IBM Software

Innovate2012

The Premier Event for Software and Systems Innovation

Next NOW!

Model-Based Development and Testing of Safety Related Software

Dr. Udo Brockmeyer
CEO
BTC Embedded Systems AG
udo.brockmeyer@btc-es.de

Please note

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Agenda

- Functional Safety
- Automotive: ISO 26262
- Model Driven System Development (MDSD) and Model Based Testing (MBT)
- Development and Testing of Safety Related Software
  - Qualification of Software Tools
  - IBM Rational Rhapsody Reference Workflow

Business and Market Drivers

Are Leading to Increased Electronic and Embedded (E/E) Software Content

Green

- Hybrid
- Electric

Product Differentiation

- Innovation
- Brand Image

Growing Customer Demands

- Comfort
- Telematics

Competition

- Overcapacity
- Price Wars

Legal Requirements

- Safety
- Environment

Increased usage of electronics and embedded software;
Highly sophisticated in-vehicle electric/electronic (E/E) systems

INSTRUMENTED

INTERCONNECTED

INTELLIGENT

Because of E/E vehicles became instrumented, interconnected and intelligent over the last 30 years

Software Functionality and Amount

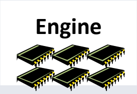
10 M LOC

1 M LOC

100 k LOC

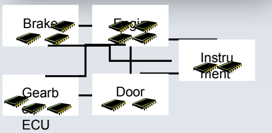
10 k LOC

1980



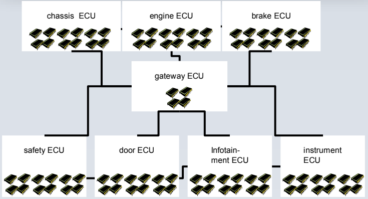
- Ignition
- Engine controls
- Instrumentation

1990



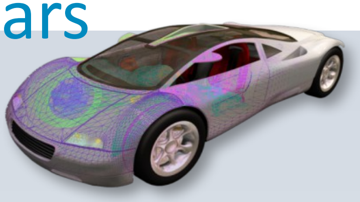
- Engine management
- ABS
- Digital dashboard
- Electronic seats / doors
- Automated climate control
- Safety sensors

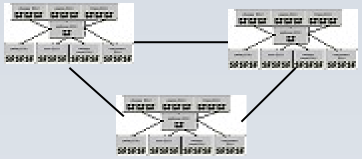
2000



- Dynamic Damping Control
- Brake Energy Regeneration
- Integral Active Steering
- Electrically controlled air vents
- Night Vision
- Lane Departure Warning
- Lane Change Warning
- Adaptive Headlights
- Head-Up Display
- Active Cruise Control
- Camera systems
- Driver assistant systems

2010





- “Intelligent” Vehicles
- Increasing levels of electrification
- Intersection control violations
- Lane/road departure
- Road surface and pavement conditions
- 360/distance vision
- Active suspension and stability control
- Dynamic route guidance / navigation
- Motorist information
- Data downloads
- Recovery of stolen vehicles
- Electronic payments

2020

Electric and Electronic (E/E) systems are realizing automotive product innovation

How can you ensure that a possible malfunction will not harm the driver, occupants or road users?

→ A „safety“ approach is needed

What is Safety?

Safety in the context of automotive embedded systems is about the prevention, detection, and response to unintended behavior that can lead to harm for the vehicle occupants and other road user

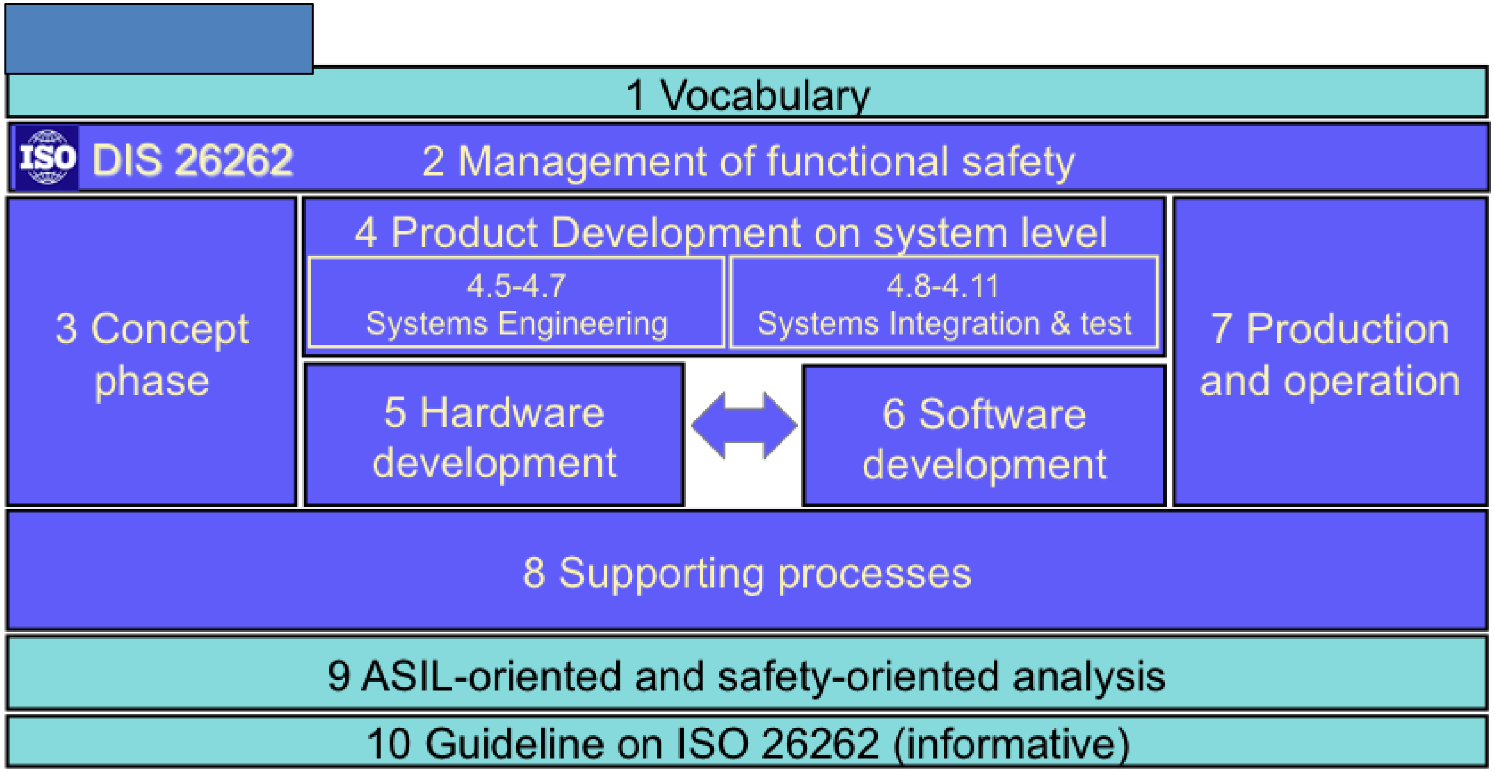
- Obvious examples:
  - anti-lock brakes, air bags, traction control, electronic cruise control, adaptive cruise control, collision avoidance, lane change control
- Less obvious examples:
  - front windshield defroster/defogger, rear windshield (backlite) defroster, auto-on headlamps, auto-on running lights, seat-belt pre-tensioners, low tire pressure warning system, engine, electric-assist power steering.

What is ISO 26262?

- ISO 26262 (derived from principles in IEC 61508)
  - Functional Safety for E/E Systems in Road vehicles (<3.5 tonnes), i.e. Automotive as opposed to Trucks and Buses
  - **Covers the management and technical aspects of the complete safety development lifecycle**
  - Takes a risk-based approach for determining risk classes (Automotive Safety Integrity Levels, ASILs).
  - **Definition of optional, recommended and highly recommended methods for development activities** within system-, hardware and software development depending on defined ASIL
- Design safety into the system from the outset
- ISO 26262 covers the complete development lifecycle from Lust to Dust

What ISO 26262 means for Automotive?

- Automotive System development for Electronic and Electrical components need to comply to ISO 26262



1 Vocabulary

DIS 26262

2 Management of functional safety

3 Concept phase

4 Product Development on system level

4.5-4.7 Systems Engineering

4.8-4.11 Systems Integration & test

5 Hardware development

6 Software development

7 Production and operation

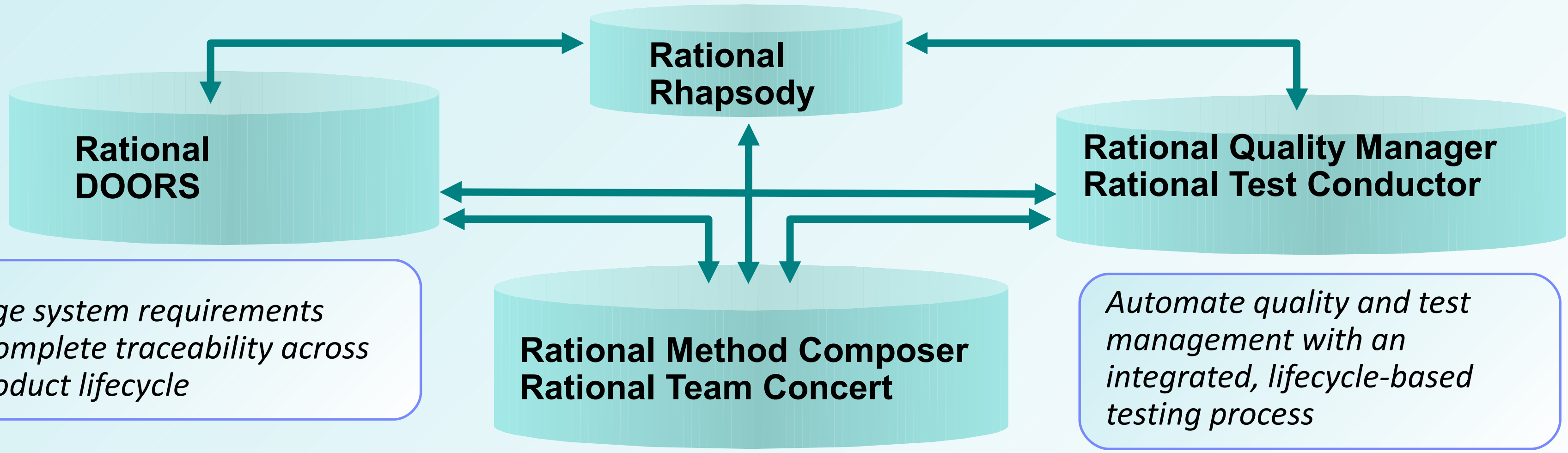
8 Supporting processes

9 ASIL-oriented and safety-oriented analysis

10 Guideline on ISO 26262 (informative)

A look to the inside: How IBM Rational Software Platform for Automotive Systems supports ISO 26262

Use modeling to validate requirements, architecture and design earlier in the development process – including Simulink integration, autocode generation and automated test case generation; use in the same way models for FMEA, FTA



Rational Rhapsody

Rational DOORS

Rational Quality Manager
Rational Test Conductor

Rational Method Composer
Rational Team Concert

Manage system requirements with complete traceability across the product lifecycle

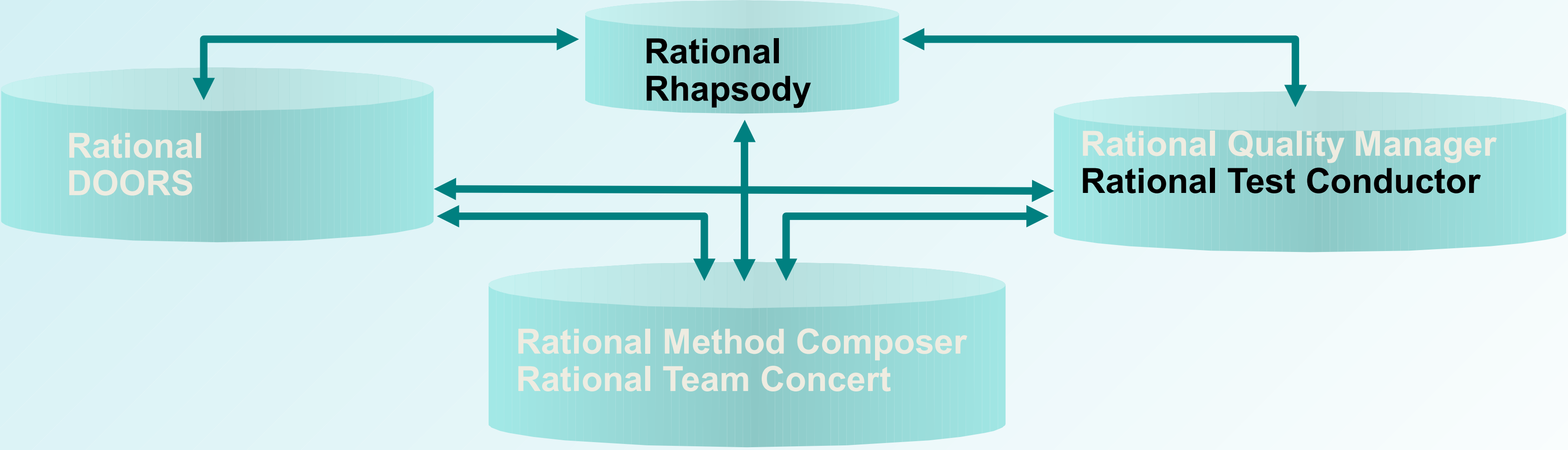
Automate quality and test management with an integrated, lifecycle-based testing process

Manage collaborative systems lifecycle management across development teams and engineering disciplines with Automotive data model based on AUTOSAR & ISO26262 process template and compliance

A look to the inside: How IBM Rational Software Platform for Automotive Systems supports ISO 26262

Software Development and Testing with
IBM® Rational® Rhapsody® and IBM® Rational® Rhapsody® TestConductor Add On

(Model Driven System Development (MDSD) and Model Based Testing (MBT))



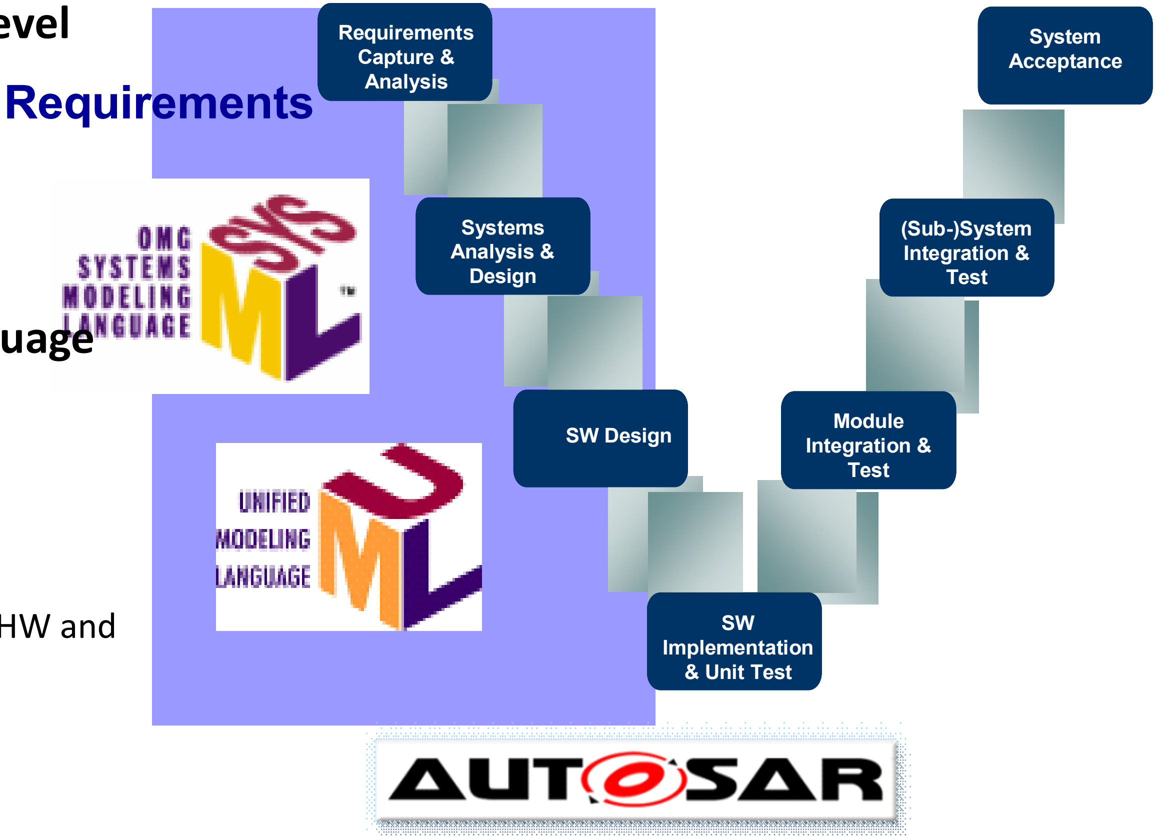
What is Model Driven Systems Development (MDSD)?

A structured approach for the development of complex systems across the mechanical, electronic and software disciplines

- Ensures that all requirements are fulfilled
- **Employs models as the primary artifacts throughout systems development**
- Facilitates improved communication among all stakeholders
- Provides a disciplined way to manage complexity through abstraction
- Improves quality through integration of testing with development
- Allows specification and development of software that controls the system and enables its use

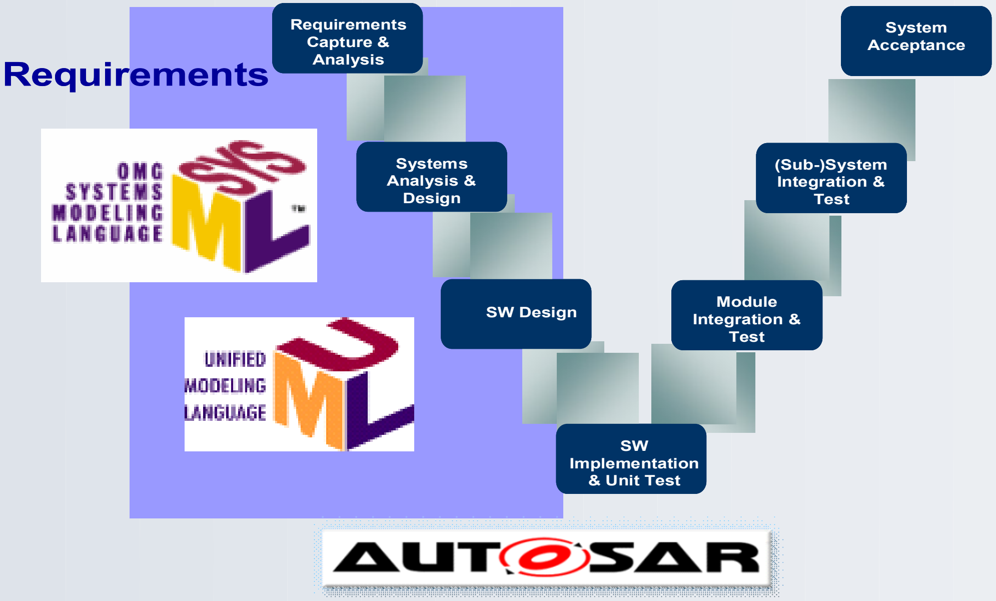
Allowing abstraction, hierarchies and modularization with domain-focused, standards-based languages

- **SysML – Systems Modeling Language for modeling high-level**
  - vehicle functions
  - logical and technical architecture
  - vehicle and E/E system behavior
- **UML – Unified Modeling Language for modeling**
  - ECU and SW architecture
  - Client-specific profiles
- **AUTOSAR**
  - Detailed E/E System and ECU HW and SW architecture


Requirements
Requirements Capture & Analysis
Systems Analysis & Design
SW Design
SW Implementation & Unit Test
Module Integration & Test
(Sub-)System Integration & Test
System Acceptance
OMG SYSTEMS MODELING LANGUAGE
SysML
UNIFIED MODELING LANGUAGE
UML
AUTOSAR

MDSD Benefit with IBM Rational Rhapsody

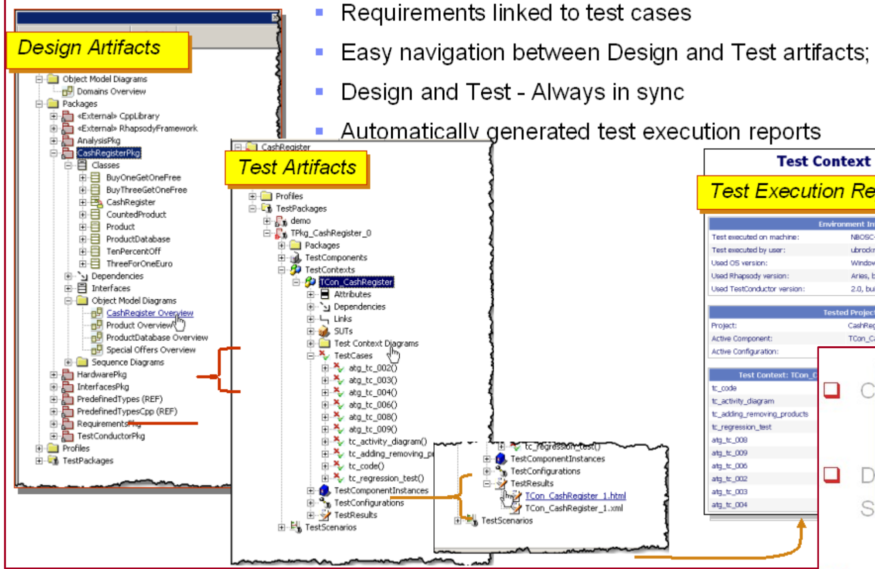
- **Execute models**
  - **Analyze behavior**
  - **Find errors early**
- Extend requirements engineering to development
  - "Traceability and more … "
- Use models for Safety Analysis
- Focus on analysis and design
  - From "system" down to "software"


Requirements
Requirements Capture & Analysis
Systems Analysis & Design
SW Design
SW Implementation & Unit Test
Module Integration & Test
(Sub-)System Integration & Test
System Acceptance
OMG SYSTEMS MODELING LANGUAGE SysML
UNIFIED MODELING LANGUAGE UML
AUTOSAR

- **Develop highly-optimized embedded C/C++/Java software**
  - Model and Code are kept in sync
- Collaborate visually
  - Integrated in the IBM Rational Software Platform for Automotive Systems

Model Driven Testing
IBM Rational Rhapsody Test Conductor Add On

- Common Browser
- Requirements linked to test cases
- Easy navigation between Design and Test artifacts;
- Design and Test - Always in sync
- Automatically generated test execution reports



Test Context Result

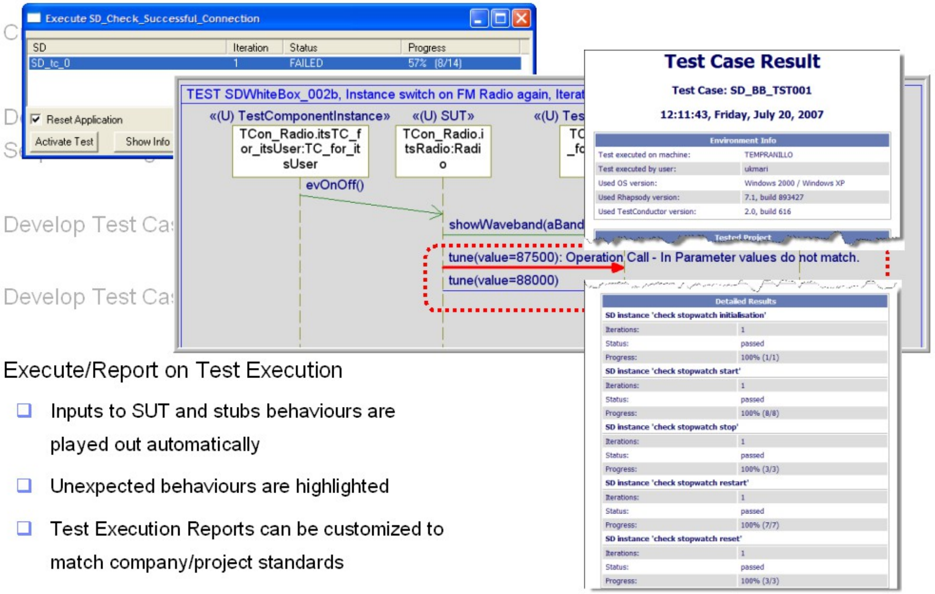
Environment Info	
Test executed on machine:	NBOSC-21-1
Test executed by user:	ubrockmeyer
Used OS version:	Windows 2000 / Windows XP
Used Rhapsody version:	Aries, build 799102
Used TestConductor version:	2.0, build 530

Tested Project	
Project:	CashRegister
Active Component:	TCon_CashRegister_5
Active Configuration:	

Test Context: TCon_C
tc_code
tc_activity_diagram
tc_adding_removing_products
tc_regression_test
atg_tc_008
atg_tc_009
atg_tc_006
atg_tc_002
atg_tc_003
atg_tc_004

Automation…
Test Execution &
Test Reporting

Design & Test Processes Fully Integrated


Execute SD_Check_Successful_Connection
☑ Reset Application
Activate Test
Show Info
TEST SDWhiteBox_002b, Instance switch on FM Radio again, Iterat
«(U) TestComponentInstance»
«(U) SUT»
«(U) Tes
TCon_Radio.itsTC_for_itsUser:TC_for_itsUser
TCon_Radio.itsRadio:Radio
TC_fo
evOnOff()
showWaveband(aBand
tune(value=87500): Operation Call - In Parameter values do not match.
tune(value=88000)

SD	Iteration	Status	Progress
SD_tc_0	1	FAILED	57% (8/14)

Test Case Result

Test Case: SD_BB_TST001

12:11:43, Friday, July 20, 2007

Environment Info	
Test executed on machine:	TEMPRANILLO
Test executed by user:	ukmari
Used OS version:	Windows 2000 / Windows XP
Used Rhapsody version:	7.1, build 893427
Used TestConductor version:	2.0, build 616

Tested Project

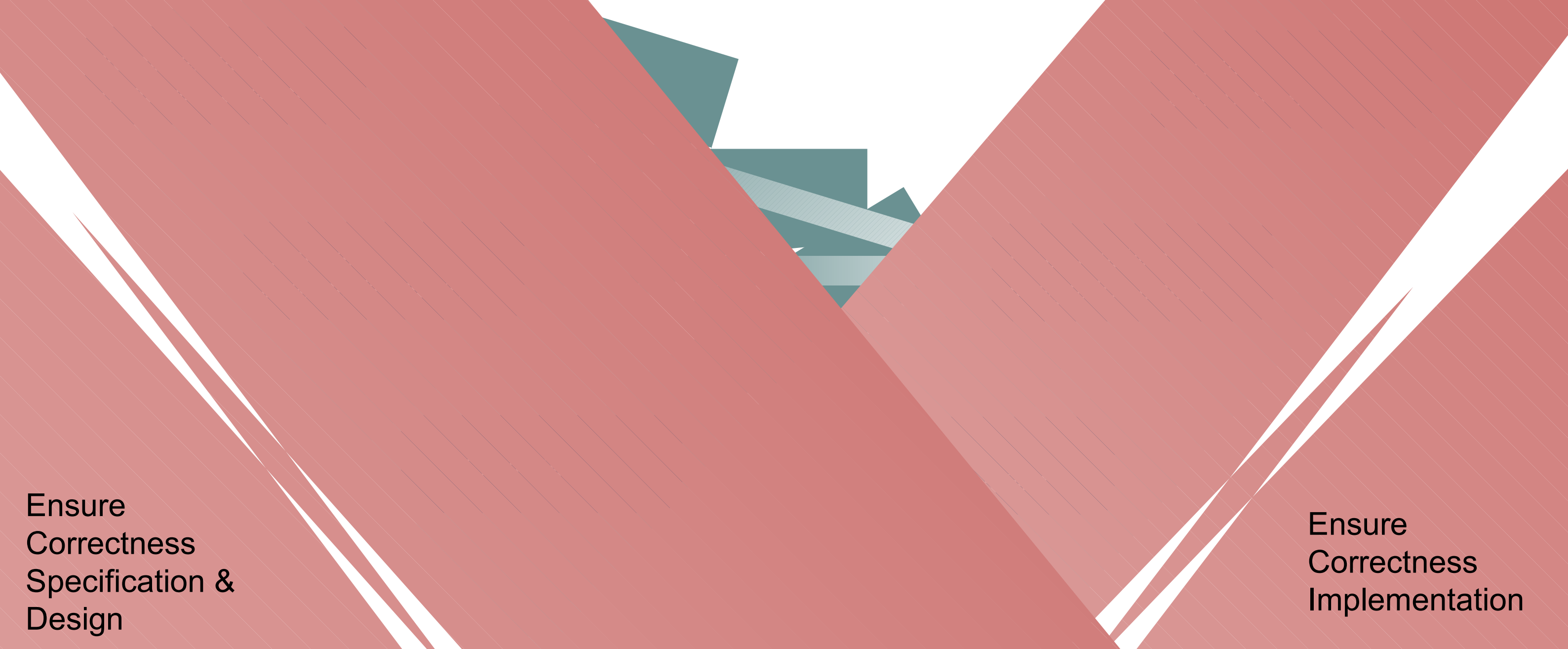
Detailed Results	
SD instance 'check stopwatch initialisation'	
Iterations:	1
Status:	passed
Progress:	100% (1/1)
SD instance 'check stopwatch start'	
Iterations:	1
Status:	passed
Progress:	100% (8/8)
SD instance 'check stopwatch stop'	
Iterations:	1
Status:	passed
Progress:	100% (3/3)
SD instance 'check stopwatch restart'	
Iterations:	1
Status:	passed
Progress:	100% (7/7)
SD instance 'check stopwatch reset'	
Iterations:	1
Status:	passed
Progress:	100% (3/3)

- ❑ C
- ❑ D
  S
- ❑ Develop Test Ca
- ❑ Develop Test Ca
- ❑ Execute/Report on Test Execution
  - ❑ Inputs to SUT and stubs behaviours are played out automatically
  - ❑ Unexpected behaviours are highlighted
  - ❑ Test Execution Reports can be customized to match company/project standards

t with Rational Rhapsody TestConductor A

definition for improved collaboration
est execution, monitoring and test a

of requirements during system
d regression testing helpin e



Model-Based Development and Testing of safety related Software

Two main topics in the remainder of this discussion:

- Functional safety is influenced by the development process (ISO 26262, part 6, Introduction)

  Using MBD/MBT and tools like IBM Rational Rhapsody require some guidance for users to enable usage of MBD/MBT and Rational Rhapsody in safety related processes

  „**Rhapsody Reference Workflow for the development of safety related software**" provides such guidance for users

- Confidence in the use of software tools is needed (ISO 26262, part 8 chapter 11)

  Qualification of software tools shall be performed if necessary

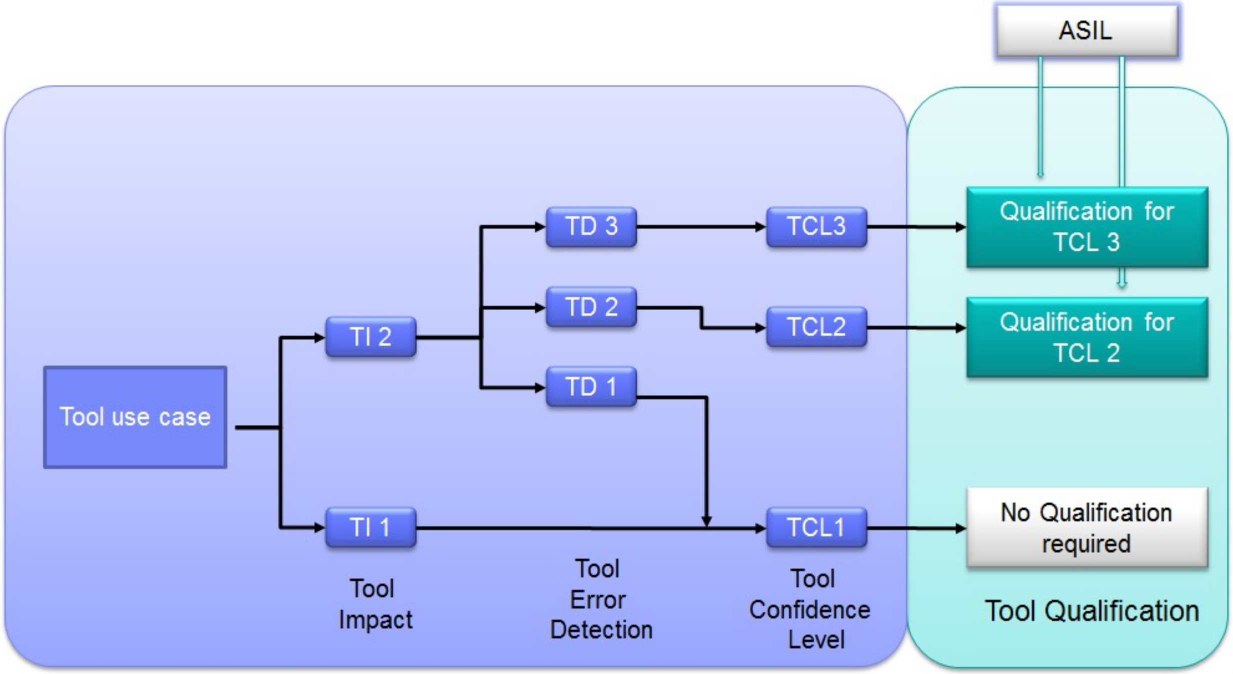
Qualification of Software Tools: Overview

- ISO 26262, part 8 chapter 11, “Confidence in the use of software tools”
  - Provides criteria to determine the required level of confidence in a software tool
  - Provide means for the qualification of a software tool
- Confidence is needed that the software tool effectively achieves the following goals:
  - The risk of systematic faults in the developed product due to malfunctions of the software tool leading to erroneous outputs is minimized
  - The development process is adequate with respect to compliance with ISO 26262, if activities or tasks required by ISO 26262 rely on the correct functioning of the software tool used
- It must be performed for individual tools, tool chains, or tool functions

Determining the Required Level of Confidence I

- To determine the required level of confidence in a software tool used within development under the conditions mentioned above, the following criteria are evaluated:
  - The possibility that the malfunctioning software tool and its corresponding erroneous output can introduce or fail to detect errors in a safety related item or element being developed, and
  - The confidence in preventing or detecting such errors in its corresponding output
- Tool Confidence Level (TCL) is based upon
  - Impact of tool failure (TI)
  - Level of Tool error detection (TD)
- TCL when combined with ASIL leads to methods for tool qualification

Determining the Required Level of Confidence II



Tool Impact = 2: the tool might have an impact on safety

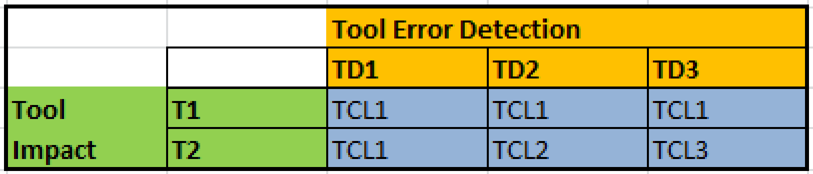
Tool Error Detection = 2 or 3: errors and malfunctions are not detected with sufficient confidence in a given process

Tool Confidence Level = 2 or 3: Qualification of a tool or feature is needed

Concrete tool qualification requirement depend on the ASIL level of the product under development

Determining the Required Level of Confidence III

The required software Tool Confidence Level shall be determined according to the following table



		Tool Error Detection		
		TD1	TD2	TD3
Tool	T1	TCL1	TCL1	TCL1
Impact	T2	TCL1	TCL2	TCL3

Examples for tools and functions

Simulation, automatic source code generation, test specification, test execution

Examples for preventing or detecting errors

Prevention or detection can be accomplished through process steps, redundancy in tasks or software tools or by rationality checks within the software tool itself

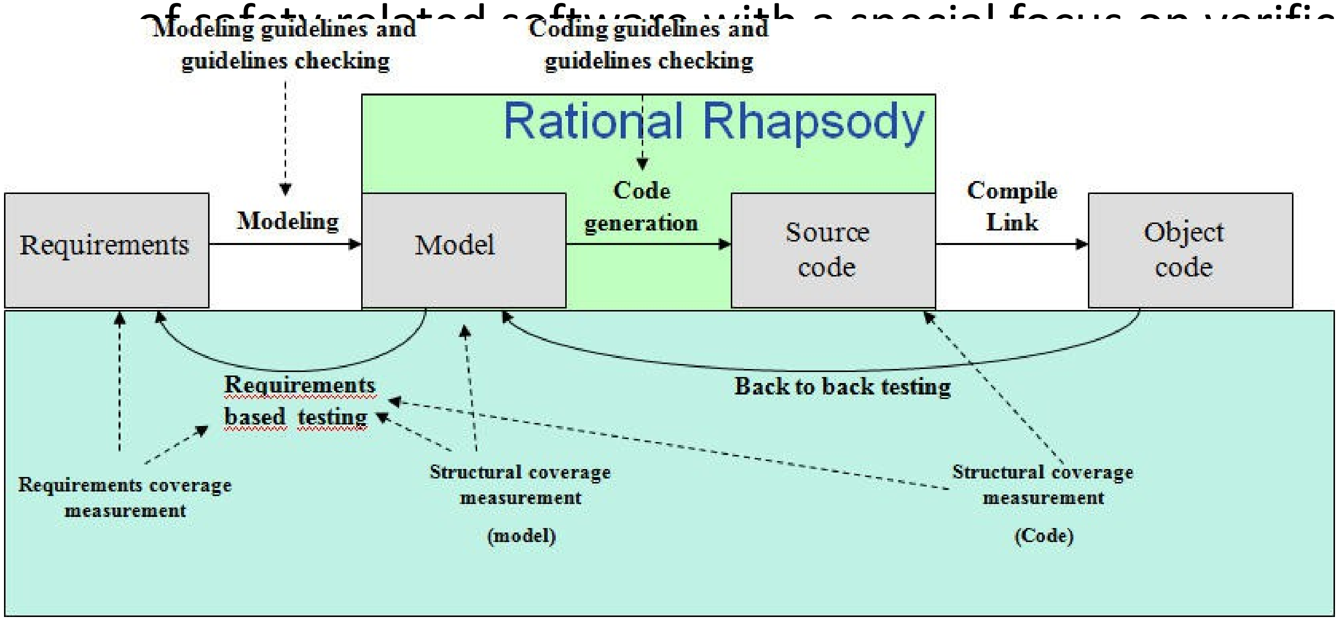
Auto Code Generator Example (IBM® Rational® Rhapsody®)

TD1 can be chosen for a code generator in case the produced source code is

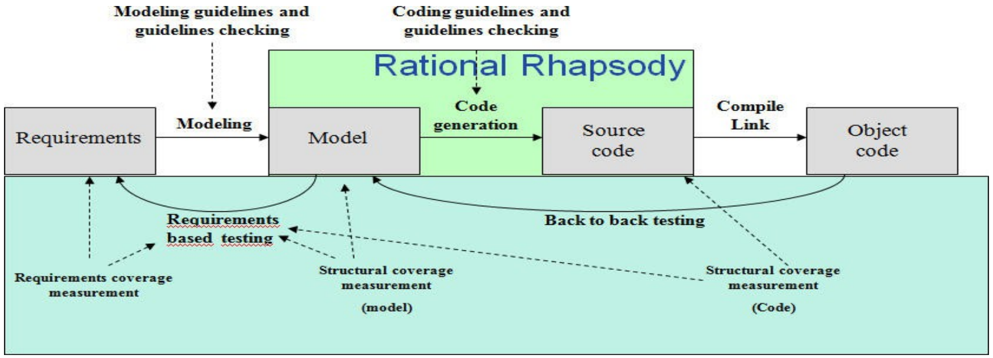
IBM Rational Rhapsody Reference Workflow: Overview

Rhapsody Reference Workflow for the development of safety related software

- provides guidance on how to fulfill functional safety requirements with model-based development methods and tools
- is based on best practices for safety related projects
- addresses various workflow activities relevant for the development of safety related software with a special focus on verification and


Modeling guidelines and guidelines checking
Coding guidelines and guidelines checking
Rational Rhapsody
Requirements
Modeling
Model
Code generation
Source code
Compile Link
Object code
Requirements based testing
Back to back testing
Requirements coverage measurement
Structural coverage measurement (model)
Structural coverage measurement (Code)

IBM Rational Rhapsody Reference Workflow: Workflow activities

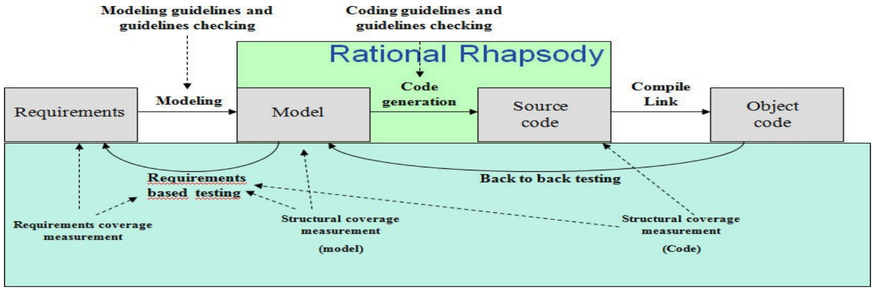

Modeling guidelines and guidelines checking
Coding guidelines and guidelines checking
Rational Rhapsody
Requirements
Modeling
Model
Code generation
Source code
Compile Link
Object code
Requirements based testing
Back to back testing
Requirements coverage measurement
Structural coverage measurement (model)
Structural coverage measurement (Code)

- Software modeling and Requirements traceability
  - Modeling guidelines and guideline checking
- Model verification
  - Requirements based testing, Requirements coverage, Model coverage
- Code generation and Rhapsody frameworks
  - Coding guidelines and guideline checking
- Code verification

  Back to back testing, Code coverage

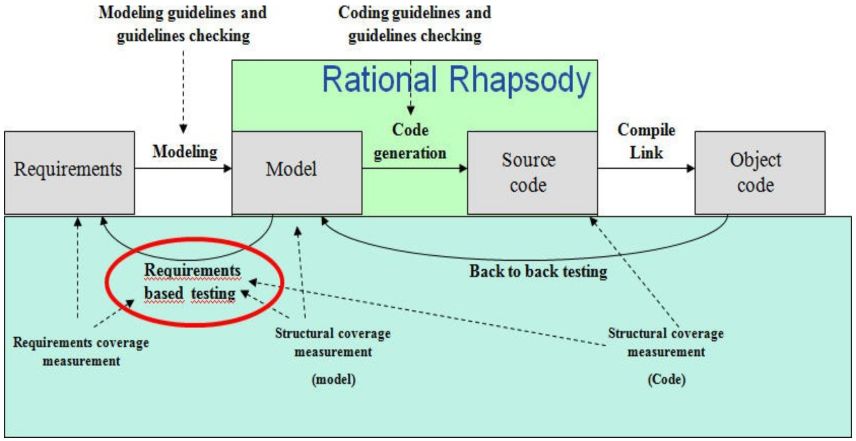
<u>Note</u>: ISO 26262 highly recommends Back to back testing to ensure that the behavior of the model units with regard to the test objectives is

Software modeling and Requirements traceability


Modeling guidelines and guidelines checking
Coding guidelines and guidelines checking
Rational Rhapsody
Requirements
Modeling
Model
Code generation
Source code
Compile Link
Object code
Requirements based testing
Back to back testing
Requirements coverage measurement
Structural coverage measurement (model)
Structural coverage measurement (Code)

- Given requirements are translated into an executable Rational Rhapsody model with appropriate modeling guidelines
    - Modeling guidelines shall be enforced and verified that they are met
- Of particular importance is traceability
- Tool supported (Rational Rhapsody) traceability information can be used to automatically generate traceability and coverage reports

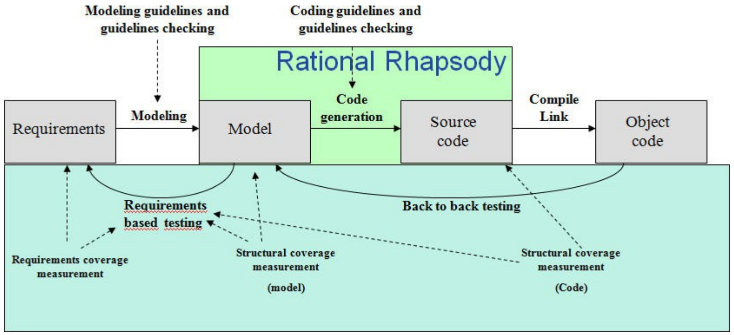
Model verification


Modeling guidelines and guidelines checking
Coding guidelines and guidelines checking
Rational Rhapsody
Requirements
Modeling
Model
Code generation
Source code
Compile Link
Object code
Requirements based testing
Back to back testing
Requirements coverage measurement
Structural coverage measurement (model)
Structural coverage measurement (Code)

Created Rational Rhapsody model is verified against the underlying requirements

- Model in the Loop Simulation (MiL Simulation) using Rational Rhapsody model animation
  - User guided (interactive) simulation of the model through different scenarios
- Requirements based testing: highly recommended by ISO 26262 for all ASIL levels
  - Rational Rhapsody TestConductor Add-On can be used to systematically test the correct implementation of the underlying requirements

Code generation and Rhapsody frameworks


Modeling guidelines and guidelines checking
Coding guidelines and guidelines checking
Rational Rhapsody
Requirements
Modeling
Model
Code generation
Source code
Compile Link
Object code
Requirements based testing
Back to back testing
Requirements coverage measurement
Structural coverage measurement (model)
Structural coverage measurement (Code)

High quality and automatic C/C++ code generation for the software model

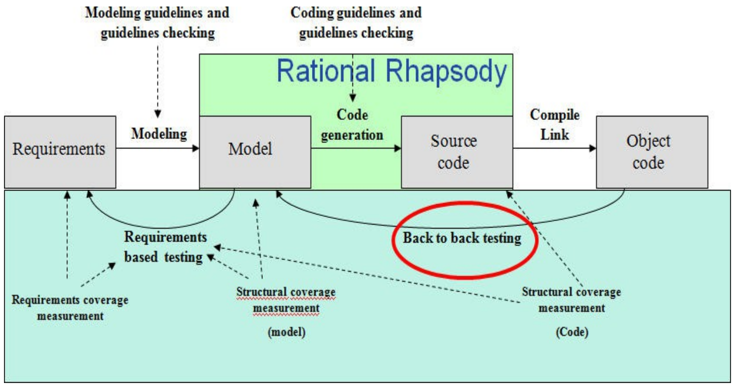
Using an execution framework coming along with Rhapsody

OXF: standard framework

SXF: simplified framework for safety critical C++

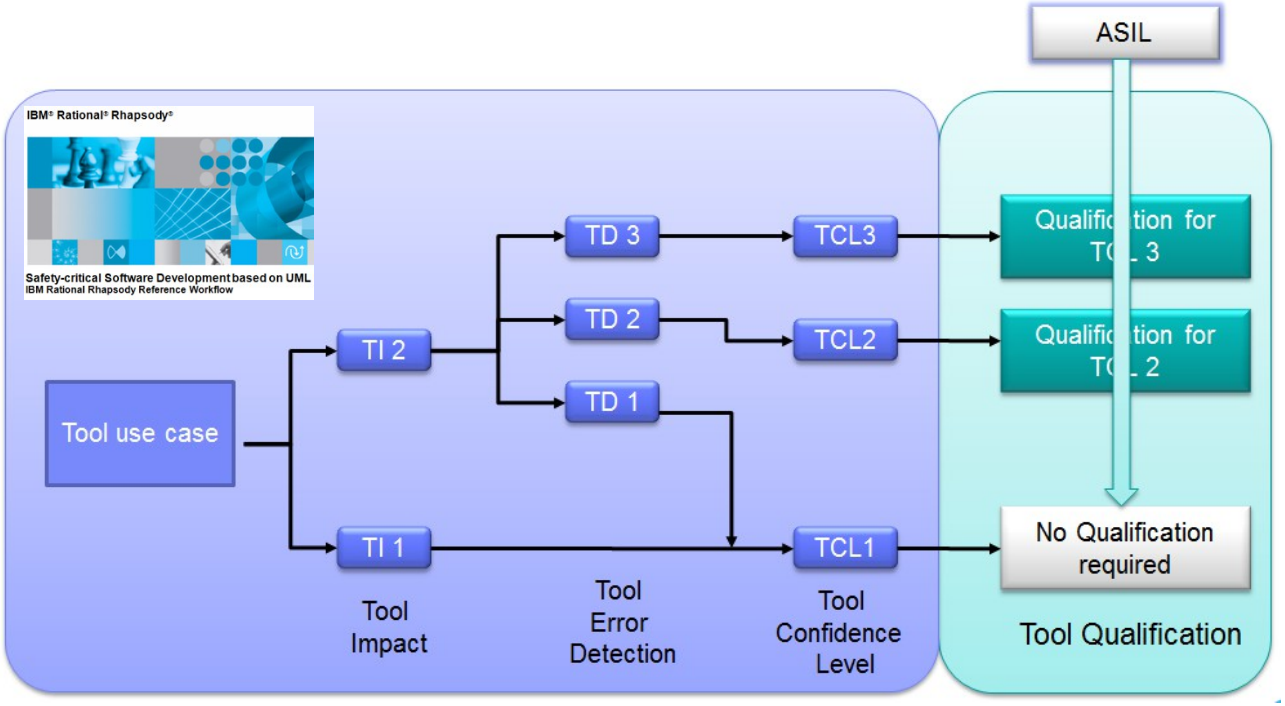
SMXF: simplified framework for safety critical C

Code verification


Modeling guidelines and guidelines checking
Coding guidelines and guidelines checking
Rational Rhapsody
Requirements
Modeling
Model
Code generation
Source code
Compile Link
Object code
Requirements based testing
Back to back testing
Requirements coverage measurement
Structural coverage measurement
(model)
Structural coverage measurement
(Code)

- With Rational Rhapsody the developed/generated production code can be tested on host computers and also on target machines
- Back to back testing is a technique to verify if MiL, SiL, and PiL execution show equivalent behavior
- Rational Rhapsody and Rational Rhapsody TestConductor provide automation to perform back to back testing

Rhapsody Code Generator Qualification for ISO 26262



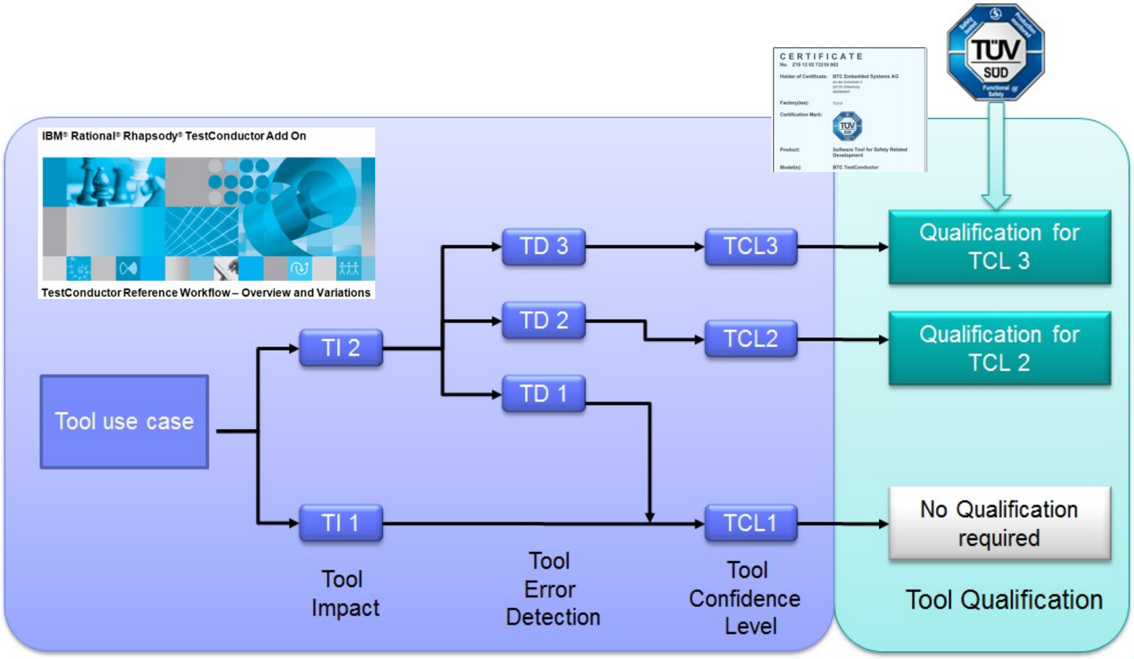
Rhapsody Tool Impact = 2: the tool might have an impact on safety

Rhapsody Tool Error Detection =1: errors and malfunctions are detected with sufficient confidence in a given process according to the *Rhapsody Reference Workflow*

Tool Confidence Level = 1 (TCL1)

Qualification of Rhapsody code generation is not needed!

Rhapsody TestConductor Qualification for ISO 26262



CERTIFICATE
No. Z10 12 02 73219 002

Holder of Certificate:	BTC Embedded Systems AG [illegible] 26135 Oldenburg GERMANY
Factory(ies):	73219
Certification Mark:	TÜV SÜD
Product:	Software Tool for Safety Related Development
Model(s):	BTC TestConductor

TestConductor Tool Impact = 2: the tool might have an impact on safety

TestConductor Tool Error Detection =3: errors and malfunctions are not detected with sufficient confidence in a given process according to the Rhapsody Reference Workflow

Tool Confidence Level = 3 (TCL3): Qualification TestConductor needed!

Qualification: Validation of Software Tool

- Süd Germany issued a certificate about successful qualification

IBM Rational Rhapsody Kit for ISO 26262 and IEC 61508

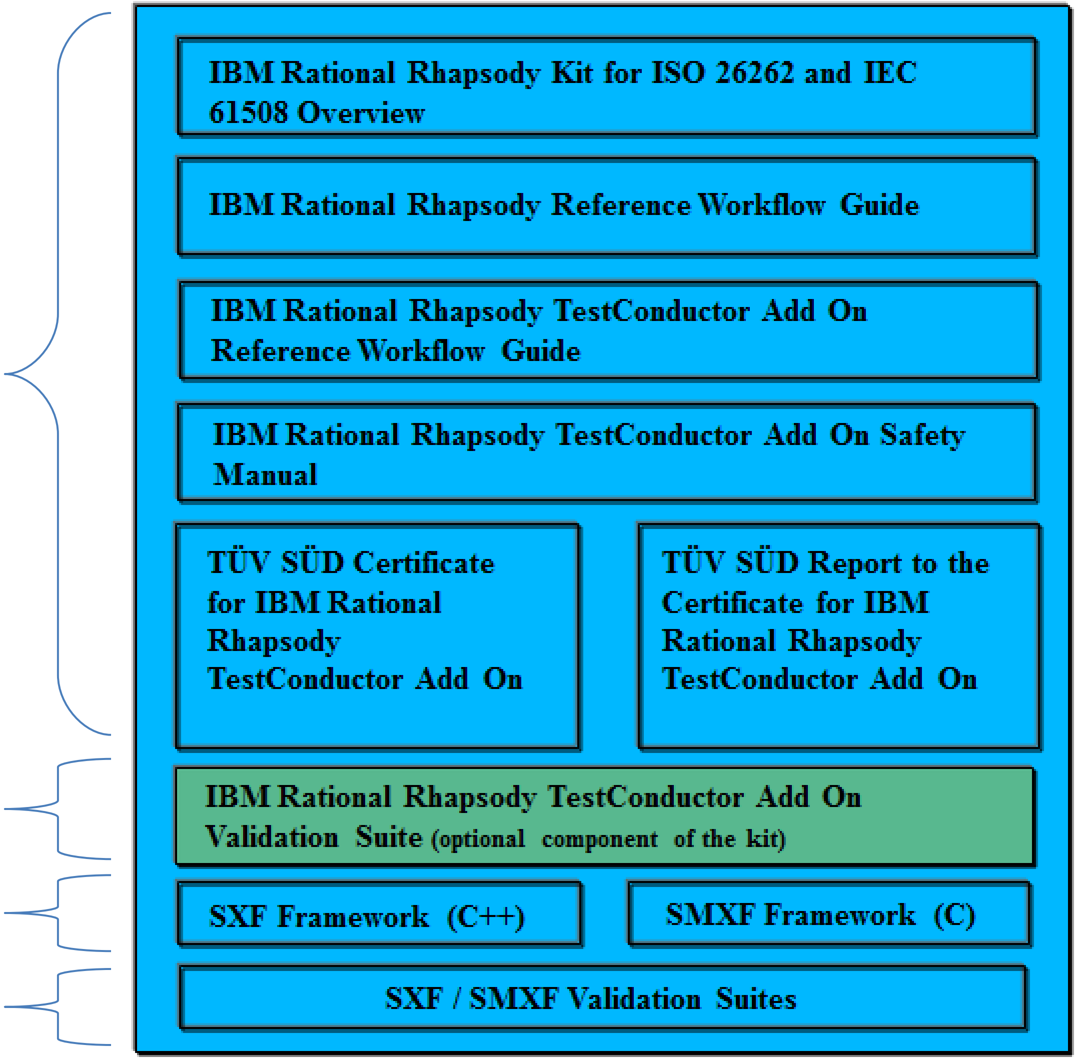
Guidance Documents

Validation Suite for

TestConductor Tool Qualification

C/C++ safety frameworks

Framework Validation Suites



IBM Rational Rhapsody Kit for ISO 26262 and IEC 61508 Overview

IBM Rational Rhapsody Reference Workflow Guide

IBM Rational Rhapsody TestConductor Add On Reference Workflow Guide

IBM Rational Rhapsody TestConductor Add On Safety Manual

TÜV SÜD Certificate for IBM Rational Rhapsody TestConductor Add On

TÜV SÜD Report to the Certificate for IBM Rational Rhapsody TestConductor Add On

IBM Rational Rhapsody TestConductor Add On Validation Suite (optional component of the kit)

SXF Framework (C++)

SMXF Framework (C)

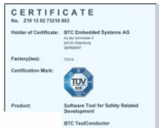
SXF / SMXF Validation Suites

Summary

- Model-based development of safety-relevant software is applied in the industry
- IBM Rational Rhapsody Reference Workflow based on best practice industry experiences provides guidance on the application of model-based development for safety related systems
- ISO 26262 defines a new approach to answer the question for software tool qualification
    - Approach has been successfully applied

IBM® Rational® Rhapsody®

Safety-critical Software Development based on UML
IBM Rational Rhapsody Reference Workflow



CERTIFICATE
No. Z10 12 02 73219 002

Holder of Certificate: BTC Embedded Systems AG
An der Schmiede 4
26135 Oldenburg
GERMANY

Factory(ies): 73219

Certification Mark:

Product: Software Tool for Safety Related Development

BTC TestConductor

IBM® Rational® Rhapsody® TestConductor Add On

TestConductor Reference Workflow – Overview and Variations

IBM Rational Rhapsody
IBM Rational Rhapsody TestCondcutor AddOn

ISO 26262

IEC 61508

www.ibm.com/software/rational

BTC

The Testing Future is Test Automation