Ansys

part of SYNOPSYS®

EN 50716: The Strategic Push Towards Certified Model-Based Engineering for the Development and Testing of SIL4 Embedded Software Applications

RAILLIVE!

IFEMA Madrid, Spain - 26-28 November 2025

Bruno Privat - System & Mission Senior MBSE Product Executive - *bruno.privat@ansys.com*

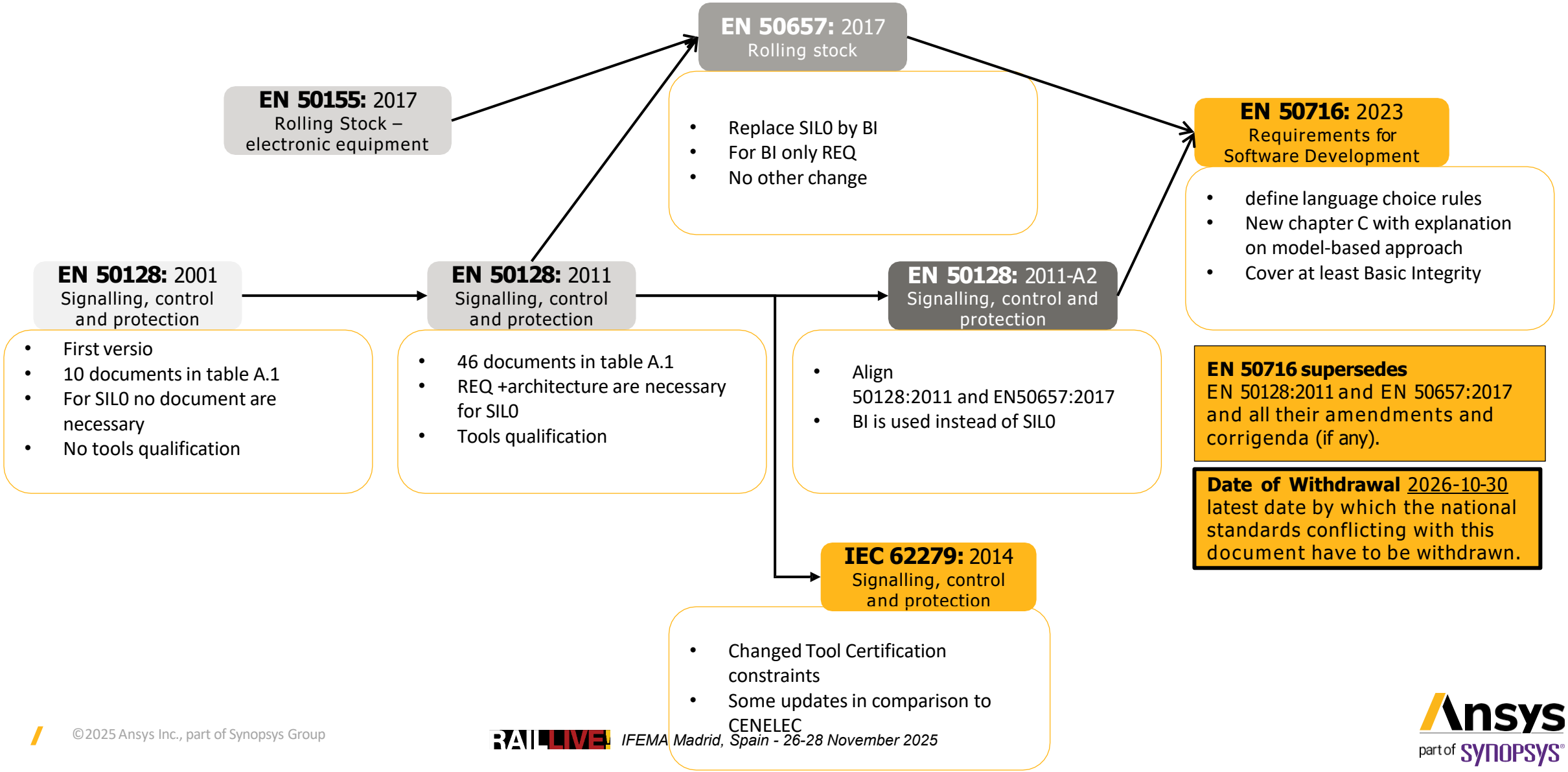
Outline

- Overall Railway Standard's Evolution
- Annex C1: Lifecycle Model Examples
- Annex C2: Modelling
- Updates in Part 5: Software Management and organization
- Updates in Part 6: Support tools and Languages
- How Ansys SCADE fully map new requirements for Safety Critical Railway Applications

©2025 Ansys Inc., part of Synopsys Group

Ansys part of SYNOPSYS®

Railway Standard's Evolution



©2025 Ansys Inc., part of Synopsys Group

Safety Integrity levels and Cost

Standard	No impact on Safety	Impact on the system		Impact on system & human		Impact on human
CELELEC 50126/50129	BI	SIL 1	SIL 2	SIL 3		SIL 4
CELELEC 50128	BI	SSIL 1	SSIL 2	SSIL 3		SSIL 4
CEI / IEC 50128	--	SIL 1	SIL 2	SIL 3		SIL 4
Automotive ISO 26262	QM	ASIL A	ASIL B	ASIL C	ASIL D	---
Aerospace DO 718C	DAL E	DAL D	DAL C	DAL B		DAL A

Level	Non-Safety Related Cost	SIL 1 Cost	SIL 2 Cost	SIL 3 Cost	SIL 4 Cost
Comparative Cost	Baseline	Base +10%	SIL 1 +36%	SIL 2 +80%	SIL 3 +30%
Cost	100	110	150	270	350+

ISO 9001

- Specification
- Overall Test
- Conf Management
- Anomalies management

Basic Integrity

- Specification
- Overall Test
- Conf Management
- Anomalies management
- Prog rules +metrics

SIL 2

- Specification
- Overall Test
- Conf Management
- Anomalies management
- Prog rules +metrics
- Architecture +Int Test
- CD +CT +coverage

SIL 4

- Specification
- Overall Test
- Conf Management
- Anomalies management
- **Prog rules +metrics**
- **Architecture +Int Test**
- **CD +CT +coverage**

SIL 0

- Specification
- Overall Test
- Conf Management
- Anomalies management
- Prog rules +metrics
- **Architecture +Int Test**

©2025 Ansys Inc., part of Synopsys Group

Ansys part of SYNOPSYS®

Typical Railway Applications / Criticality Level

Low Criticality: SIL 1&2

- Passenger Information System
- Train Monitoring System (basic fault logging)
- Basic Communication System
- Trackside Monitoring System
- Automatic Train Identification (**ATI**)
- Non-critical Diagnostic System
- Basic Energy Management System

High Criticality: SIL3 & SIL4

- European Train Control System (**ETCS**)
- Automatic Train Control (**ATC**)
- Automatic Train Protection (**ATP**)
- Automatic Train Operation (**ATO**)
- Signaling and Interlocking
- Train Control and Management Systems (**TCMS**)
- Communication-Based Train Control (**CBTC**)
- Braking System
- Driver Machine Interface (**DMI**)

Ansys, part of Synopsys ©2025

EN 50128 → **EN 50716** Evolution

Simplifies Roles and Responsibilities

Provides guidance for the use of Model Based Languages

Provides guidance for Iterative lifecycle

Makes emphasis on Formal Methods

Examines Artificial Intelligence

Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

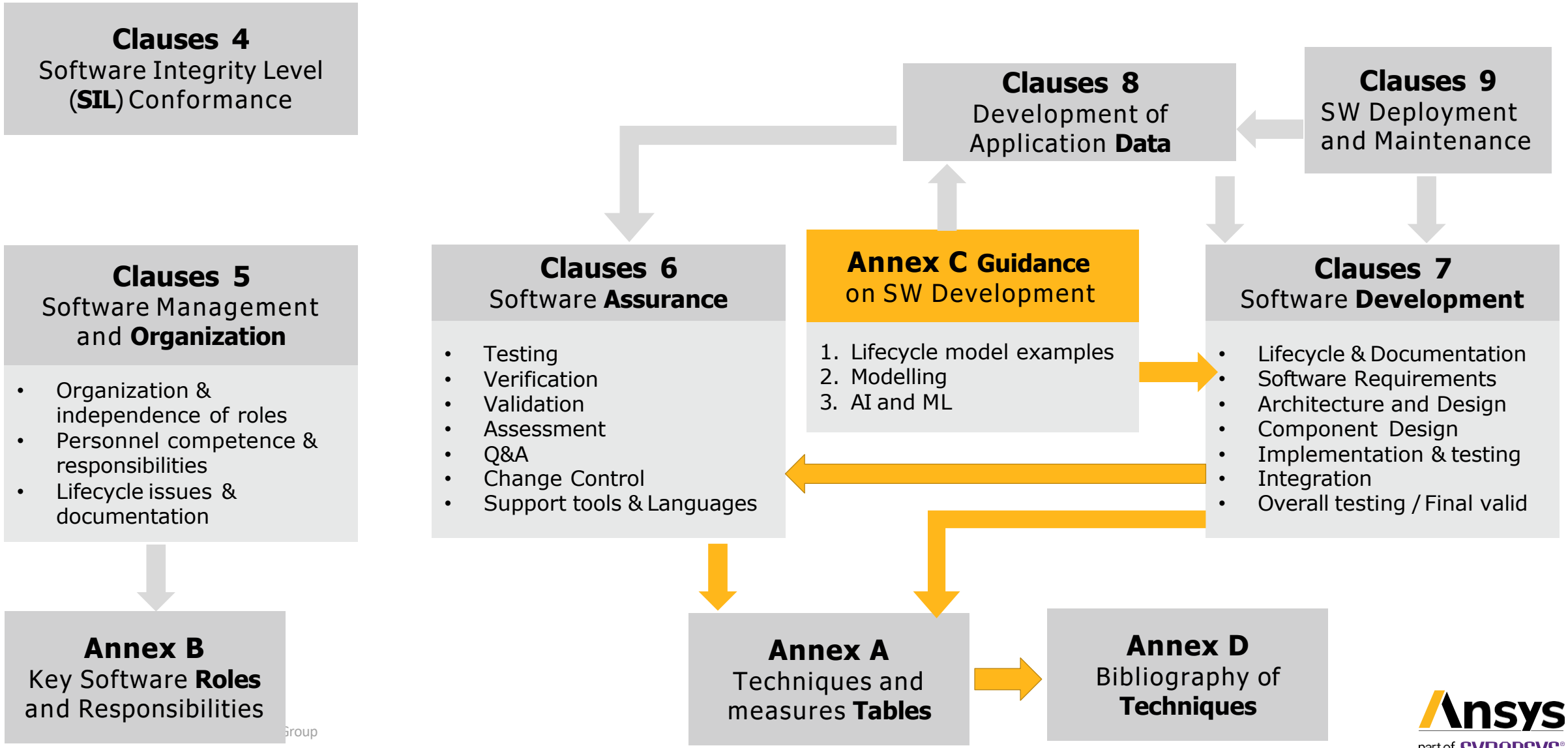
EN 50716 Basic Design Principles

EN 50716:2023 Introduction

Modelling	Top-down design	Components	Verification at each phase
Formal Methods	Traceability	Auditable Documents	Configuration Management and Change Control
	Organization and Personnel Competency	Assessment	

Ansys, part of Synopsys ©2025

EN 50716 Clauses and Annexes

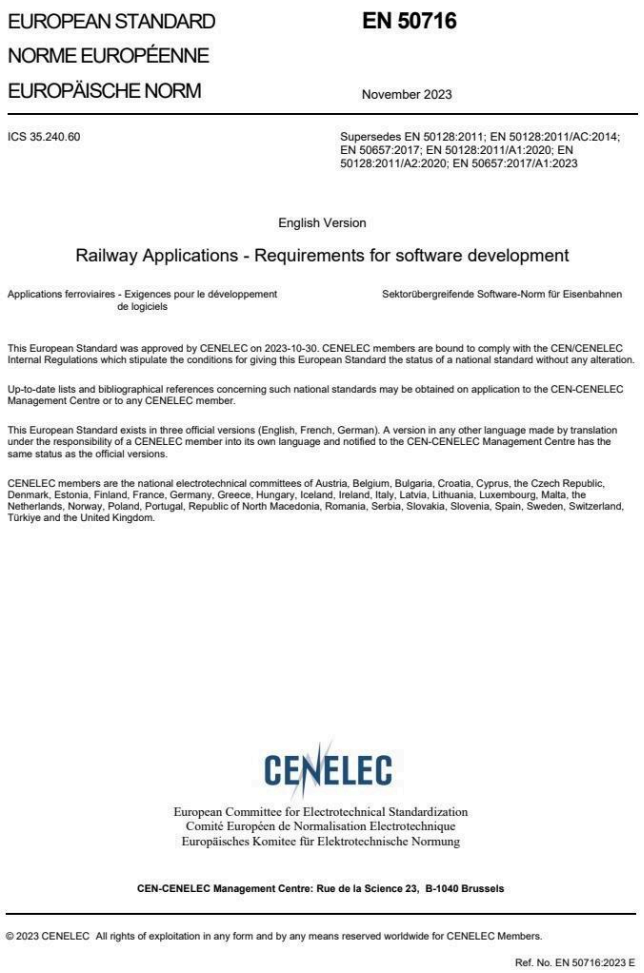


Ansys part of SYNOPSYS®

EN 50716 Main Technical Changes

wrt EN 50128:2011 and EN 50657:2017

SIST EN 50716:2024



EUROPEAN STANDARD **EN 50716**

NORME EUROPÉENNE

EUROPÄISCHE NORM November 2023

ICS 35.240.60

Supersedes EN 50128:2011; EN 50128:2011/AC:2014; EN 50657:2017; EN 50128:2011/A1:2020; EN 50128:2011/A2:2020; EN 50657:2017/A1:2023

English Version

Railway Applications - Requirements for software development

Applications ferroviaires - Exigences pour le développement de logiciels

Sektorübergreifende Software-Norm für Eisenbahnen

This European Standard was approved by CENELEC on 2023-10-30. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Republic of North Macedonia, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Türkiye and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

© 2023 CENELEC All rights of exploitation in any form and by any means reserved worldwide for CENELEC Members.

Ref. No. EN 50716:2023 E

- Better aligned with EN 50126-1:2017 and EN 50126-2:2017, including definitions
- Clause 5 **rewritten for simpler readability; organizational options unchanged**
- Annex A updated for improved lifecycle phase alignment
- Annex C.1 **added with guidance on lifecycle models** Annex C.2

added with guidance on software modelling More guidance

for software components of different SIL **Programming**

language requirements generalized

Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

Software Integrity Levels (Clauses 4)

"Classification which determines the techniques and measures that have to be applied to software"
EN 50716:2023 3.1.32

From EN 50126-2017 Table 2

Safety Integrity Level **(SIL)**	Train Failure and Fault Report $[h^{-1}]$ **(TFFR)**
4	$\geq 10^{-9}\ to\ < 10^{-8}$
3	$\geq 10^{-8}\ to\ < 10^{-7}$
2	$\geq 10^{-7}\ to\ < 10^{-6}$
1	$\geq 10^{-6}\ to\ < 10^{-5}$
Basic Integrity	$\geq 10^{-5}$

Mean Time Between Failures (**MTBF**) = 1/PFH
for SIL 4 is between 114,155 years and 11,415 years.

The higher the risk resulting from software failure, the higher the software integrity level will be.

- "[4.3] The required software integrity level shall be **decided and assessed at system level**, on the basis of the system safety integrity level and the level of risk associated with the use of the software in the system."
- "[4.4] **At least the Basic Integrity** requirements of this document shall be fulfilled for the software part of functions that have a safety impact below SIL 1"
- Software that was developed in accordance with the previous software standards can still be re- used for new projects."

Ansys, part of Synopsys ©2025

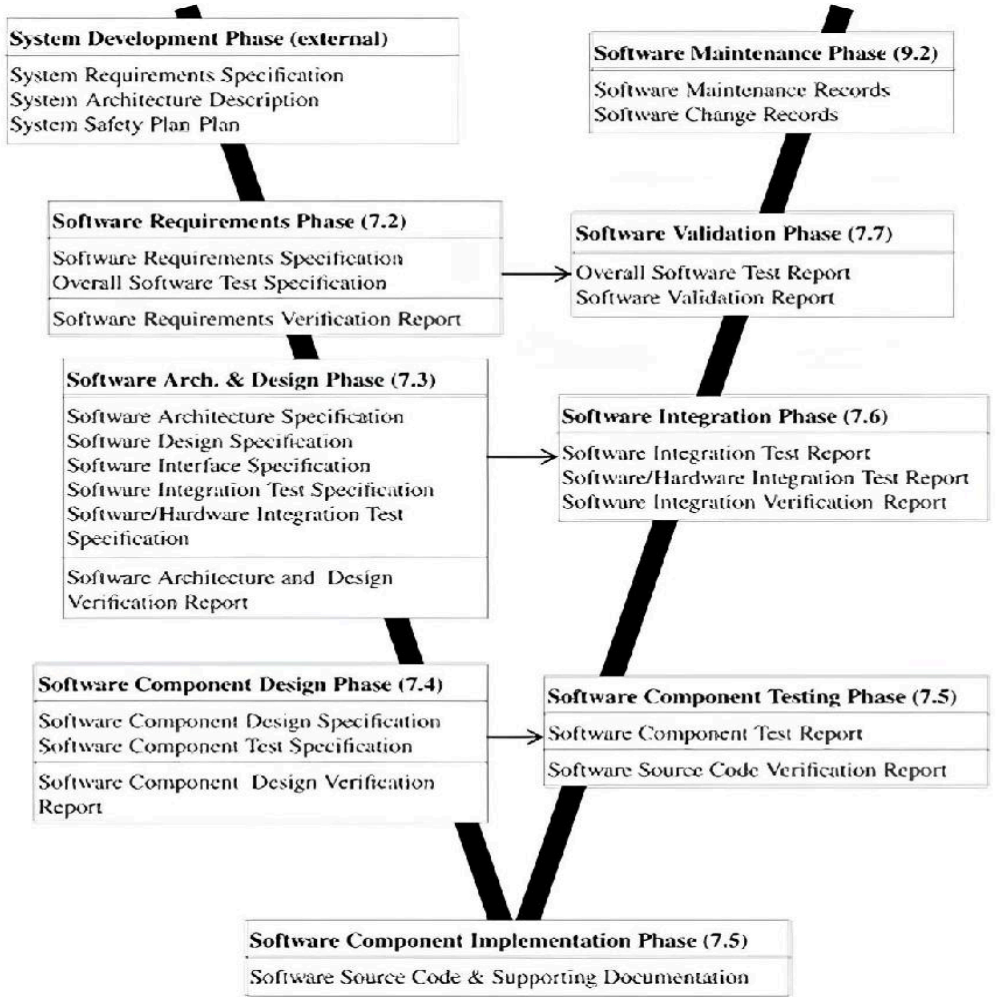
Ansys
part of SYNOPSYS®

Annex C.1

Lifecycle Model Examples

Annex C.1 Guidance on Lifecycle Models

C.1.2 Linear lifecycle models



Linear lifecycle models have implicit feedback loops, any change can force a return to earlier phases.

Examples:

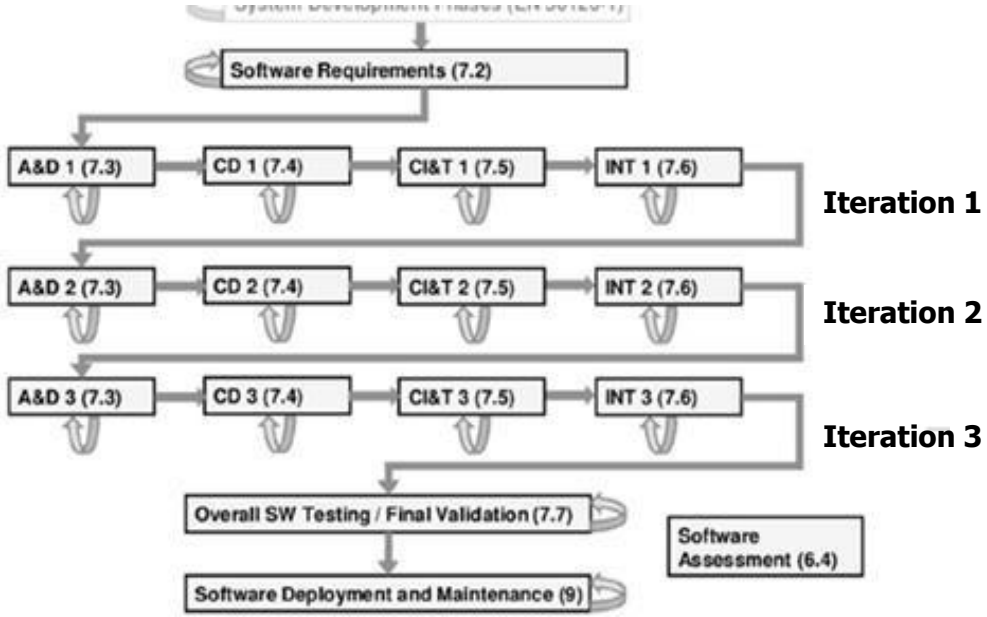
- An architecture review uncovers a missing requirement.
- Component testing exposes implementation/design defects or missing test cases.
- In the worst case, a system-requirements change during overall testing triggers re-doing all phases starting from requirements.
- These repetitions are unplanned and require updating the project plan.
- The standard's normative sections assume a linear lifecycle to model phase dependencies, reflecting the structure of activities and results at completion, regardless of the actual process used to produce validated software and documentation.

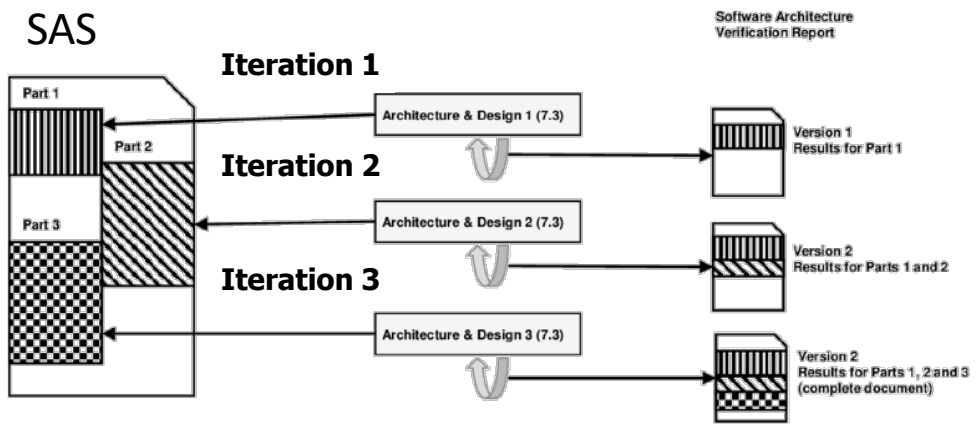
Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

Annex C.1 Guidance on Lifecycle Models

C.1.3 Iterative lifecycle models





- Use the same phases/activities as linear models but apply them repeatedly to smaller scope packages until the goal is met.
- Final deliverables are the same as in a linear lifecycle and must meet the same design, verification, and validation requirements.
- When reusing outputs from earlier iterations, verify they're still valid and haven't been invalidated by intervening changes.
- Unlike linear approaches, repetition is intentional and planned upfront, not just a reaction to unplanned events.

Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

Ansys
part of SYNOPSYS®

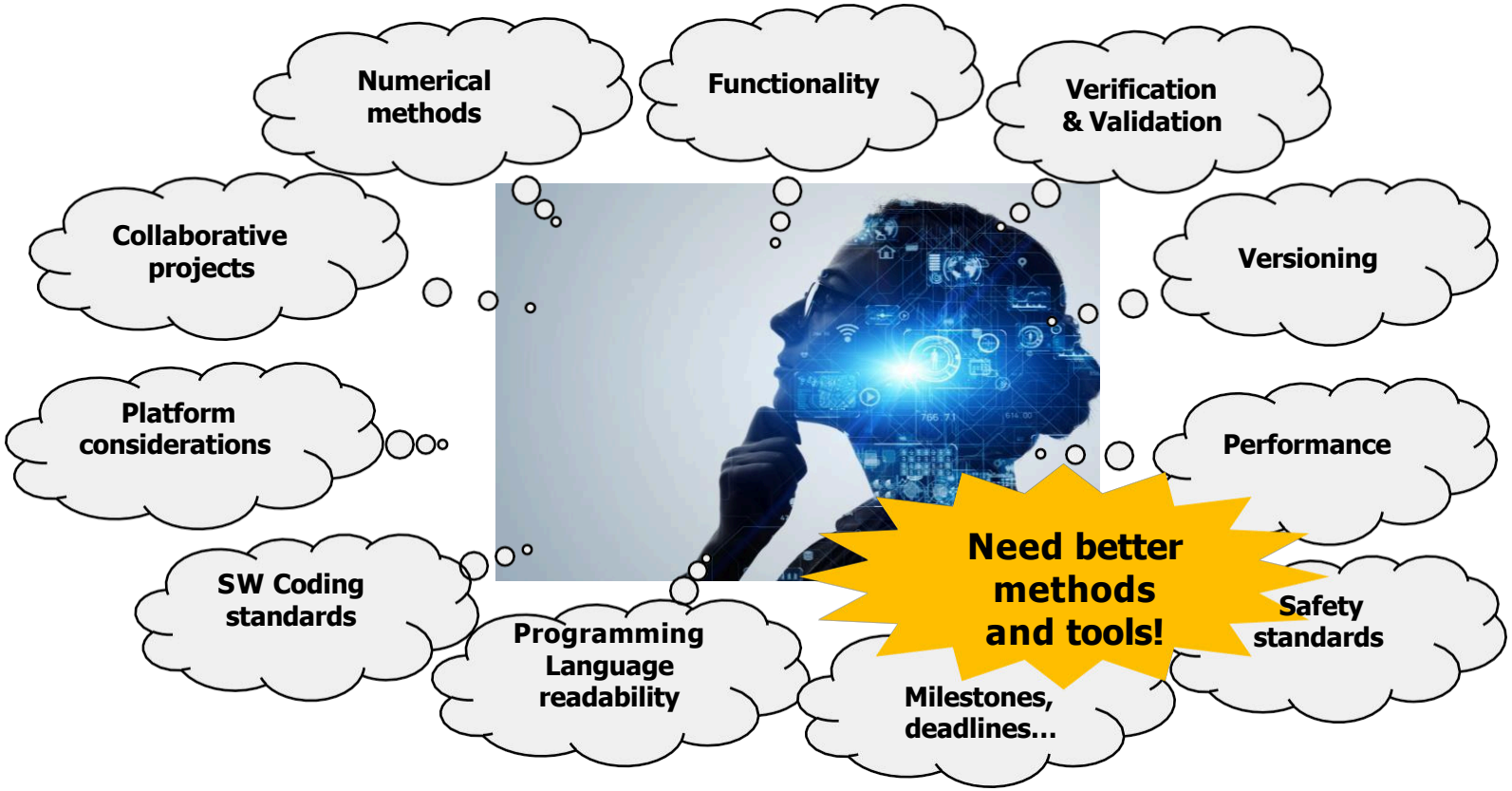
Annex C.2

Modelling

Annex C.2.1 Modelling

"*Nowadays, **software is ubiquitous in railway**. It provides more services to the operator by implementing more functions and by increasing the complexity of existing functions.* " **EN 50716 [C.2.1]**

- *"At the same time, the time schedule is more challenging while software needs to comply with the same or more demanding requirements"*
- *"This Annex explains possible usage benefits of modelling approaches during development at software level as well as specific aspects to consider."*
- *"It also provides a guidance for the application of modelling in software development in compliance to EN 50716."*

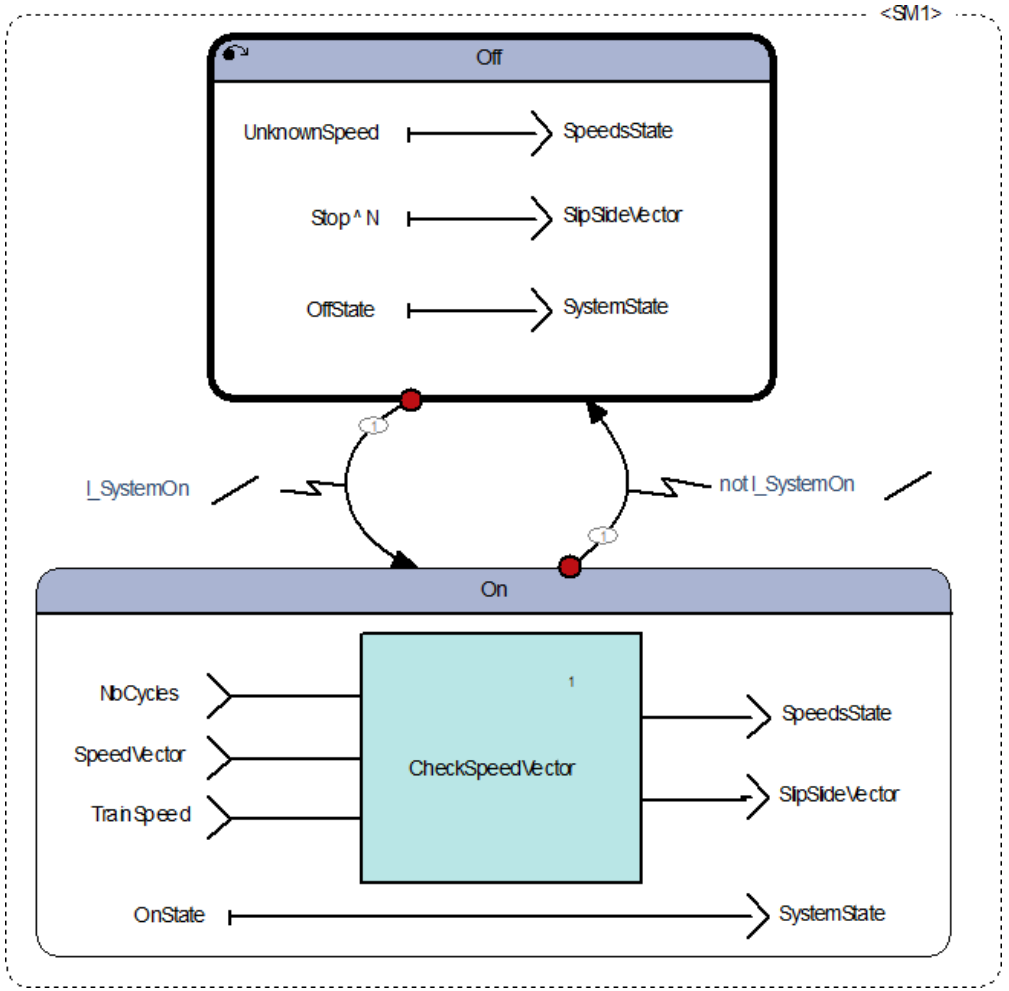


Ansys, part of Synopsys ©2025

Annex C.2.2 Modelling Definition

"A Model is a logical representation aimed at developing, understanding, communicating, or explaining aspects of a system, entity, or process."

- *"Modelling notations can be Graphical or Textual, Formal or Semiformal"*
- *"Modelling notations with **well-defined** syntax and semantics allow to decrease workload, complexity, project risks and costs*
  - *Easier evaluation of alternative scenario*
  - *Early discovery of defects*
  - *Automatic code and document generation*
  - *Enhanced verification means*
  - *Reduced test effort by use of model consistency*
  - *Built-in simulation capabilities*
  - *Integration with support tools (e.g. traceability)*
  - *Reuse on different hardware platforms"*
  - *increased consistency leading to improved change management and maintainability*

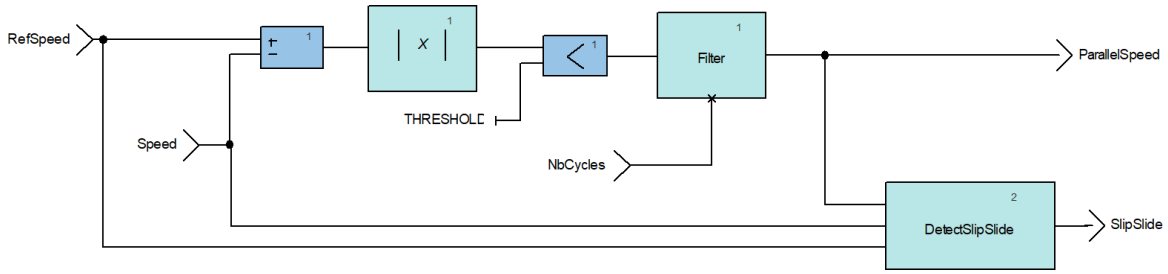


Ansys, part of Synopsys ©2025

C.2.3 Modelling – Lifecycle Issues & Documentation

A document is information and the medium on which it is contained. Information is not limited to natural language and medium is not limited to paper. Thus, models stored in databases are actual documents. Table A.1 can then be applied as is." C.2.6.1

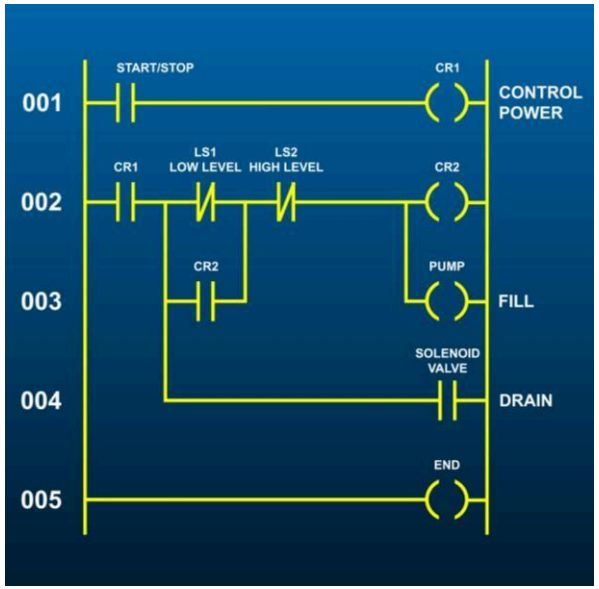
*"Models can better **integrate phases** together and **minimize overlaps** between them mainly because one given model may contain the deliverables of several phases of the development, **reducing duplication** and **improving consistency**. Models can also **reduce** the need for certain activities."* **C.2.3**



"All tools used when modelling need to comply with the requirements of 6.7." Tool qualification

*"Some models are not easily viewed, edited and printed without dedicated (*closed*) tools.*

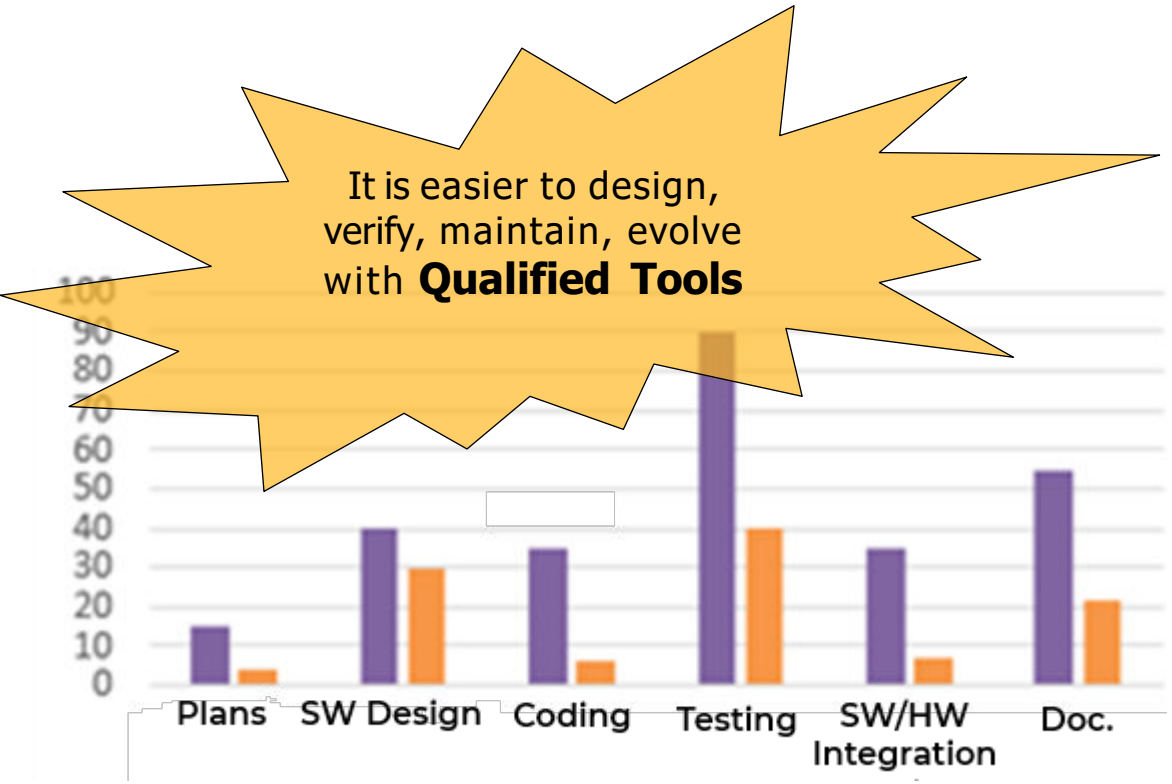
Even if document generation is possible, a lot of the information contained in the models, the metadata associated to the elements they contain, may be lost in the process." **C.2.3**



Ansys, part of Synopsys ©2025

Annex C.2.5 Modelling – Support Tools & Languages

All tools used when modelling need to comply with the requirements of **6.7**.



The objective is to provide evidence that potential failures of tools do not adversely affect the integrated toolset output in a safety related manner [6.7.1]

- **Tool class T1** tool which is not used to generate outputs which can directly or indirectly contribute to the executable code (including data), e.g. **model editors** ✓
- **Tool class T2** tool which is used to support the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software

  ✓**Verification tools**
  ✓**Diagnostic tools**
  ✓**Report tools**
  ✓**Traceability tools**

- **Tool class T3** tool which is used to generate outputs which can directly or indirectly contribute to the executable code of the system

  ✓**Code Generators**

Ansys, part of Synopsys ©2025

C.2.6.2 Modelling – Software Requirements

Table A.2 Software Requirements Specification, EN 50128:2011

TECHNIQUE/MEASURE	Ref	SIL 0	SIL 1	SIL 2	SIL 3	SIL 4
1. Formal Methods (based on a mathematical approach)	D.28	-	R	R	HR	HR
2. Modelling	Table A.17	R	R	R	HR	HR
3. Structured methodology	D.52	R	R	R	HR	HR
4. Decision Tables	D.13	R	R	R	HR	HR
Requirements: 1) The Software Requirements Specification shall include a description of the problem in natural language and any necessary formal or semiformal notation. 2) The table reflects additional requirements for defining the specification clearly and precisely. One or more of these techniques shall be selected to satisfy the Software Safety Integrity Level being used.						

Table A.2 Software Requirements Specification, EN 50716:2023

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Modelling	Table A.17	R	R	R	HR	HR
2. Structured methodology	D.52	R	R	R	HR	HR
3. Decision Tables	D.13	R	R	R	HR	HR
Requirements: 1) The Software Requirements Specification shall include a description of the problem in natural language and any necessary formal or semiformal notation. 2) The table reflects additional requirements for defining the specification clearly and precisely. One or more of these techniques shall be selected to satisfy the Software Integrity Level being used.						

Table A.17 — Modelling

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Data Modelling	D.65	R	HR	HR	HR	HR
2. Data Flow Diagrams	D.11	-	HR	HR	HR	HR
3. Control Flow Diagrams	D.66	R	HR	HR	HR	HR
4. Finite State Machines or State Transition Diagrams	D.27	-	HR	HR	HR	HR
5. Petri Nets	D.55	-	HR	HR	HR	HR
6. Decision/Truth Tables	D.13	R	HR	HR	HR	HR
7. Formal Methods	D.28	-	HR	HR	HR	HR
8. Performance Modelling	D.39	-	HR	HR	HR	HR
9. Prototyping/Animation	D.43	-	R	R	R	R
10. Structure Diagrams	D.51	-	HR	HR	HR	HR
11. Sequence Diagrams	D.67	R	HR	HR	HR	HR
12. Cause Consequence Diagrams	D.6	R	R	R	R	R
13. Event Tree Diagrams	D.22	-	R	R	R	R
Requirements: 1) For SIL 1-4, modelling guidance shall be defined and used. 2) For SIL 1-4, at least one of the HR techniques shall be chosen.						

Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

C.2.6.3 Modelling – Architecture and Design

*"Most of the modelling techniques of **Table A.17** can be used for the architecture and design of the software, particularly time **Petri nets**, **control flow diagrams**, and **finite state machines**."*

*"Techniques of Table A.4 related to suitable programming languages are either not applicable or need to be adapted (see **Table C.1**)."*

Table A.4 — Software design and implementation (7.3, 7.4 and 7.5)

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Modelling	Table A.17	R	HR	HR	HR	HR
2. Structured methodology	D.52	R	HR	HR	HR	HR
3. Modular Approach	D.38	HR	M	M	M	M
4. Components	Table A.20	HR	HR	HR	HR	HR
5. Design and Coding Standards	Table A.12	HR	HR	HR	M	M
6. Analysable Programs	D.2	HR	HR	HR	HR	HR
7. Structured Programming	D.53	R	HR	HR	HR	HR
8. Suitable Programming Languages	Table A.15	R	HR	HR	HR	HR
Requirements: 1) An approved combination of techniques for software integrity levels 3 and 4 is 1, 3, 4, 5. 2) An approved combination of techniques for software integrity levels 1 and 2 is 2, 3, 4, 5 and one from 7 or 8.						

Table C.1 — Architecture and design typical adaptation for modelling

TECHNIQUES / MEASURES OF TABLE A.4	TYPICAL ADAPTATION FOR MODELLING
1. Modelling	applicable as is
2. Structured methodology	applicable as is
3. Modular approach	A module is an element of organization of the source code to improve its understanding and separate the concerns. If the source code is not automatically generated, if it's modified after generation or if it's the input of manual analysis, this technique remains applicable.
4. Component	applicable as is
5. Design and coding standards	The design standard shall cover the modelling notation (i.e. it shall encourage good modelling practices and avoid poorly-defined features of the modelling language) and the structure, organization and hierarchy of the model. The coding standard is fully applicable when the source code is not automatically generated, when it's modified after generation or when it's input of manual analysis. Otherwise, the coding standard may be present as part of a validated translator, particularly the rules contributing to the avoidance of poorly-defined features of the programming language.
6. Analysable programs	If analysis are performed on the source code, the technique is applicable as-is. If all or parts of the analysis are made on the models, the models also need to be analysable.
7. Structured programming	This technique is used to limit the structural complexity of the source code. An equivalent technique need then to be similarly applied on models. The technique is also applicable to the source code if specific analysis are done on it.
8. Suitable Programming language	Criteria for programming languages are also applicable to modelling notations. If the source code is automatically generated but neither modified after generation nor the input of subsequent analysis and depending on the guarantees provided by the generator, some of the criteria can be useless.

Ansys, part of Synopsys ©2025

C.2.6.4 Modelling –Component Design & Testing

- "Most of the modelling techniques of Table A.17 can also be used for software component design. As for architecture and design, depending on the context, techniques of Table A.4 related to suitable programming languages need to be adapted." (see **Table C.1**).

- Techniques for software component analysis and testing of **Table A.5** are directly applicable as-is to modelling except test coverage for code. Indeed, techniques of **Table A.21** are very specific to imperative programming languages. Alternative coverage criteria need then to be defined depending on the modelling notation used with the objective to cover all parts of the model (e.g. components, interfaces, data flow, control flow) with the same level as with programming languages

Table A.5 — Software component analysis and testing (6.2 and 7.4)

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Formal Proof	D.29	-	R	R	HR	HR
2. Static Analysis	Table A.19	-	HR	HR	HR	HR
3. Dynamic Analysis and Testing	Table A.13	HR	HR	HR	M	M
4. Metrics	D.37	-	R	R	R	R
5. Test Coverage for code	Table A.21	-	HR	HR	HR	HR
6. Performance Testing	Table A.18	-	HR	HR	HR	HR
7. Interface Testing	D.34	HR	HR	HR	HR	HR

Table A.21 — Test coverage for code

Test coverage criterion	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Statement	D.50	R	HR	HR	HR	HR
2. Branch	D.50	-	R	R	HR	HR
3. Compound Condition	D.50	-	R	R	HR	HR
4. Data flow	D.50	-	R	R	HR	HR
5. Path	D.50	-	R	R	HR	HR

Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

C.2.6.5 Modelling – Component Implementation & Testing

"When using models at software component level, the subclauses applicable to the source code can be contextualized in similar subclauses applicable to models (see Table C.2)."

"If the source code is automatically generated and neither modified after generation nor input of analysis, it is no longer necessary to apply the subclauses listed in the **Table C2**, as 6.7 would apply for the automatic code generation tool."

As for architecture and design, techniques of Table A.12 related to coding standards need to be transposed to the modelling notations used (see **Table C.3**)

Table C.3 — Coding standards techniques / measures typical adaptation for modelling

TECHNIQUES / MEASURES OF TABLE A.12	TYPICAL ADAPTATION FOR MODELLING
1. Coding Standard	Modelling Standard
2. Coding Style Guide	Modelling Style Guide
3. Limited size and complexity of Functions, Subroutines and Methods	Limited size and complexity of model parts
4. Entry/Exit Point strategy for Functions, Subroutines and Methods	Not applicable since models generally don't have the concept of unconditional jump
5. Defined control of Global Variables	Defined control of input and output data in model parts (e.g. signals, information flows)

Table C.2 — Component implementation and testing typical adaptation for modelling

SUBCLAUSES	TYPICAL ADAPTATION FOR MODELLING
7.5.4.2 The size and complexity of the developed source code shall be balanced.	7.5.4.2 The size and complexity of the developed model shall be balanced.
7.5.4.3 The Software Source Code shall be readable, understandable and testable.	7.5.4.3 The model shall be readable, understandable and testable.
7.5.4.4 The Software Source Code shall be placed under configuration control before the commencement of documented testing.	7.5.4.4 The model shall be placed under configuration control before the commencement of documented testing.
7.5.4.10 After the Software Source Code and the Software Component Test Report have been established, verification shall address a) the adequacy of the Software Source Code as an implementation of the Software Component Design Specification, b) the correct use of the chosen techniques and measures from Table A.4 as a set satisfying 4.8 and 4.9, c) determining the correct application of the coding standards, d) that the Software Source Code meets the general requirements for readability and traceability in 5.3.2.7 to 5.3.2.10 and in 6.5.4.14 to 6.5.4.17, as well as the specific requirements in 7.5.4.1 to 7.5.4.4, e) the adequacy of the Software Component Test Report as a record of the tests carried out in accordance with the Software Component Test Specification	7.5.4.10 After the model and the Software Component Test Report have been established, verification shall address a) the adequacy of the model as an implementation of the Software Component Design Specification, b) the correct use of the chosen techniques and measures from Table A.4 as a set satisfying 4.8 and 4.9, c) determining the correct application of the modelling standards, d) that the model meets the general requirements for readability and traceability in 5.3.2.7 to 5.3.2.10 and in 6.5.4.14 to 6.5.4.17, as well as the specific requirements in 7.5.4.1 to 7.5.4.4, e) the adequacy of the Software Component Test Report as a record of the tests carried out in accordance with the Software Component Test Specification

Ansys, part of Synopsys ©2025

C.2.6.6 Modelling – Integration

*"Techniques for software integration analysis and testing of **Table A.6** are all directly as-is applicable to modelling."*

Table A.6 — Software integration analysis and testing (7.3 and 7.6)

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Dynamic Analysis and Testing	Table A.13	HR	HR	HR	HR	HR
2. Performance Testing	Table A.18	-	R	R	HR	HR

Table A.13 — Dynamic analysis and testing

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Test Case Execution from Error Guessing	D.20	R	R	R	R	R
2. Test Case Execution from Error Seeding	D.21	-	R	R	R	R
3. Structure-Based Testing	D.50	-	R	R	HR	HR
4. Test Case Execution from Cause Consequence Diagrams	D.6	-	-	-	R	R
5. Prototyping / Animation	D.43	-	-	-	R	R
6. Test Case Execution from Boundary Value Analysis	D.4	-	HR	HR	HR	HR
7. Equivalence Classes and Input Partition Testing	D.18	R	HR	HR	HR	HR
8. Process Simulation	D.42	R	R	R	R	R

Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

C.2.6.7 Modelling – Overall Software Testing

*"Techniques for overall software analysis and testing of **Table A.7** are all directly as-is applicable to modelling"*

Table A.7 — Overall software analysis and testing (6.2 and 7.2)

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Performance Testing	Table A.18	-	HR	HR	M	M
2. Dynamic Analysis and Testing	Table A.13	HR	HR	HR	M	M
3. Modelling	Table A.17	-	R	R	R	R
NOTE At the overall software level the "Dynamic Analysis and Testing" technique is based on Software Requirements Specification (Functional) and is applied on the whole integrated software (Black-box).						

Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

Ansys
part of SYNOPSYS®

Updates in Part 5

Software Management & Organization

Organizational Structure Evolution

EN 50128:2011

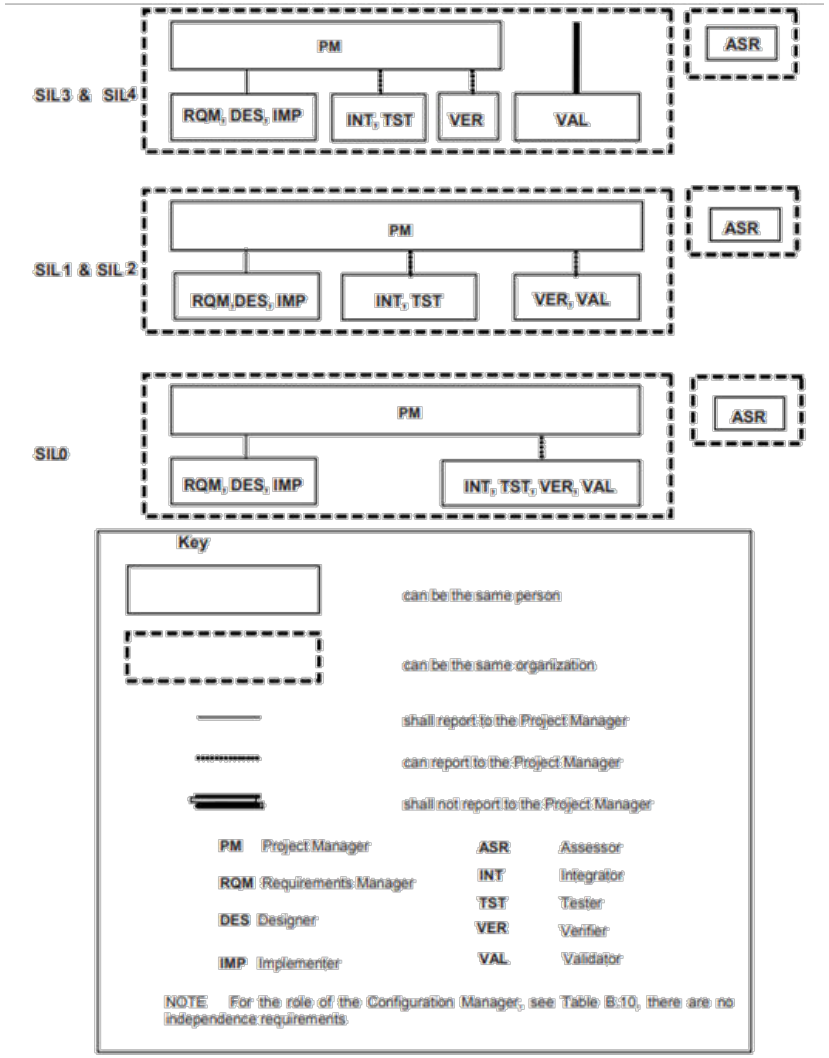


Figure 2 – Illustration of the preferred organisational structure

EN 50716:2023

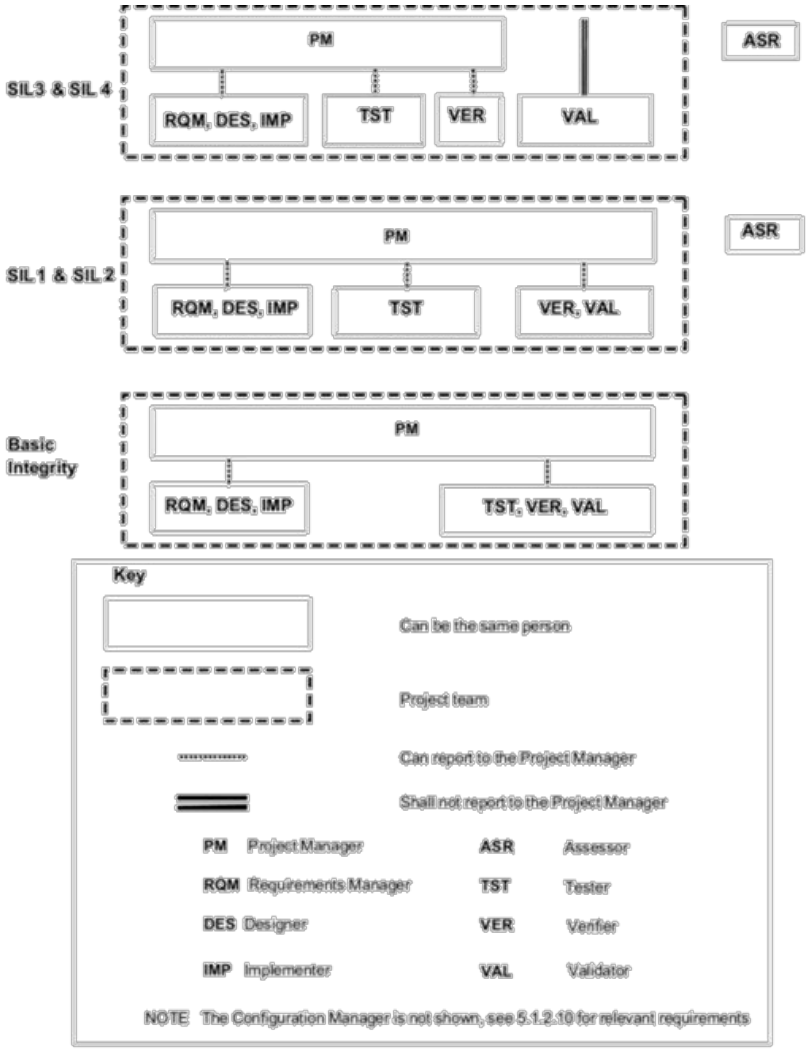


Figure 2 — Illustration of the organizational structure

Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

Organizational Structure changes

1. From 3 org charts to 1 baseline + add-ons.
   - 50128 had separate "preferred structures" by SIL.
   - 50716 collapses to: a common baseline (5.1.2.10), extra test-independence rules (5.1.2.11), and a short SIL3–4 add-on (5.1.2.12).
2. Principle-based independence.
   - Keeps core separations: Validator/Verifier ≠ RM/Designer/Implementer (b); no dev ↔ Validator reporting (c); Validator ≠ PM (d); Config Manager ≠

     Validator (e). Drops rigid role trees; focuses on demonstrable independence.
3. Stable V&V roles.
   - **Verifier/Validator is not just a person, but an entity**
   - Verifier/Validator fixed at project level; any personnel change must be justified and not jeopardize activities.
4. Clear testing independence.
   - Devs can't test their own component; may test others/higher levels if independence is shown (5.1.2.11c). If Validator tests →another Validator

     reviews (a). If Verifier tests →a Verifier or Validator reviews (b).
5. High-SIL firewall kept.
   - Validator shall not report to PM (5.1.2.12a).
6. Assessor modernized.
   - Required for SIL1–4; can be from any stakeholder but organizationally independent of the project team and empowered (5.1.2.4–7).
7. Terminology.
   - SIL0 replaced by "Basic Integrity" but covered by the same organization.

Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

Ansys
part of SYNOPSYS®

Updates in Part 6

Support tools and Languages

6.7 Support tools and languages

6.7.4.12 The relation between the tool classes and the applicable subclauses is defined within Table 1.

EN 50716 **Table 1 — Relation between tool class and applicable subclauses**

Tool class	Applicable requirements for SIL 1 to SIL 4	Applicable requirements for Basic Integrity
T1	6.7.4.1	6.7.4.1
T2	6.7.4.1, 6.7.4.2, 6.7.4.3, 6.7.4.10, 6.7.4.11	6.7.4.1
T3	6.7.4.1, 6.7.4.2, 6.7.4.3, 6.7.4.4, 6.7.4.5, 6.7.4.9, 6.7.4.10, 6.7.4.11	6.7.4.1, 6.7.4.3, 6.7.4.10, 6.7.4.11

EN 50128 **Table 1 - Relation between tool class and applicable sub-clauses**

Tool class	Applicable sub-clauses
T1	6.7.4.1
T2	6.7.4.1, 6.7.4.2, 6.7.4.3, 6.7.4.10, 6.7.4.11
T3	6.7.4.1, 6.7.4.2, 6.7.4.3, 6.7.4.4, 6.7.4.5 or 6.7.4.6, 6.7.4.7, 6.7.4.8, 6.7.4.9, 6.7.4.10, 6.7.4.11

- Applicability by Integrity level
- SIL-specific wording
- Basic Integrity Track, for non-SIL developments, obligations are lighter, especially for T2.
- **Evidence for T3 (Code Generators)**
- EN 50716 6.7.4.4 requires evidence via one or more named techniques and spells out what “history of use” means (documented projects, anomaly lists, version lineage, periodic confirmation)
- Fall-back /compensating measures
- Fewer escape hatches; expectations are clearer and
- stricter for T3 in EN 50716. It removes “Plan B.” 6.7.4.6–
- 6.7.4.8 subclauses (marked “intentionally left blank”) and expects you to meet 6.7.4.4 using its defined evidence paths.
- Operational tools exclusion.
- EN 50717 explicitly says that *tools after development as part of the system are treated as software*

Ansys, part of Synopsys ©2025

Software Component

"Constituent part of software which has well-defined interfaces and behavior with respect to the software architecture and design " **EN 50716 [3.1.4]**

A software component fulfils the following criteria:

- *it is designed according to "Components" (see Table A.20);*
- *it covers a specific subset of software requirements;*
- *it is clearly identified and has an independent version inside the configuration management system or is a part of a collection of components (e.g. subsystems) which have an independent version.*

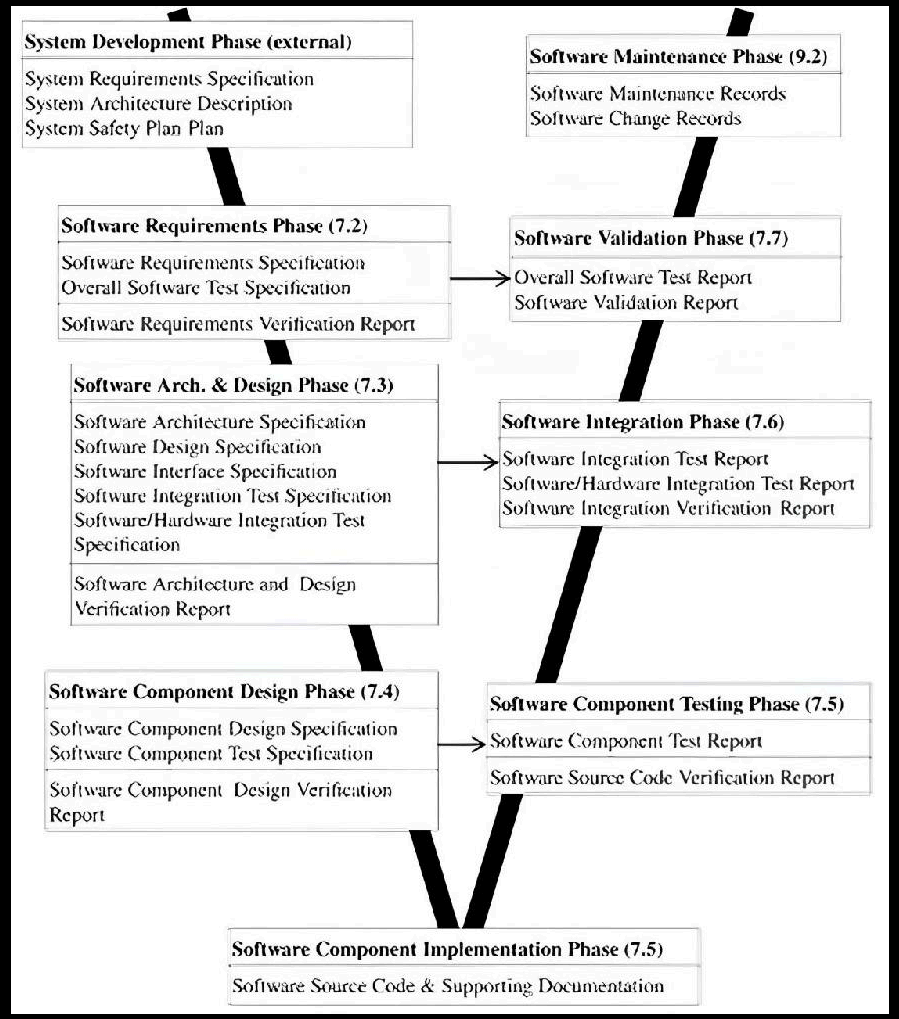
Table A.20 — Components

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Information Hiding / Encapsulation	D.33	R	HR	HR	HR	HR
2. Parameter Number Limit	D.38	R	R	R	R	R
3. Fully Defined Interface	D.38	R	HR	HR	M	M
Requirement: 1) Information encapsulation are only highly recommended if there is no general strategy for data access.						
NOTE Technique/measure 3 is for Internal Interfaces.						

©2025 Ansys Inc., part of Synopsys Group

Ansys part of SYNOPSYS®

Top-Down Design and iterative



The principle of "top-down design" is fundamental to EN 50716, Reflected in a lifecycle approach that implies successive refinements:

1. Starting with high-level system safety requirements.
2. Decomposing these into detailed software requirements.
3. Structuring the software into a hierarchy of components with well-defined interfaces.
4. Refining these components through detailed design.
5. Maintaining strict traceability throughout the process.

The systematic decomposition and refinement are essential for managing the complexity of safety-critical software and ensuring that safety is built into the system from the highest level down to the lowest lines of code.

©2025 Ansys Inc., part of Synopsys Group

Ansys part of SYNOPSYS®

Verification at Each Phase

Objective

The objective of software verification is to examine and arrive at a judgment based on evidence that output items (process, documentation, software or application) of a specific development phase fulfil the requirements and plans with respect to completeness, correctness and consistency. **[6.2.1]**

Requirements

*Each **Software Verification Report** shall document the following:*

a) the identity and configuration of the items verified, as well as the Verifier names;

b) items which do not conform to the specifications;

c) components, data, structures and algorithms poorly adapted to the problem;

d) detected errors or deficiencies;

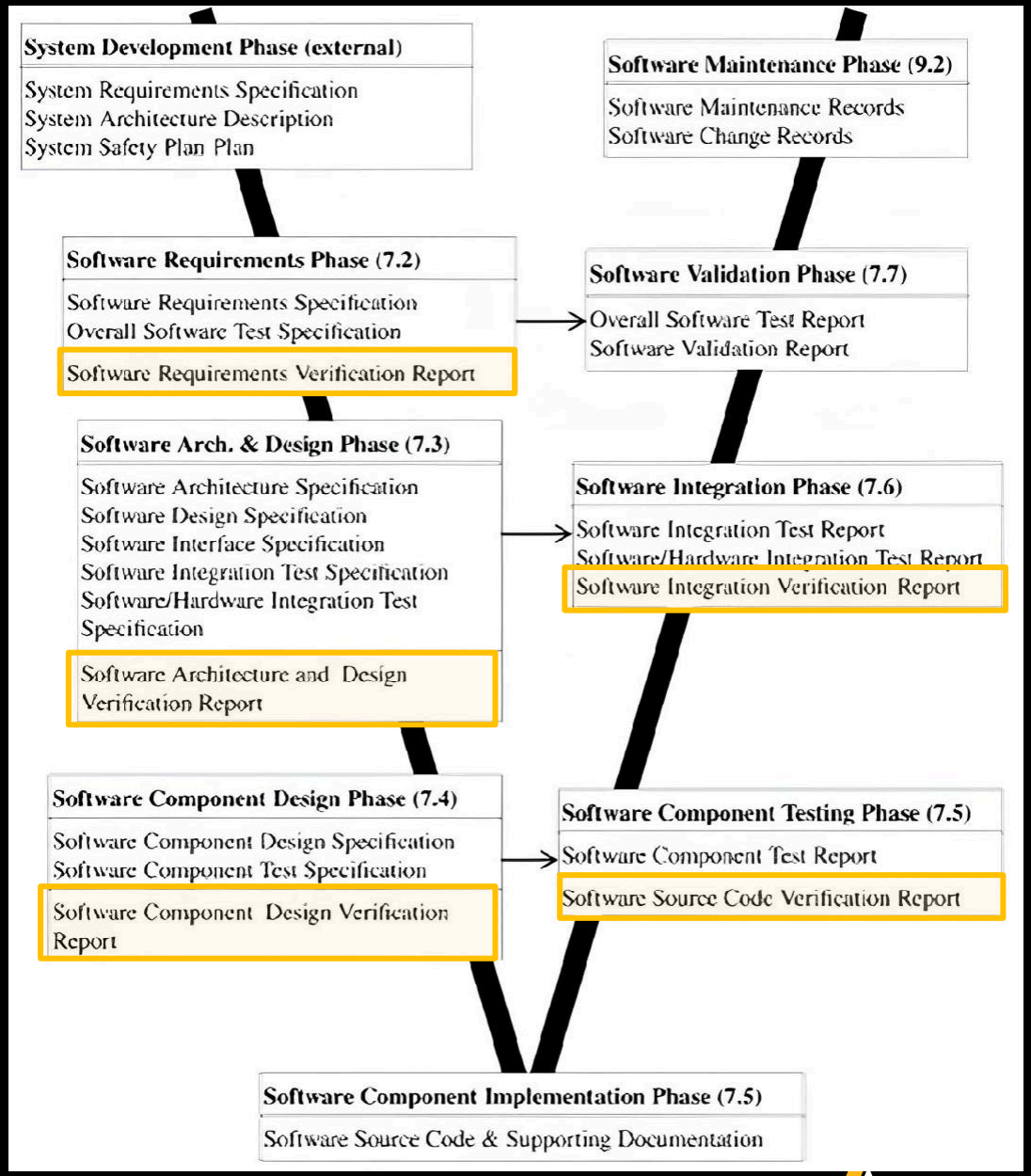
e) the fulfilment of, or deviation from, the Software Verification Plan (in the event of deviation the Verification Report shall explain whether the deviation is critical or not);

f) the correct use of the chosen techniques and measures;

g) assumptions if any;

h) a summary of the verification results

EN 50716 [6.2.4.13]



©2025 Ansys Inc., part of Synopsys Group

Ansys part of SYNOPSYS®

Lifecycle and Documentation (Clauses 5.3)

Table A.1 — Lifecycle issues and documentation (5.3)

DOCUMENTATION	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
Planning					
1. Software Quality Assurance Plan	HR	HR	HR	HR	HR
2. Software Planning Verification Report	R	HR	HR	HR	HR
3. Software Configuration Management Plan	HR	HR	HR	HR	HR
4. Software Verification Plan	HR	HR	HR	HR	HR
5. Software Validation Plan	HR	HR	HR	HR	HR
Software requirements					
6. Software Requirements Specification	HR	HR	HR	HR	HR
7. Overall Software Test Specification	HR	HR	HR	HR	HR
8. Software Requirements Verification Report	R	HR	HR	HR	HR
Architecture and design					
9. Software Architecture Specification	R	HR	HR	HR	HR
10. Software Design Specification	R	HR	HR	HR	HR
11. Software Interface Specifications	HR	HR	HR	HR	HR
12. Software Integration Test Specification	R	HR	HR	HR	HR
13. Software/Hardware Integration Test Specification	R	HR	HR	HR	HR
14. Software Architecture and Design Verification Report	R	HR	HR	HR	HR
Component Design					
15. Software Component Design Specification	-	HR	HR	HR	HR

(47 Documents)

Objectives

- *"To structure the development of the software into defined phases and activities*
- *To record all information pertinent to the software throughout the lifecycle of the software."* Extract from EN 50716 [5.3.1]

Requirements

- *A lifecycle model for the development of software shall be selected.*
- *For each document, traceability shall be provided in terms of a unique reference number and a defined and documented relationship with other documents*
- *Documents may be combined or divided. Some development steps may be combined, divided or, when justified, eliminated.*
- *Documents can be generated from databases or modelling tools, in which case, traceability shall be preserved. Extract from EN 50716 [5.3.2]*

Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

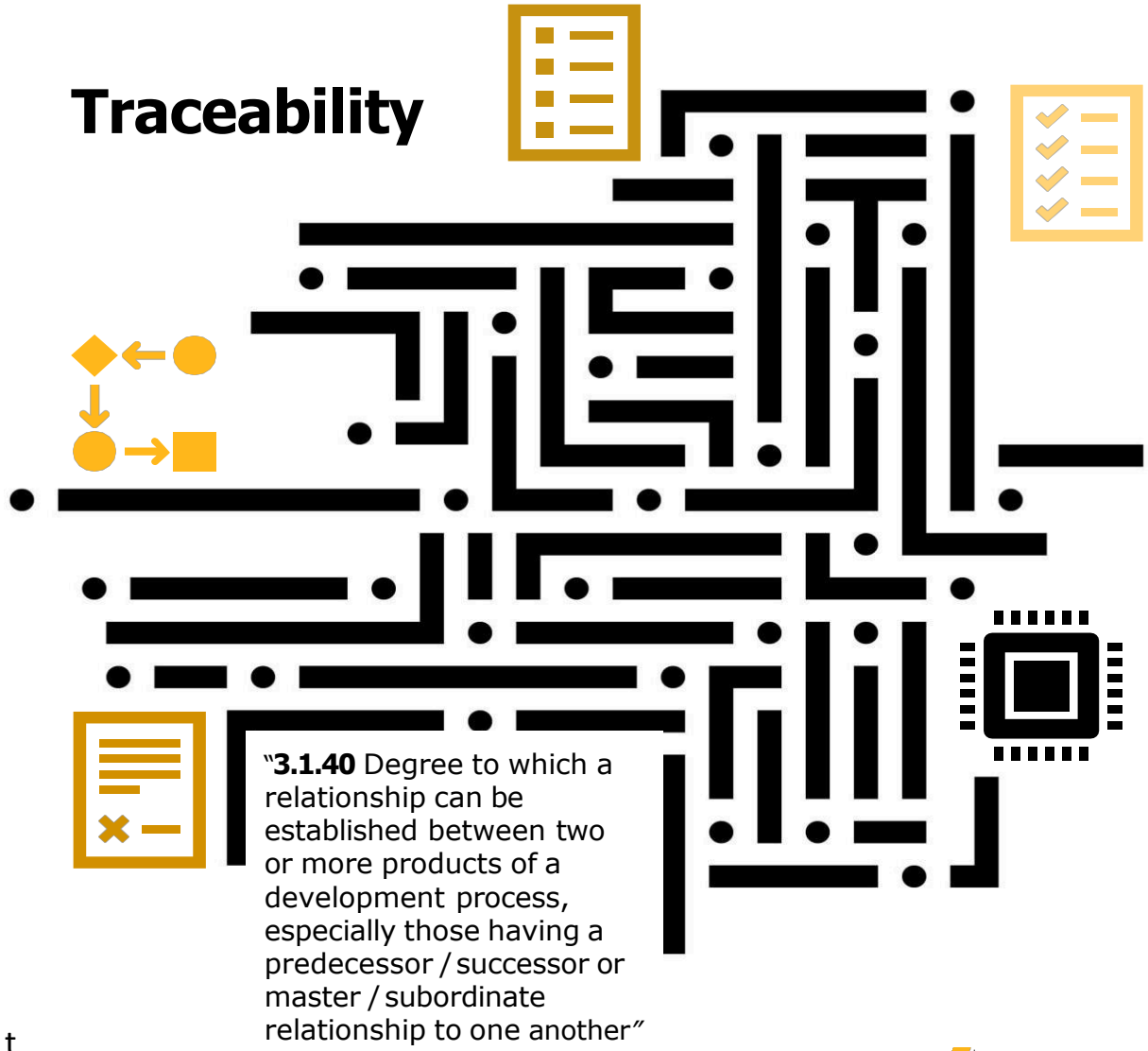
Traceability

5.3.2.7 For each document, traceability shall be provided in terms of a unique reference number and a defined and documented relationship with other documents.

6.5.4.15 Within the context of this document, and to a degree appropriate to the specified software integrity level, traceability shall particularly address

a) traceability of requirements to the design or other objects which fulfil them,
b) traceability of design objects to the implementation objects which instantiate them,
c) traceability of requirements and design objects to the tests (component, integration, overall test) and analyses that verify them

Traceability shall be the subject of configuration management



Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

Techniques and Measures - illustration

Table A.4 — Software design and implementation (7.3, 7.4 and 7.5)

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Modelling	Table A.17	R	HR	HR	HR	HR
2. Structured methodology	D.52	R	HR	HR	HR	HR
3. Modular Approach	D.38	HR	M	M	M	M
4. Components	Table A.20	HR	HR	HR	HR	HR
5. Design and Coding Standards	Table A.12	HR	HR	HR	M	M
6. Analysable Programs	D.2	HR	HR	HR	HR	HR
7. Structured Programming	D.53	R	HR	HR	HR	HR
8. Suitable Programming Languages	Table A.15	R	HR	HR	HR	HR

Requirements:

1) An approved combination of techniques for software integrity levels 3 and 4 is 1, 3, 4, 5.

2) An approved combination of techniques for software integrity levels 1 and 2 is 2, 3, 4, 5 and one from 7 or 8.

Table A.17 — Modelling

TECHNIQUE/MEASURE	Ref	Basic Integrity	SIL 1	SIL 2	SIL 3	SIL 4
1. Data Modelling	D.65	R	HR	HR	HR	HR
2. Data Flow Diagrams	D.11	-	HR	HR	HR	HR
3. Control Flow Diagrams	D.66	R	HR	HR	HR	HR
4. Finite State Machines or State Transition Diagrams	D.27	-	HR	HR	HR	HR
5. Petri Nets	D.55	-	HR	HR	HR	HR
6. Decision/Truth Tables	D.13	R	HR	HR	HR	HR
7. Formal Methods	D.28	-	HR	HR	HR	HR

D.53 Structured programming

Aim

To design and implement the software component in a way which makes practical the analysis of the software component. This analysis should be capable of discovering all significant component behaviour.

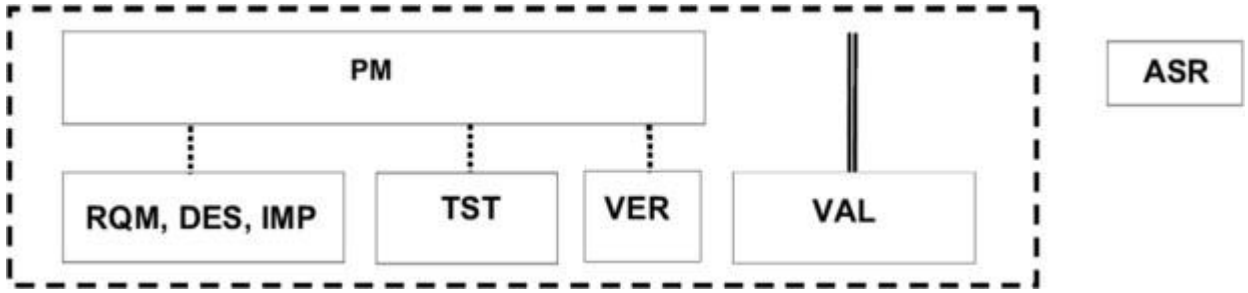
Description

The software component should contain the minimum of structural complexity. Complicated branching should be avoided. Loop constraints and branching should (where possible) be simply related to input parameters. The software component should be divided into appropriately small modules, and the interaction of these modules should be explicit. Features of the programming language which

©2025 Ansys Inc., part of Synopsys Group

Ansys part of SYNOPSYS®

Organizational Structure for SIL 3 & SIL 4 (Clauses 5)



ASR

- **PM** Project Manager
- **RQM** Requirements Manager
- **DES** Designer
- **IMP** Implementer

Can be the same person

- **TST** Tester
- **VAL** Validator -- shall not report to **PM**
- **VER** Verifier

Team

- **ASR** Assessor

Objectives [5.1.1]

*"**a)** To reduce the probability of people in different roles suffering from the same misconceptions or making the same mistakes by ensuring independence between roles.*

***b)** To ensure that people in roles which involve making judgements about the acceptability of a product or process from the point of view of safety should not be influenced by pressure from their peers or supervisors, or by considerations of commercial gain"*

"[5.1.2.4] An Assessor shall be appointed"

"[5.1.2.6] The Assessor shall be independent from the project team and shall be a different entity, organizationally independent, from those undertaking other roles in the project."

Integrator desperation integrated within tester

Ansys, part of Synopsys ©2025

Personnel Competence and Responsibilities

5.2.2.5 Responsibilities shall be compliant with the requirements defined in **Annex B** (**Tables B.1 to B.9**).

5.2.1 Objectives

To ensure that all personnel who have responsibilities for the software are competent, empowered and capable of fulfilling their responsibilities by demonstrating the ability to perform relevant tasks correctly, efficiently and consistently to a high quality and under varying conditions.

1. The key competencies required for each role in the software development are defined in Annex B.
2. Documented evidence of personnel competence, including technical knowledge, qualifications, relevant experience and appropriate training, shall be maintained by the supplier's organization in order to demonstrate appropriate safety organization.

5.2.2.4 Once it has been proved to the satisfaction of an **Assessor** or by a certification that competence has been demonstrated for all personnel appointed in various roles, each individual will need to show continuous maintenance and, if needed, development of competence.

Table B.2 — Designer role specification

Role: Designer
Responsibilities: 1. shall transform specified software requirements into acceptable solutions 2. shall own the architecture and downstream solutions 3. shall define or select the design methods and supporting tools 4. shall apply appropriate design principles and standards 5. shall develop component specifications where appropriate 6. shall maintain traceability to and from the specified software requirements 7. shall develop and maintain the design documentation 8. shall ensure design documents are under change and configuration control
Key competencies: 1. shall be competent in engineering appropriate to the application area 2. shall be competent in safety design principles (only for safety-related functions) 3. shall be competent in design analysis and design test methodologies 4. shall be able to work within design constraints in a given environment 5. shall be competent in the problem domain 6. shall understand all the constraints imposed by the hardware platform, the operating system and the interfacing systems 7. shall be competent in application engineering for development of application data

Ansys, part of Synopsys ©2025

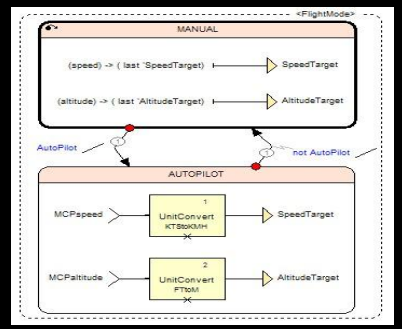
Ansys part of SYNOPSYS®

Ansys

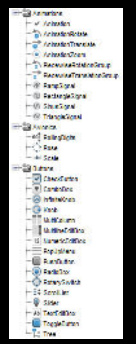
SCADE

Proven Model-Based Solution for Safety Critical SIL4 Railway Embedded Software Applications supported by Formal Language

For Embedded SW Control Applications

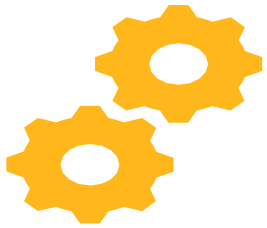


For Embedded HMI Applications



RAILLIVE! *IFEMA Madrid, Spain - 26-28 November 2025*

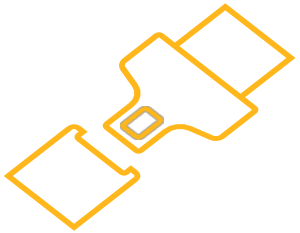
Key Elements of EN 50716



Key Process Areas:

Requirements Engineering Software Architecture & Design Coding & Integration Verification & Validation Configuration & Change Management

Software Safety Assurance

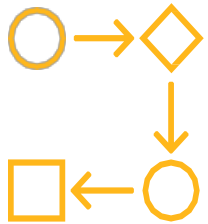


SIL-Dependent Requirements:

Stricter documentation, verification, and independence criteria as SIL goes from 0 to 4.

Emphasis on Tool Qualification:

Tools must be qualified if they contribute to safety-related outputs.



Advocates for Modelling Languages

Explains possible usage benefits of modelling approaches during all phases of the development of Safety-critical components.

Ansys, part of Synopsys ©2025

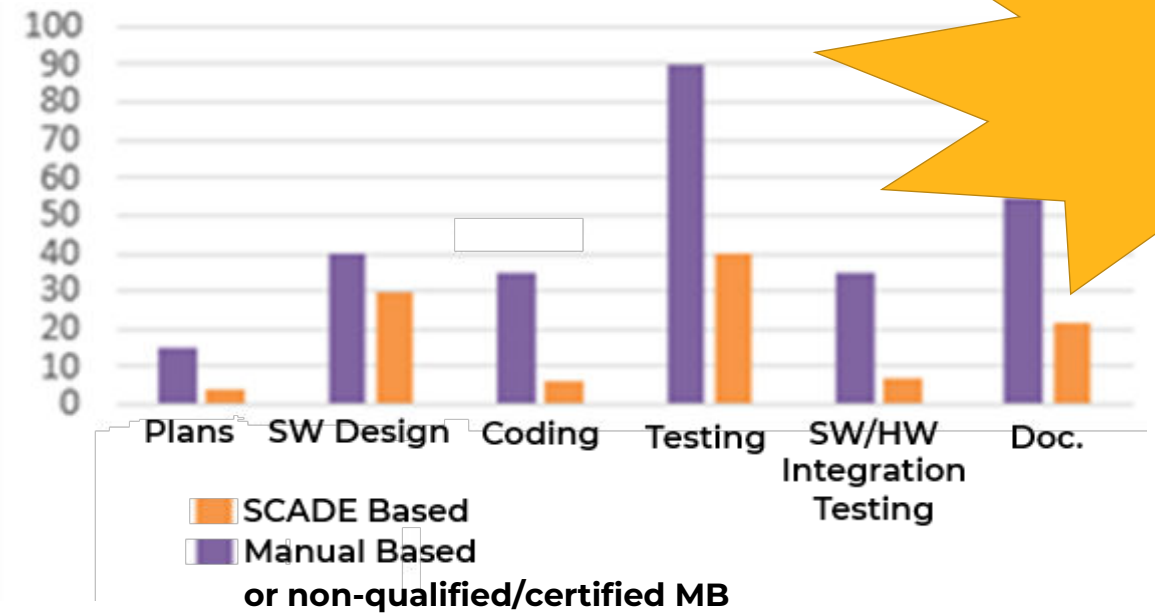
Ansys part of SYNOPSYS®

Benefits of Ansys SCADE Model-Based Design Tools

- **Formal and Deterministic Language**
- **Automatic Generation of Code**
- **Automatic Generation of Documents**
- **Reduce or Elimination of Efforts**
  - Plan
  - Design
  - Coding
  - Testing
  - Documentation

With MBD:
It is easier to design, verify, maintain, evolve, etc.

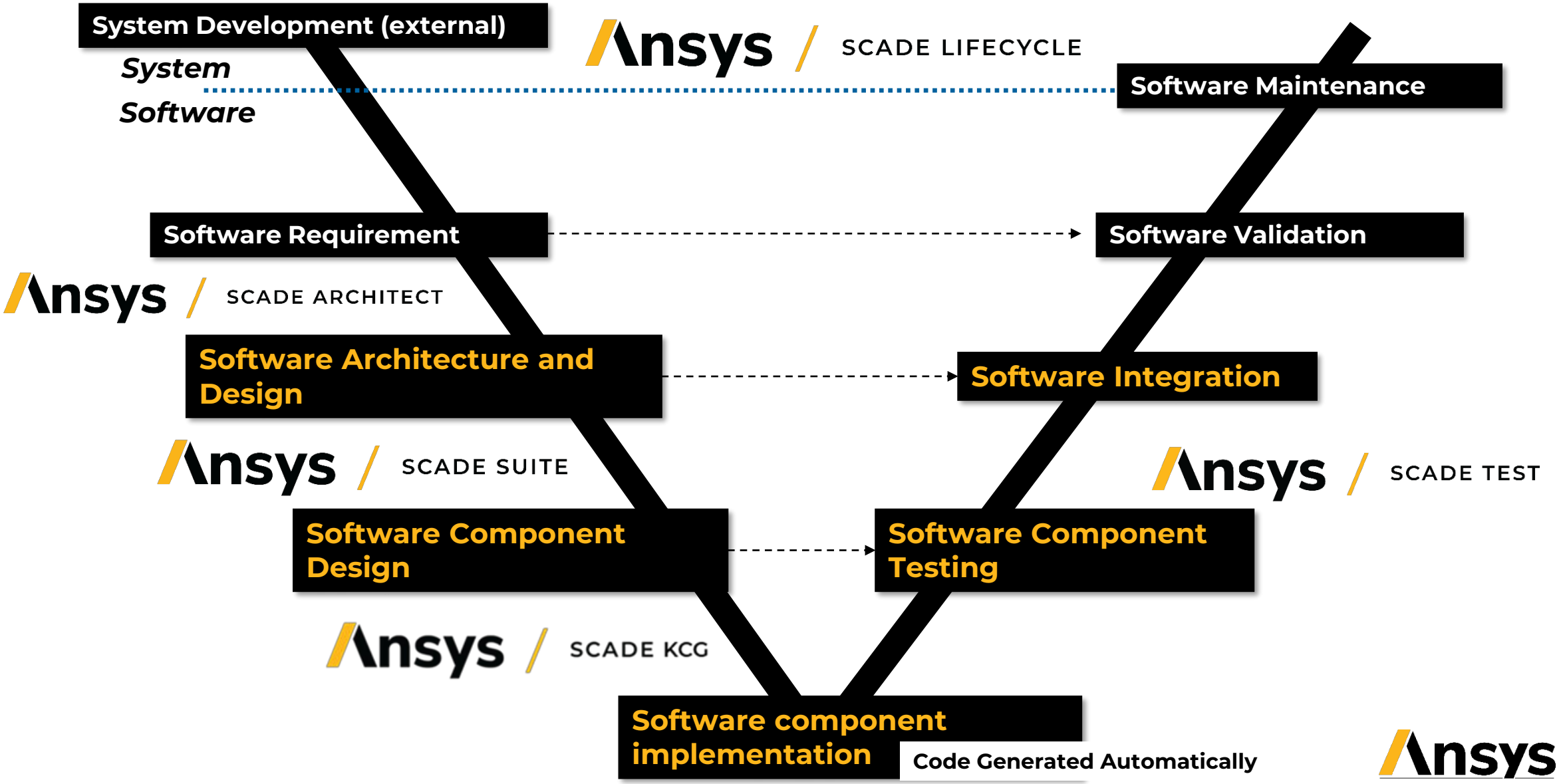
If the MBD tools are Qualified / Certified !



Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

EN-50716:2023 Guidance with SCADE



Ansys, part of Synopsys ©2025

Tool Qualification

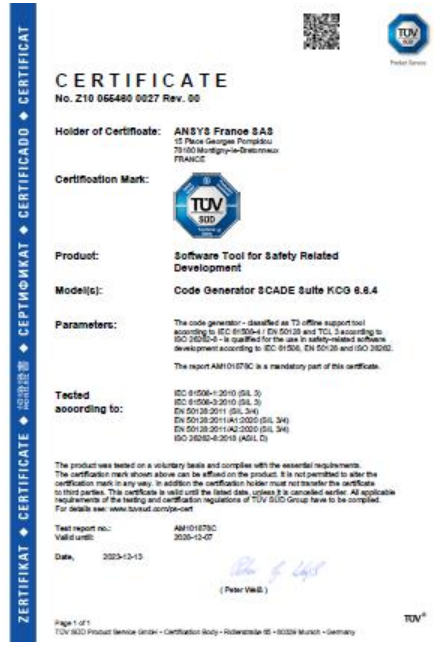
The objective is to provide evidence that potential failures of tools do not adversely affect the integrated toolset output in a safety related manner [6.7.1]

- **Tool class T1** tool which is not used to generate outputs which can directly or indirectly contribute to the executable code (including data) of the software
- **Tool class T2** tool which is used to support the test or verification of the design or executable code (including data), where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software (including data)
  - **SCADE Test on Host** for requirements-based test
  - **SCADE Model Coverage** for code and model coverage
  - **SCADE Harness Generator** for SW-HW integration test
  - **SCADE Lifecycle Reporter** for automatic doc generation
- **Tool class T3** tool which is used to generate outputs which can directly or indirectly contribute to the executable code (including data) of the system
  - **SCADE KCG** for automatic code generation

Ansys KCG Code Generator is classified as **T3** offline support tool according to Railway & Industrial standard, qualified for the use in safety-related software development.

- IEC 61508-1:2010 (SIL 3)
- IEC 61508-3:2010 (SIL 3)
  ISO 26262-8:2018 (ASIL D)
- EN 50128:2011 (SIL 3/4)
- EN 50128:2011/A1:2020 (SIL 3/4)
- EN 50128:2011/A2:2020 (SIL 3/4)
- **EN 50716:2023 (SIL 3/4) ready**

(official TUV' stamp expected these days, by end of 2025 the latest)


CERTIFICATE

No. Z10 066480 0027 Rev. 00

Holder of Certificate: ANSYS France SAS

Certification Mark:

Product: Software Tool for Safety Related Development

Model(s): Code Generator SCADE Suite KCG 6.6.4

Parameters:

Tested according to:

Date, 2025-12-13

Ansys, part of Synopsys ©2025

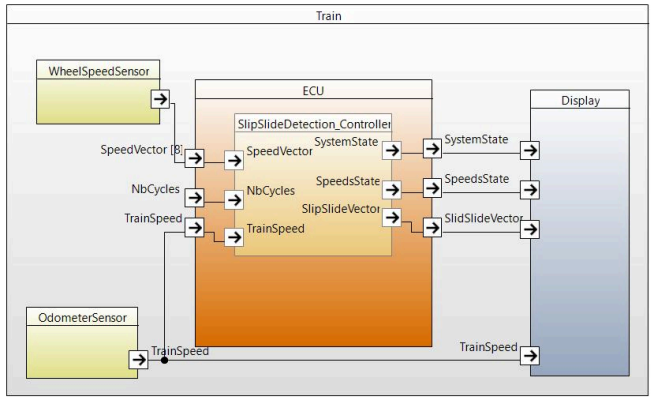
Software Architecture and Design

Clauses 7.3
Guidelines C.2.6.3

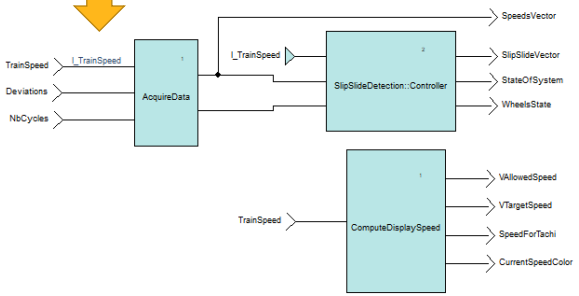
Option #1:

- Synchronization ensures consistency of interfaces and Internal Block Diagrams (IBD)
- between the SCADE Architect Design components and the SCADE Suite operators

Ansys
SCADE Architect



Ansys
SCADE Suite

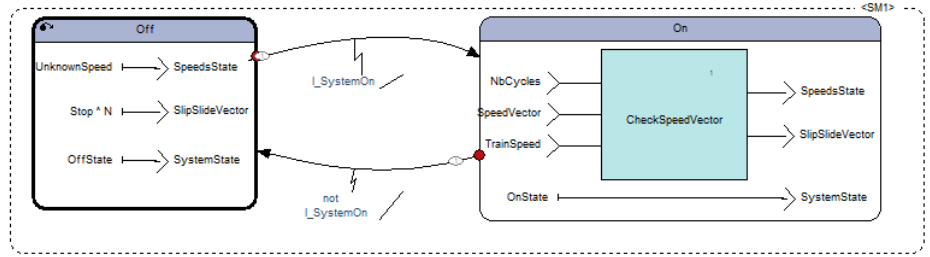


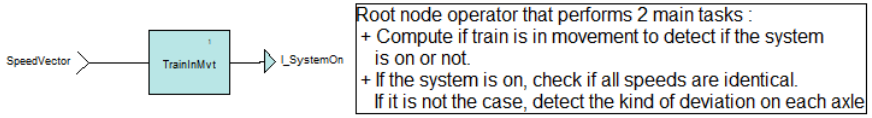
Option #2:

Ansys
SCADE Suite

- Identify high-level functions / components
- Define the component interfaces: names, data types
- Model the data flow and control flow between functions
- Verify the model consistency using SCADE Suite semantic checks
- Prepare the framework for the detailed design process:

define the top-level components while ensuring consistency of their interfaces





Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

Software Architecture and Design

*"Most of the modelling techniques of **Table A.17** can be used for the architecture and design"* **EN 50716 C.2.6.3**

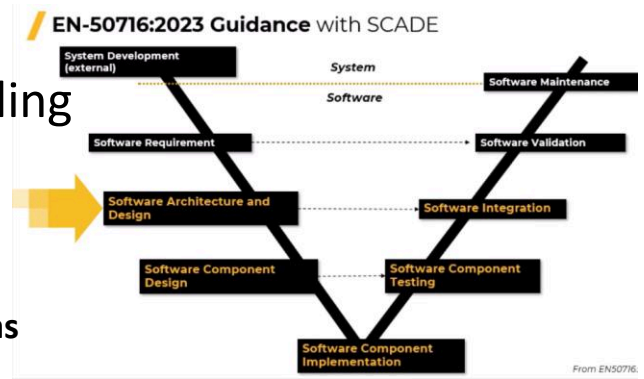
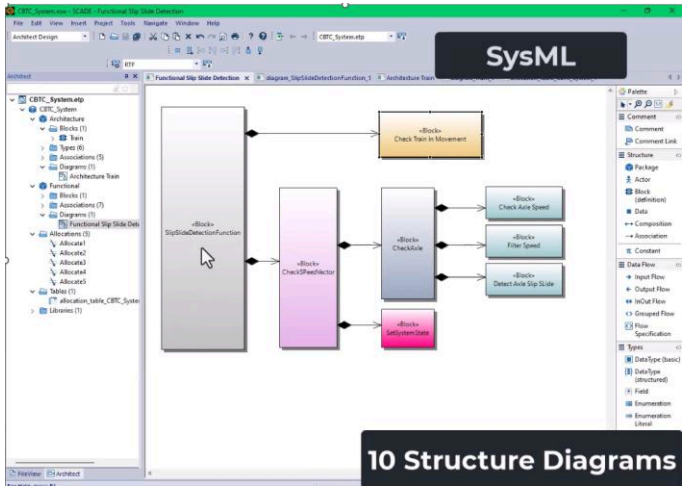
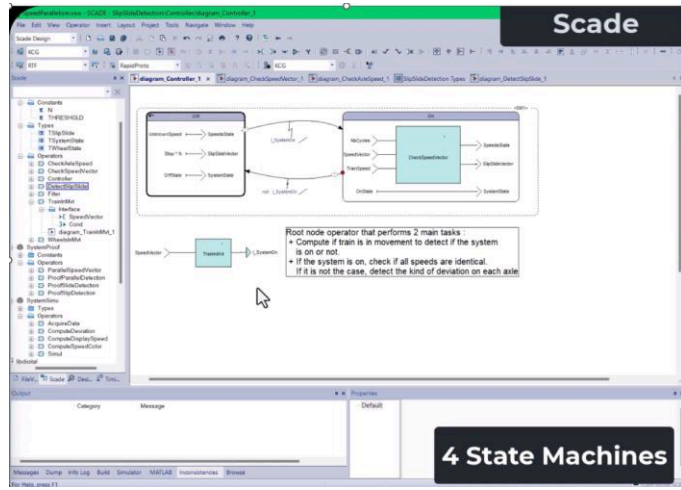
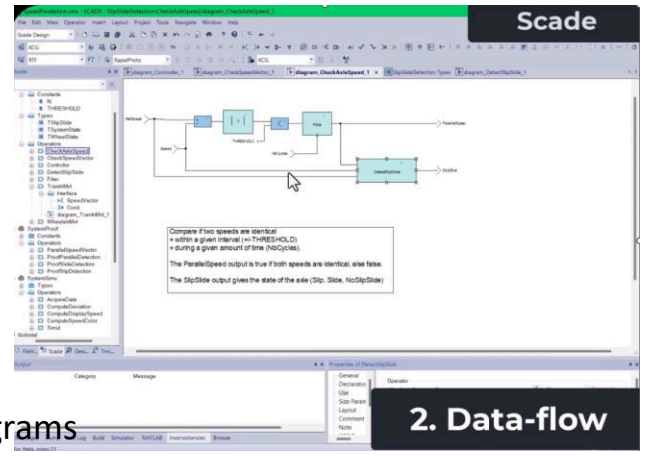


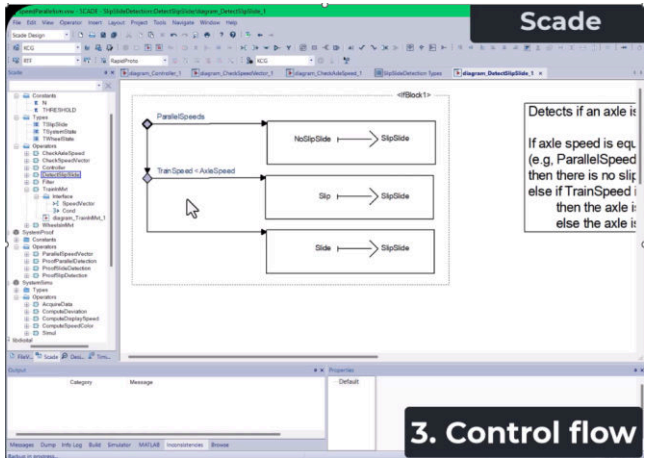
Table A.17 -- Modelling

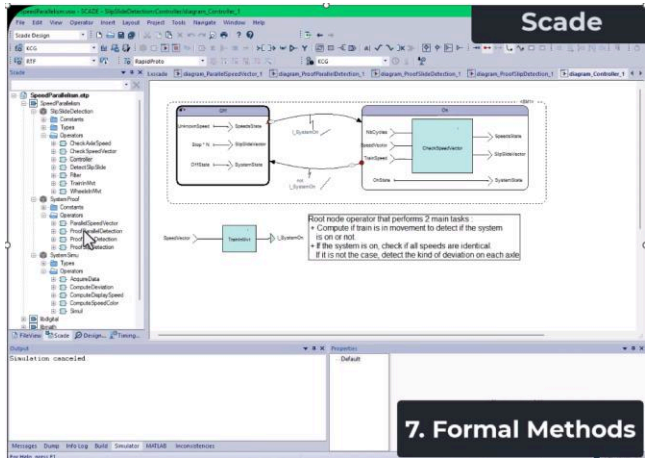
1. **Data Modelling**
2. **Data Flow Diagrams**
3. **Control Flow Diagrams**
4. **Finite State Machines**
5. Petri Nets
6. **Decision/Truth Tables**
7. **Formal Methods**
8. **Performance Modelling**
9. **Prototyping/Animation**
10. **Structure Diagrams**
11. **Sequence Diagrams**
12. Cause Consequence Diagrams
13. Event Tree Diagrams











Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

Software Component Design

"techniques of Table A.4 related to suitable programming languages need to be adapted ***(Table C.1)"*** **EN 50716 C.2.6.3**

Table A.4 (C1) – SW Design

1 **Modelling** (Table A.17)

2 **Structured Methodology**

3 **Modular Approach**

4 **Components** (Table A.20)

- **Encapsulation**
- **Parameter number limit**
- **Fully Defined interface**

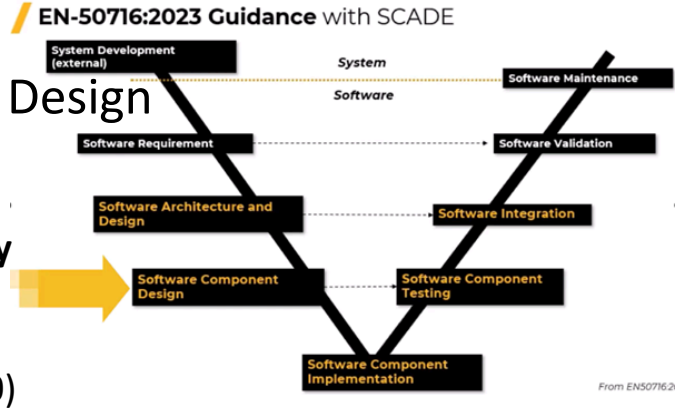
5 **Design Standard** (Table A.12)

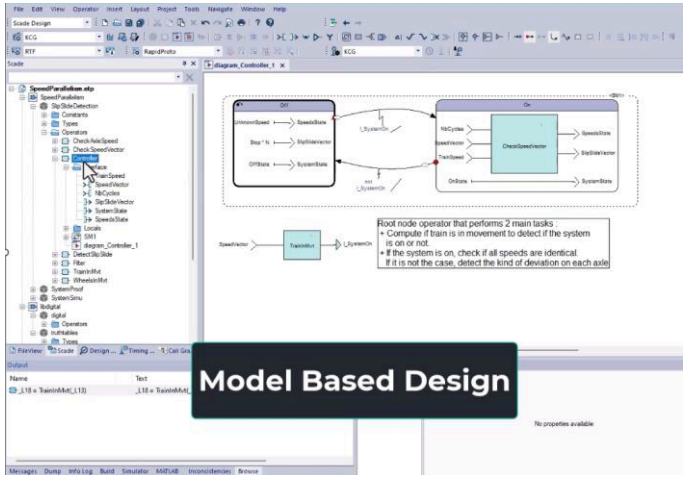
6 **Analyzable programs**

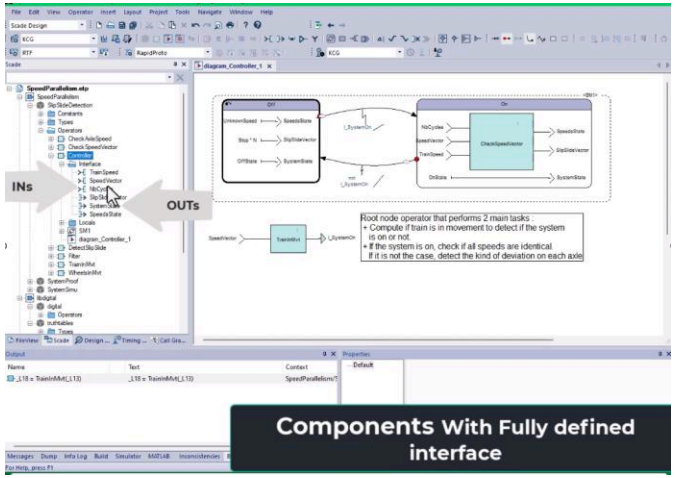
7 **Structured Programming**

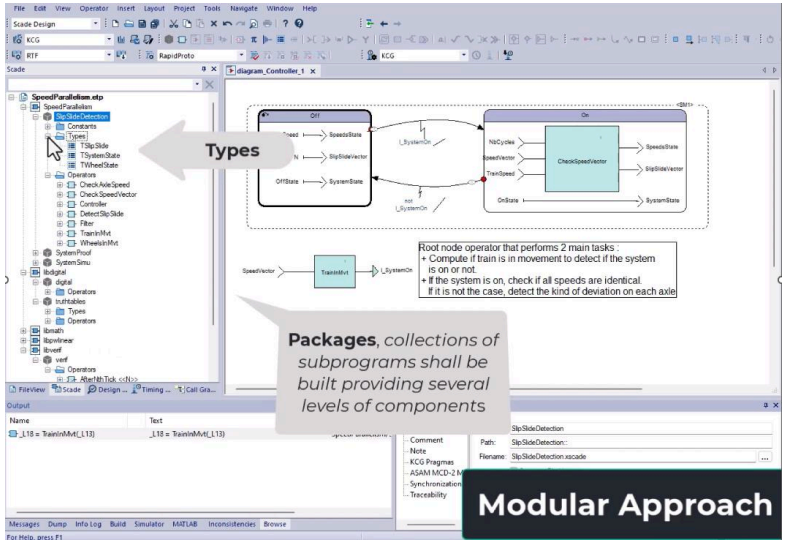
8 **Suitable Programming Languages** (Table A.15)

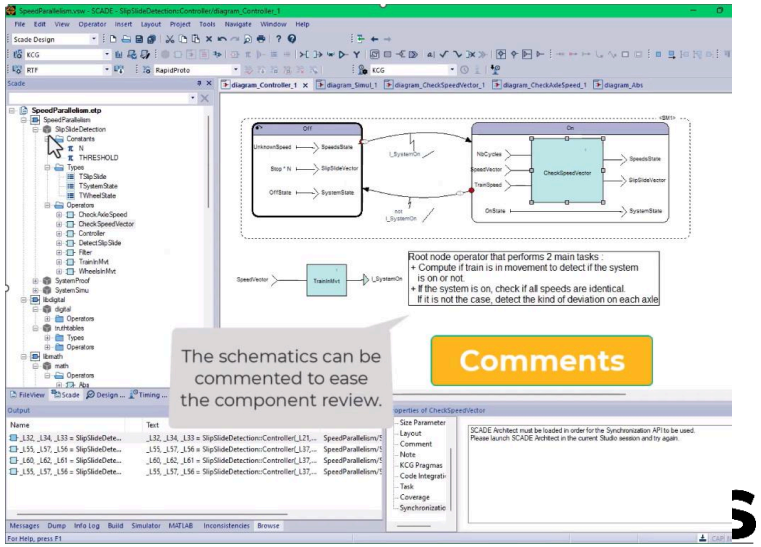
- **Defined operational semantics**
- **Supports commenting**
- **Strong type checking**











Ansys, part of Synopsys ©2025

part of SYNOPSYS®

Design Verification

SW Design Table A.4 (C1)

1 **Modelling** (Table A.17)

2 **Structured Methodology**

3 **Modular Approach**

4 **Components** (Table A.20)

Encapsulation

Parameter number limit

Fully Defined interface

5 **Design Standard** (Table A.12)

6 **Analysable programs**

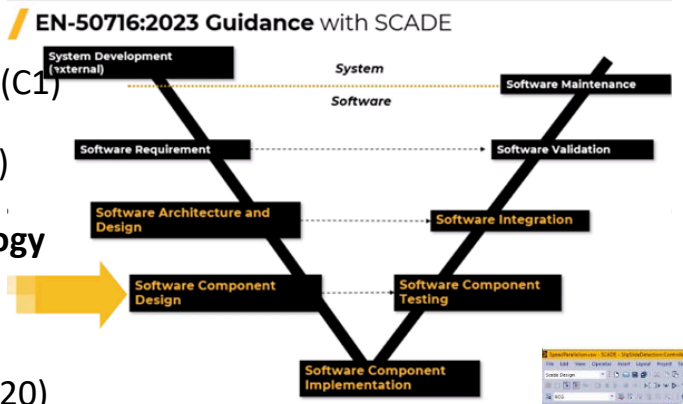
7 **Structured Programming**

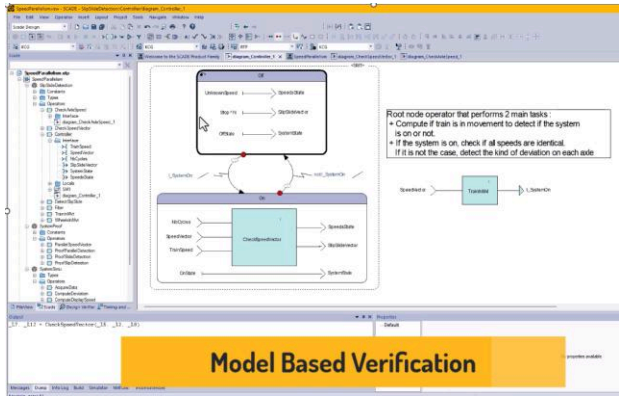
8 **Suitable Programming Languages** (Table A.15)

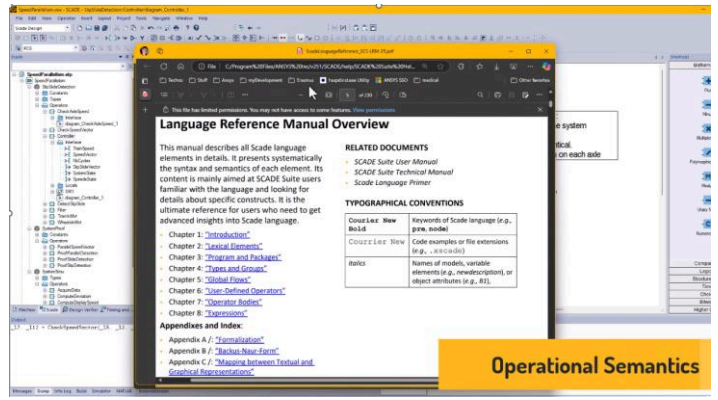
Defined operational semantics

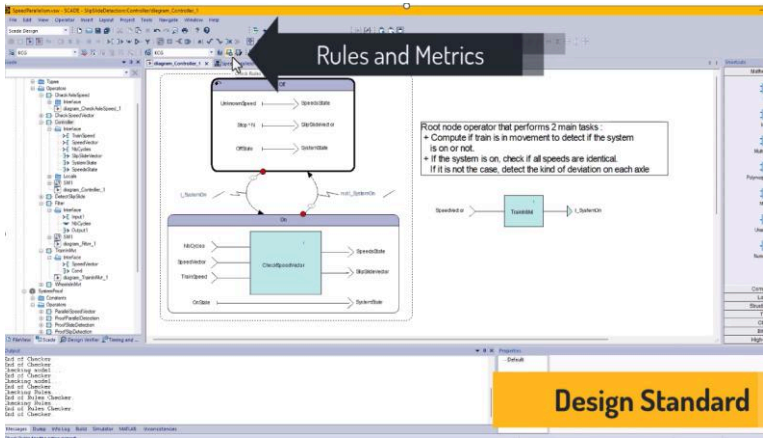
Strong type checking

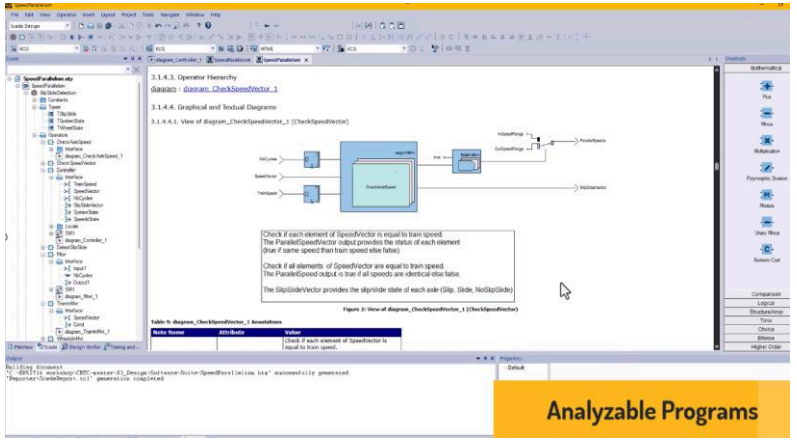
Supports Static Analysis

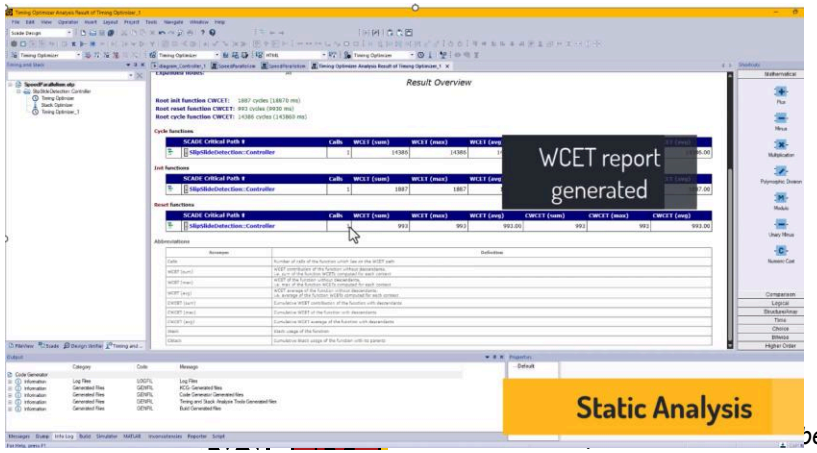


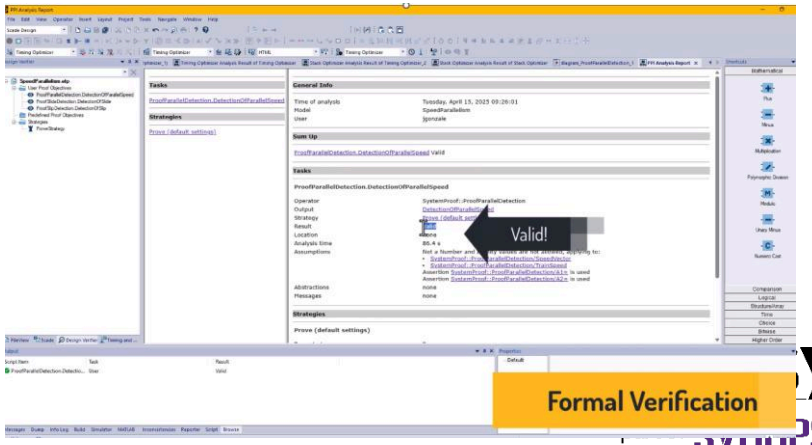










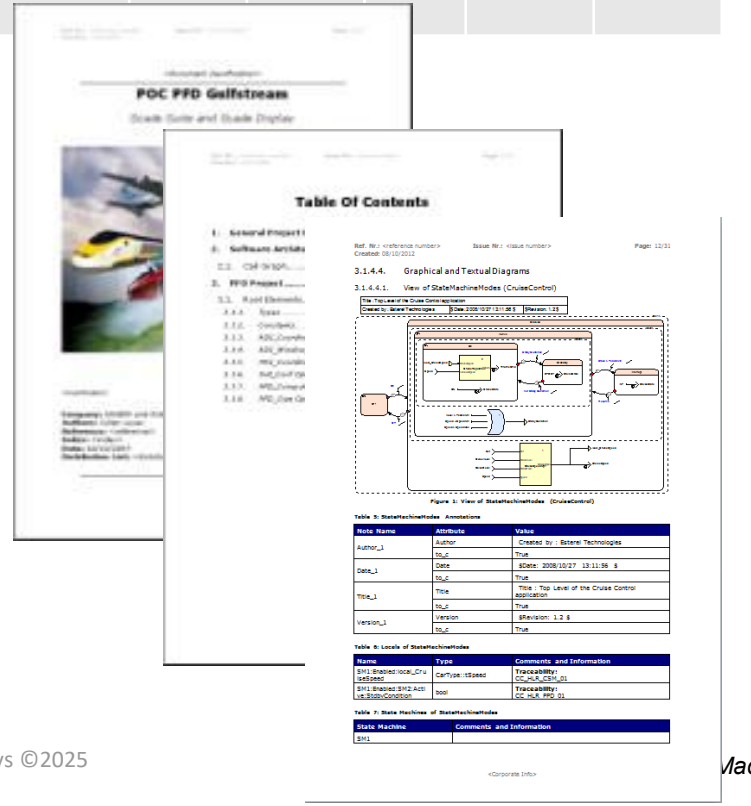


Ansys, part of Synopsys ©2025

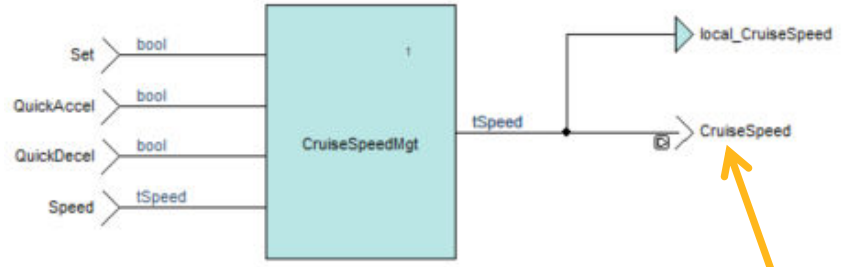
Design Verification Static Analysis Table A.19

TABLE A.19 Technique / measure	Ref	SIL 1	SIL 2	SIL 3	SIL 4
1. Control Flow Analysis	D.8	HR	HR	HR	HR
2. Data Flow Analysis	D.10	HR	HR	HR	HR
3. Software Error Effect Analysis	D.25	R	R	HR	HR
3. Walkthroughs / Design Reviews	D.56	HR	HR	HR	HR

SCADE LifeCycle Reporter automatically generates the Design Review Documentation



SCADE Suite Semantics Checker performs control flow and data flow analysis of SCADE models. It also checks Strong data typing, Initializations, Data dependencies, Cycle detection



Friday January 11 2008 14:30:55

Result of check for operator CruiseControl::CruiseControl/ in model CruiseControl

1 error(s) detected - 0 warning(s) detected

Category	Code	Message
Semantic Error	ERR_100	**Type: Type mismatch at** CruiseControl::CruiseControl/CruiseSpeed/ This expression has type (real) but is here used with type (bool)

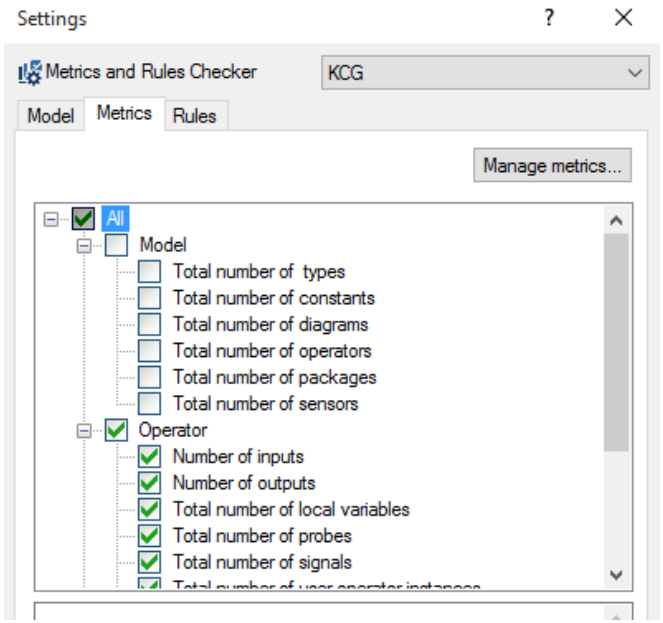
Ansys, part of Synopsys ©2025

Design Verification Metrics and Design Standard Rules & Formal Proof

Checker: check models against:

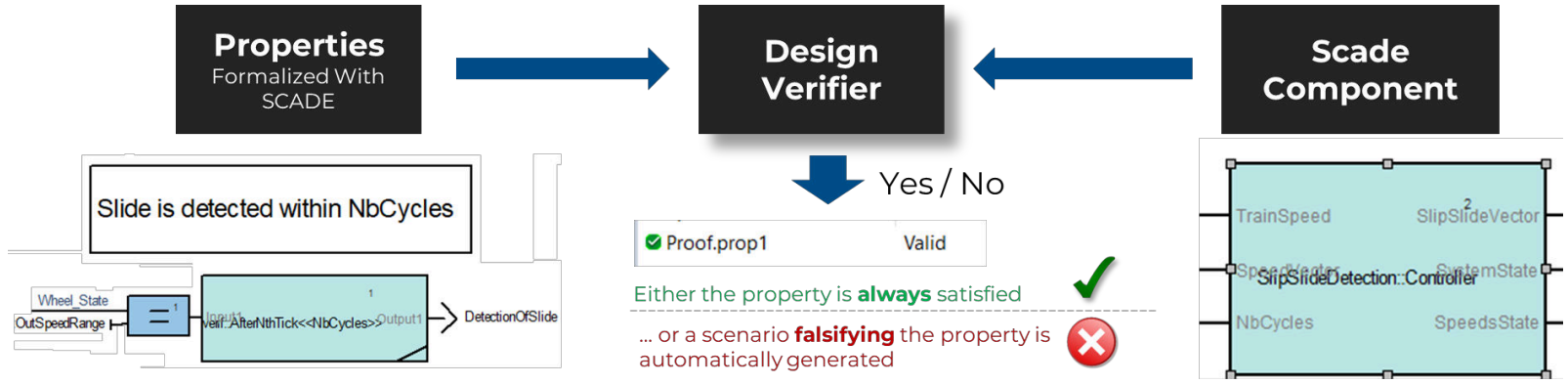
- **Design Standard**
  . Find non-compliances to the **modelling standard**
  . A **full flexible** Python **framework**
  . **Extensible**, with dedicated report

- **Metrics**: complexity analysis
  . **Get numbers** on your models
  . Get high-level metrics
  . **Derive your own analyses**



SCADE Suite Design Verifier (DV) is a **Formal Verification tool powered by Prover**

- Identify **arithmetic issues** such as division by zero, overflows, Infinite or NaN values
- Formally express and assess **safety properties**
- **Verify equivalence** between two design choices/implementation
- Analyze **model coverability** & **generate testing sequences** covering uncovered parts



Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

Software Component Implementation

EN 50716:2023 Annex C.2.6.5

"When using models at software component level, the subclauses applicable to the source code can be contextualized in similar subclauses applicable to models (see Table C.2)"

*"If the source code is automatically generated and neither modified after generation nor input of analysis, **it is no longer necessary to apply the subclauses listed in the table C.2**, as 6.7 would apply for the automatic code generation tool"*

SCADE KCG generated code, does not need to be modified nor analyzed.

Table C.2 — Component implementation and testing typical adaptation for modelling

SUBCLAUSES	TYPICAL ADAPTATION FOR MODELLING
7.5.4.2 The size and complexity of the developed source code shall be balanced.	7.5.4.2 The size and complexity of the developed model shall be balanced.
7.5.4.3 The Software Source Code shall be readable, understandable and testable.	7.5.4.3 The model shall be readable, understandable and testable.
7.5.4.4 The Software Source Code shall be placed under configuration control before the commencement of documented testing.	7.5.4.4 The model shall be placed under configuration control before the commencement of documented testing.
7.5.4.10 After the Software Source Code and the Software Component Test Report have been established, verification shall address a) the adequacy of the Software Source Code as an implementation of the Software Component Design Specification, b) the correct use of the chosen techniques and measures from Table A.4 as a set satisfying 4.8 and 4.9, c) determining the correct application of the coding standards, d) that the Software Source Code meets the general requirements for readability and traceability in 5.3.2.7 to 5.3.2.10 and in 6.5.4.14 to 6.5.4.17, as well as the specific requirements in 7.5.4.1 to 7.5.4.4, e) the adequacy of the Software Component Test Report as a record of the tests carried out in accordance with the Software Component Test Specification	7.5.4.10 After the model and the Software Component Test Report have been established, verification shall address a) the adequacy of the model as an implementation of the Software Component Design Specification, b) the correct use of the chosen techniques and measures from Table A.4 as a set satisfying 4.8 and 4.9, c) determining the correct application of the modelling standards, d) that the model meets the general requirements for readability and traceability in 5.3.2.7 to 5.3.2.10 and in 6.5.4.14 to 6.5.4.17, as well as the specific requirements in 7.5.4.1 to 7.5.4.4, e) the adequacy of the Software Component Test Report as a record of the tests carried out in accordance with the Software Component Test Specification

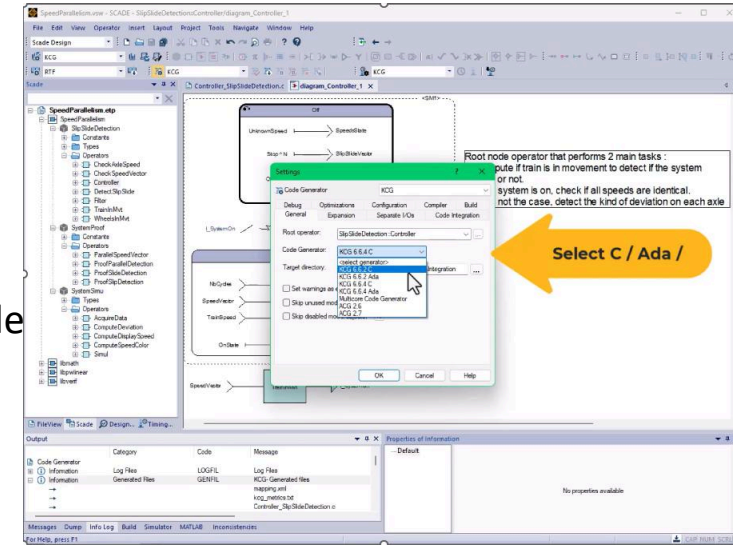
Ansys, part of Synopsys ©2025

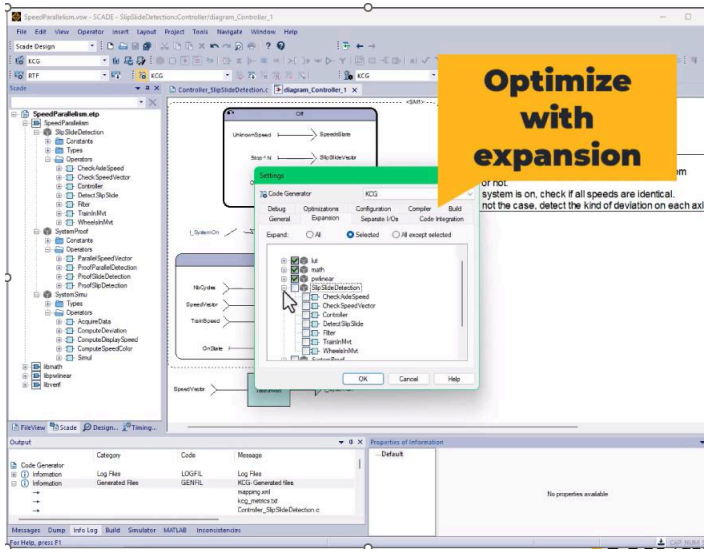
part of SYNOPSYS®

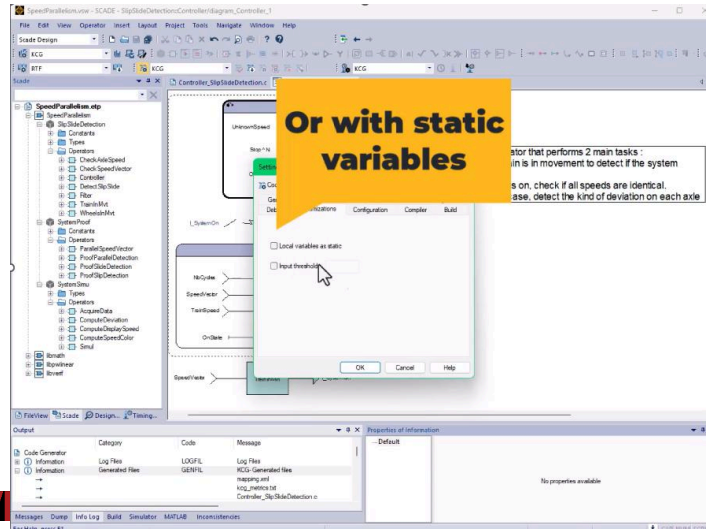
Ansys KCG Suite Qualified Code Generator

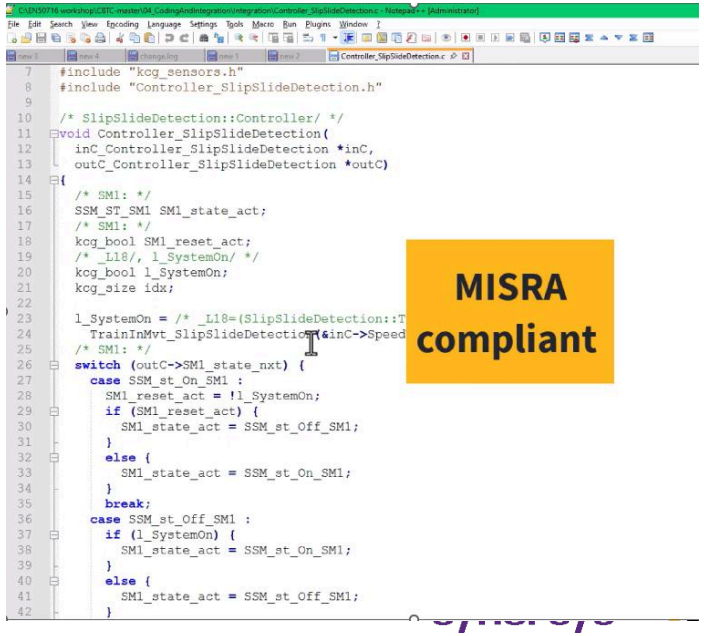
Qualified at SIL3 / SIL4 => There is no need to verify the generated C code

- Full support of Scade formal language
- KCG generates code that satisfies the constraints of safety-critical embedded software:
  - **Portable** (target and OS independent)
  - **Readable** and **traceable** with respect to the de (name / annotation propagation)
  - **Optimized** code for all constructs
  - **Structured**
  - **Static** memory allocation
  - **No pointer arithmetic**
  - **No recursion**, bounded loops only
  - **Bounded execution** time
- Standard ANSI C code generation:
  - **MISRA C** compliant (2012)
  - **SEI CERT-C** compliant (2016)
  - Compatible with standard C compilers
- Limited size/complexity/parameters:
  - SCADE Suite Rule Checkers to check these rules









Ansys, part of Synopsys ©2025

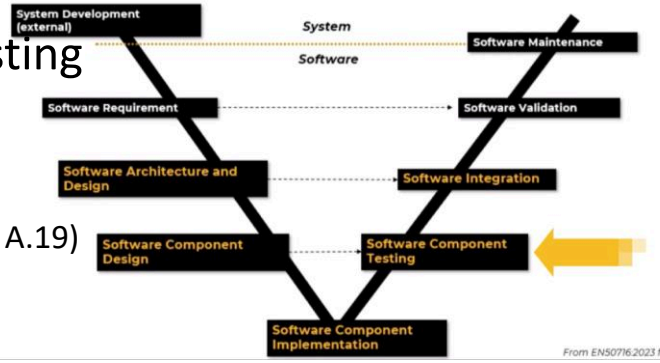
Software Component Analysis and Testing

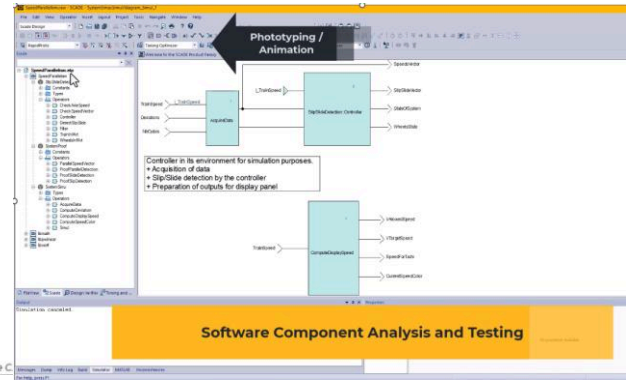
"Techniques for software component analysis and testing of Table A.5 *are directly applicable as-is to modelling"*

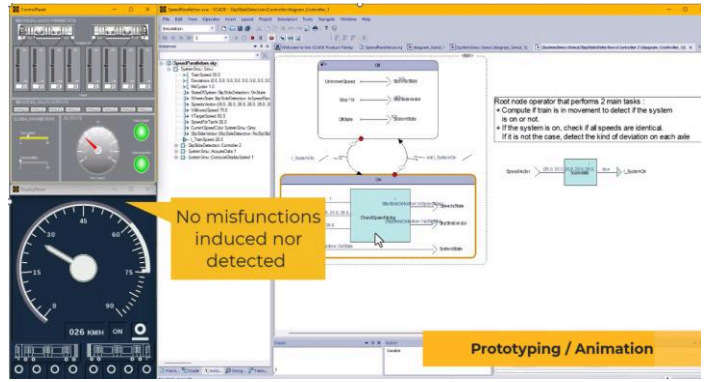
EN 50716 C.2.6.5 / **EN-50716:2023 Guidance** with SCADE

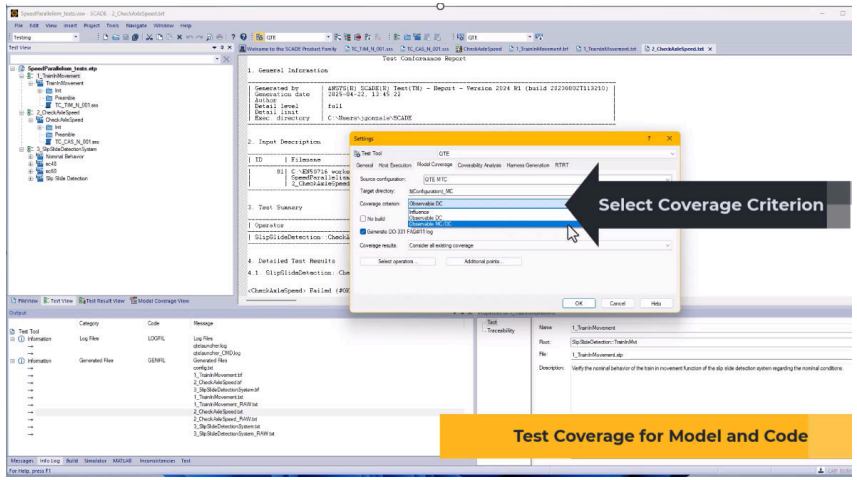
SW Analysis and Testing
Table A.5

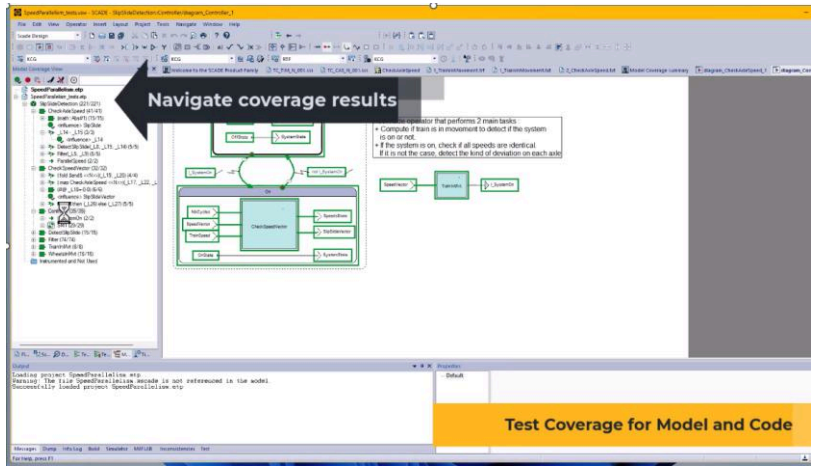
1. **Formal proof**
2. **Static Analysis** (Table A.19)
   - **Control Flow Analysis**
   - **Data Flow Analysis**
   - **Walkthroughs / Design review**
3. **Dynamic Analysis and Testing** (Table A.13)
   - **Prototyping / Animation**
   - **Test Case Execution Boundary value**
   - **Equivalent Classes and Input**
   - **Partition Testing**
4. **Metrics**
5. **Test coverage for Code** (Table A.21)
   Statement, Branch, Data Flow
6. **Performance Testing** (Table A.18)
   - **Response Timing and Memory**
   - **Performance Requirements**











Ansys, part of Synopsys ©2025

Ansys part of SYNOPSYS®

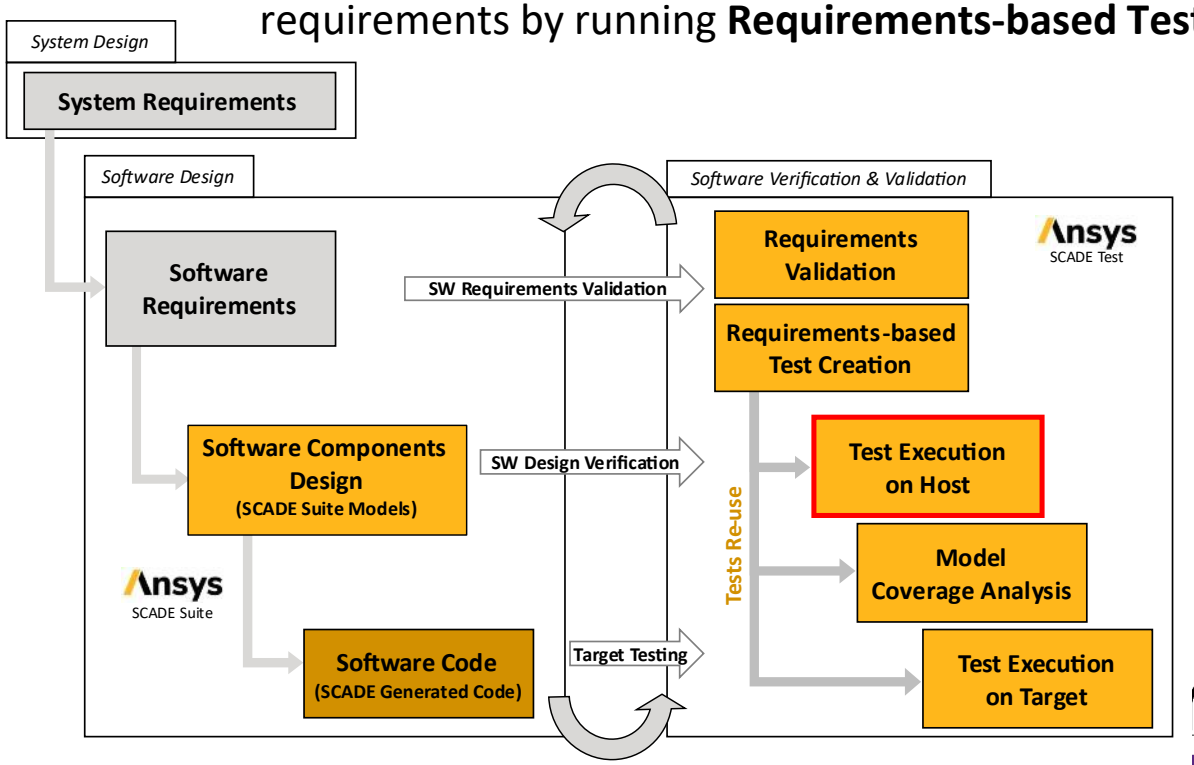
Software Component Analysis and Testing

Dynamic Analysis and Testing **Table A.13**

*Techniques for Models supporting testing activities can be used to simulate the environment of the software under test (i.e., **model-in-the-loop**)* **EN 50716 Annex C.2.4.2 Software Testing**

TABLE A.13 Technique / measure	Ref	SIL 1	SIL 2	SIL 3	SIL 4
1. Test Case Execution from Error Guessing	D.20	R	R	R	R
2. Test Case Execution from Error seeding	D.21	R	R	R	R
3. Structure-Based Testing	D.50	R	R	HR	HR
4. Test Case Execution from Cause Consequence Diagrams	D.6	-	-	R	R
5. Prototyping / Animation	D.43	-	-	R	R
6. Test Case Execution from Boundary Value	D.4	HR	HR	HR	HR
7. Equivalence Classes and Input Partition Testing	D.18	HR	HR	HR	HR
8. Process Simulation	D.42	R	R	R	R

MiL allows to exercise the behavior of the model to provide repeatable evidence of compliance of the model to the software requirements by running **Requirements-based Testing**



Ansys, part of Synopsys ©2025

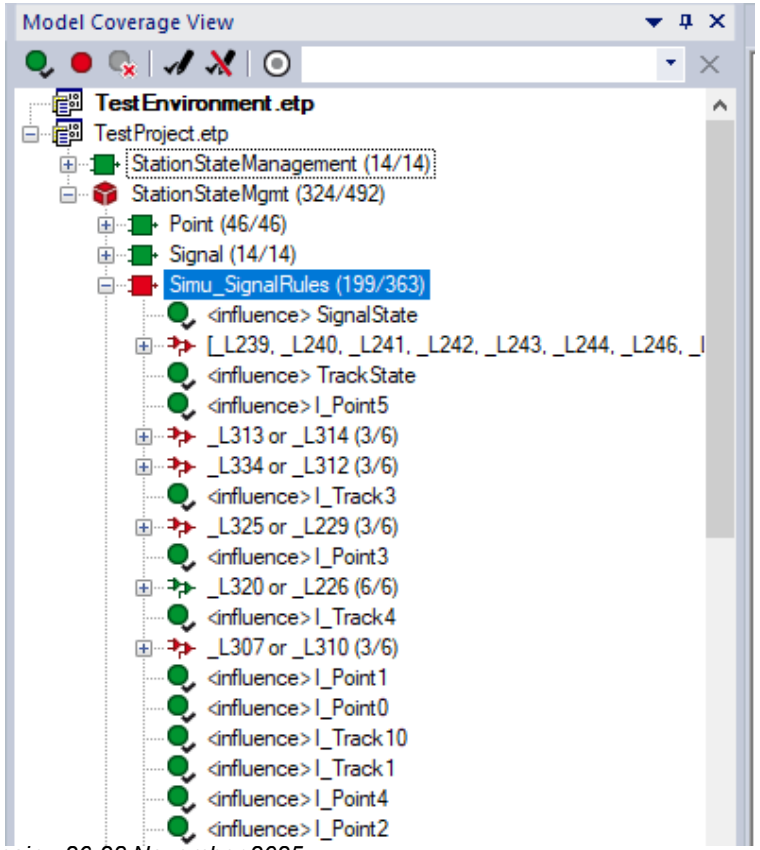
Software Component Analysis and Testing

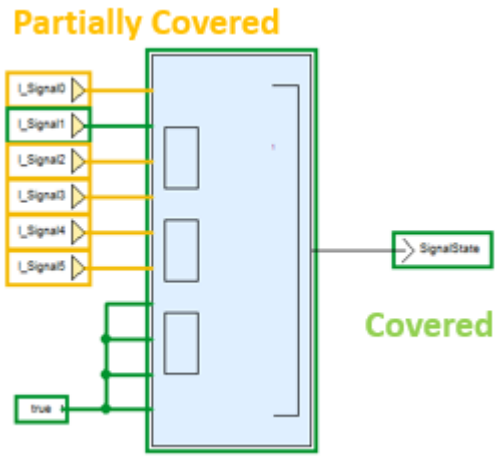
Model Coverage

*"techniques of **Table A.21** are very specific to imperative programming languages. Alternative coverage criteria need then to be defined depending on the modelling notation used with the objective to cover all parts of the model (e.g. components, interfaces, data flow, control flow) with the same level as with programming languages"* **EN 50716 C.2.6.5**

SCADE Model Coverage Criterion	Corresponding criterion sense for code coverage A.21
Influence	Statement Coverage
Observable Decision Coverage (ODC)	Decision Coverage
Observable Modified Condition / Decision Coverage (OMC/DC)	Modified Condition / Decision Coverage (MC/DC)

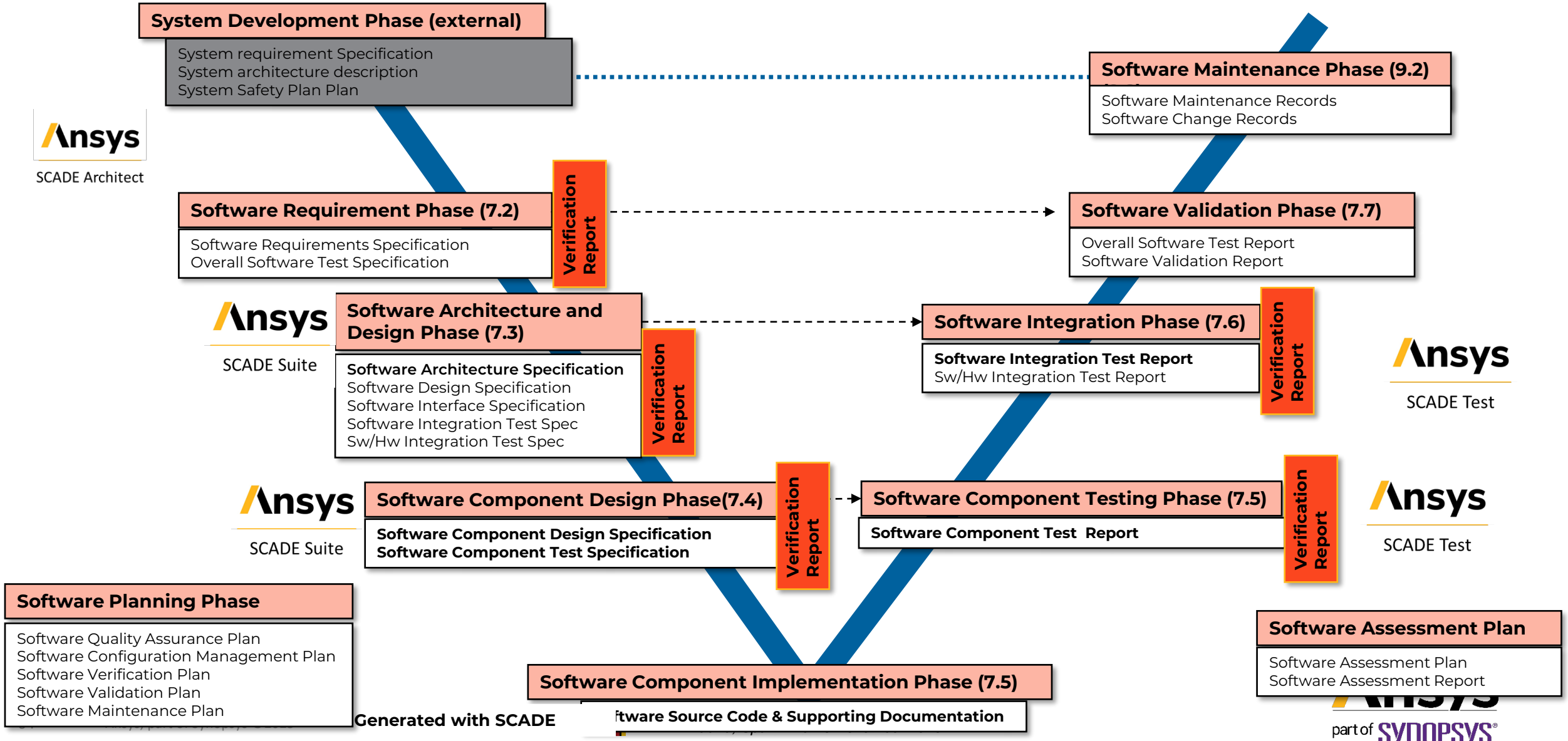
Benefit: **Once 100% model coverage** is achieved with SCADE Test Model Coverage and a given criterion, user can claim **100% code coverage** of the SCADE Suite KCG generated code in corresponding criterion sense



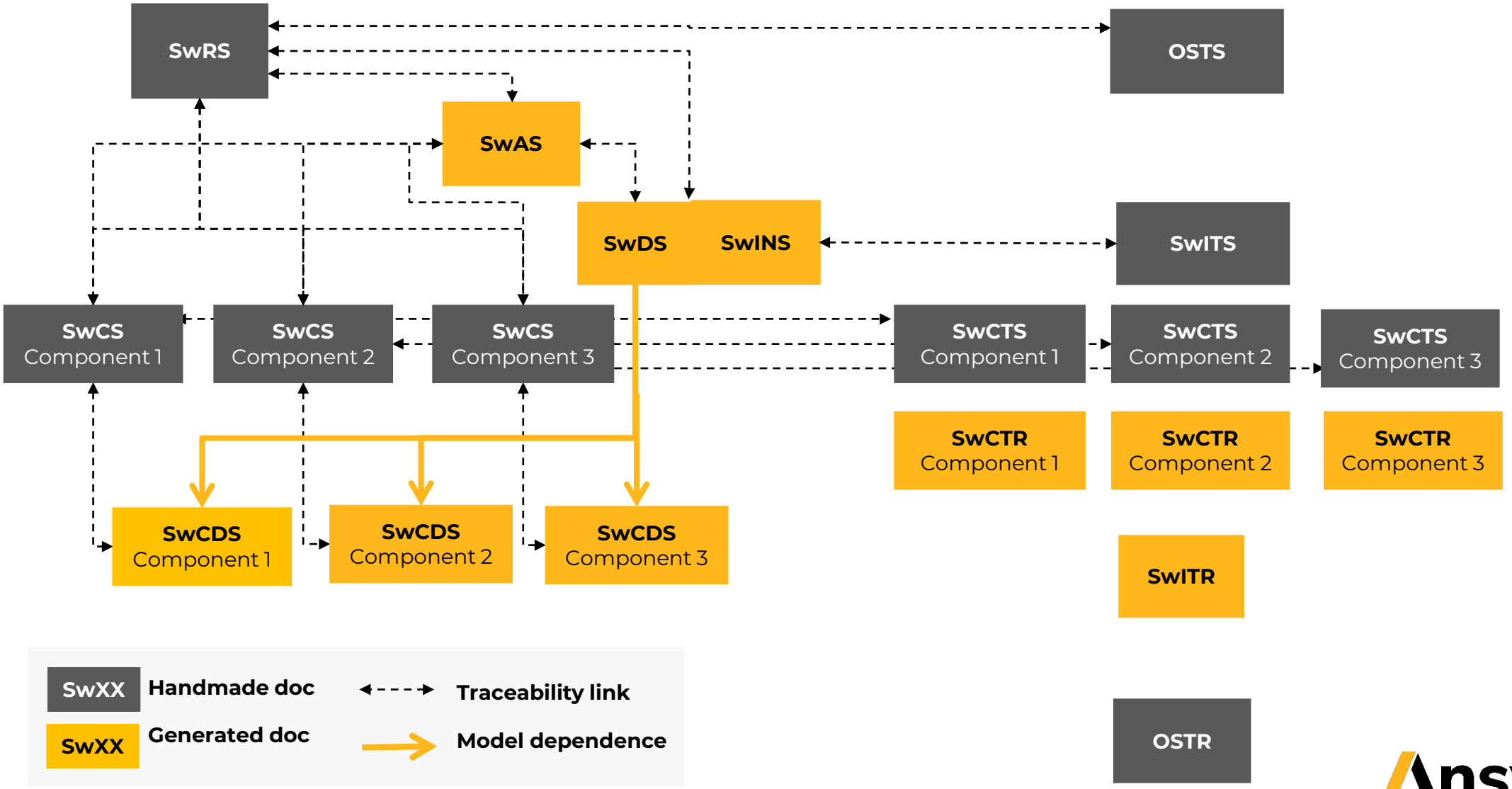


Ansys, part of Synopsys ©2025

EN-50716:2023 V-cycle with Ansys SCADE LifeCycle & Documentation *From EN50716:2023 figure C.2*



Documentation with SCADE



Ansys, part of Synopsys ©2025

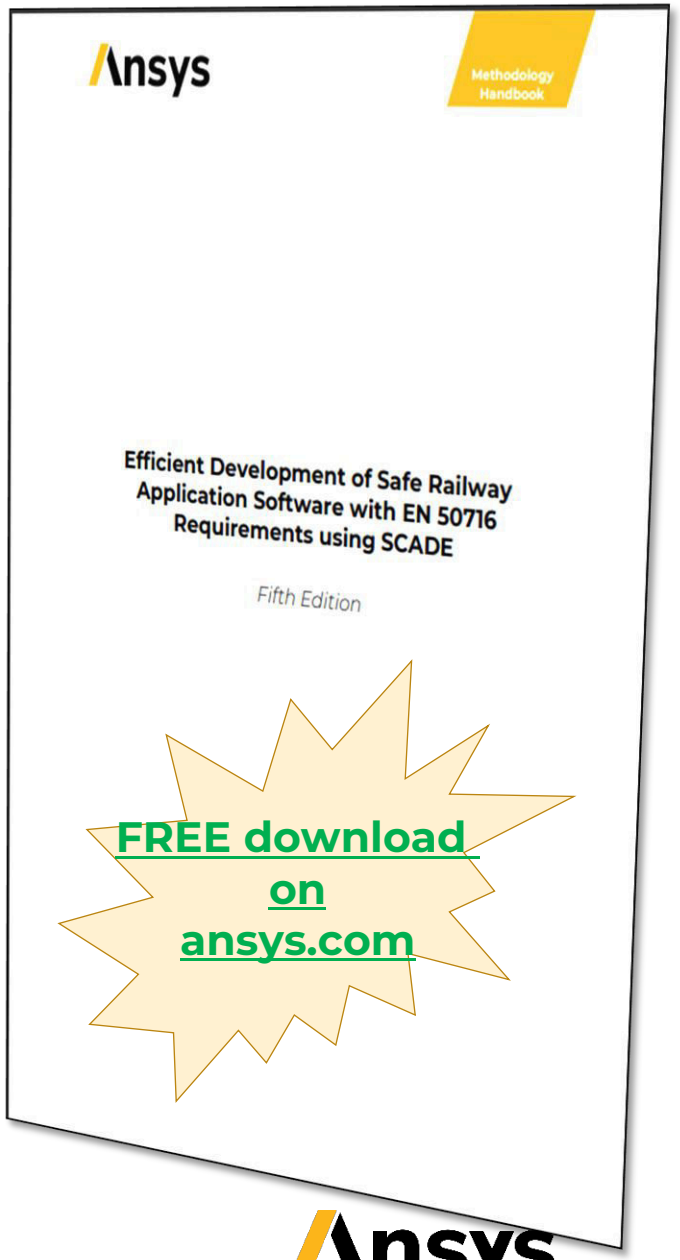
EN 50716 Methodology Handbook

Efficient Development of Safe Railway Application Software with EN 50716 Requirements using SCADE

This methodology handbook provides detailed explanations on how to fully satisfy requirements of EN 50716 with a SCADE model-based development approach to achieve safe and reliable software, while promoting an efficient model-based development and verification strategy

- Model-based development with **SCADE Suite**
- Simulation and Model Coverage
- Formal verification
- Automatic code generation with KCG

Set of guidelines for developing efficient models, generating efficient code, setting up efficient V&V strategy, etc



Ansys, part of Synopsys ©2025

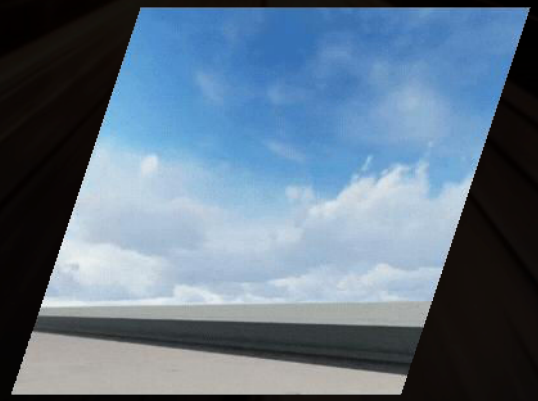
Ansys part of SYNOPSYS®

What you'd love about Ansys SCADE for EN 50716

Develop Safe & Reliable Embedded Software

Reduce Development Time & Costs

Secure your Certification journey



Ansys

part of SYNOPSYS®

RAILLIVE!

IFEMA Madrid, Spain - 26-28 November 2025

Bruno Privat - System & Mission Senior MBSE Product Executive - *bruno.privat@ansys.com*