SIS 202 - Functional Design

15 minutes

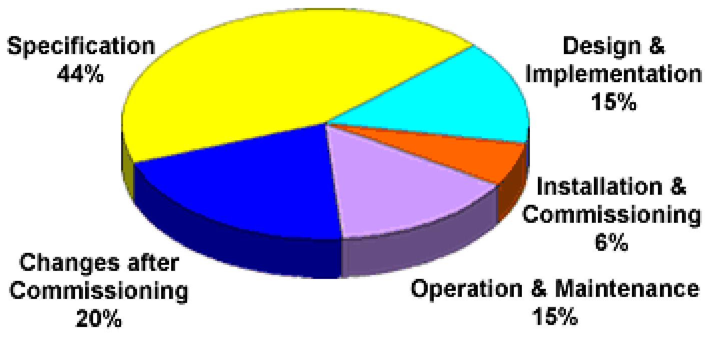
In this course:

Overview

In the previous course you learned about the physical (hardware) aspects of an SIS. Now we'll turn our attention to the functional (software) aspects.

You may think of software as something that's easy to fix if there's a problem. But in safety systems, getting it right is essential. Approximately half of accidents caused by control and safety-system failures originate from mistakes or poor decisions made before the systems ever left the drawing board. So good engineering at this point makes a big difference in how well — or even whether — your SIS will do its job.

Root causes of failures in control and safety systems



Source: Health & Safety Executive

This course outlines key aspects of ensuring your software enables the safety system to carry out its functions.

Hint

As you go through the topics in this course, watch for answers to these questions:

- What are the three types of safety system software?
- What are the two sides of the software-development "V-model"?
- What are the benefits of using SIS certified software modules?

Software Types

The IEC 61511 standard identifies three types of software:

1. **Application**: The software you develop specifically for your SIS solution — in other words, the system configuration.
2. **Utility**: The software tools you use to develop, verify, and maintain the application software.
3. **Embedded**: The software (also called firmware) that is "built in" to SIS products.

Utility and embedded software is usually provided by instrument and control system manufacturers as part of their products. When these products are certified for SIS applications, the suppliers typically take primary responsibility for ensuring that this software complies with IEC 61508 standards.

Responsibility for ensuring that the application software meets requirements, on the other hand, is all yours — although consultants and integrators can help.

Development Life Cycle

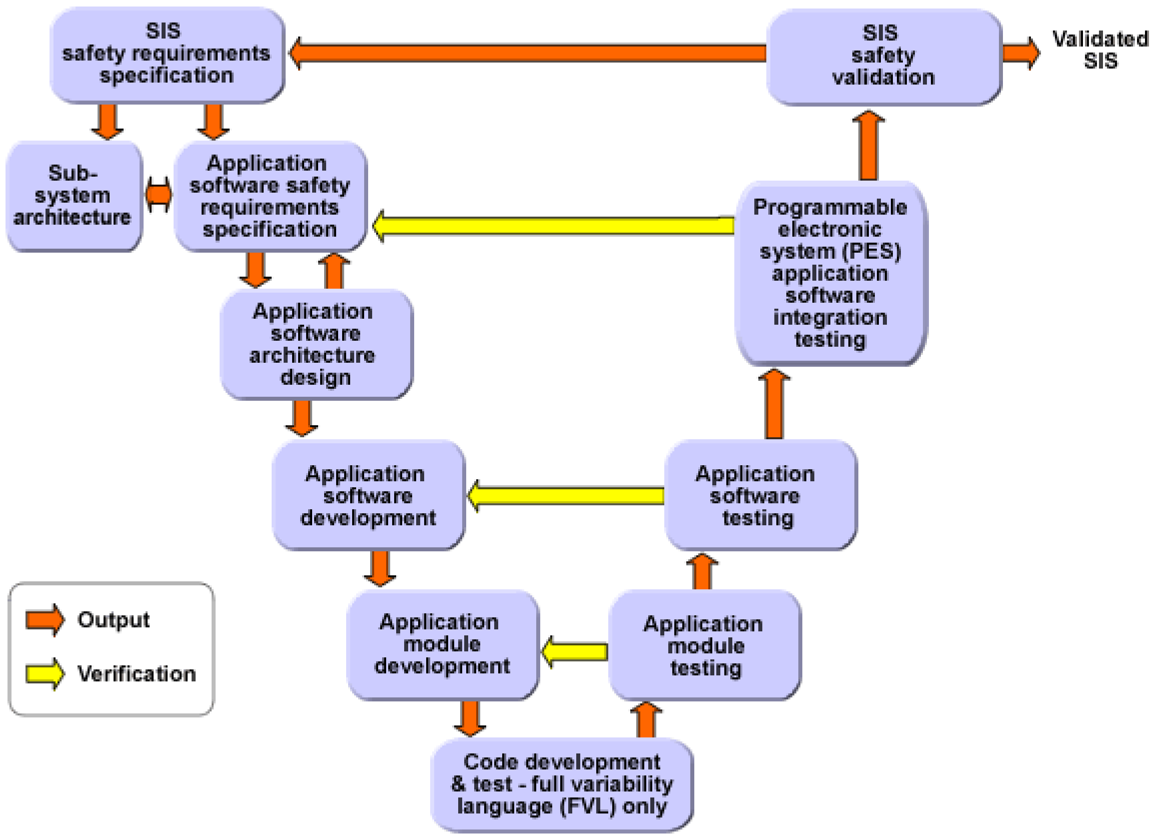
Computer giant IBM uses the "1-10-100 rule" to explain the importance of good software design: For every software error that can be removed for $1 during the design phase, waiting to remove the error during testing will cost $10, and waiting until the software has been delivered will cost $100.

In the safety world, the costs may be much higher — and measured in terms of life and limb. That's why it's important to ensure software quality right from the start, and keep verifying it throughout the development process.

IEC 61511 allows some flexibility in how you develop application software for your SIS. However, it does require that the development process be carefully structured to avoid engineering errors that result in dangerous failures during operation. It also requires verifying and validating that the software application solution performs as defined in the design documentation.

The standard includes the popular software development "V-Model" to illustrate the activities necessary to ensure this happens. The left side of the V shows software development activities, and the right side shows corresponding verification and validation activities.

IEC 61511-1 software development life cycle (V-model)


SIS safety requirements specification
SIS safety validation
Validated SIS
Sub-system architecture
Application software safety requirements specification
Programmable electronic system (PES) application software integration testing
Application software architecture design
Application software development
Application software testing
Application module development
Application module testing
Code development & test - full variability language (FVL) only
Output
Verification

The process begins with the safety requirements specification (SRS) and progresses through increasingly detailed design and development stages. Then a series of increasingly broad-based tests verifies that the work done at each stage has met safety requirements. At the end of the process, successful integration testing leads to validated software.

This process — including test planning and documentation — is covered in more detail in the next course, **SIS 203 - SIS Verification and Validation**.

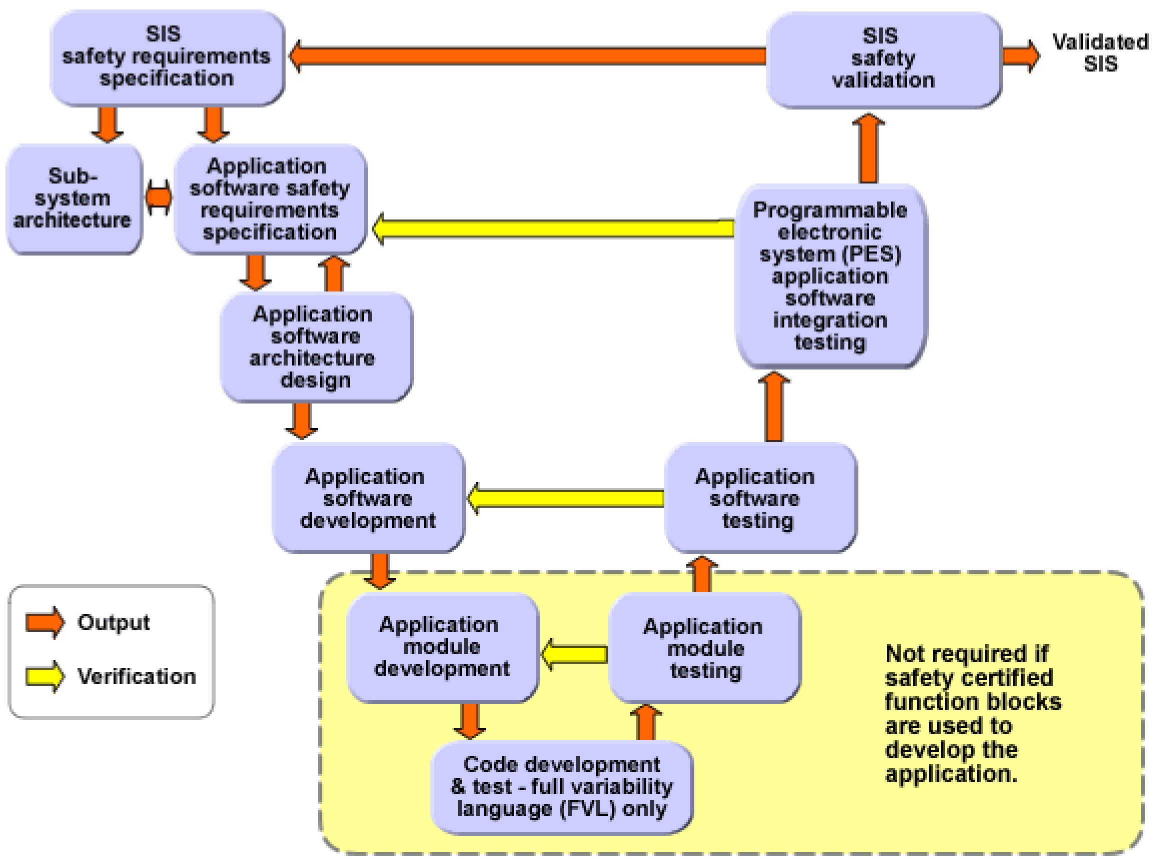
Certified Software Modules

Modern software design typically uses modular code that is developed once but used repeatedly — avoiding the time, cost, and errors that can result from "reinventing the wheel."

The IEC standard gives you two options: creating and validating your own library of application software modules, or using pre-developed, pre-tested, third-party-certified modules.

When you use certified modules provided by SIS product manufacturers, you not only avoid the time and cost associated with developing that code. You also avoid the requirement to test each module. The supplier and third-party certifying organization have already done the work for you.

IEC 61511-1 software development life cycle (V-model)


SIS safety requirements specification
SIS safety validation
Validated SIS
Sub-system architecture
Application software safety requirements specification
Programmable electronic system (PES) application software integration testing
Application software architecture design
Application software development
Application software testing
Output
Verification
Application module development
Application module testing
Code development & test - full variability language (FVL) only
Not required if safety certified function blocks are used to develop the application.

If you choose to create your own application development modules, you should understand that the IEC standard lays out very strict requirements for designing, developing, and testing these re-usable software modules.

The bottom line is that application software modules must be "bulletproof," and whoever develops them assumes the responsibility and risk of ensuring that happens.

The PlantWeb Advantage

The DeltaV SIS system that's part of Emerson's smart SIS includes a full palette of TÜV-certified function blocks including Voter, Cause and Effect Matrix, Step Sequencer, and State Transition Table.

Powerful, smart function blocks, such as extensible MooN (M out of N) Voter blocks with built-in bypass functionality reduce what once required developing pages and pages of ladder logic to a simple drag-and-drop configuration activity.

Other DeltaV SIS software capabilities, such as an alarm state engine built to the EEMUA 191 standard, off-line simulation, sequence of events recorder, bypass handling, and override bundling, make SIS maintenance easy and less complex.

All these built-in capabilities help automate IEC 61511 compliance, thus simplifying documentation requirements and reducing your life-cycle costs and risks.

Software Utility Tools

Like any other craft, software development has tools to speed and simplify the work.

IEC 61511 groups things such as application programming languages, configuration management tools, simulations, test harnesses, and automatic test coverage measurement under the heading of **software utility tools**. The standard permits you considerable flexibility in selecting these tools — including using or developing your own.

However, before any tool (purchased or developed) can be used to help achieve IEC compliance, the tool's user manual must fully document

- How to use the tool
- Usage constraints
- Known weaknesses
- Release limitations

…and similar topics.

Because of the importance of application development tools, the standard requires you use a **proven software language translator/compiler** that can detect programming and syntax errors — and doesn't introduce errors of its own.

The PlantWeb Advantage

A key software utility tool in Emerson's smart SIS is AMS™ Suite: Intelligent Device Manager, which enables you to verify proper setup and installation of field instruments. Its QuickCheck capability also simplifies interlock validation by enabling you to put several instruments in fixed checkout mode at the same time. And during SIS modification it can be used to compare new device configurations with previous ones.

For a third-party study of these and other capabilities, see "AMS Safety Analysis: Using AMS Suite in Safety Instrumented System Applications."

Summary

In this course you've learned that:

- Good software engineering is essential to avoid safety problems that are "designed into" the SIS.
- IEC 61511 identifies three types of software: application software, utility software, and embedded software (also called firmware).
- While suppliers usually have primary responsibility for utility and embedded software, it's up to you to ensure that application-specific software complies with the standard.
- For each phase of software design and development, there's a corresponding testing phase to verify that the software meets requirements.
- Using pre-certified software modules can reduce development and testing time and cost.