligence are now up to the engineers based on their experiences to determine the priority and importance. To avoid this issue, development organizations must now define their ISO compliance process up front and ensure all defined activities are executed with evidence proving the results match those of the activity objectives.

Another common issue MathWorks has observed is that development organizations tend to document their existing processes based only on a high-level understanding of ISO

requirements. The realization usually occurs as the product matures and in the final preparation phase of an assessment. This approach usually drives rework, redesign, or even delay of production programs. For applications that target the functional safety standard, it is important to map the development organization's process to the actual standard at the start of the program. For example, ISO 26262-6:2018 has approximately 90 different principles and methods with varying degrees of recommendations for embedded software development. The development organization should first decide which activities in the ISO standard are going to be executed, map those activities to internal processes, and specify which artifacts should be archived as proof from each activity.

The following example document provides an example of various tables in ISO 26262 Section 6 and how they could be mapped to the development organization's internal process.



Example ISO 26262 Section 6 mapping to development organization's process.

Mapping documents such as this can be part of the gap analysis service to show missing activities and capabilities. MathWorks Consulting Services can work with you on an implementation plan to use as the production program continues. Such a step is important because when the development organization goes through an audit, this document will provide a way to demonstrate which ISO 26262 activities are being carried out and can also be used as part of the safety plan evidence. This document provides information on why some activities were skipped and where the corresponding artifacts will reside in the configuration management system.

Therefore, it is crucial for development organizations to plan three main steps for all of the engineering activities:

- **Define:** Create process documents that describe which activities need to occur at each stage of the development cycle.
- **Execute:** Consistently execute these activities. Checklists and design reviews should be in place to ensure compliance.

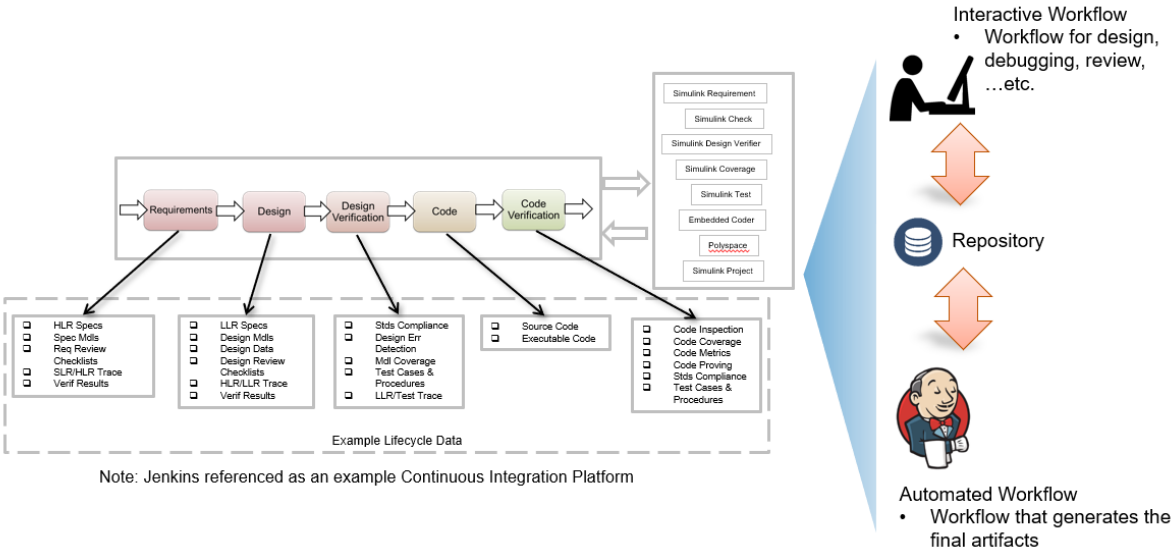
- **Archive:** Identify a list of artifacts (work products) that need to be archived at each stage of the development cycle.

An ISO-compliant process should be defined as early in the product development cycle as possible. These tasks can drive design decisions and implementation methodologies. Defining the process early on avoids unnecessary rework during assessment or prior to the start of production.

3. Lack of Automation Infrastructure and Environment

Common feedback from development organizations that are executing their first ISO-compliant project is the planning and execution rigor required is greatly above their expectations. Not only are the development organizations new to some of the activities, but the added processes around task execution and artifact management can add constraints to the organizations' bandwidth. As a result, automation should be considered part of the overall ISO strategy. Development processes should be optimized such that most of the developer's time is spent creating the algorithm and the test cases used to verify the algorithm against the requirement. Everything else beyond that should be automated. Automation is a common industry practice to improve ISO compliance.

An enabler or a prerequisite to automation is to have a defined process with inputs, activities, and outputs clearly defined and documented as mentioned earlier. It is important to note that automation should be done in two different modes. First, to support the user in understanding what needs to be done, an interactive workflow where a desktop automation tool is also considered should be included. An easy way to achieve this is to have a set of APIs or GUIs that automatically configure the work products based on project needs. These can include model/code guidelines, model or code generation configurations, or report templates. Second, the same set of activities supported through desktop automation should also be available through a continuous integration (CI) type of workflow.

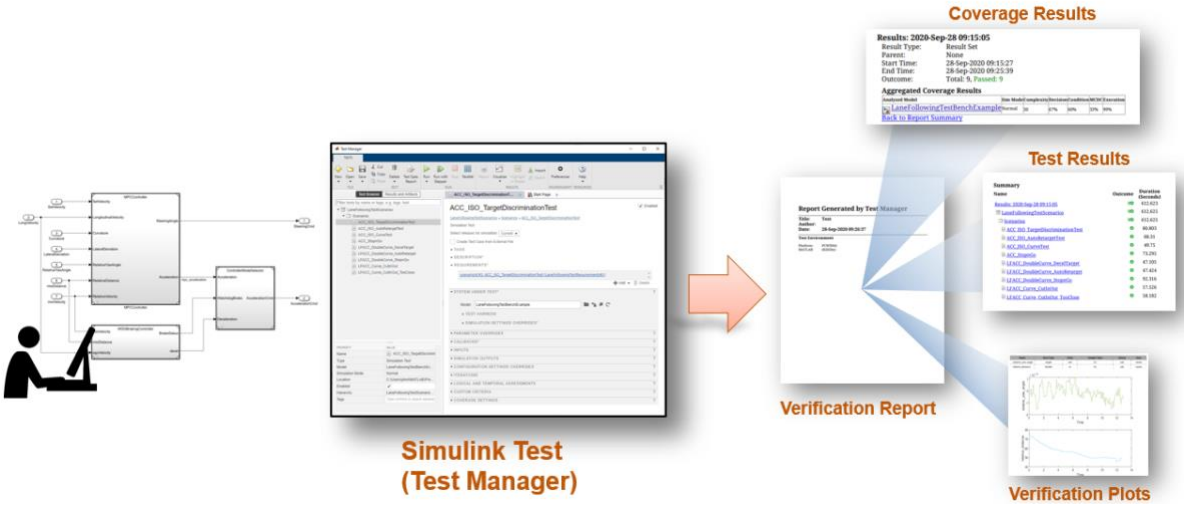


Automation support for both interactive and automated workflows.

Desktop automation support allows the users to interact with the model in the native Simulink environment while simplifying the necessary configuration through a “single-button clicks” level of automation. That way, developers are focusing on creating the algorithm and review/update the algorithm based on design guidelines. The interactive workflow adds a layer of security, ensuring the final artifacts are generated correctly and objectively verified against defined guidelines and metrics.

4. No Artifact Archiving Strategy

When organizations adopt an ISO 26262–compliant workflow it is crucial to decide which artifacts should be stored in the configuration management system. Even if some of the process is automated in a CI server or results are being checked manually, proof of these activities should be stored in a configuration management system to ensure that the activities were completed. To accomplish this, each checklist and/or process document should detail which artifacts should be saved.



Example Simulink Test workflow with artifacts.

The above screenshot demonstrates an example of a verification report that can be generated and saved in a configuration management system for proof-of-test-case execution. This artifact can be reviewed later during an ISO audit, as part of a release checklist, or if confirmation is needed that a given release was tested. A baselining strategy should be in place that details which artifacts were used and when they need to be reproducible. Not all design increments or verification reports are needed.

Through our Consulting Services engagements, we often see customers having difficulties capturing all the necessary artifacts. However, this information can be known only once there is a good understanding of ISO 26262 Part 6, the Model-Based Design workflow, and the corresponding MathWorks toolchain outputs. This is where the MathWorks IEC Certification Kit serves as a valuable resource. The kit provides a mapping of applicable tools against the tables of topics, principles, and methods. Also contained in the kit is a reference workflow that

will help developers visualize a high-level ISO 26262-6 software development workflow. Examples of the mapping table and reference workflow are shown below.

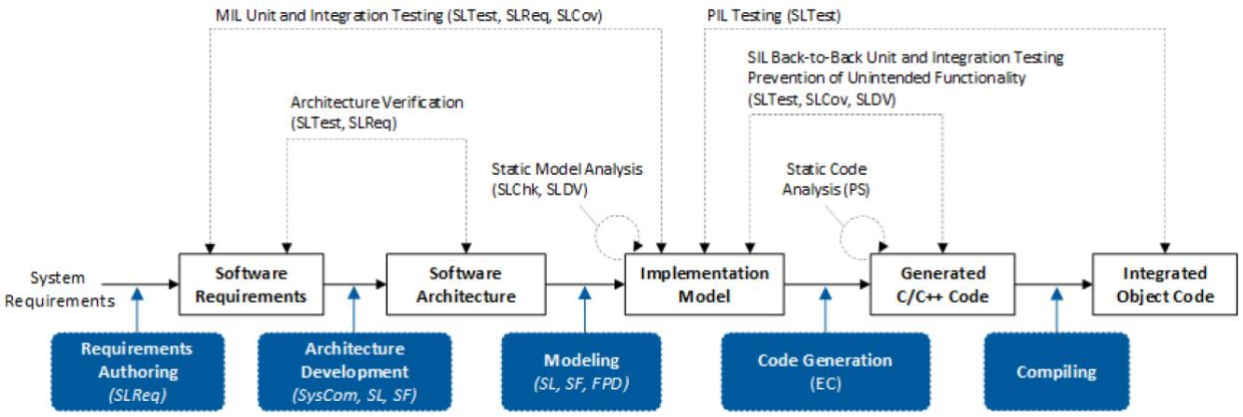
Table 9 — Structural Coverage Metrics at the Software Unit Level

Methods		ASIL A	ASIL B	ASIL C	ASIL D	Applicable Model-Based Design Tools and Processes	Comments
1a	Statement coverage	++	++	+	+	Simulink Coverage – Model coverage analysis Simulink Coverage – Code coverage analysis	During model testing, Simulink Coverage can collect execution coverage at the model level. During SIL and PIL execution, Simulink Coverage can measure the statement coverage of the generated code.
1b	Branch coverage	+	++	++	++	Simulink Coverage – Model coverage analysis Simulink Coverage – Code coverage analysis Simulink Desi… – Test case ge…	During model testing, Simulink Coverage can collect decision coverage (also known as branch coverage) at the model level. During SIL and PIL execution, Simulink Coverage can measure the decision coverage of Simulink Design … can generate test cases … the model level.

ISO 26262 Recommended Activities

IEC Certification Kit Recommendations

Example of tool mapping for ISO 26262-6:2018 Table 9.

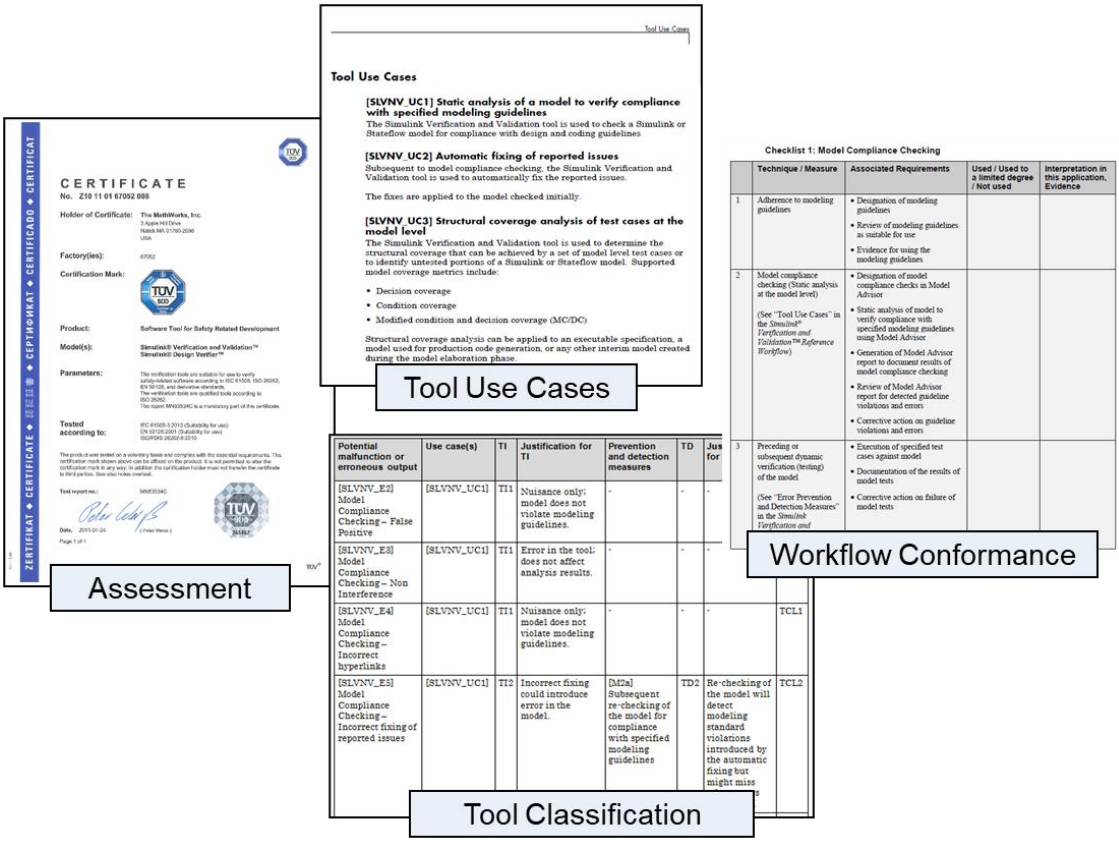


IEC Certification Kit reference workflow.

5. Not Leveraging Supplier-Provided Tool Qualification Kit

ISO 26262-8:2018 discusses the supporting processes that are used during the development of an ISO 26262–compliant application. This section also discusses a key concept that is often unfamiliar to the automotive industry: need for tool qualification. Without a tool qualification process, it is possible for automated tools to introduce errors into the final product. If a tool is

used in the development of an ASIL-rated application, then the tool will need to be classified based on its tool confidence level (TCL), followed by necessary steps proving that it is safe to use. The work is tedious but can be mitigated because many tool vendors do provide tool certification kits. This is also true for MathWorks.



IEC Certification Kit sample prequalification artifacts.

The above screenshot details the MathWorks IEC Certification Kit documentation. The kit provides the analysis done to classify each tool based on its use cases, potential malfunctions, and error detection and prevention methods. For tools that are classified as other than TCL 1, additional artifacts such as independent assessment of the tool development process and validation of the software tool through test cases are also available.

Through MathWorks ISO gap analysis, we found that many customers often overlook the availability of such certification kits and end up repeating much of their work. When reviewed, much of the tool qualification work had missed certain artifacts or had incorrect analysis, incorrect justifications, or both. All of these factors can lead to unnecessary rework during assessment time.

Summary

We have shared a set of common pitfalls that we observed through recent ISO 26262 projects. The hope is for this information to help other automotive engineering teams that need to achieve ISO 26262 compliance. We believe the key to avoiding these pitfalls lies with planning, process capture, software architecture, reporting/archiving, automation, and tool qualification.

Contact

We are interested in your observations and feedback.

John Lee, johnl@mathworks.com

Jason Moore, jasonm@mathworks.com

Learn More

ISO 26262 Process Deployment Advisory Service – Consulting Service Overview

ISO 26262 Support in MATLAB and Simulink - Resources