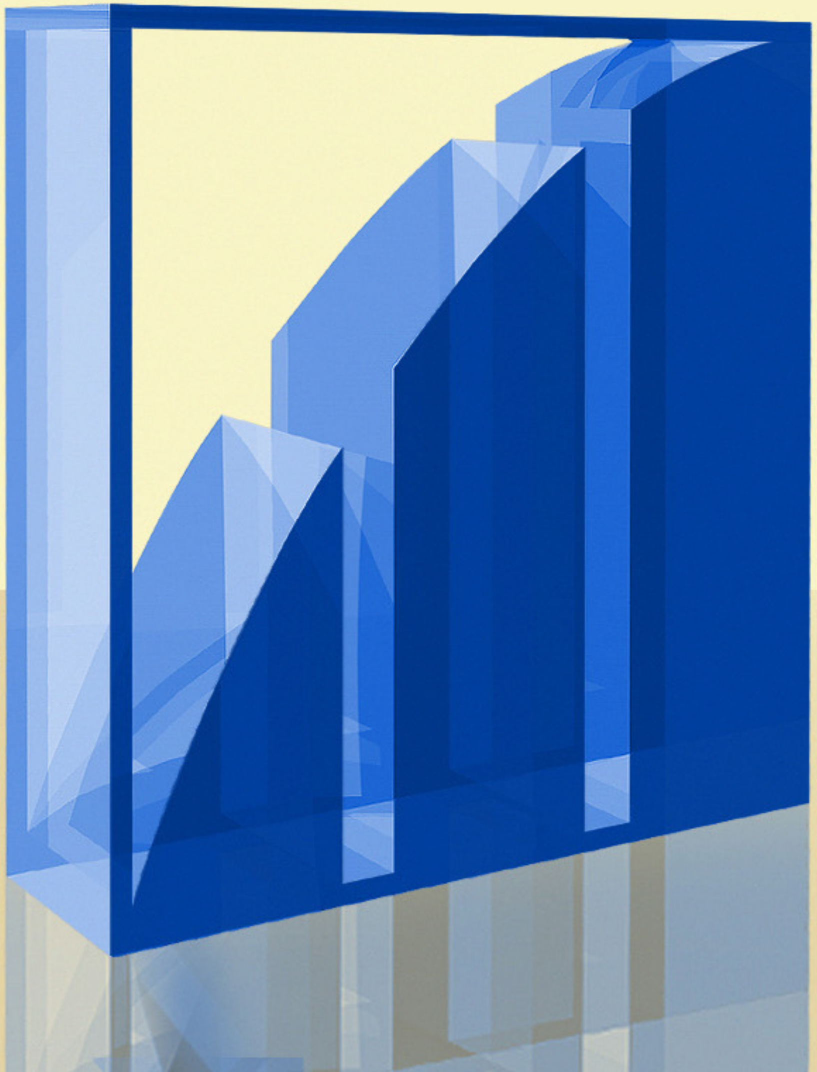
ISO 26262 Conform Model Based Development and Verification Process



dSPACE User Conference
India 2010

◆

Adrian Valea
BTC Embedded Systems AG

BTC Embedded Systems

Agenda

- **Introduction**
- **Theoretical aspects of the ISO 26262 standard, its terminology, methodology and mapping**
  - ISO 26262 – New Functional Safety Standard
  - Enhanced Model Based Development and Testing
  - Model-Based Reference Workflow
  - Modeling and Coding Guidelines
  - Formal Specifications and Formal Verification
  - Automatic Test Generation and Execution
  - Requirements Based Testing and Traceability
  - Qualification of software tools in the context of ISO 26262
- **Conclusions**

BTC Embedded Systems AG proprietary · All rights reserved

BTC Embedded Systems AG

- **OSC GmbH Company established in 1999**
- **OSC – Embedded Systems AG founded in 2002**
- **Beginning 2009 OSC became BTC-ES**
  - as part of BTC AG Corporation with1400 Employees
- **BTC-ES Headquarter in Oldenburg (D)**
  - Subsidiary in Munich (D)
  - BTC Japan Co., Ltd.
- **Expert in Automatic Test- and Validation Technologies**

dSPACE **Strategic Partner provider of Automatic Test and Verification Products for TargetLink**

- Common Activities especially together with dSPACE GER/JP/FRA

All rights reserved

ISO 26262 – New Functional Safety Standard

- New Automotive Standard addressing functional safety
- Derived from IEC 61508
- Draft International Standard (DIS) published in July 2009
- Official release planned for 2011
- But already used by OEMs and suppliers

BTC Embedded Systems AG proprietary · All rights reserved

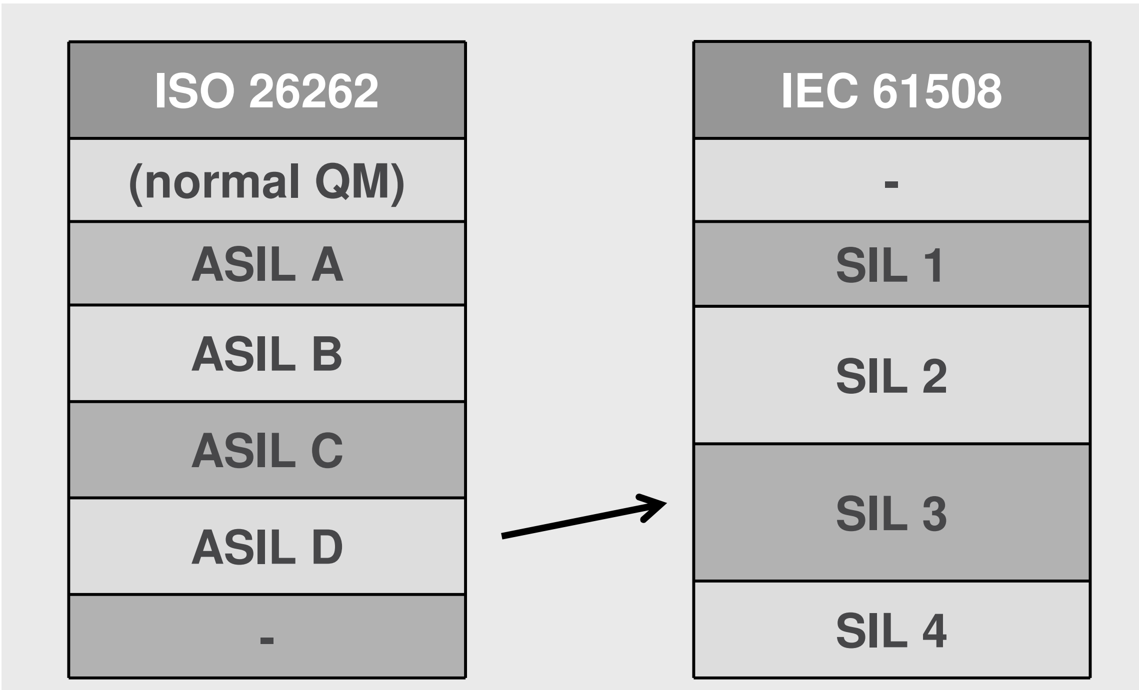
BTC Embedded Systems

ISO 26262 – Automotive Safety Integrity Levels

- ISO 26262 defines four Automotive Safety Integrity Levels (ASIL)
- Definition of ASIL: one class to specify the necessary safety requirements items for achieving an acceptable residual risk with D representing the highest and A the lowest class.

International Organization for Standardization

ISO 26262



Functional safety of electrical/electronic/programmable electronic safety-related systems

IEC 61508

BTC Embedded Systems AG proprietary · All rights reserved

ISO 26262 – Model-based Development

- ISO 26262 specifically addresses model-based development and testing

International Organization for Standardization

ISO 26262

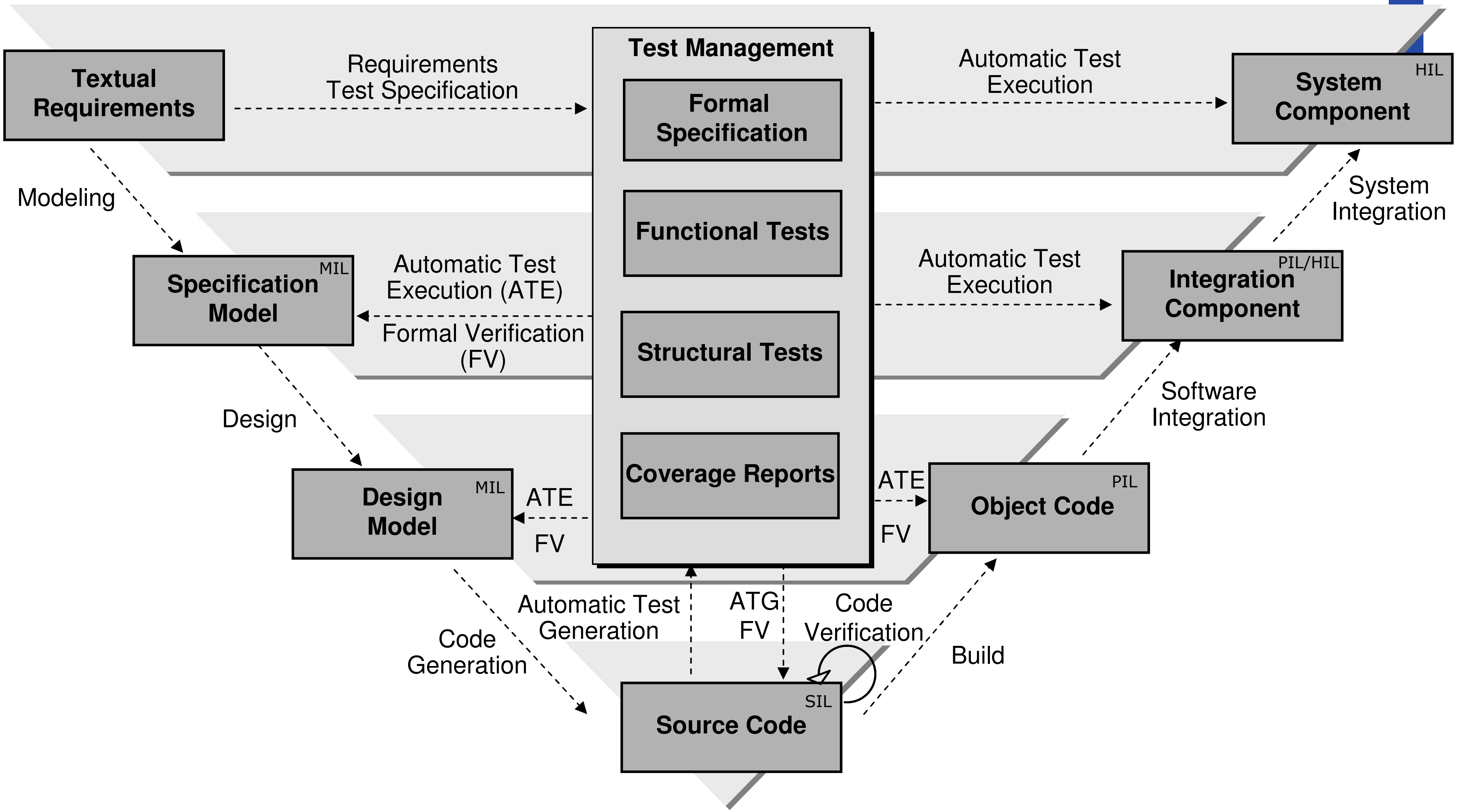
One characteristic of the model-based development paradigm is the fact that the functional model not only specifies the desired function but also provides design information and finally even serves as the basis of the implementation by means of code generation.

…

In contrast to code-based software development with a clear separation of phases in model-based development a stronger coalescence of the phases Software Safety requirements, Software Architectural Design, and Software unit design and implementation can be noted. Moreover, one and the same graphical modeling notation is used during the consecutive development stages. Testing activities are also treated differently since models can be used as a useful source of information for the testing process (model-based testing). The seamless utilization of models facilitates a highly consistent and efficient development.

BTC Embedded Systems AG proprietary · All rights reserved

Enhanced Model Based Development and Testing Process

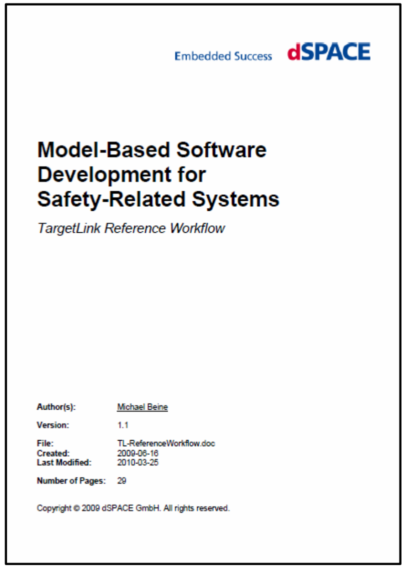


BTC Embedded Systems AG proprietary · All rights reserved

BTC Embedded Systems

Model-Based Reference Workflow

- Well suited to develop safety-related software according to ISO 26262 and IEC 61508
- Many of the proposed methods are directly recommended by ISO 26262 and IEC 61508

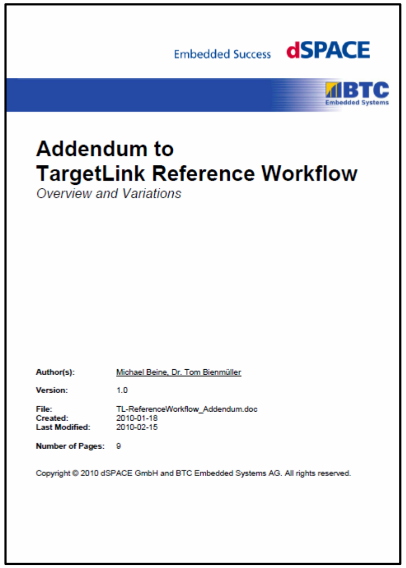

Embedded Success dSPACE

Model-Based Software Development for Safety-Related Systems

TargetLink Reference Workflow

Author(s): Michael Beine
Version: 1.1
File: TL-ReferenceWorkflow.doc
Created: 2009-06-16
Last Modified: 2010-03-25
Number of Pages: 29

Copyright © 2009 dSPACE GmbH. All rights reserved.


Embedded Success dSPACE

BTC Embedded Systems

Addendum to TargetLink Reference Workflow

Overview and Variations

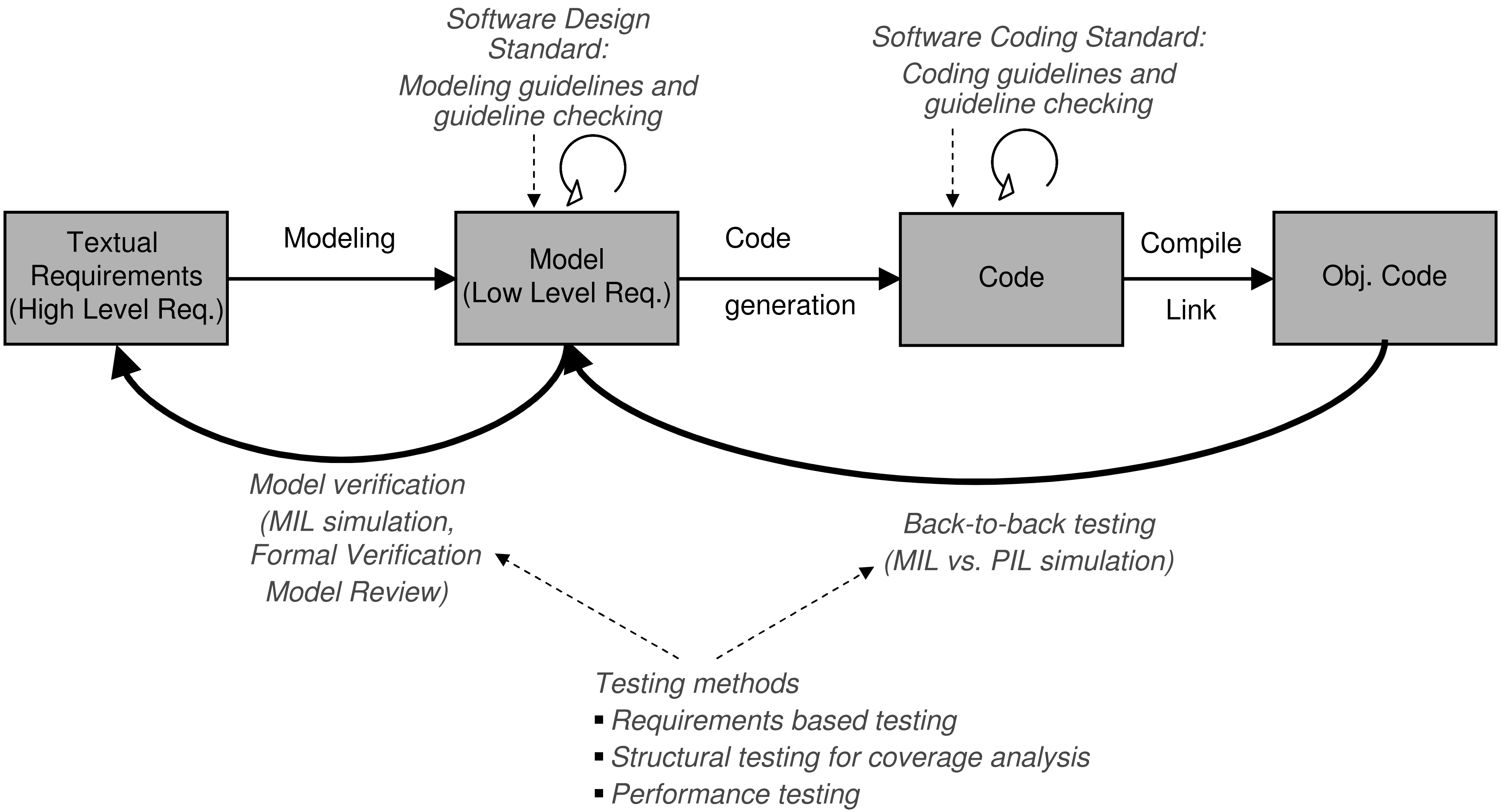
Author(s): Michael Beine, Dr. Tom Bienmüller
Version: 1.0
File: TL-ReferenceWorkflow_Addendum.doc
Created: 2010-01-18
Last Modified: 2010-02-15
Number of Pages: 9

Copyright © 2010 dSPACE GmbH and BTC Embedded Systems AG. All rights reserved.

- TÜV Certification
  - Workflow has been approved by TÜV
  - TargetLink and EmbeddedTester are fit for purpose to develop safety-related software according to ISO DIS 26262, IEC 61508 and derivative standards such as EN 50128[1]

[1] EN 50128, standard for software for railway control and protection systems, is considered as a sector-specific standard derived from IEC 61508.

Model-Based Reference Workflow

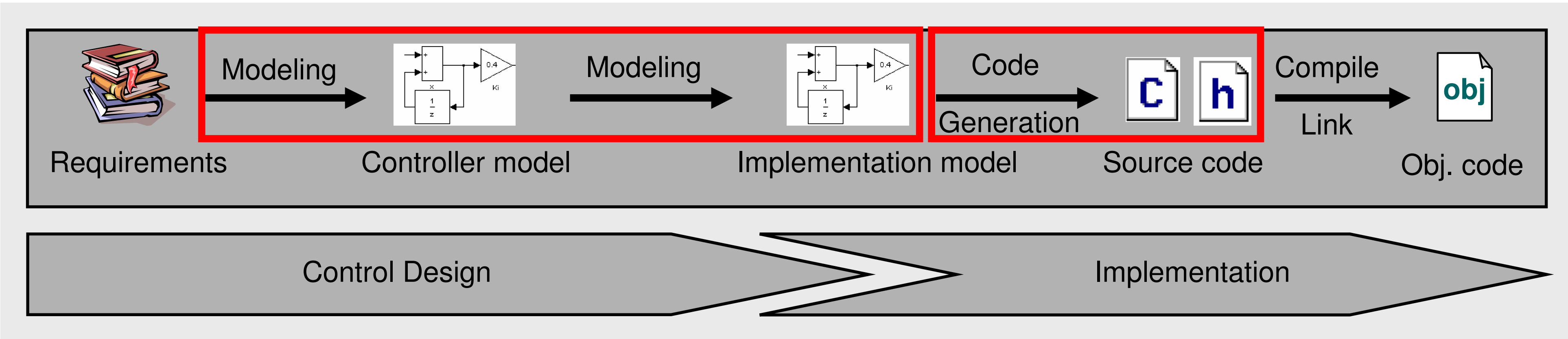


BTC Embedded Systems AG proprietary · All rights reserved

Modeling and Coding Guidelines

Table 1 — Topics to be covered by modelling and coding guidelines

Topics		ASIL A	ASIL B	ASIL C	ASIL D
1a	Enforcement of low complexity	++	++	++	++
1b	Use of language subsets[b]	++	++	++	++
1c	Enforcement of strong typing[c]	++	++	++	++
1d	Use of defensive implementation techniques	o	+	++	++
1e	Use of established design principles	+	+	+	++
1f	Use of unambiguous graphical representation	+	++	++	++
1g	Use of style guides	+	++	++	++
1h	Use of naming conventions	++	++	++	++



BTC Embedded Systems AG proprietary · All rights reserved

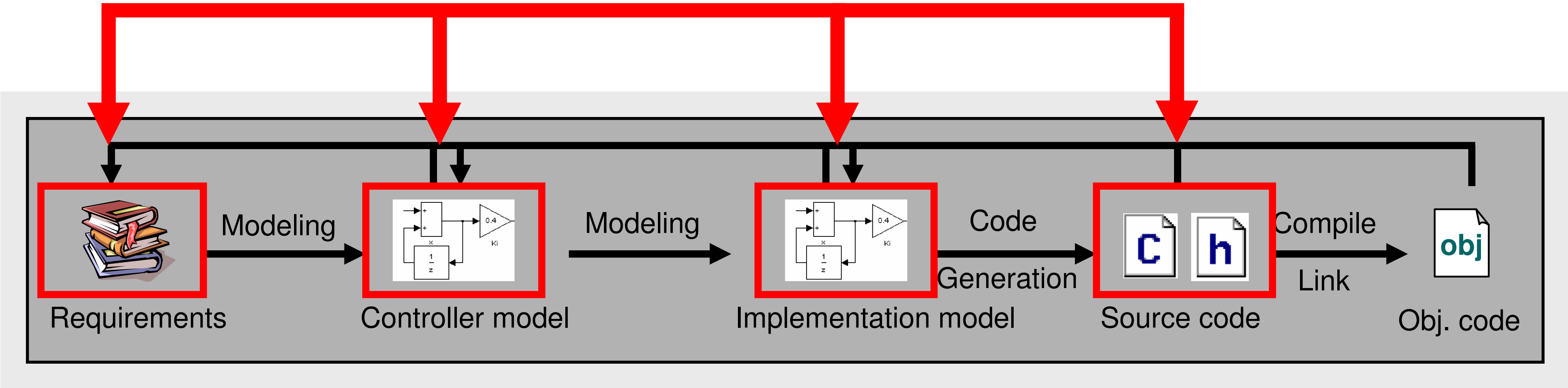
Requirements Traceability

- Requirements in DOORS Excel Word, etc. can be linked to the model
- Links: Model ↔ Code
- Bi-Directional Traceability between Requirements and Test-Cases

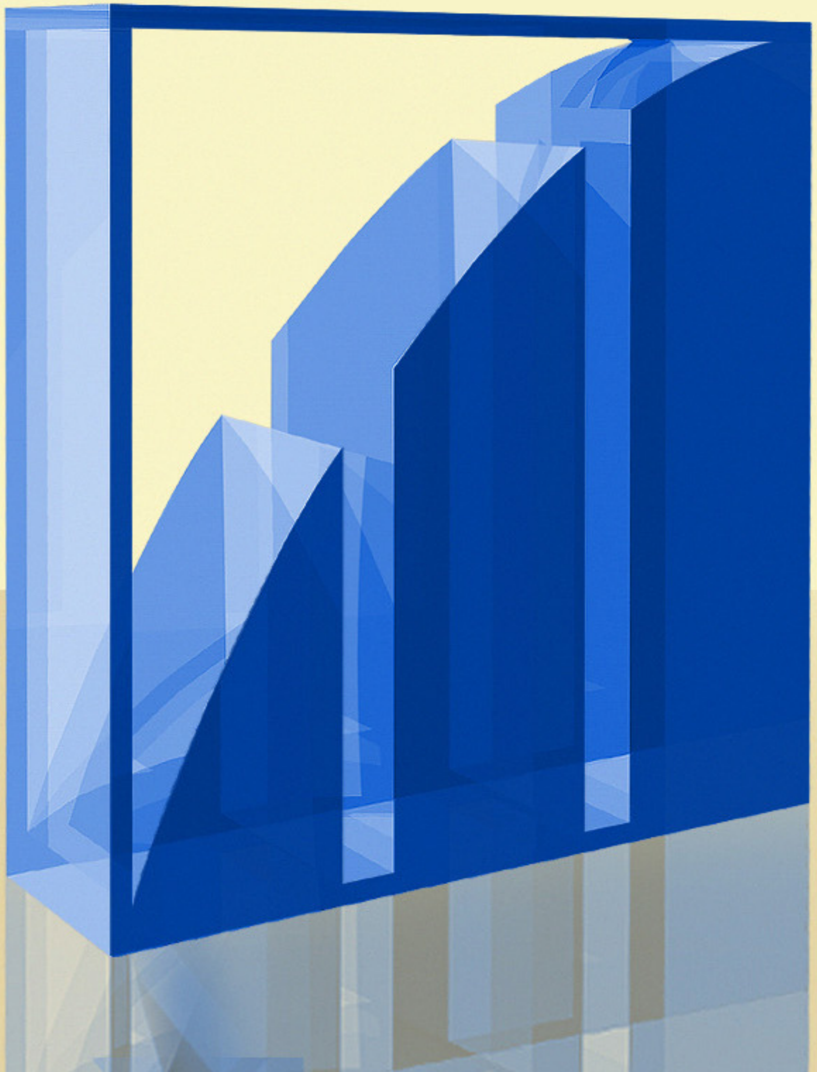
RMI Simulink V&V

dSPACE TargetLink

BTC Embedded Systems EmbeddedTester

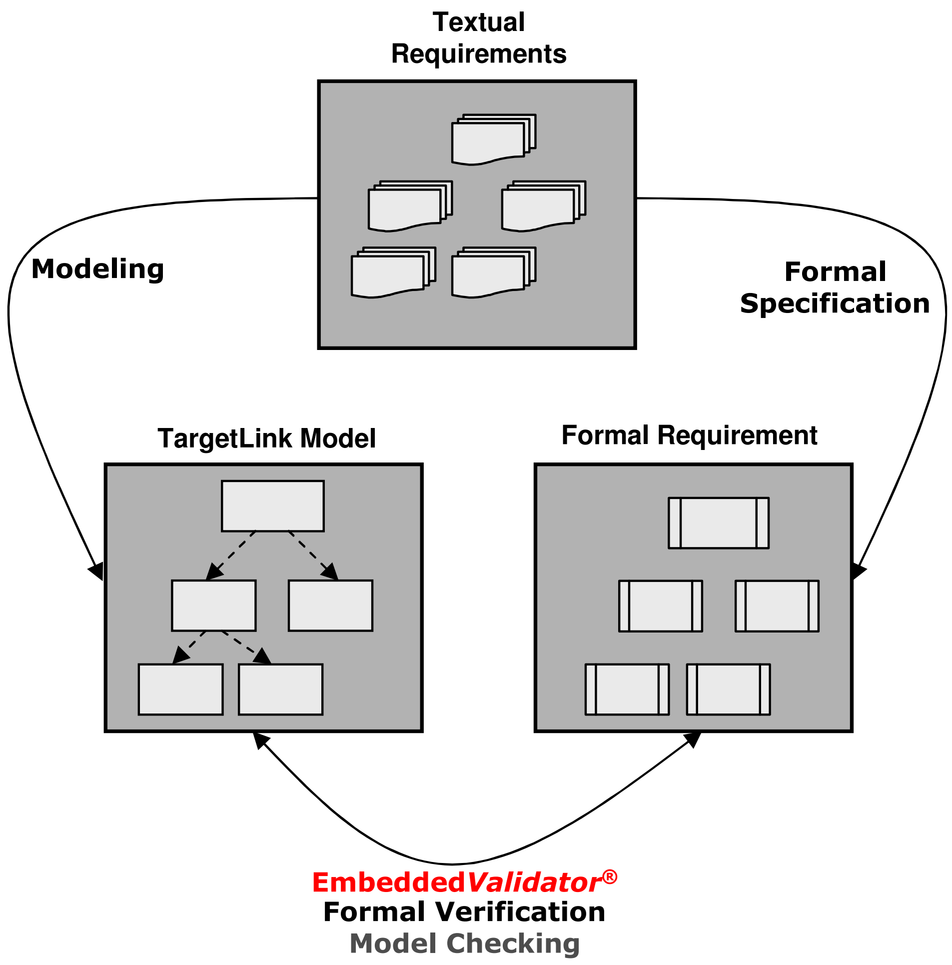


BTC Embedded Systems AG proprietary · All rights reserved
BTC Embedded Systems



Formal Specifications and Formal Verification in the context of ISO 26262

Formal Specification and Formal Verification Workflow



BTC Embedded Systems AG proprietary · All rights reserved

ISO 26262 - Notation Recommendations

Table 3 — Notations for software architectural design

Methods		ASIL A	ASIL B	ASIL C	ASIL D
1a	Informal notations	++	++	+	+
1b	Semi-formal notations	+	++	++	++
1c	Formal notations	+	+	+	+

Table 8 — Notations for software unit design

Methods		ASIL A	ASIL B	ASIL C	ASIL D
1a	Documentation of the software unit design in natural language	++	++	++	++
1b	Informal notations	++	++	+	+
1c	Semi-formal notations	+	++	++	++
1d	Formal notations	+	+	+	+

- Formal Notations are recommended for all Design levels starting with ASIL A

BTC Embedded Systems AG proprietary · All rights reserved

BTC Embedded Systems

ISO 26262 – Formal Verification Recommendations

Table 2 — Methods for the verification of requirements

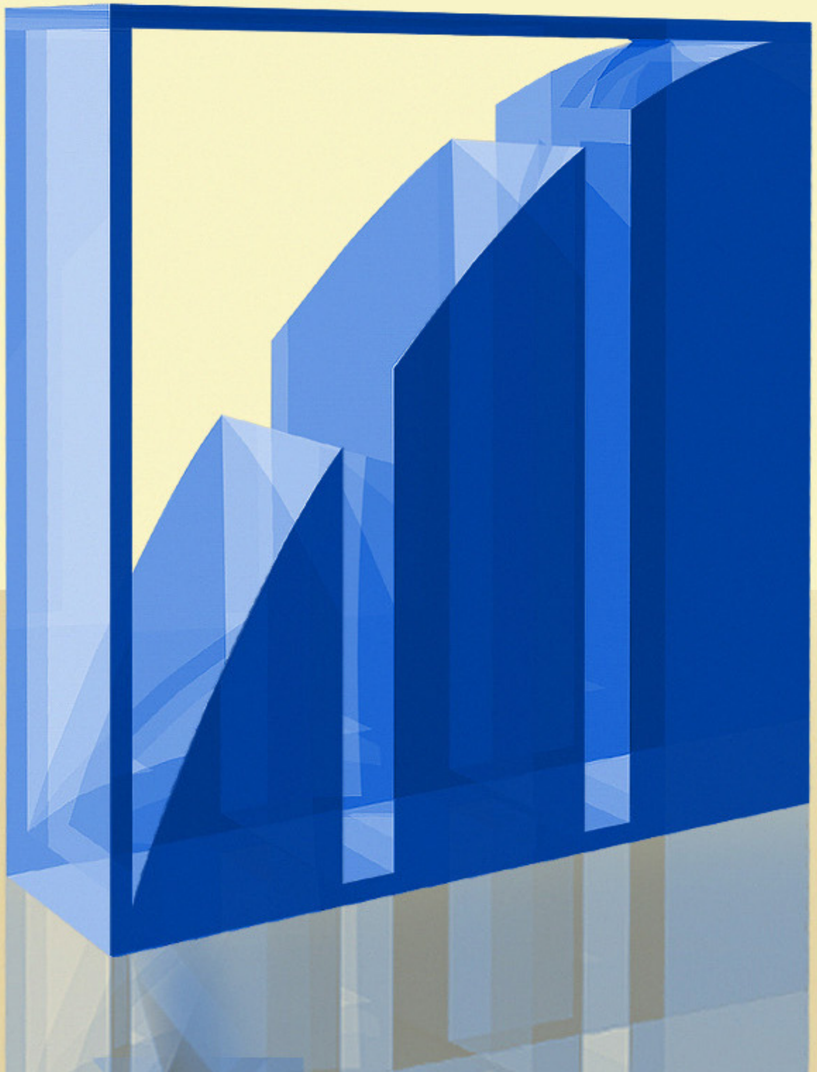
Methods		ASIL			
		A	B	C	D
1a	Informal verification by walkthrough	++	+	o	o
1b	Informal verification by inspection	+	++	++	++
1c	Semi-formal verification[a]	+	+	++	++
1d	Formal verification	o	+	+	+

[a] Method 1c can be supported by executable models.

- **Semi-formal Verification** (→ Simulation) of Requirements is even highly recommended for levels greater than ASIL B
- **Formal Verification** recommended from ASIL B
- ⇒ **inline with Model-based Development**
  - Executable Specification/Model allows Semi-formal verification
  - Formal Verification becomes applicable in early Development stages

BTC Embedded Systems AG proprietary · All rights reserved

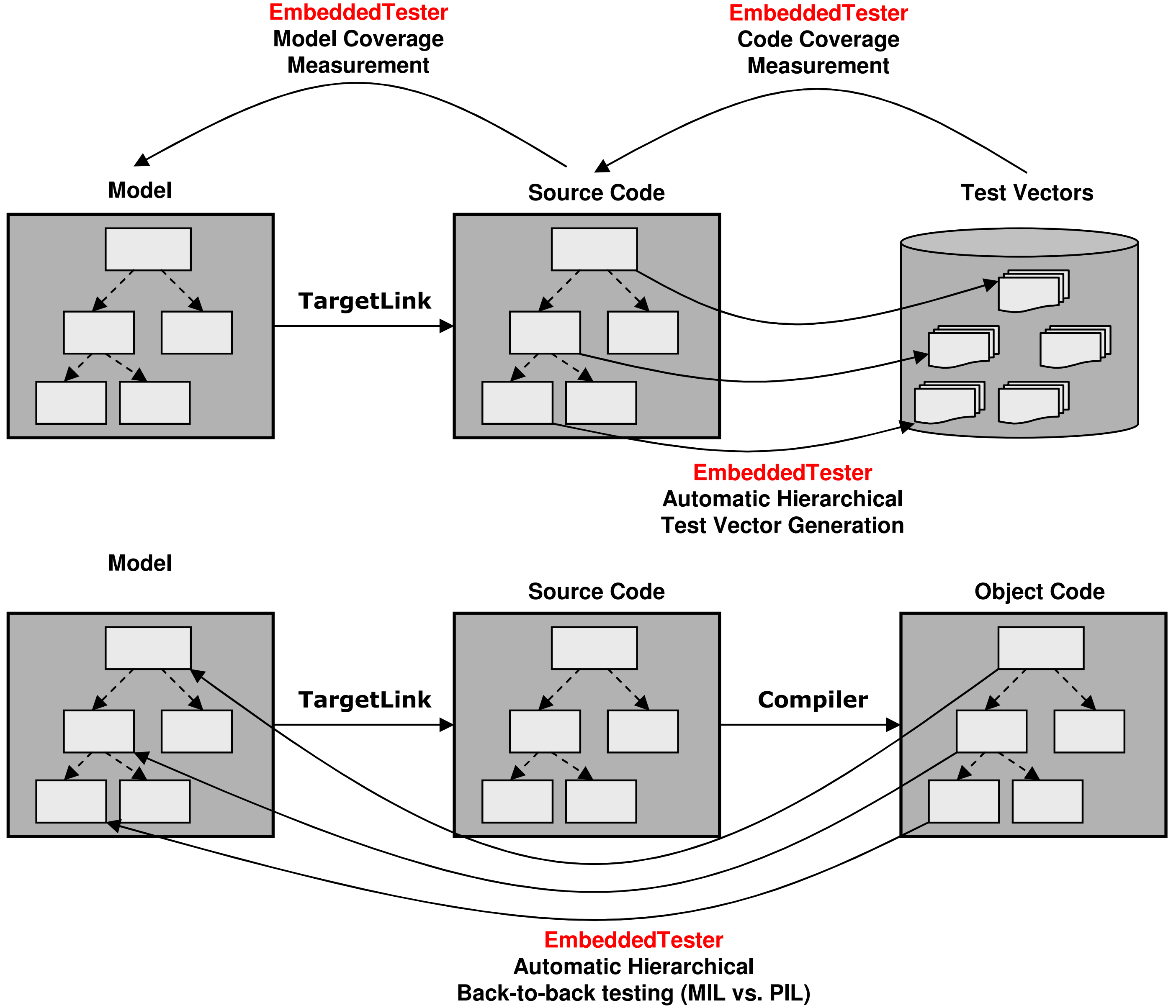
Automatic Test Generation and Execution



Automatic Test Generation and Execution in the context of ISO 26262

BTC Embedded Systems

Automatic Test Generation and Execution Workflow



BTC Embedded Systems AG proprietary · All rights reserved

ISO 26262 - Back-to-Back Testing

Table 12 — Methods for software unit testing

Methods		ASIL			
		A	B	C	D
1a	Requirement-based test	++	++	++	++
1b	Interface test	++	++	++	++
1c	Fault injection test[a]	+	+	+	++
1d	Resource usage test[b]	+	+	+	++
1e	Back-to-back test between model and code, if applicable[c]	+	+	++	++

[a] This includes injection of arbitrary faults in order to test safety mechanisms (e.g. by corrupting values of variables)

[b] Some aspects of the resource usage test can only by evaluated properly when the software unit tests are executed on the target hardware or if the emulator for the target processor supports resource usage tests.

[c] This method requires a model that can simulate the functionality of the software units. Here, the model and code are stimulated in the same way and results compared with each other.

- **For Testing of SW-Units, from ASIL C back-to-back-Tests are highly recommended**
  - Model-based and Code Testing in MIL,SIL and PIL

BTC Embedded Systems AG proprietary · All rights reserved

ISO 26262 - Coverage Metrics (SW-Unit)

9.4.4 To evaluate the completeness of test cases and to demonstrate that there is no unintended functionality, the coverage of requirements at the software unit level shall be determined and the structural coverage shall be measured in accordance with the metrics listed in Table 14. If necessary, additional test cases shall be specified or a rationale shall be available.

Table 14 — Structural coverage metrics at the software unit level

Methods			ASIL			
			A	B	C	D
1a	Statement coverage		++	++	+	+
1b	Branch coverage		+	++	++	++
1c	MC/DC (Modified Condition/Decision Coverage)		+	+	+	++

- **Quality of Test Cases measured**
  - … by coverage of Requirements (just informally)
  - … by structural Coverage metrics
    - The higher the ASIL-Level, the stronger the Metrics
  - Structural coverage metrics highly recommended for all ASIL Levels.

BTC Embedded Systems AG proprietary · All rights reserved

ISO 26262 - Target Testing

9.4.5 The test environment for software unit testing shall correspond as far as possible to the target environment. If the software unit testing is not carried out in the target environment, the differences in the source and object code, and the differences between the test environment and the target environment, shall be analysed in order to specify additional tests on the target environment during the subsequent test phases.

NOTE 2 Depending on the scope of the tests, it can be useful to carry out the software unit testing on the processor of the target system. If this is not possible, a processor emulator can be used. Otherwise the software unit testing is executed on the development system.

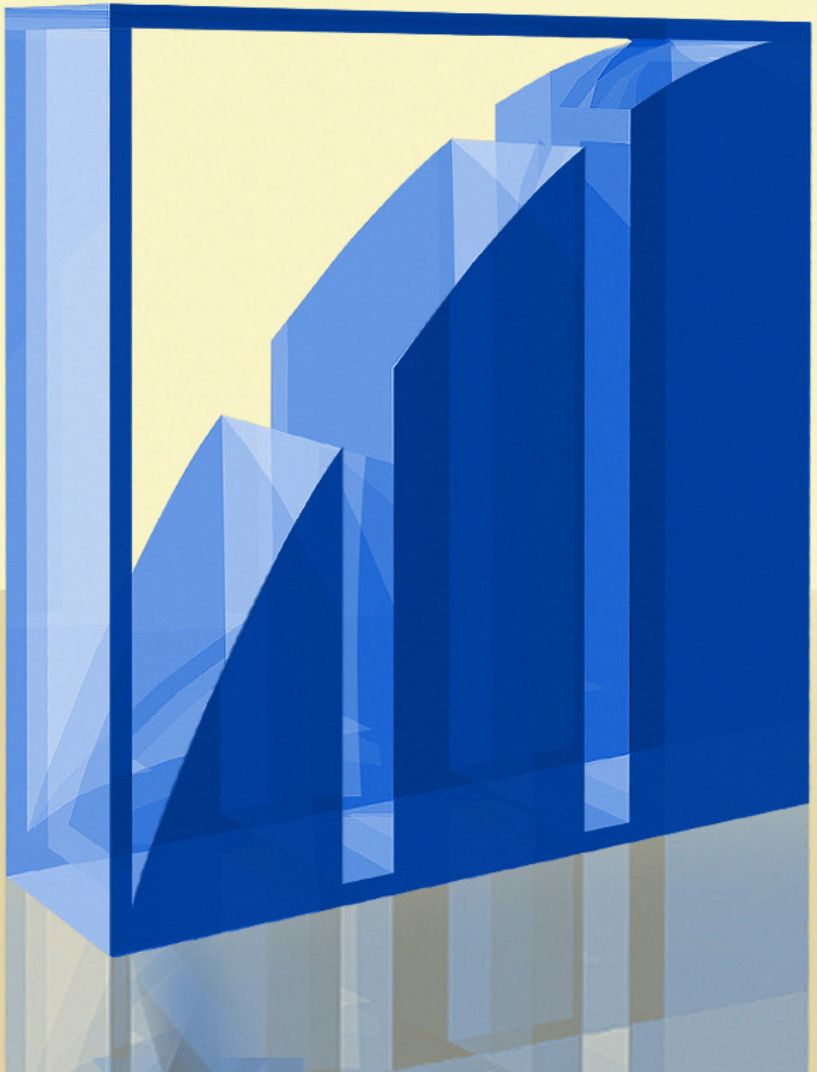
NOTE 3 Software unit testing can be executed in different environments, for example:

- Model-in-the-loop tests;
- Software-in-the-loop tests;
- Processor-in-the-loop tests; and

NOTE 4 For model-based development, software unit testing can be carried out at the model level followed by back-to-back tests between the model and the code. The back-to-back tests are used to ensure that the behaviour of the models with regard to the test objectives is equivalent to the automatically-generated code.

- **Perfect Match for model-based Development**
- **PIL-Tests are appropriate**

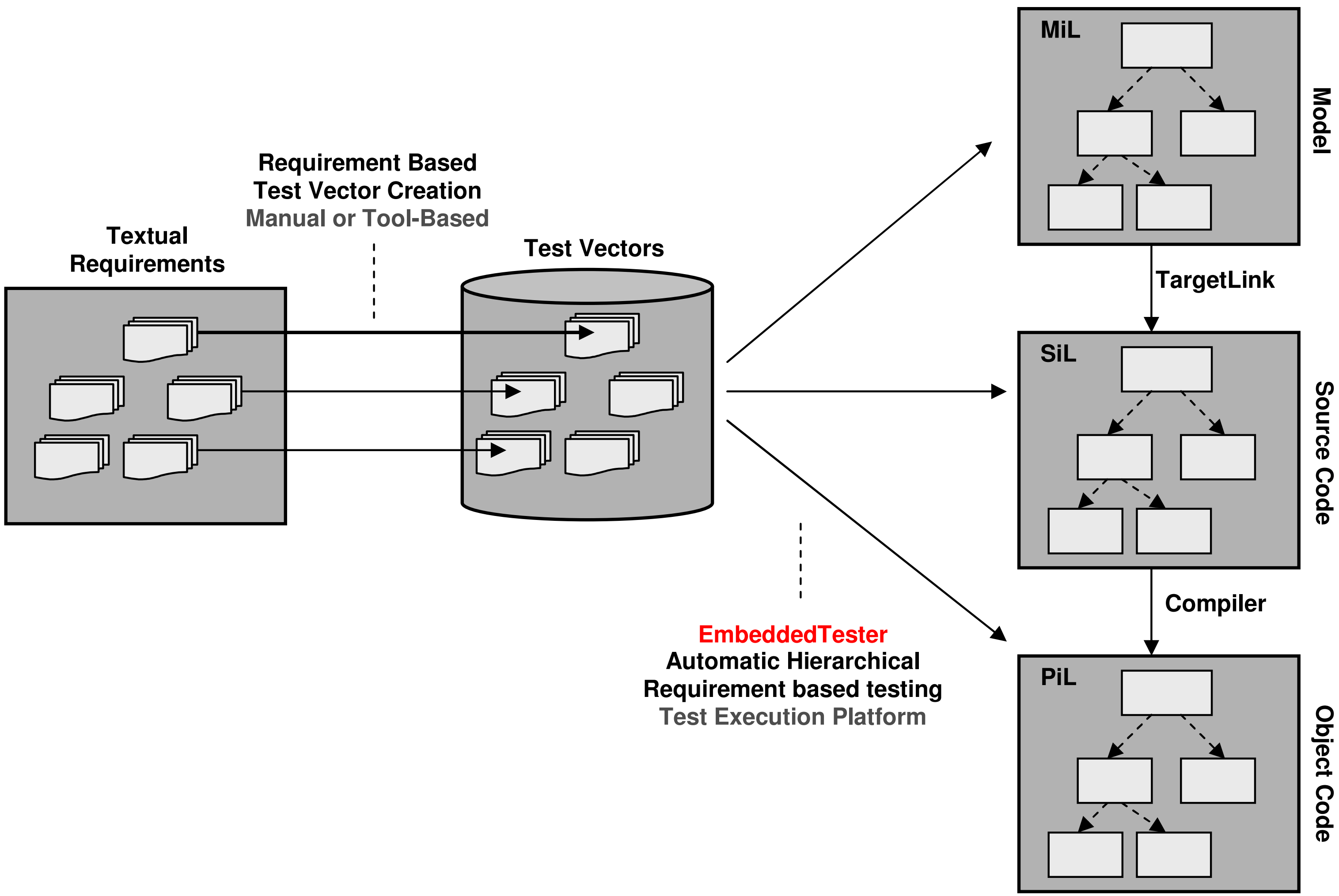
Requirements-Based Testing



Requirements Based Testing and Traceability in the context of ISO26262

BTC Embedded Systems

Requirements-Based Testing Workflow



BTC Embedded Systems AG proprietary · All rights reserved

BTC Embedded Systems

ISO 26262 - Requirements-based Test

Table 1[illegible] — Methods for software unit testing

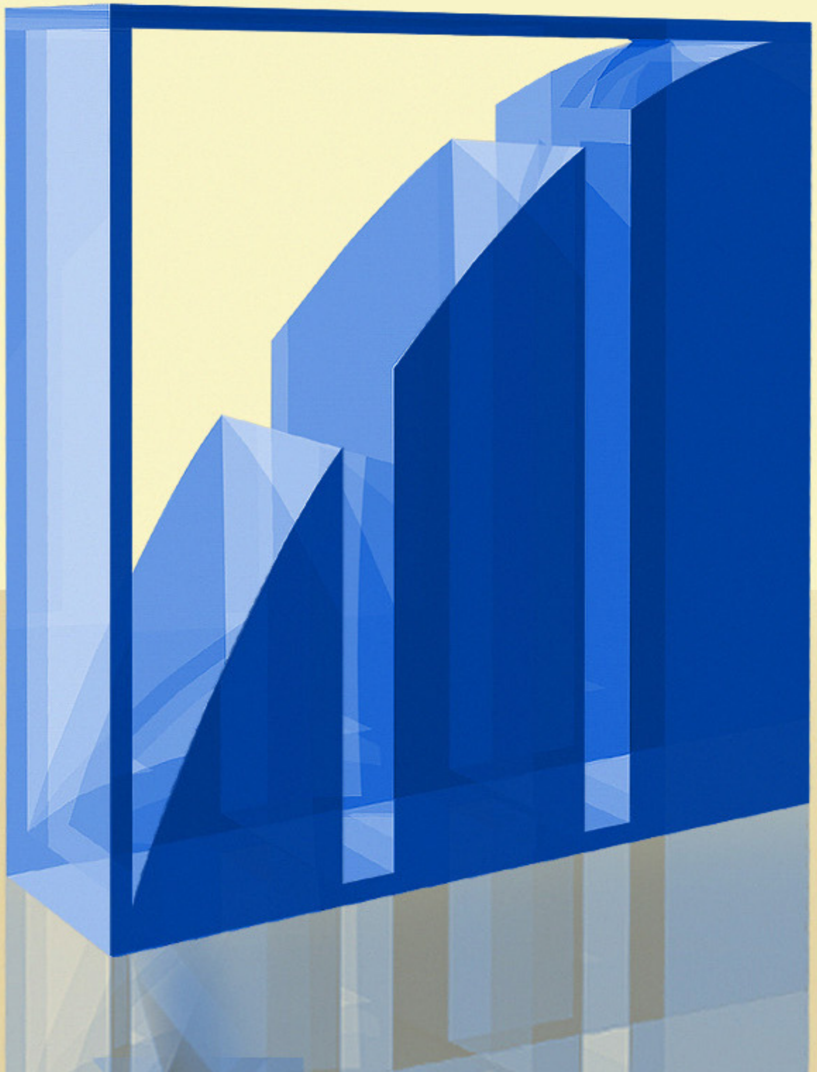
	Methods	ASIL			
		A	B	C	D
1a	Requirement-based test	++	++	++	++
1b	Interface test	++	++	++	++
1c	Fault injection test[a]	+	+	+	++

9.4.4 To evaluate the completeness of test cases and to demonstrate that there is no unintended functionality, the coverage of requirements at the software unit level shall be determined and the structural coverage shall be measured in accordance with the metrics listed in Table 14. If necessary, additional test cases shall be specified or a rationale shall be available.

- **Requirements-based Test is highly recommended for all ASIL–Levels (also Integration Testing)**
- **Metrics for Quality of Tests just intuitively defined**

BTC Embedded Systems AG proprietary · All rights reserved

Tools coverage of ISO 26262 standard methods

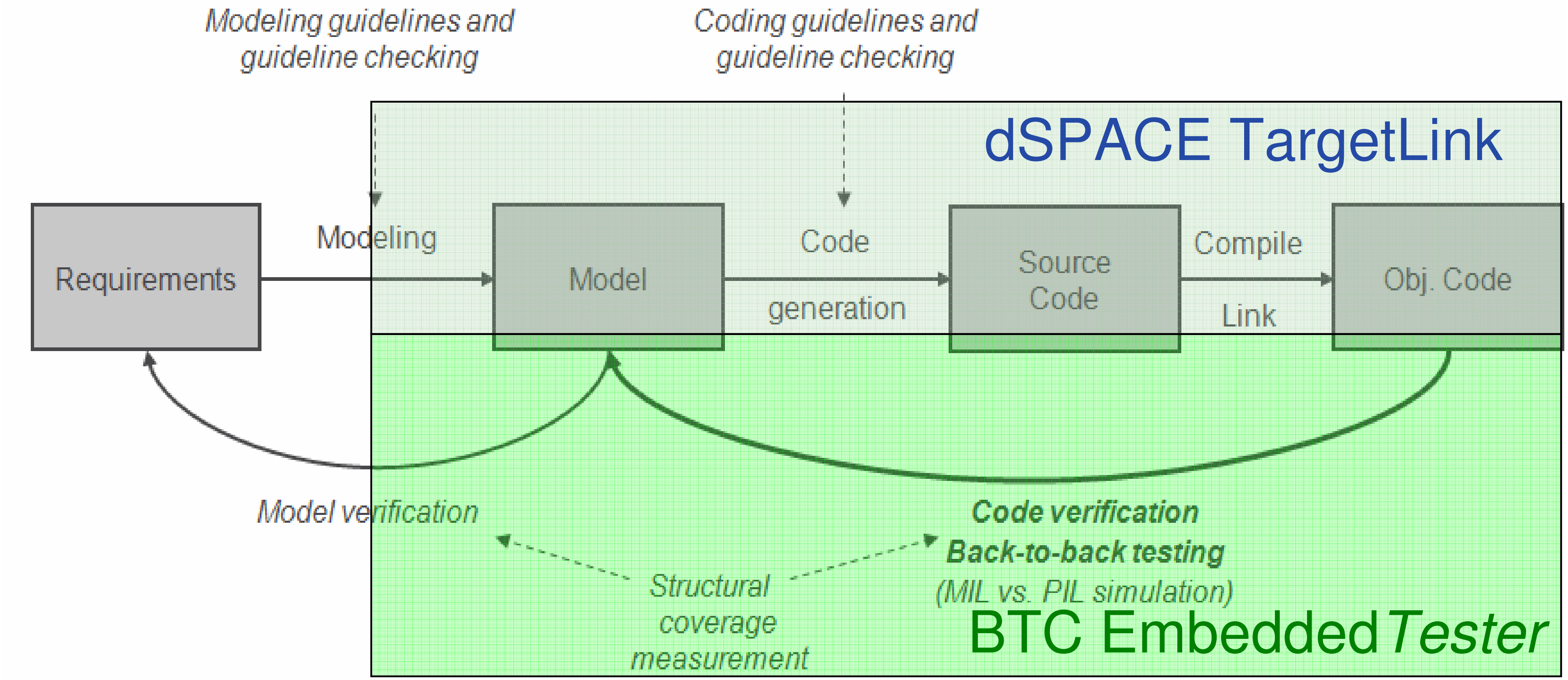


TargetLink and EmbeddedTester features mapping on ISO26262

BTC Embedded Systems

Tools mapping to the Workflow

Which portion of that workflow is covered by a tool?



BTC Embedded Systems

TargetLink Coverage of ISO26262 standard

SW Dev. Subphase	ISO 26262-6 Reference	ASIL A	ASIL B	ASIL C	ASIL D	Model Level	Code Level
Initiation of product development at the software level	Table 1 – Topics to be covered by modelling and coding guidelines 1a Enforcement of low complexity	++	++	++	++	▪ MISRA AC TL guidelines ▪ dSPACE TargetLink guidelines ▪ MAAB guidelines	▪ MISRA C:2004 guidelines ▪ TargetLink MISRA C compliance document
	Table 1 – 1b Use of language subsets	++	++	++	++		
	Table 1 – 1c Enforcement of strong typing	++	++	++	++		
	Table 1 – 1d Use of defensive implementation techniques	o	+	++	++		
	Table 1 – 1e Use of established design principles	+	+	+	++		
	Table 1 – 1f Use of unambiguous graphical representation	+	++	++	++		
	Table 1 – 1g Use of style guides	+	++	++	++		
	Table 1 – 1h Use of naming conventions	++	++	++	++		
Specification of software safety requirements	Table 2 – Methods for the verification of requirements 1c Semi-formal verification	+	+	++	++	▪ Supported by creating models that are executed in order to verify that it meets its requirements. ▪ Simulink/TargetLink MIL simulation	
	1d Formal Verification	o	+	+	+	▪ Embedded Validator supports formal verification of TargetLink models	

BTC Embedded Systems AG proprietary · All rights reserved

TargetLink Coverage of ISO26262 standard

SW Dev. Subphase	ISO 26262-6 Reference	ASIL A	ASIL B	ASIL C	ASIL D	Model Level	Code Level
Software architectural design	Table 3 – Notations for soft-ware architectural design 1b Semi-formal notations	+	++	++	++	▪ Using Simulink for the creation of a model adressing software architectural design aspects.	
	Table 4 – Principles for software architectural design 1a Hierarchical structure of software components	++	++	++	++	▪ Simulink allows modularization using hierarchical subsystems ▪ Simulink provides means to measure module complexity	▪ TargetLink allows flexible configuration of assignment of model parts to separate functions and C-files
	1b Restricted size of software components	++	++	++	++		
	Table 7 – Methods for the verification of the software architectural design 1a Informal verification by walkthrough of the design	++	+	o	o	▪ Informal verification is used to assess whether the software requirements are completely and correctly refined and rea-lised. Being able to link requirements and model and navigate back and forth supports this assessment and facilitates verifying that all requirements are covered.	
	Table 7 – 1b Informal verifica-tion by inspection of the design	+	++	++	++		
	Table 7 – 1c Semi-formal verification by simulating dynamic parts of the design	+	+	+	+	▪ Simulink/TargetLink MIL simulation	

BTC Embedded Systems AG proprietary · All rights reserved

TargetLink Coverage of ISO26262 standard

SW Dev. Subphase	ISO 26262-6 Reference	ASIL A	ASIL B	ASIL C	ASIL D	Model Level	Code Level
Software Unit Design and Imple-mentation	Table 8 – Notations for soft-ware unit design 1c Semi-formal notations	+	++	++	++	▪ The implementation model serves as the software unit design specification	
	Table 11 1c Model Inspection	+	++	++	++	▪ Direct visual inspection of the model ▪ TargetLink report generation for modules together with implementation information like data types, scalings, and simulation plots ▪ Rule-based guideline checkers to assure compliance with modelling guidelines	
	Table 11 - 1d Model Walk-through	++	+	o	o		
	Table 11 - 1e Inspection of the source code	+	++	++	++		▪ MISRA C compliance checkers to assure compliance to MISRA C ▪ TargetLink generated code in HTML-format with navigable links from code to model ▪ Can be replaced by automated methods and techniques
	Table 11 - 1f Walkthrough of the source code	++	+	o	o		

BTC Embedded Systems AG proprietary · All rights reserved

BTC Embedded Systems

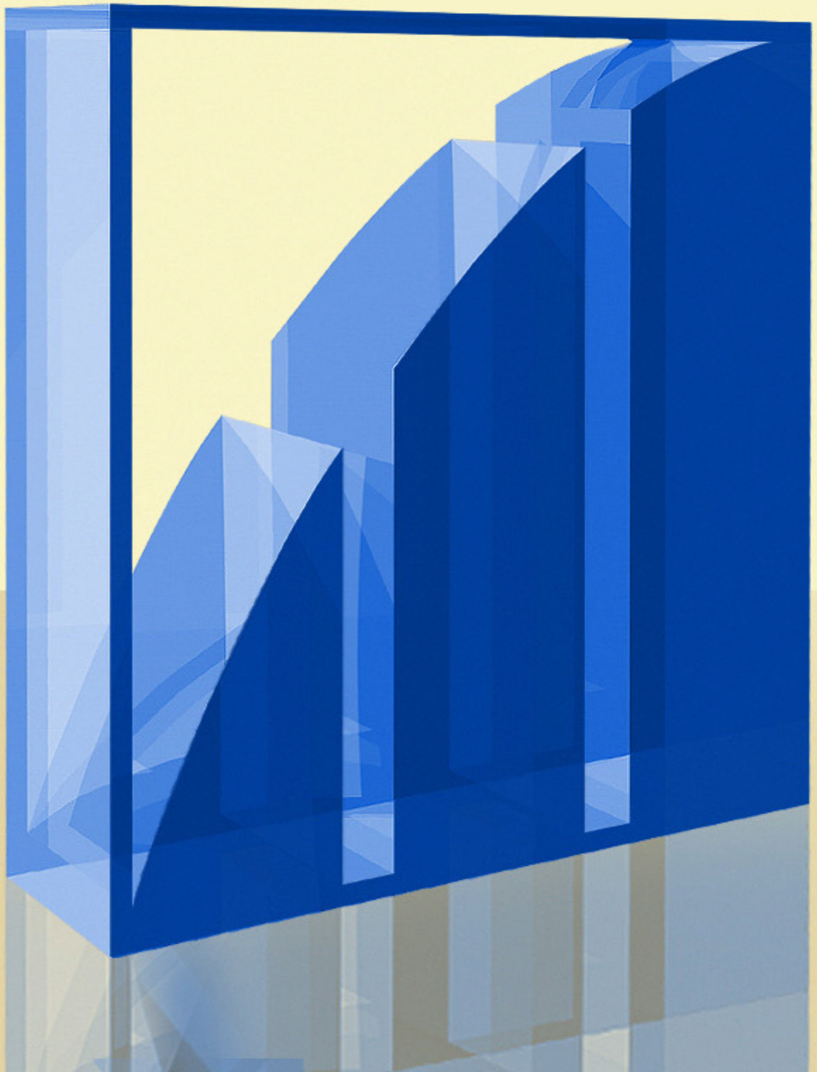
EmbeddedValidator/EmbeddedTester Coverage of ISO26262 standard

Process Phase	ISO 26262 Reference	ISO 26262 Method	ASIL A	ASIL B	ASIL C	ASIL D	EmbeddedValidator Coverage	EmbeddedTester Coverage
Requirements Specification	Table 8 — Notations for software unit design	1d Formal notations for requirements specification	+	+	+	+	Formal specification of functional and safety requirements based on patterns	
	Table 3 — Notations for software architectural design	1c Formal notations for requirements specification	+	+	+	+	Formal specification of functional and safety requirements based on patterns	
Requirements Verification	Table 2 — Methods for the verification of requirements	1c Semi-formal verification	+	+	++	++		Self-monitoring validity of the C-Observes from Patterns under MIL/SIL/PIL simulation.
		1d Formal verification	o	+	+	+	Formal verification based on model checking	
	Table 7 — Methods for the verification of the software architectural design	1d Semi-formal verification by simulating dynamic parts of the design	+	+	+	+		Self-monitoring validity of the C-Observes from Patterns under MIL/SIL/PIL simulation.
		1e Formal verification	o	o	+	+	Formal verification based on model checking	
	Table 10 — Methods for the verification of software unit design and implementation	1b Semi-formal verification	+	+	++	++		Self-monitoring validity of the C-Observes from Patterns under MIL/SIL/PIL simulation.
		1c Formal verification	o	o	+	+	Formal verification based on model checking	
Software integration and testing	Table 15 — Methods for software integration testing	1a Requirements-based test	++	++	++	++	Requirements based test generation based on the pattern mutation	Import and Execution of Functional Tests from different formats e.g. CTE, EXCEL, Signal Builder. Requirements based test generation based on C-Observers Patterns coverage
		1e Back-to-back test between model and code	+	+	++	++		Automatic MIL/SIL/PIL regression test execution and results comparison
Software unit testing	Table 14 — Structural coverage metrics at the software unit level	1a Statement coverage	++	++	+	+		Part of the code coverage report
		1b Branch coverage	+	++	++	++		Part of the code coverage report
		1c MC/DC (Modified Condition/Decision Coverage)	+	+	+	++		Part of the code coverage report
Software integration and	Table 17 — Structural coverage metrics at the	1a Function coverage	+	+	++	++		Part of the code coverage report
		1b Call coverage	+	+	++	++		Part of the code coverage report

BTC Embedded Systems AG proprietary · All rights reserved

BTC Embedded Systems

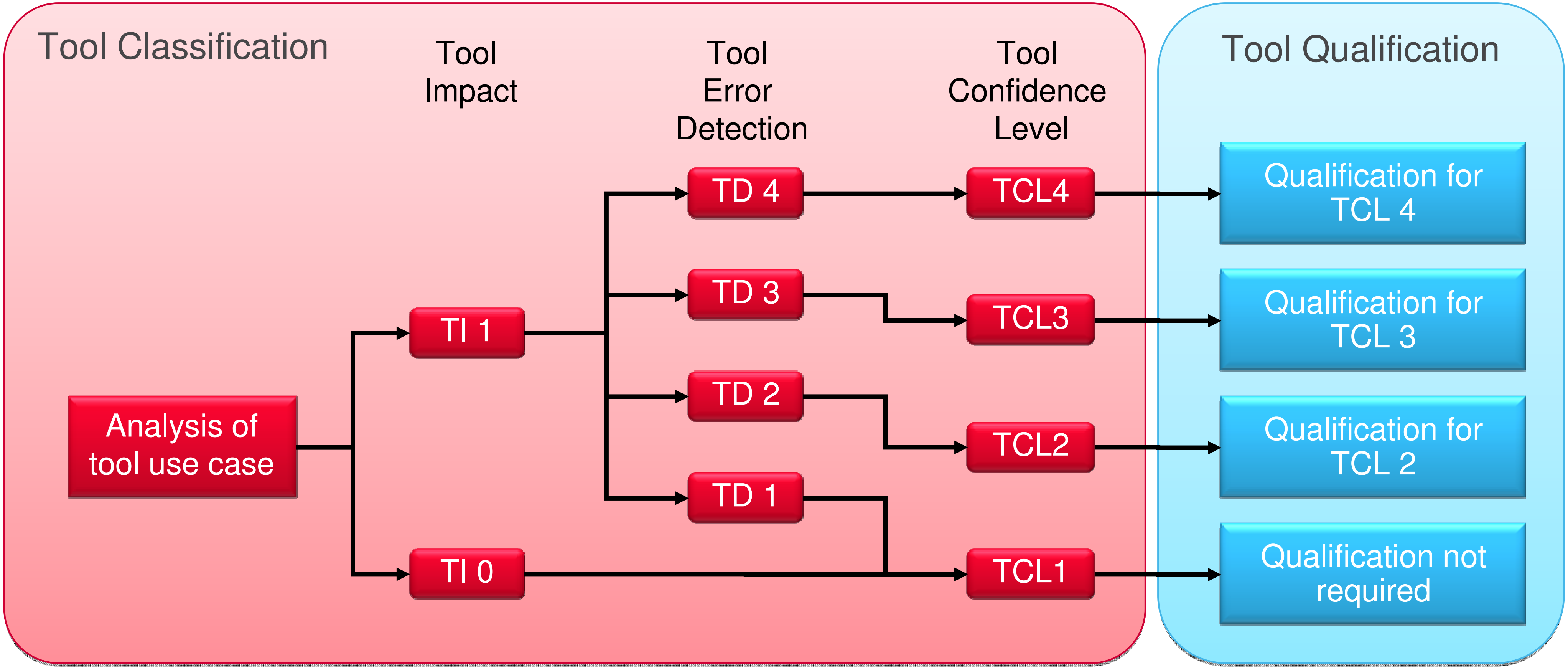
Qualification of software tools in the context of ISO26262



TargetLink and EmbeddedTester
Qualified for ISO26262

ISO 26262: Software Tool Qualification

- Tool Confidence Level (TCL): defines need for qualification and appropriate measures
- Tool Impact (TI): impact of tool errors on the software/system
- Tool Error Detection (TD): probability of preventing or detecting tool errors

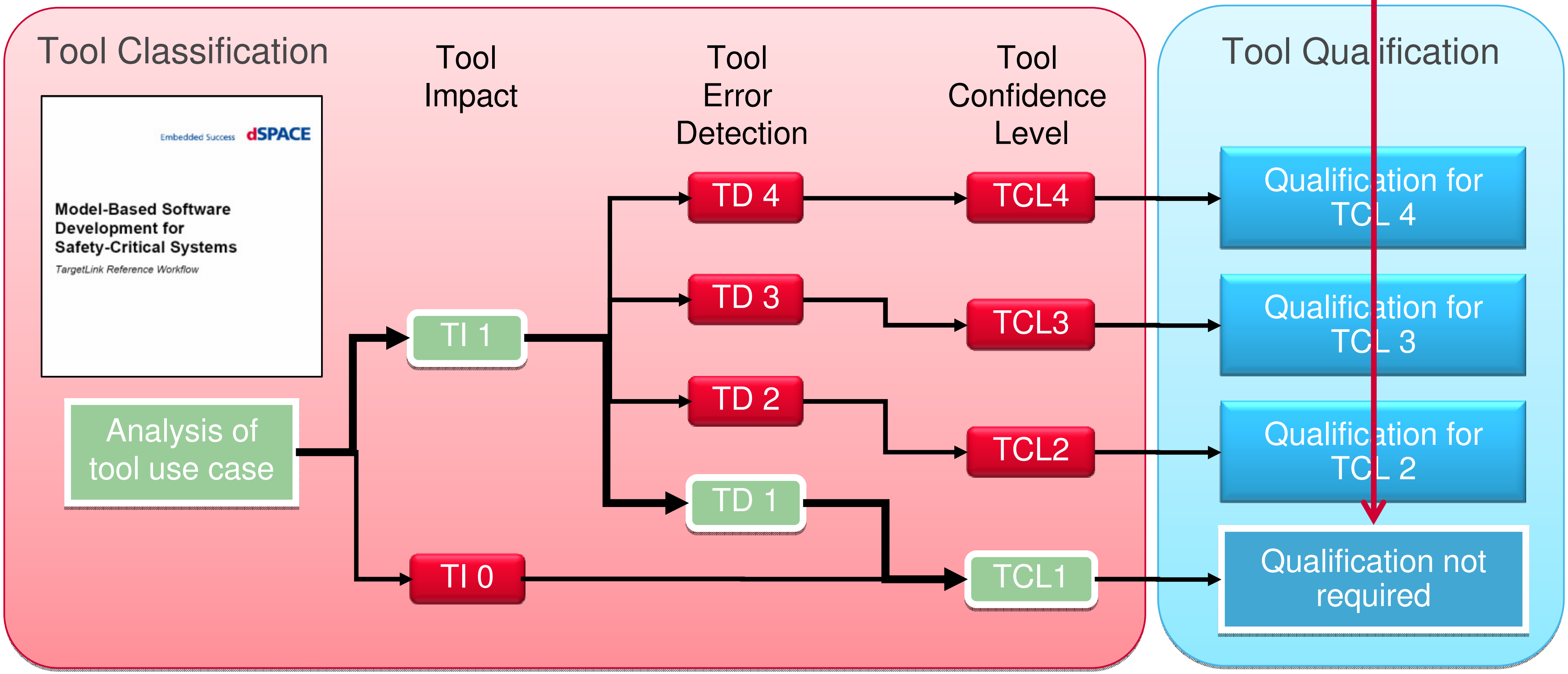


BTC Embedded Systems AG proprietary · All rights reserved

Code Generator Qualification for ISO 26262

TargetLink

- TCL based on Reference Workflow
- “Fit-for-Purpose” Certification



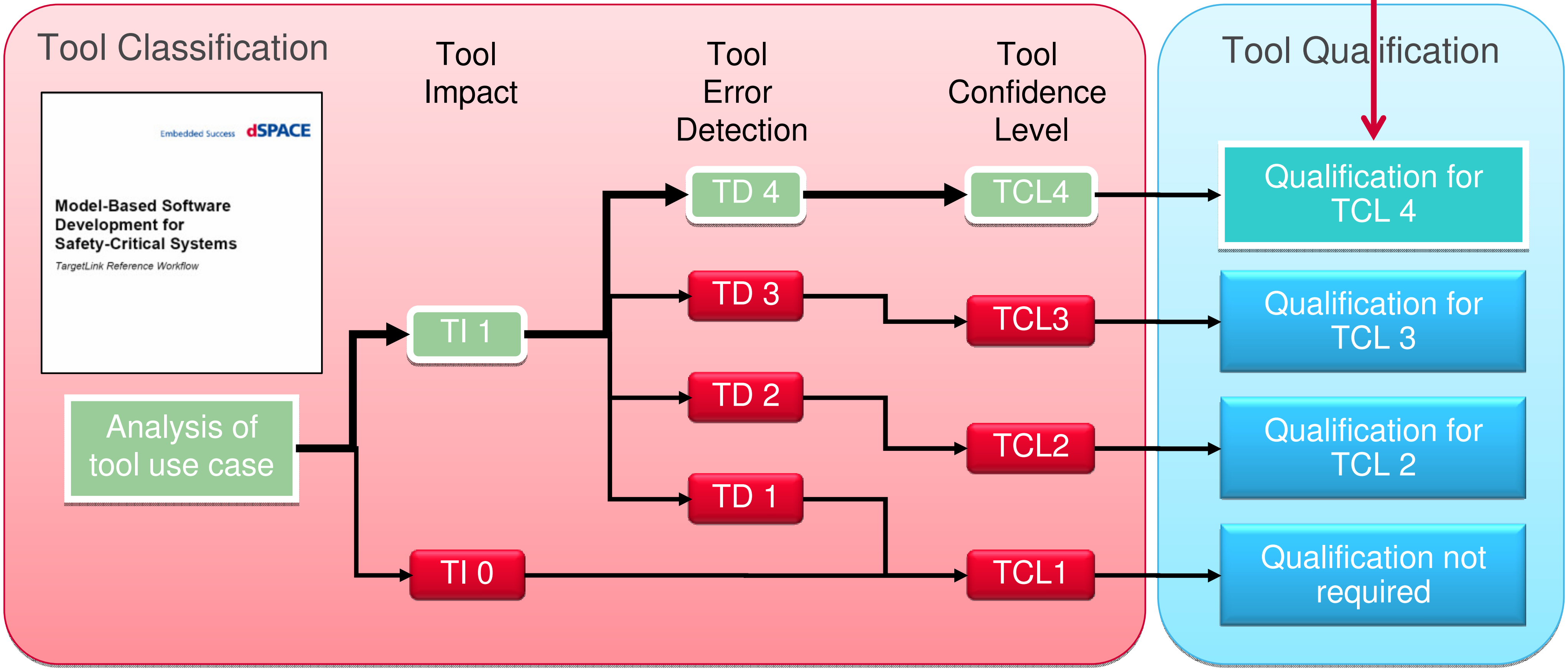
BTC Embedded Systems AG proprietary · All rights reserved

BTC Embedded Systems

Test Tool Qualification for ISO 26262

BTC EmbeddedTester

- TCL based on Reference Workflow
- Certification: Validation of Software Tool and Evaluation of Tool Development Process



BTC Embedded Systems AG proprietary · All rights reserved

EmbeddedTester Qualified for ISO26262

- **ISO/DIS 26262 (highly) recommends**
  - Back-to-Back test between Model and Code
  - Structural Coverage Metrics for Software-Unit-Testing
- **ISO/DIS 26262 demands Tool-Qualification**
  - Also for Testing Tools used for *revealing* errors
- **Embedded*Tester* offers**
  - Automated Back-to-Back tests between MIL/SIL/PIL
  - Different Structural Coverage Metrics up to MC/DC

ZERTIFIKAT ◆ CERTIFICATE ◆ CEPTИФИKAT ◆ CERTIFICADO ◆ CERTIFICAT

CERTIFICATE

No. Z10 10 02 73219 001

Holder of Certificate: BTC Embedded Systems AG
Buschstraße 1
26127 Oldenburg
GERMANY

Factory(ies): 73210

Certification Mark:

Product: Software Tool for Safety Related Development

Model(s): BTC EmbeddedTester

Parameters: The model-based verification tool is fit for purpose to verify target code by back-to-back testing including coverage criteria according to IEC 61508, ISO 26262, and derivative standards.
The tool is qualified for all ASIL-Levels according to ISO 26262.

The report BO75961C is a mandatory part of this certificate.

Tested according to: IEC 61508-3:1998 (Fitness for purpose)
ISO/DIS 26262-8:2009

The product was tested on a voluntary basis and complies with the essential requirements. The certification mark shown above can be affixed on the product. It is not permitted to alter the certification mark in any way. In addition the certification holder must not transfer the certificate to third parties. See also notes overleaf.

Test report no.: BO75961C

Date, 2010-02-25 (Frank Rauch)

Page 1 of 1

404584

TÜV SÜD Product Service GmbH · Zertifizierstelle · Ridlerstraße 65 · 80339 München · Germany

Qualify Embedded*Tester*
for the automated Application of
Back-to-Back Tests and
Structural Coverage Measurement
in 26262-compliant Processes

BTC Embedded Systems AG proprietary · All rights reserved

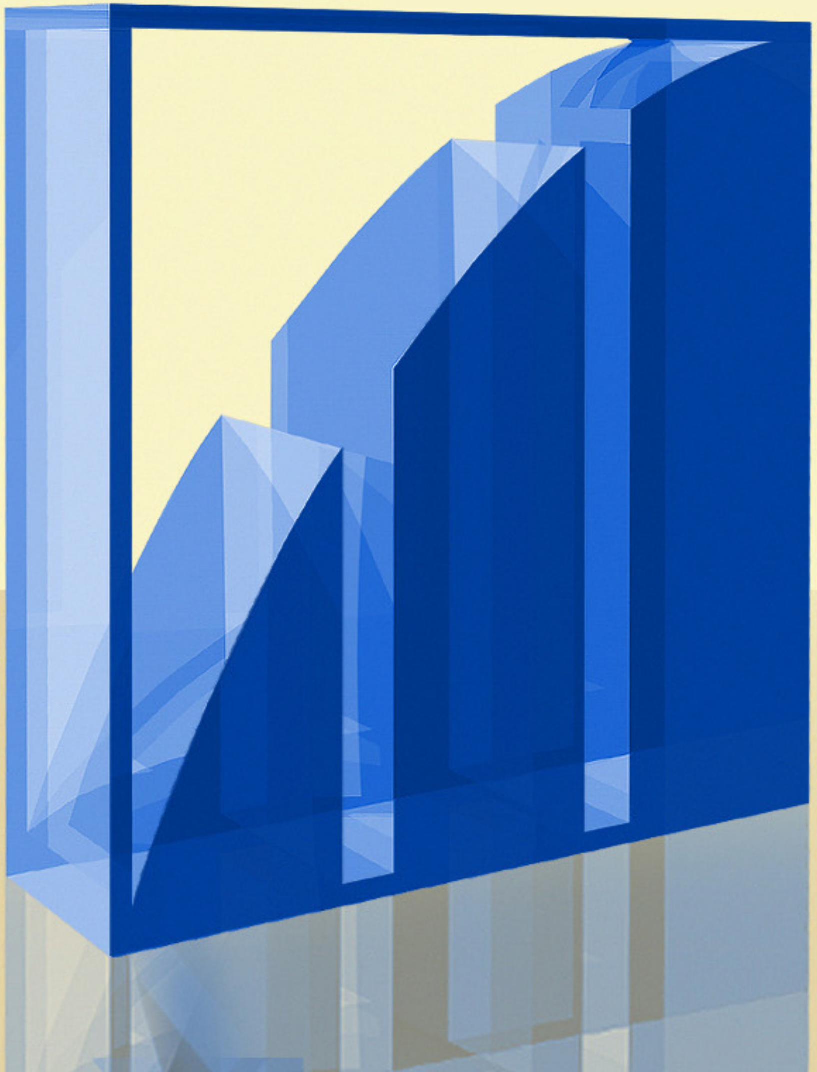
BTC Embedded Systems

Qualification with Validation Suite

- **In 26262 Method „Validation of the Software Tool“ on ASIL-D is considered as „highly recommended“**
- **”Validation of the software tool can be automated largely by using a *validation suite*.“** **[ISO/DIS 26262-8]**
- **A Validation Suite (VS) contains**
  - Feature Specifications for the relevant Features
  - Test Specifications for these Features incl. Feature Coverage
  - Test Implementation for the Test Specifications
- **Qualification is achieved by executing the VS at the User‘s site**
  - Added value: this approach also assures Quality of the Tool to be qualified in the User‘s environment

BTC Embedded Systems AG proprietary · All rights reserved

Conclusions



Benefits of an ISO 26262 Conform Model Based Development and Testing Process

BTC Embedded Systems AG proprietary · All rights reserved

BTC Embedded Systems

Benefits of Model Based Development

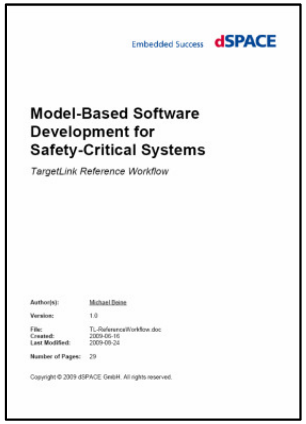
- Development of **embedded systems** is a **time and cost** consuming procedure under growing **time-to-market** and new **quality and safety** standards pressure.
- **Model-Based Development** and **Autocoding** of safety-relevant software is widely applied for gaining **efficiency**.
- ISO 26262 explicitly **acknowledges** the paradigm of Model-based development with Autocoding to improve quality and ensure the safety needs.
- **TÜV approved** that **TargetLink** and **EmbeddedTester** are **fit** for purpose to develop and test safety-related software according to **ISO 26262, IEC 61508 and derivative standards**.

TargetLink

EmbeddedTester

International Organization for Standardization

ISO 26262


Embedded Success dSPACE

Model-Based Software Development for Safety-Critical Systems

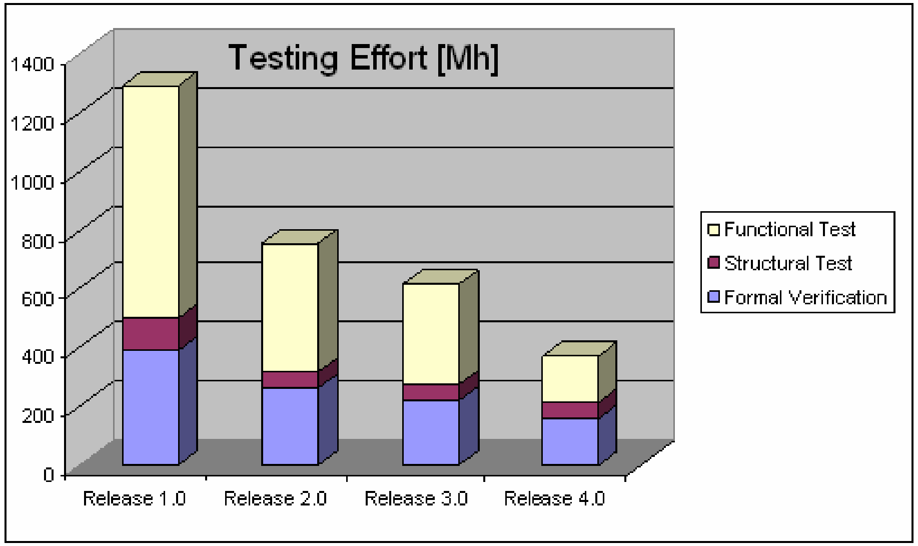
TargetLink Reference Workflow

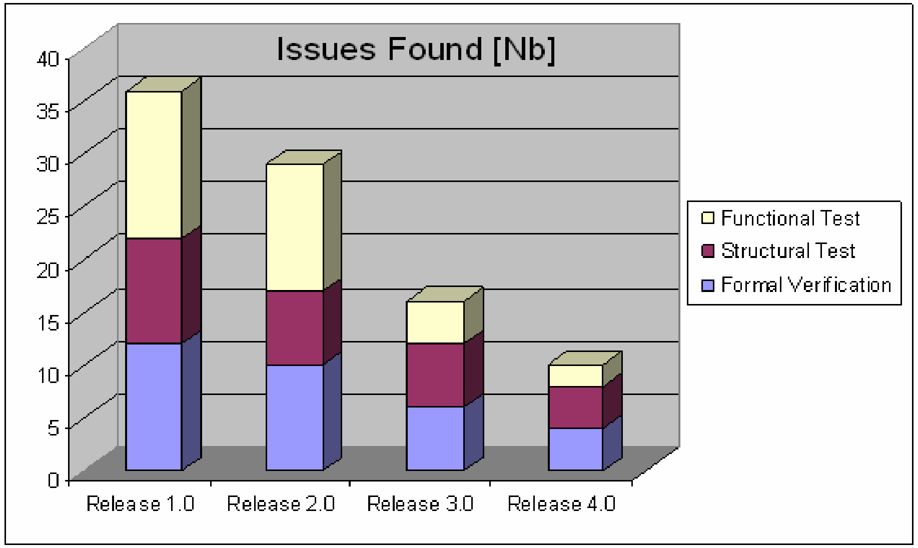
BTC Embedded Systems AG proprietary · All rights reserved

BTC Embedded Systems

Benefits of advanced and integrated test method

- Functional testing finds about 20-40% of the problems.
- 30-40% of the software problems can be directly found by using the structural testing and back-to-back comparison.
- Formal verification is relevant for testing of safety-relevant software as it finds additional problems that might not be found by traditional testing methods.





BTC Embedded Systems AG proprietary · All rights reserved

Thanks for your attention!

- dSPACE and BTC Embedded Systems through DynaFusion in India can be your trustful partners in providing ISO 26262 conform products and know-how.

- We are looking forward to contacting us!

BTC Embedded Systems AG proprietary · All rights reserved