



Quality Management

Alberto Sardini – Senior Systems Specialist

alberto_sardini@it.ibm.com



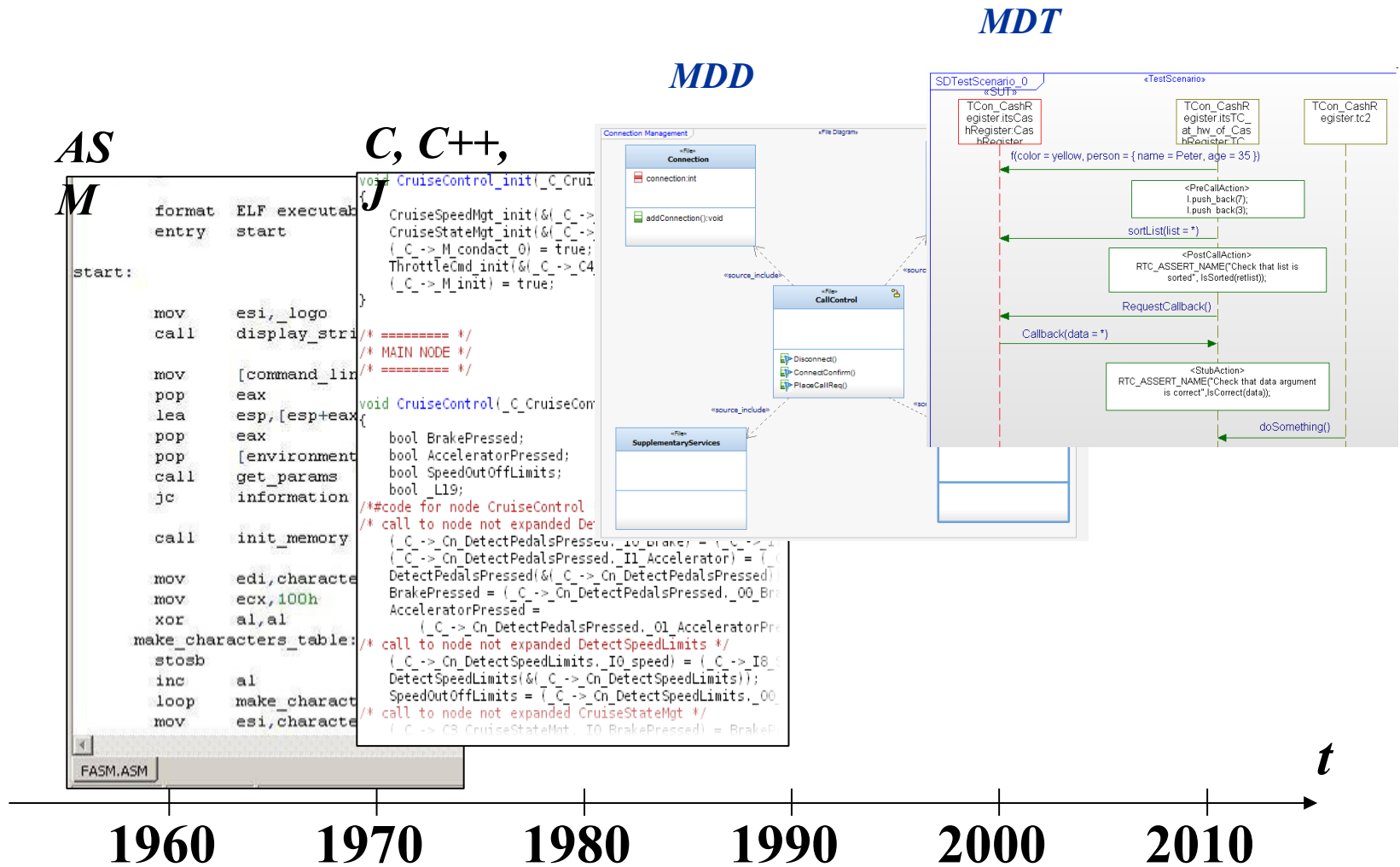
Agenda

- Development and Test Process
- Key Enabler for Model Based Test
- Rational Solution for System and Software Engineering

Agenda

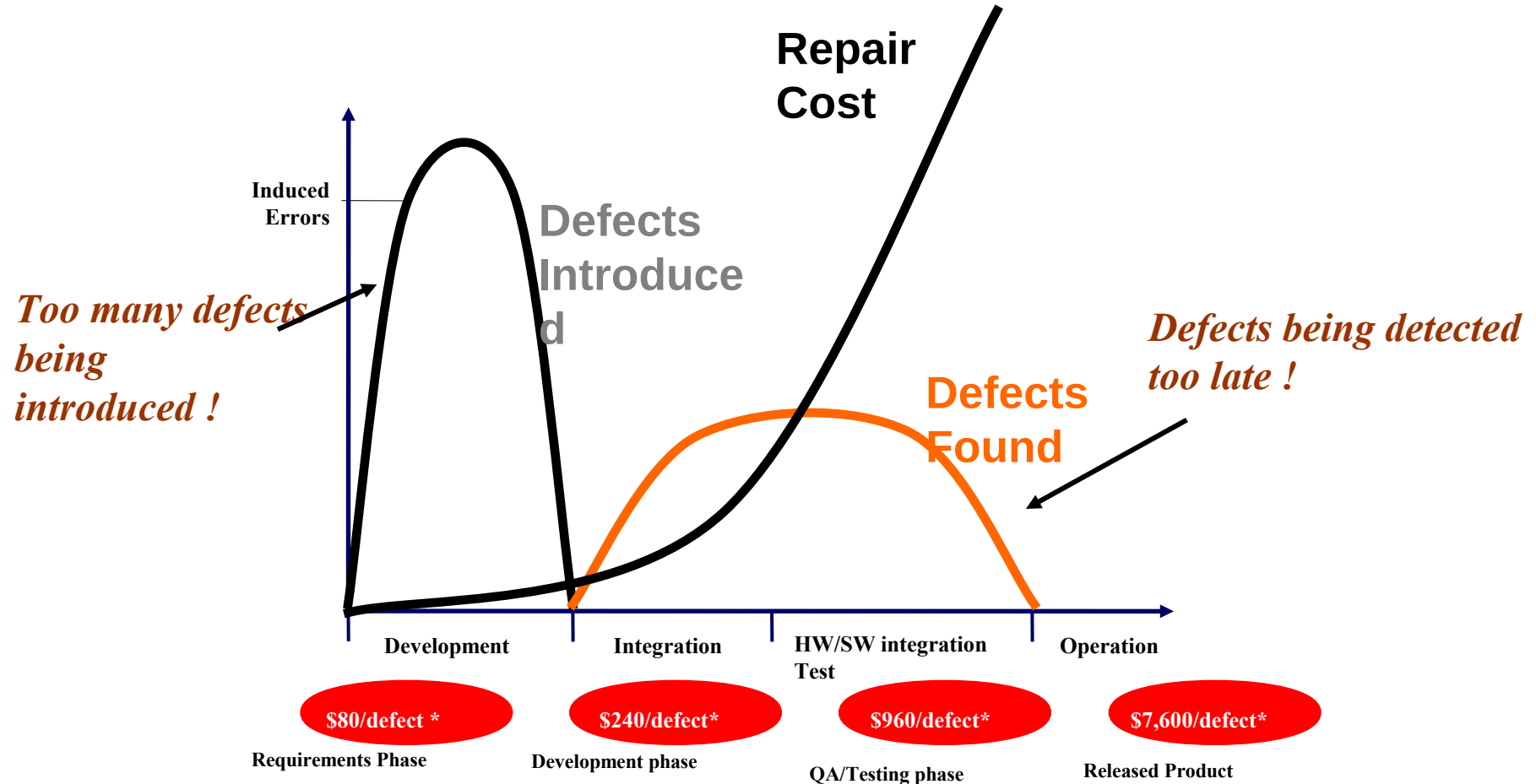
- **Development and Test Process**
- Key Enabler for Model Based Test
- Rational Solution for System and Software Engineering

Development Process is Evolving...



Traditional testing methodologies are insufficient

80% of development costs are spent identifying and fixing defects



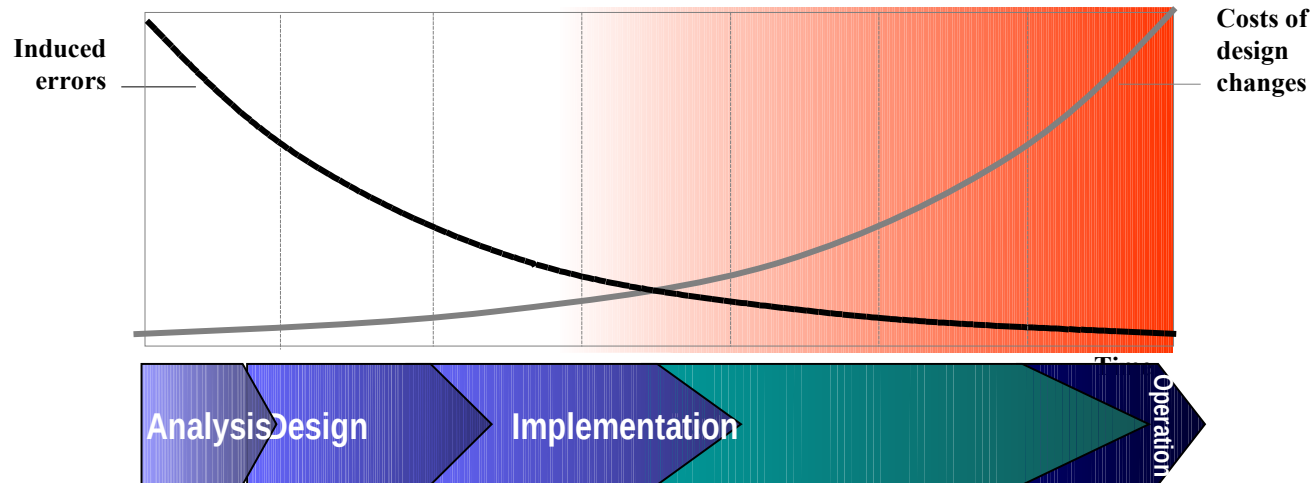
*Source: 2008 GBS Industry standard study

Defect cost derived in assuming it takes 8 hrs to find, fix and repair a defect when found in code and unit test.

Defect FFR cost for other phases calculated by using the multiplier on a blended rate of \$80/hr

Current practices....

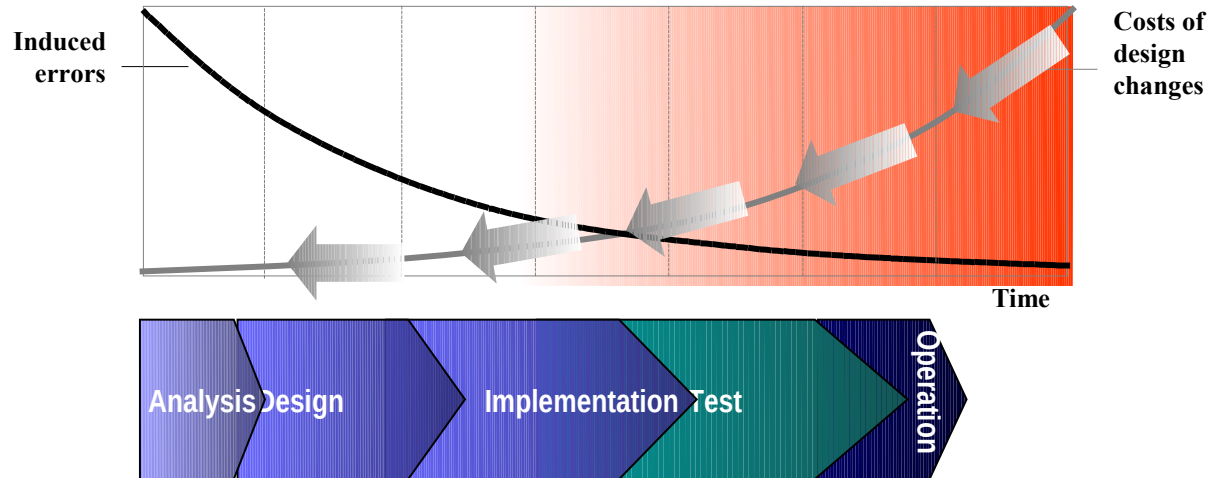
- Implement code from **textual requirements**
- Test only on target **late in development cycle**
 - Defects introduced early but detected late during testing and validation
- Large part of development time spent manually testing
 - Difficult to communicate script/code tests to others and not traced to requirements



An optimal approach would be to fix errors as they are introduced, thereby helping to reduce your cost of development.

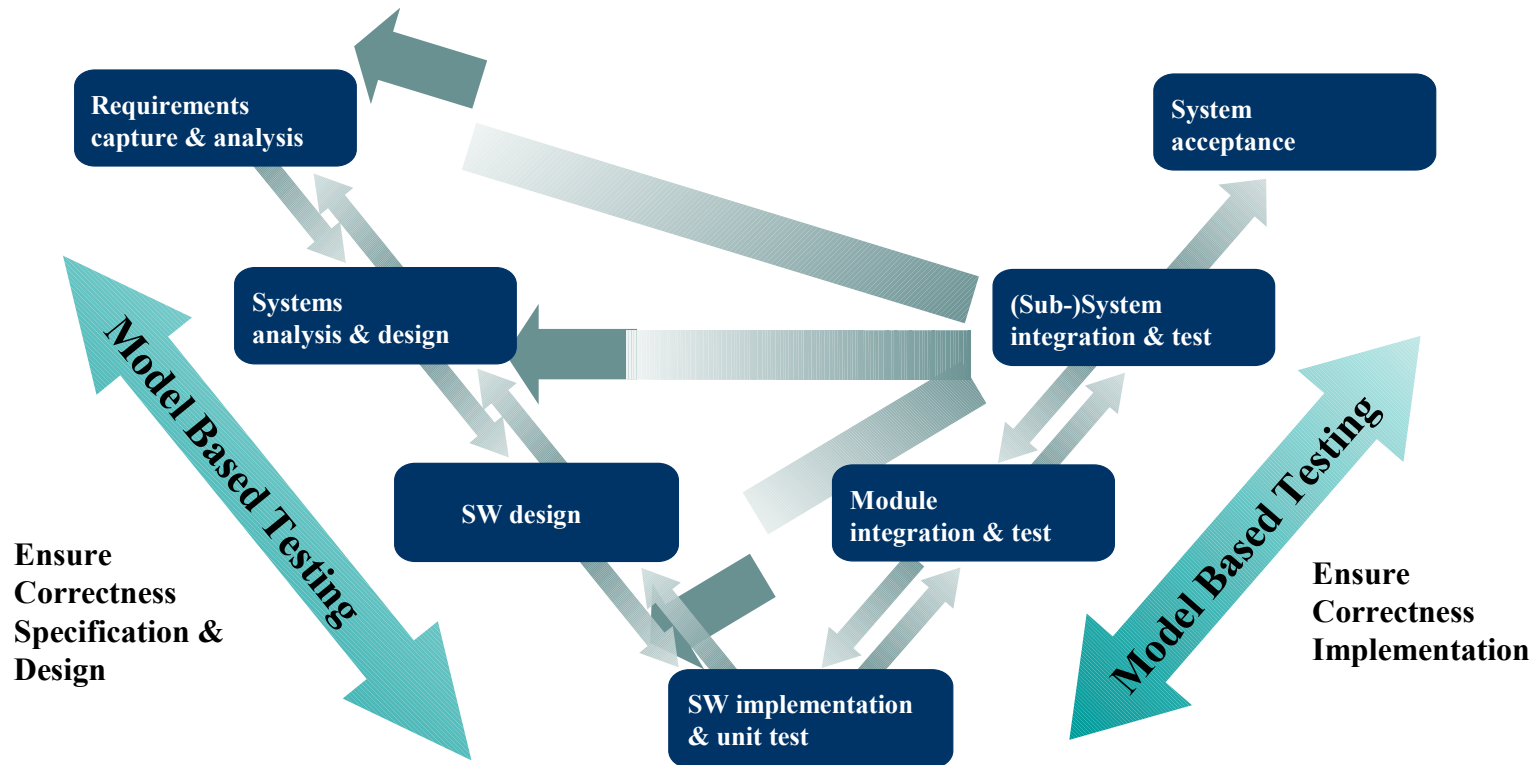
Quality by Design - “shift left” to identify and fix errors early

The Benefits of Model Driven Development with Model Based Testing



Model Based Testing

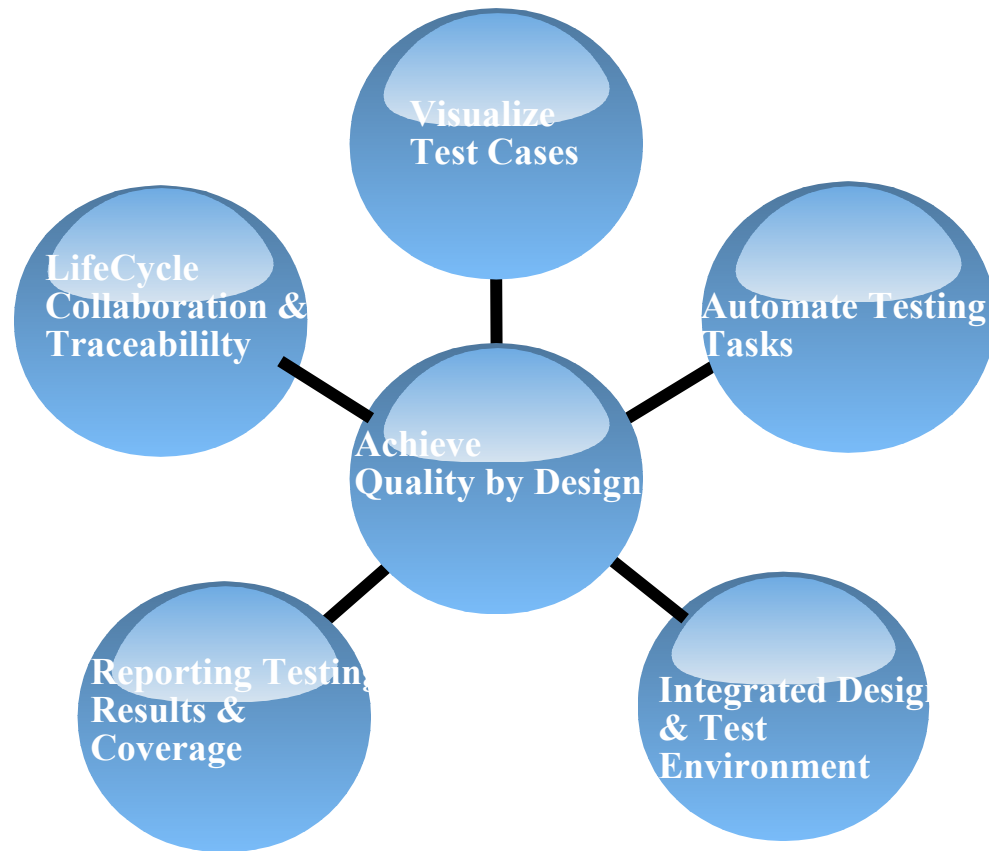
- **Visual test definition for improved collaboration**
- **Automated test execution, monitoring and test architecture creation**
- **Early validation of requirements during systems engineering**
- **Automate unit and regression testing helping improve software quality**



Agenda

- Development and Test Process
- **Key Enabler for Model Based Test**
- Rational Solution for System and Software Engineering

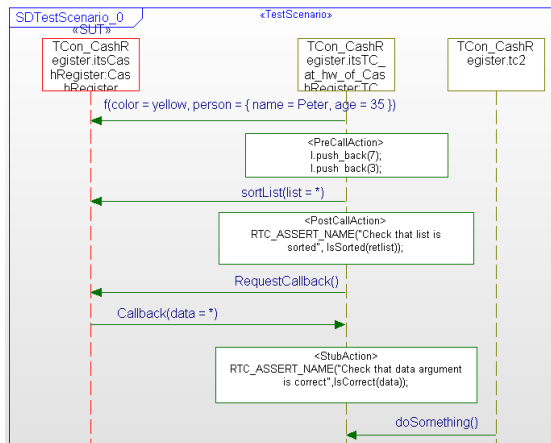
Key Enablers of Model Based Testing



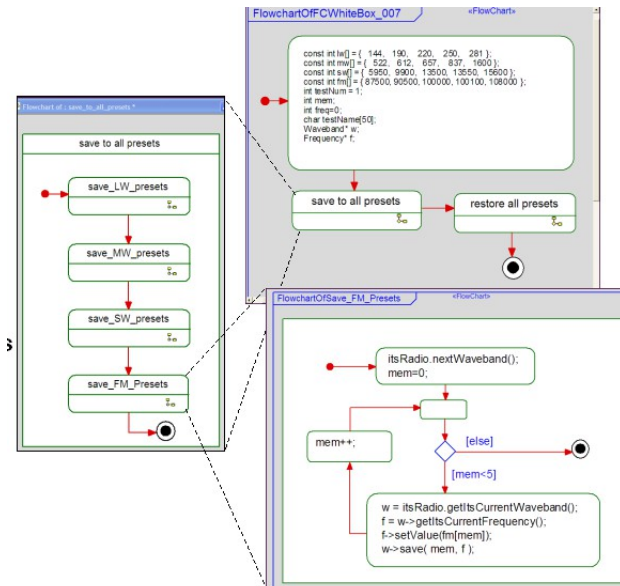
Capture test cases with UML

Visualize
Test Cases

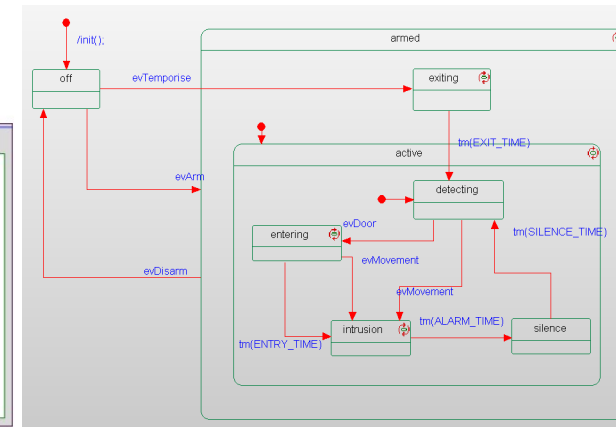
- Based on OMG's standard UML Testing Profile (www.omg.org)
- Specify test cases visually for better communication across teams
- Creating code tests cases or importing Cunit/Cpp unit tests also possible



Sequence Diagram Test Case



Flowchart Test Cases

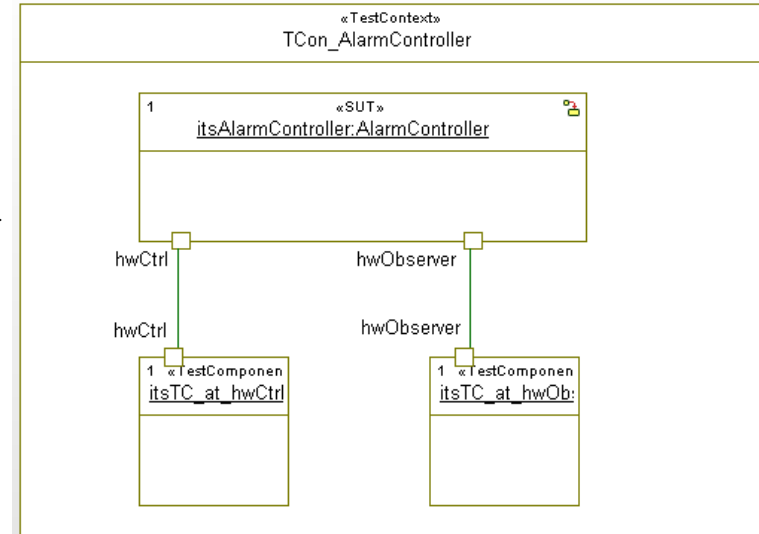
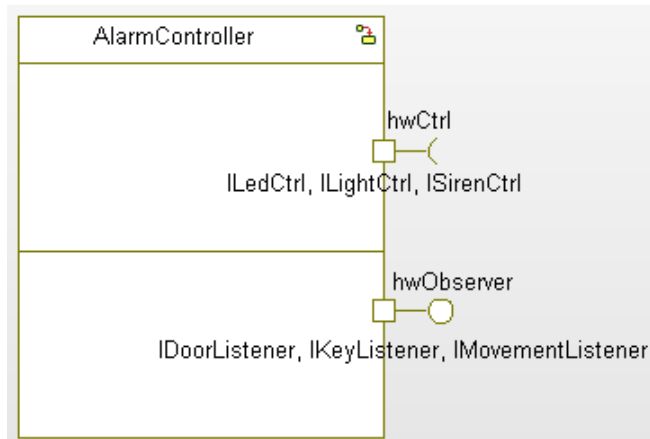


Statechart Test Case

Automate Quality

Automate
Testing Tasks

- **Automatically create test architecture**
 - **Creates a System Under Test (SUT), test components and test context**
- **Apply model based testing to external code**
 - **Code is developed outside of Rhapsody**
 - **Visualize code interfaces in Rhapsody and apply model based testing**



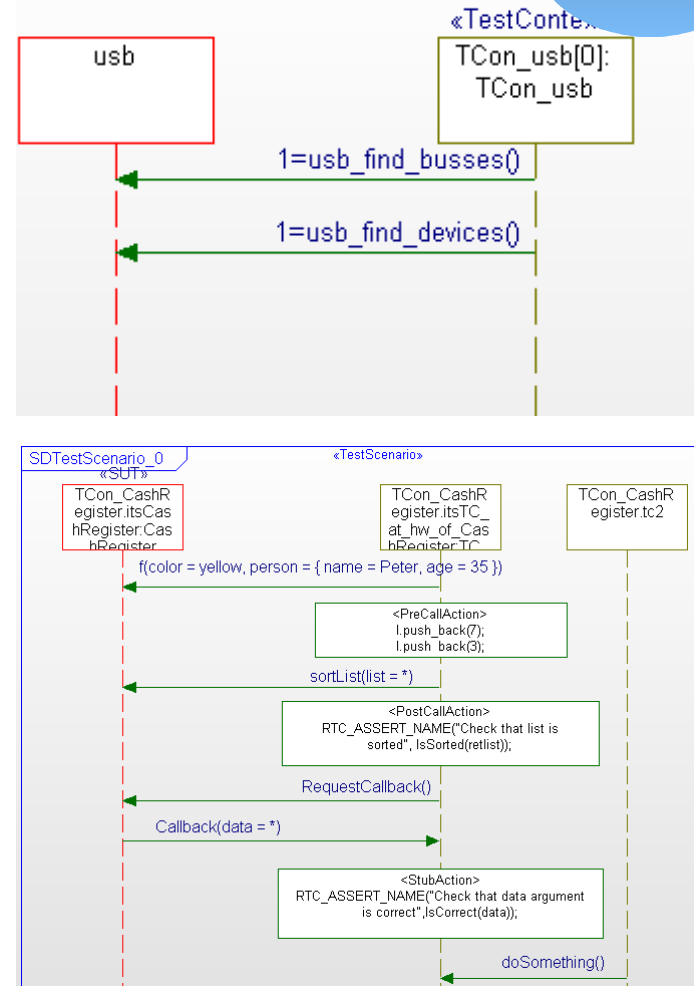
Automatically Created Test Architecture

Model-Based Testing

IBM Rational Rhapsody Test Conductor Add On

Automate
Testing Tasks

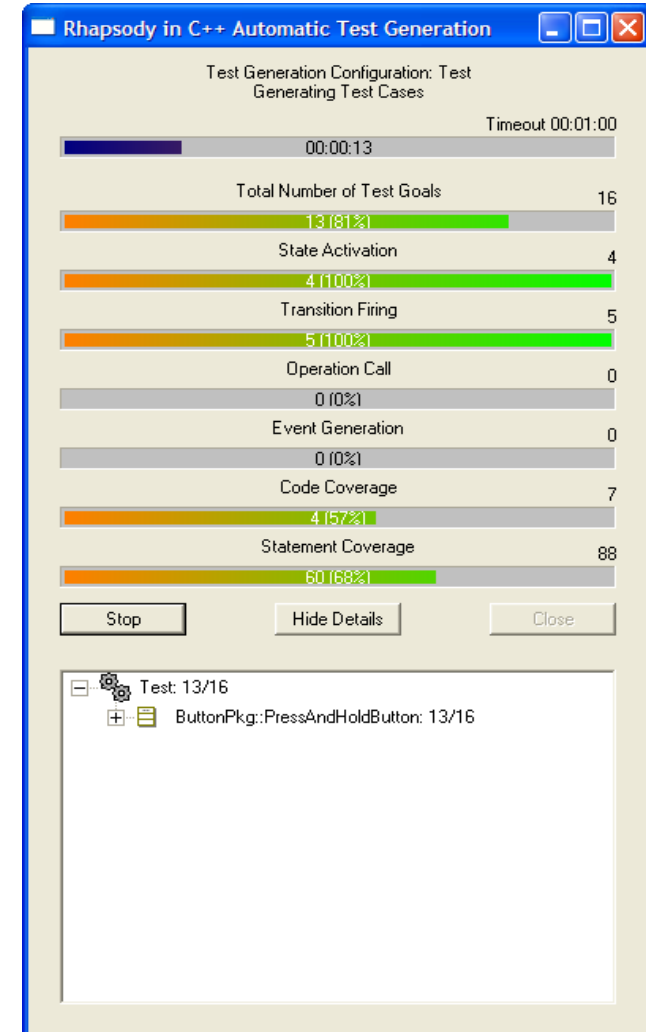
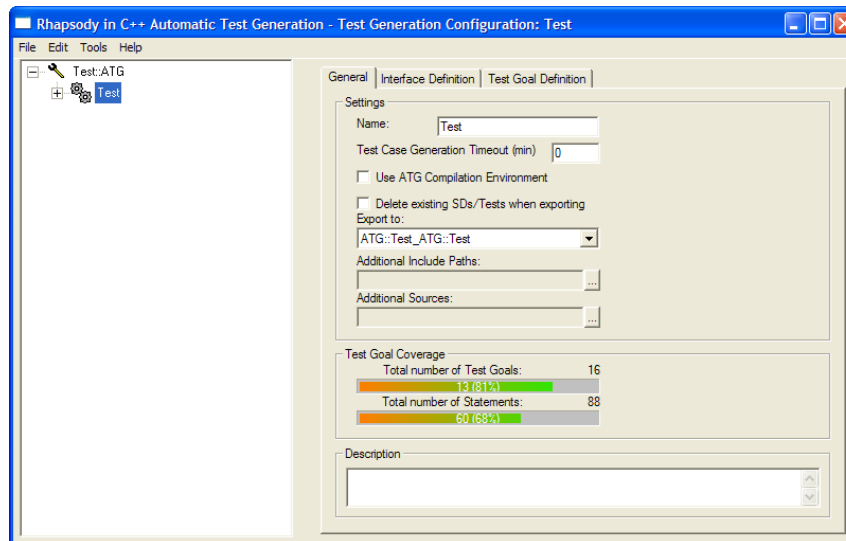
- ▶ Visually capture test cases for better communication
 - Define test cases with sequence diagrams, statecharts, flowcharts or even code
 - Avoid manual programming
- ▶ Automate testing tasks
 - Create test architecture
 - Execute and monitor tests
 - Interactive for debugging,
 - Batch test suites for nightly regression
 - Include CUnit/CppUnit tests
- ▶ Traceability across lifecycle – from requirements to integration
 - Integration with Rational lifecycle solution
- ▶ Host level and target based execution
 - White-box , black-box for design validation
 - “Offline testing” mode:–for testing on target
 - C++, C, Java, Ada Supported
- ▶ Definition and management of regression tests
- ▶ Reporting of results, coverage and traceability



Model Driven Test Generation

Automate
Testing Tasks

- **Automates generation of test cases for complete model and code coverage**
 - **Export test cases for scenario-based testing using Test Conductor**
 - **Supported for C++**
- **Automatically generates test cases with high coverage of the design**
 - **Model coverage: Covers states, transitions, operations, event generation**
 - **Code coverage: Generates all relevant combinations of inputs for MC/DC**



Code Verification with Rational Test RealTime integration

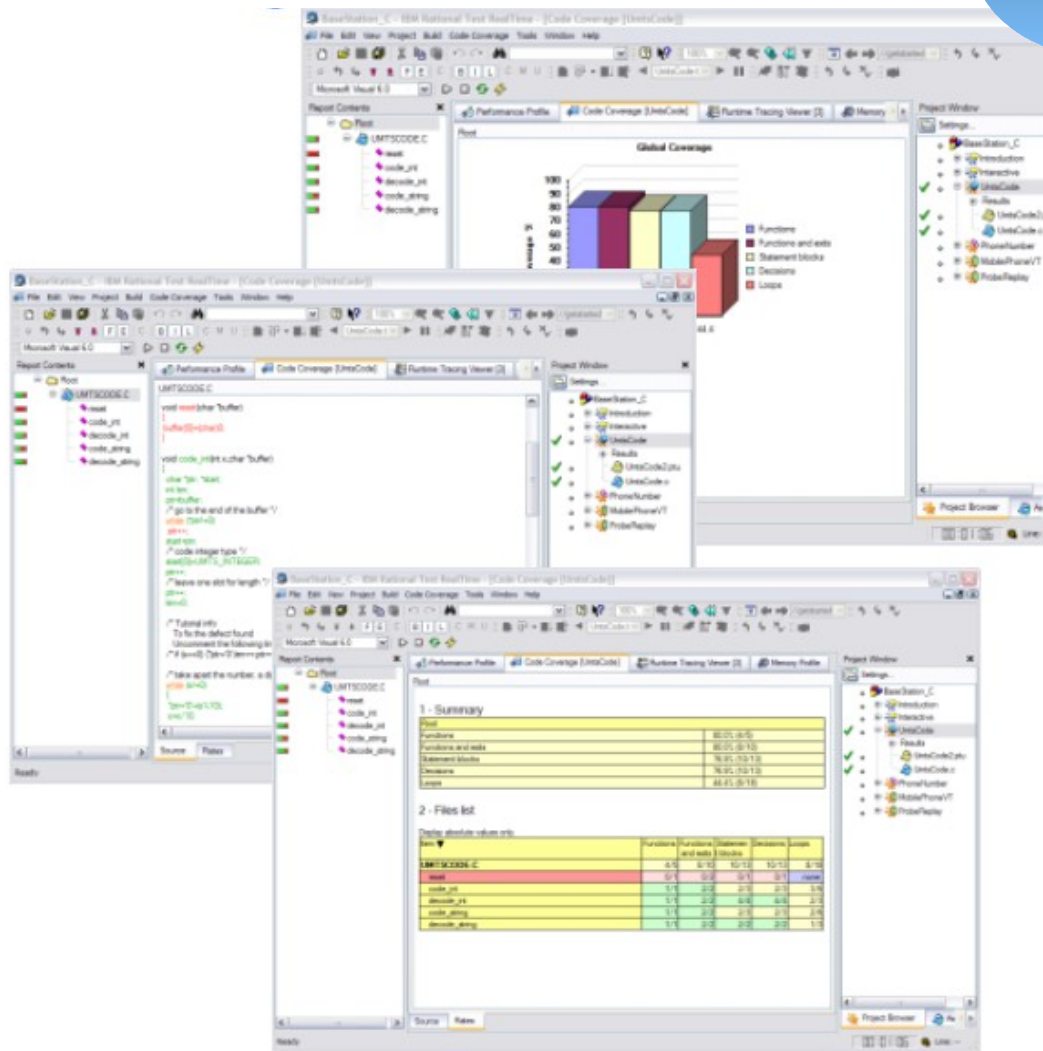
Automate Testing Tasks

Runtime Analysis:

Code Coverage

Performance Analysis

Memory Profiling

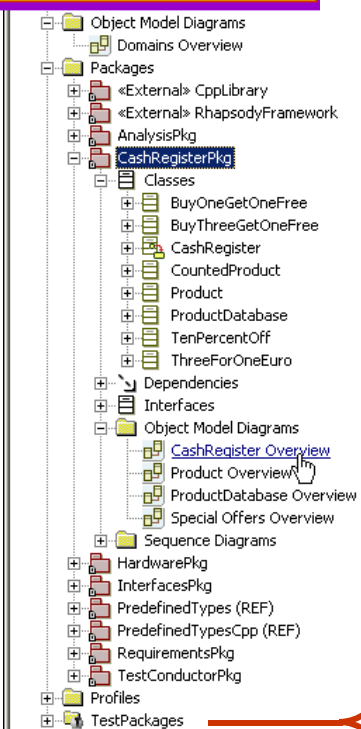


Integrated Design and Test Environment

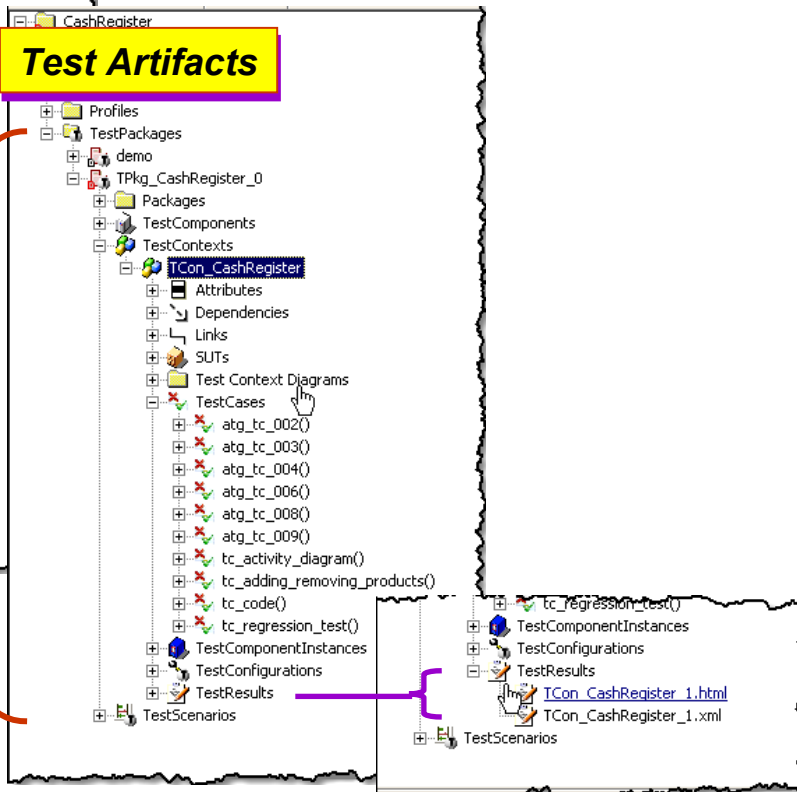
Manage test cases within Rational Rhapsody

Integrated Design
& Test
Environment

Design Artifacts



Test Artifacts



- **Common Browser for design and test information**
 - Syncs information to maintain consistency between design and test
- **Apply model based testing to external code**
 - Visualize interfaces in Rhapsody

Test Context Result

Test Execution Reports

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:	DefaultConfig

Test Context: TCon_CashRegister	Summary: PASSED
tc_code	PASSED
tc_activity_diagram	PASSED
tc_adding_removing_products	PASSED
tc_regression_test	PASSED
atg_tc_008	PASSED
atg_tc_009	PASSED
atg_tc_006	PASSED
atg_tc_002	PASSED
atg_tc_003	PASSED
atg_tc_004	PASSED

Requirements to Test Results Coverage

Reporting Testing Results & Coverage

- **Automated reporting of test results**
 - Requirement to test coverage table
 - Test Coverage results
 - Complete test results in the reports

To: Requirement	Scope: CashRegister
REQ1	
REQ2	
REQ3	
REQ4	
REQ5	
REQ6	
REQ7	
REQ8	
REQ9	
REQ10	

From: TestCase	Scope: CashRegister
SD_tc_0	REQ10
atg_tc_007	
atg_tc_006	
atg_tc_002	
atg_tc_003	REQ1
atg_tc_004	
atg_tc_016	
atg_tc_01	
FC_tc_0	
Code_tc_	
SD_tc_0	

All Requirements

Name	Specification	Covered by Test Case
REQ1	A small stand-alone Cash Register needs to be designed that reads barcodes of products that a Customer has selected.	atg_tc_003 (Passed)
REQ10	After receiving a start event Cash Register will send a message "show(Ready)" to its display.	SD_tc_0 (Failed)
REQ2	When a product has been identified, its name and price are displayed on a display.	not covered
REQ3	If the barcode cannot be read automatically then the message "Unknown product" will be displayed and the barcode can be entered via the Cashier's keyboard.	not covered
REQ4	When all the selected products have been read, a ticket is generated containing the list of all the selected products with the unit price, quantity and total price.	Code_tc_0 (Passed)

Rational Quality Manager

Home View Test Plans TestPlan_CashRegiste... TestCase_01_SD_InitC... Execution Result

Execution Result
Command Line Result

Test Case Result
Test Case: SD_tc_0
10:20:31, Monday, April 27, 2009

Actual Result: ✔ **Passed**

Host Name: jekyllslave
Owner: Mary, Test Manager

Test Milestone: TestCase_01_SD_InitCashRe
Test Case: SD_tc_0
Test Script: Unassigned
Test Data: Unassigned
Weight: 100

Environment Info

Test executed on machine:	JEKYLSSLAIVE
Test executed by user:	Administrator
Used OS version:	Windows 2000 / Windows XP
Used Rhapsody version:	7.5, build 1135117
Used TestConductor version:	2.4, build 1406

Tested Project

Project:	CppCashRegister
Active Component:	TRkg_CashRegister_Comp
Active Configuration:	DefaultConfig

SDs used in test

TRkg_CashRegister::SDTestScenario_0	
Summary Info Summary: passed	
Total number of SD instances used:	1
Total number of SD instances in test:	1
Total number of executed SD instances:	1
Total number of PASSED SD instances:	1 (100%)
Total number of FAILED SD instances:	0 (0%)
Total number of ACTIVE SD instances:	0 (0%)
Total number of NOT ACTIVE SD instances:	0 (0%)

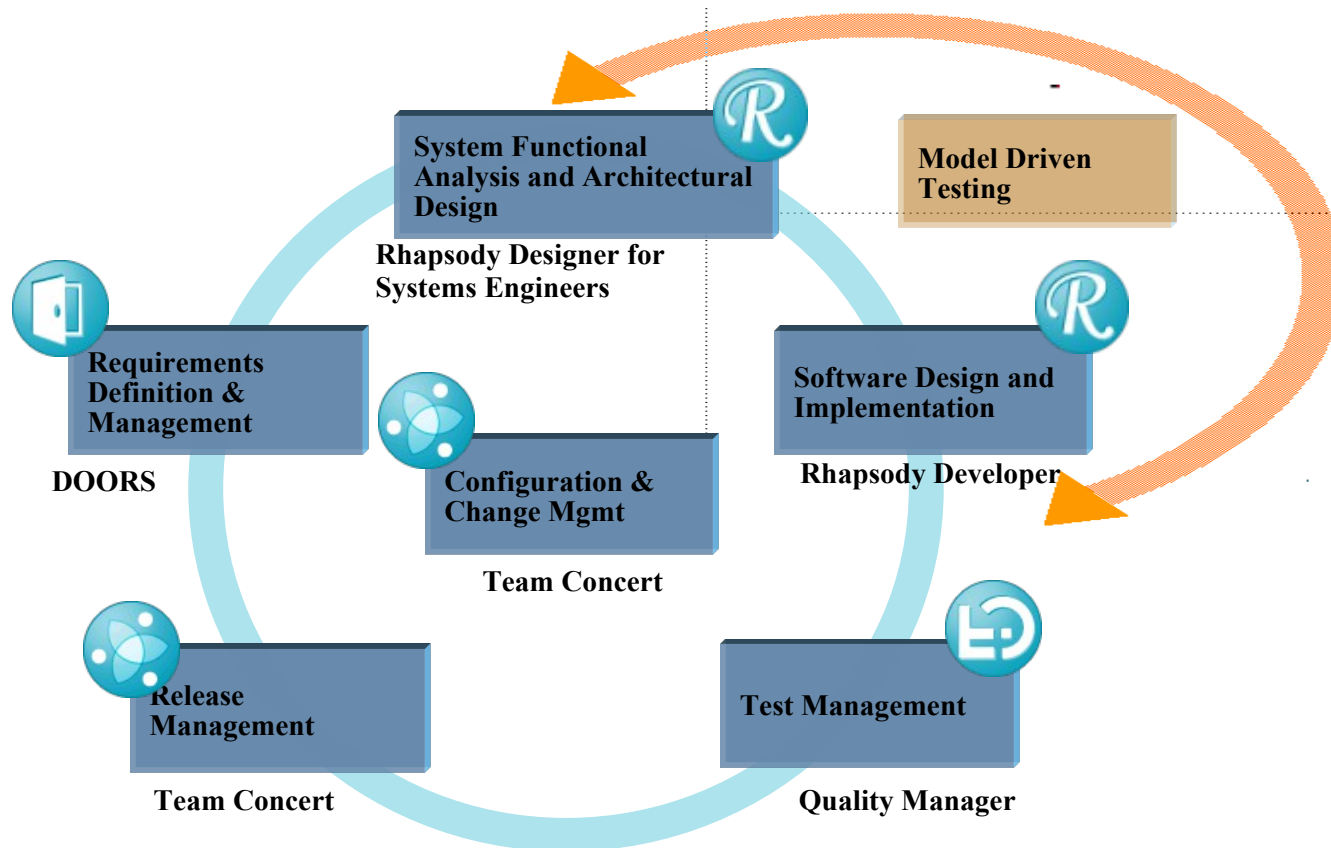
Result Details

TCon_CashRegister__SD_tc_0_0.html
TestConductorAdapter20844.out
TestConductorAdapter20845.err
TestLog20843.log

Name	Verdict
TCon_CashRegister__SD_tc_0_4.html	Failed
TCon_CashRegister__atg_tc_007_7.html	Passed
TCon_CashRegister__atg_tc_006_9.html	Passed
TCon_CashRegister__atg_tc_002_9.html	Passed
TCon_CashRegister__atg_tc_003_9.html	Passed
TCon_CashRegister__atg_tc_004_9.html	Passed
TCon_CashRegister__FC_tc_0_0.html	Passed
TCon_CashRegister__Code_tc_0_0.html	Passed
TCon_CashRegister_7.html	Failed

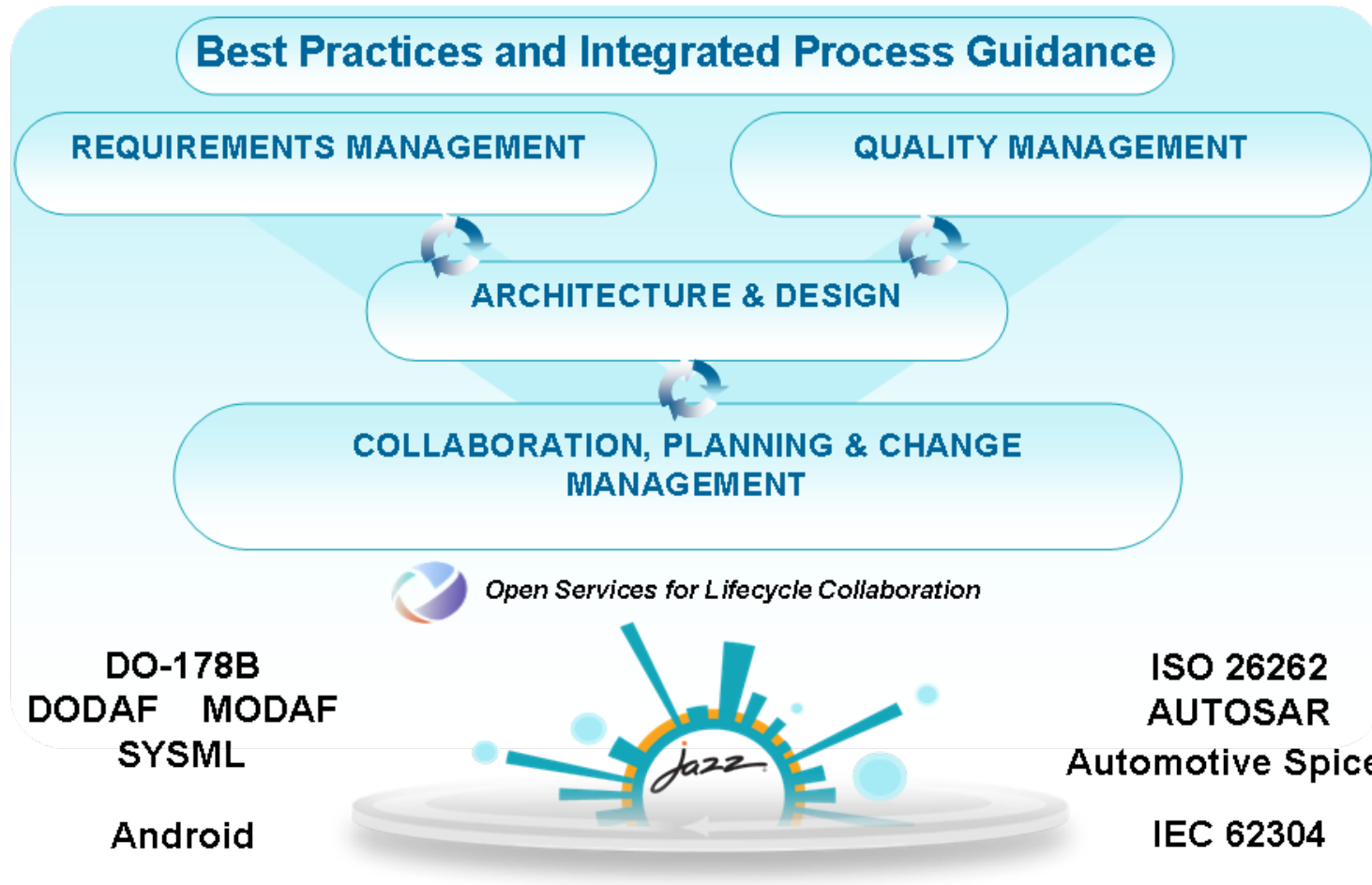
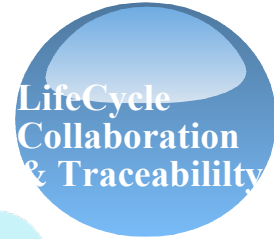
Model Based Test Spans the Development Lifecycle

- **Integrated traceability from requirements to design to test cases**
- **Manage execution of test cases and results reports using Rational Quality Manager**
- **Mitigate project risks with continuous QA statistics**



Unify Lifecycle Disciplines across Systems and Software Engineering

Best Practices, Tools and Services on an open platform



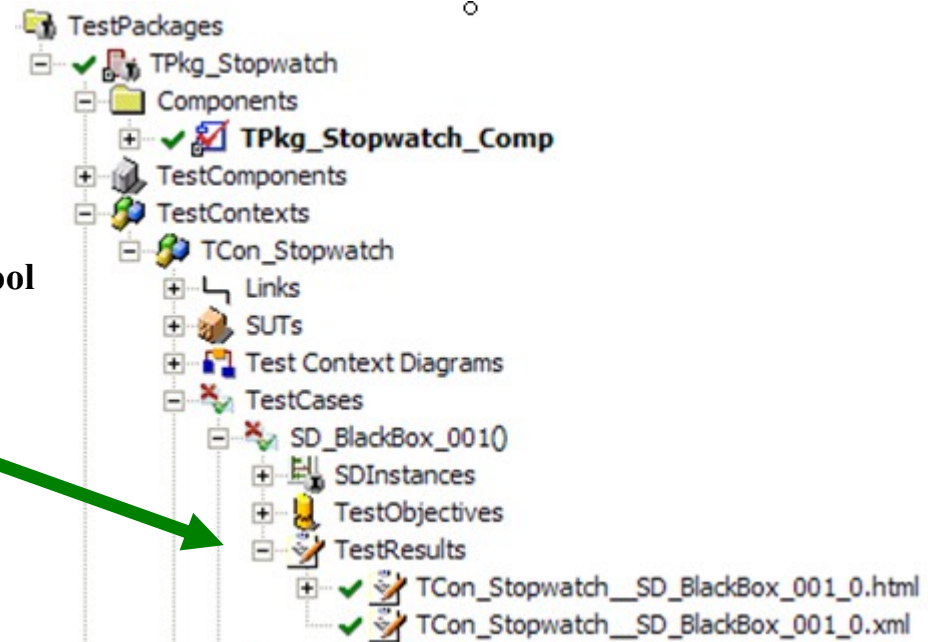
Requirements-driven testing



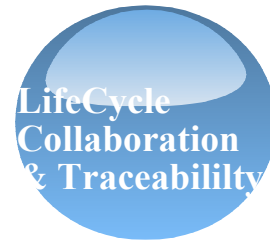
- **Quick definition and execution of model and requirement-aware tests**
 - Unit testing and regression testing
 - Reuse design scenarios as test cases
- **Requirement change impact and analysis**
 - Know which part of the model or which tests are affected by changing requirements

Req. ID	Req. Description	Design	Test Case
3 Requirements			
This section contains the user requirements.			
3.1 Capability Requirements			
3.1.1 Carrying Capacity			
3.1.1.1 Number of People			
Req. 304	Free average size adults shall be able to travel in comfort for a period of 3 hours. This level of comfort is defined as being equivalent to the standard of comfort provided by the top 40% of cars produced in 1999.	D-544	Verify Number of People
Req. 304	The car shall be able to carry 4 average size adults in average comfort for a period of 3 hours. Last modified: 11 February 1997	D-544	Verify Number of People
Req. 304	The top level of cars are those in the price range \$30,000 to \$40,000 at 1999 prices.		
Req. 304	Free average size adults shall be able to travel in comfort for a period of 3 hours.		
Req. 304	Users shall have easy entry and exit.	D-57	Verify support for Customers
Req. 304	A simple interior light shall be placed in the front of the vehicle.	D-57	Verify support for Customers
Req. 304	The car shall be able to...	D-57	Verify support for Customers

Pass/fail results can be synchronized with RM tool



Rational Rhapsody TestConductor integration with Rational Quality Manager



- Enables full execution control & management of model based Rhapsody TestConductor test cases from RQM
- Execution status (passed/failed) and result reports (Execution Results, Coverage Results) accessible through RQM
- RQM can utilize TestConductor execution results to continuously provide transparent & up to date QA statistics and QA reports

Rational Quality Manager

Home View Test Plans TestPlan_CashRegiste... TestCase_01_SD_InitC... Execution Result

Execution Result
Command Line Result

Test Case Result
Test Case: SD_tc_0
10:20:31, Monday, April 27, 2009

Actual Result: **Passed**
Host Name: jekyllslave
Owner: Mary, Test Manager

Test Milestone:
Test Case: TestCase_01_SD_InitCashRe
Test Script: SD_tc_0
Test Data: Unassigned
Weight: 100

Result Details
TCon_CashRegister__SD_tc_0_0.html
TestConductorAdapter20844.out
TestConductorAdapter20845.err
TestLog20843.log

Test Case Result
Test Case: SD_tc_0
10:20:31, Monday, April 27, 2009

Environment Info
Test executed on machine: JekyllSLAVE
Test executed by user: Administrator
Used OS version: Windows 2000 / Windows XP
Used Rhapsody version: 7.5, build 1155117
Used TestConductor version: 2.4, build 1406

Tested Project
Project: CppCashRegister
Active Component: TPkg_CashRegister_Comp
Active Configuration: DefaultConfig

SDs used in test
TPkg_CashRegister__SDTestScenario_0

Summary Info	Summary: passed
Total number of SDs used:	1
Total number of SD instances in test:	1
Total number of executed SD instances:	1
Total number of PASSED SD instances:	1 (100%)
Total number of FAILED SD instances:	0 (0%)
Total number of ACTIVE SD instances:	0 (0%)
Total number of NOT ACTIVE SD instances:	0 (0%)

SDTestScenario_0 «TestScenario»

«SUT»

TCon_CashRegister.itsCashRegister

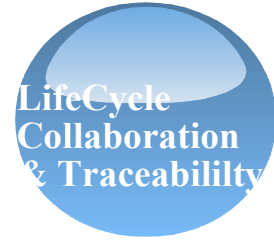
TCon_CashRegister.itsTC_at_hw_of_CashRegister:TC_at_hw_of_CashRegister

evStart()

show(aMsg = Ready)

evEnd()

Integration with Rational Quality Manager



- Allows linking Tests and test results in RQM with model based tests in Rhapsody
- Test execution, test reporting, test statistics etc. from RQM for model based tests
- Enables quality driven development throughout the development cycle

The screenshot displays the Rational Quality Manager interface for configuring a test script. The main window is titled "Test Script: New - Rational Quality M...". The left sidebar shows navigation options: Requirements, Planning, Construction, Lab Management, Builds, Execution, Reports, and Defects. The central area is titled "Create Test Script" and shows details for "TestCase0815".

Key interface elements and annotations:

- Adapter Type "Rhapsody TestConductor":** A red arrow points to the "Type" dropdown menu, which is currently set to "Rhapsody TestConductor".
- Interactive Test Case Selection:** A red arrow points to a list of test cases. The list has columns for "Name", "Full Path", and "Test Type". The selected item is "SD_tc_0".
- Buttons & List Boxes for Test Execution Settings:** A red arrow points to the "Start Rhapsody in Silent Mode" checkbox and the "Performance Setting" dropdown menu.

Name	Full Path	Test Type
<input type="checkbox"/> atg_tc_002	TPkg_CashRegister:TCon_CashRegister.atg_tc_002	TestCase
<input type="checkbox"/> atg_tc_003	TPkg_CashRegister:TCon_CashRegister.atg_tc_003	TestCase
<input type="checkbox"/> atg_tc_004	TPkg_CashRegister:TCon_CashRegister.atg_tc_004	TestCase
<input type="checkbox"/> atg_tc_006	TPkg_CashRegister:TCon_CashRegister.atg_tc_006	TestCase
<input type="checkbox"/> atg_tc_008	TPkg_CashRegister:TCon_CashRegister.atg_tc_008	TestCase
<input type="checkbox"/> FC_tc_0	TPkg_CashRegister:TCon_CashRegister.FC_tc_0	TestCase
<input checked="" type="checkbox"/> SD_tc_0	TPkg_CashRegister:TCon_CashRegister.SD_tc_0	TestCase
<input type="checkbox"/> TCon_CashRegister	TPkg_CashRegister:TCon_CashRegister	TestContext
<input type="checkbox"/> TPkg_CashRegister	TPkg_CashRegister	TestPackage

Agenda

- Development and Test Process
- Key Enabler for Model Based Test
- **Rational Solution for System and Software Engineering**

Rational Solutions for Systems and Software Engineering

Built on a core product set

Use modeling to validate requirements, architecture and design throughout the development process

Rational Rhapsody

Rational DOORS

Rational Quality Manager

Manage all system requirements with full traceability across the lifecycle

Achieve “quality by design” with an integrated, automated quality management and testing process

Rational Team Concert

Collaborate across diverse engineering disciplines and development teams

Collaborate

Automate

Report

Rational Workbench for Systems and Software Engineering



Ensure Success with Rational

Rational software

Process and methodology

- Process framework workshops
- Rational Harmony family of Best Practices and Processes
- Process training

Implementation services

- Adoption quick starts
- Deployment support
- Project architecture workshops
- Project management
- Planning support
- Escalation/risk mitigation

Training and mentoring

- Product familiarity
- Product expertise and specialization
- Technology transfer
- Adoption mentoring
- e-Learning

Technical services

- Measured Capability Improvement Framework (MCIF)
- Product optimization and customization
- Tool configuration

What's next?

Track 15,45 – 16,30

Model-Driven Development e

Model-Based Testing per codice Safety Critical

Track 16:30-17:15

Agile and RTC Adoption@Rome

Demo point 15,45 – 17,00

Collaborative Life-Cycle Management



Learn more at:

- IBM Rational software
- IBM Rational Software Delivery Platform
- Process and portfolio management
- Change and release management
- Quality management
- Architecture management
- Rational trial downloads
- Leading Innovation Web site
- developerWorks Rational
- IBM Rational TV
- IBM Business Partners
- IBM Rational Case Studies