Quality Management

Alberto Sardini – Senior Systems Specialist

alberto_sardini@it.ibm.com

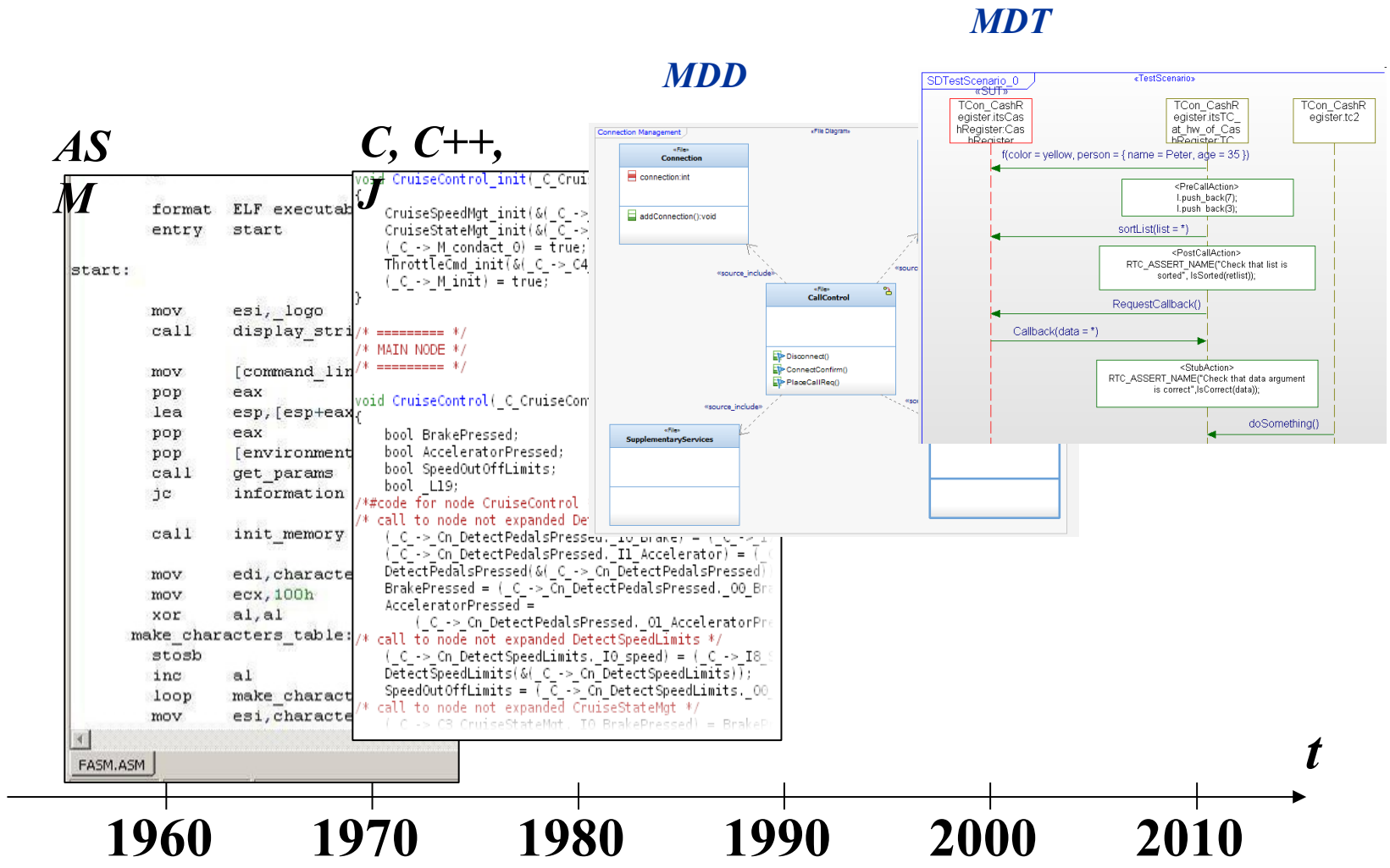
Agenda

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

Agenda

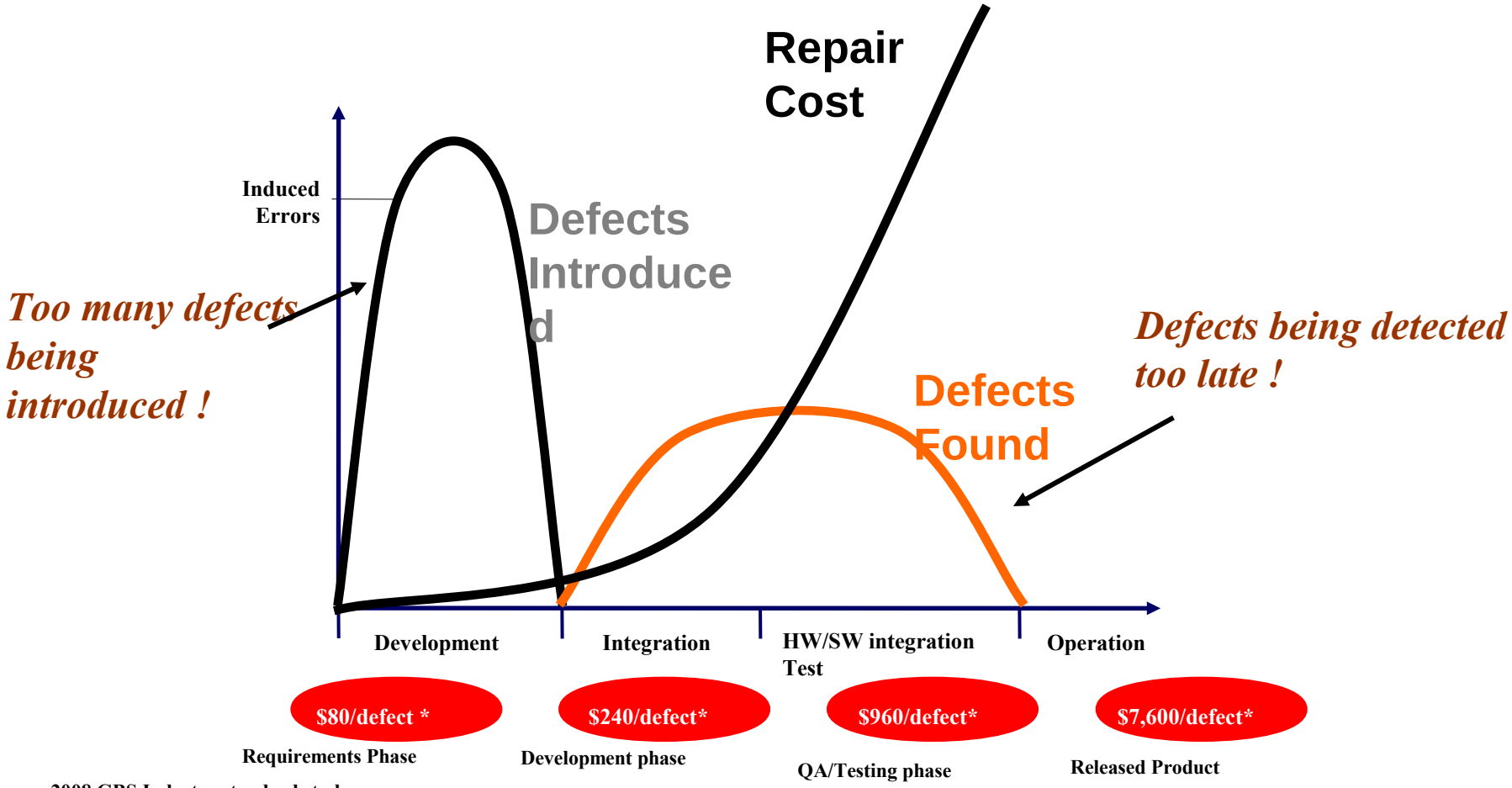
- **Development and Test Process**
- Key Enabler for Model Based Test
- Rational Solution for System and Software Engineering

Development Process is Evolving…



Traditional testing methodologies are insufficient

80% of development costs are spent identifying and fixing defects



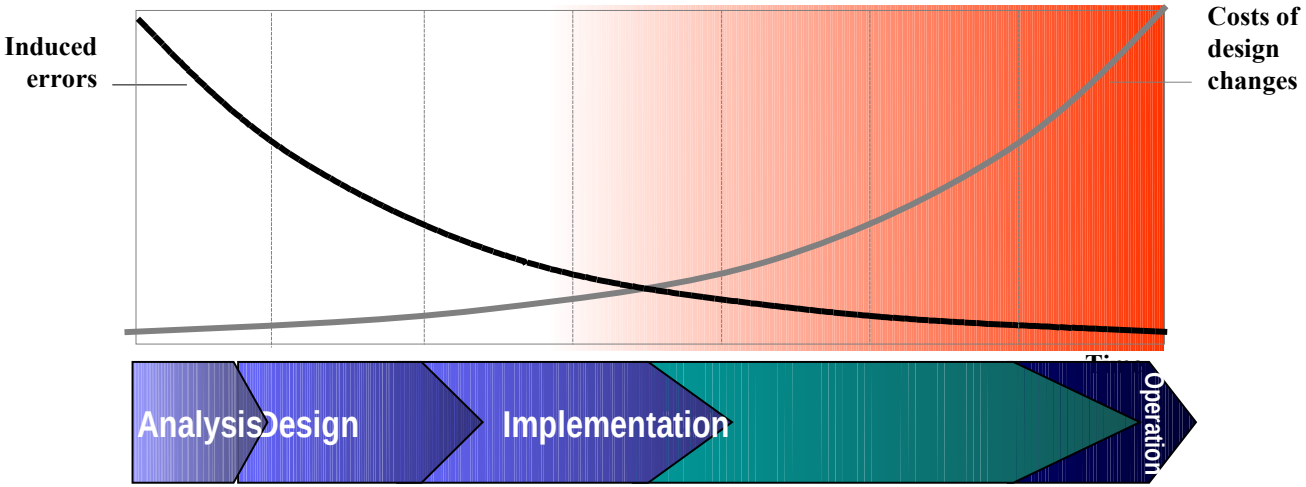
*Source: 2008 GBS Industry standard study

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….

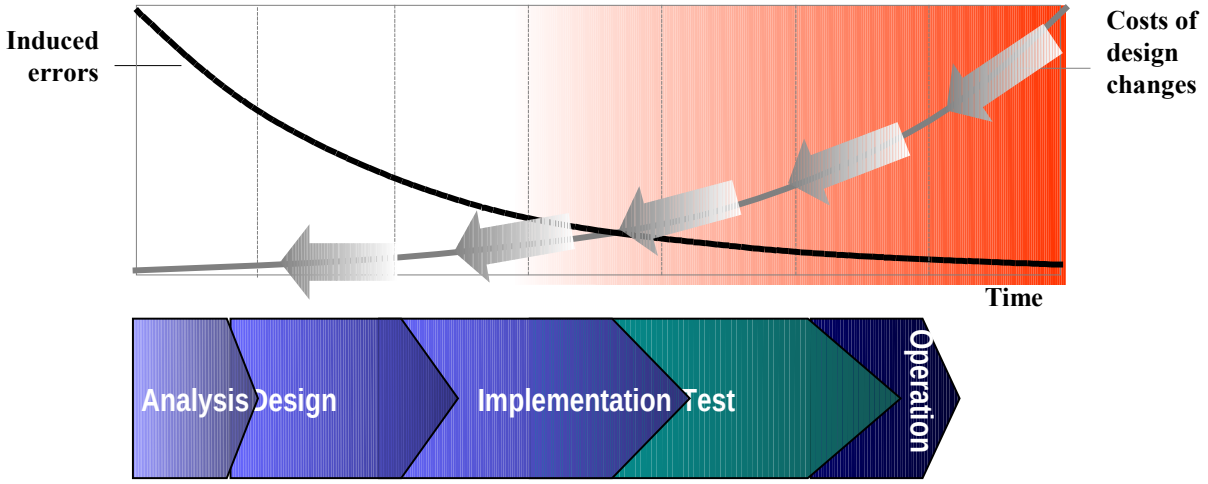
- Implement code from **textual requirements**
- Test only on target **late in development cycle**
  - Defects introduced early but detected late during testing and validation
- Large part of development time spent manually testing
  - Difficult to communicate script/code tests to others and not traced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



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

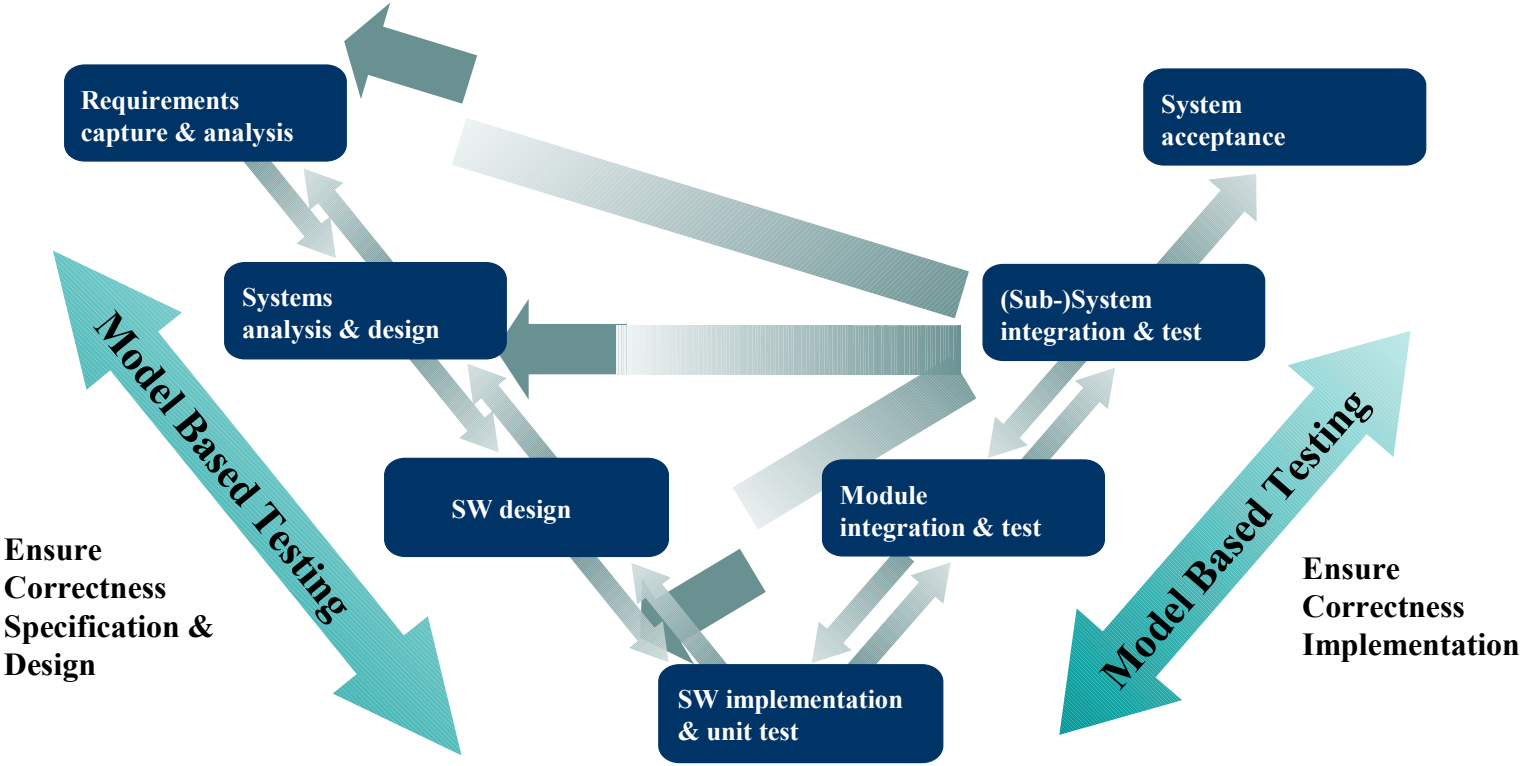
Quality by Design - "shift left" to identify and fix errors early

The Benefits of Model Driven Development with Model Based Testing



Model Based Testing

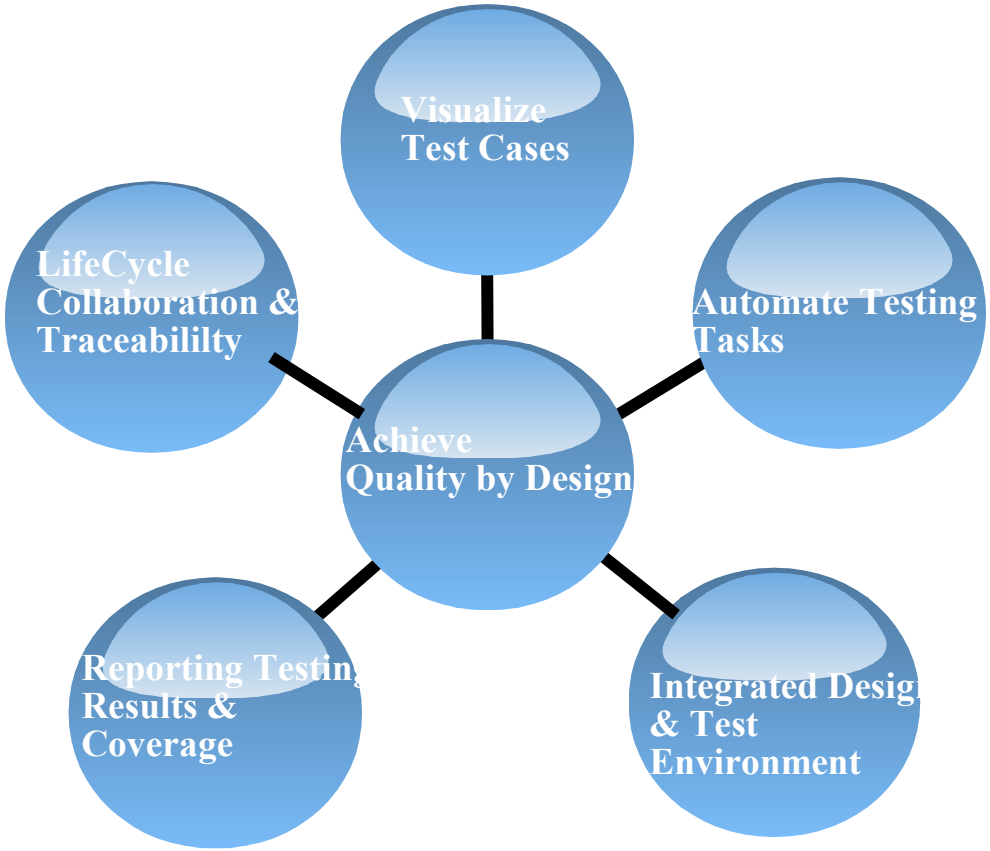
- Visual test definition for improved collaboration
- Automated test execution, monitoring and test architecture creation
- Early validation of requirements during systems engineering
- Automate unit and regression testing helping improve software quality



Agenda

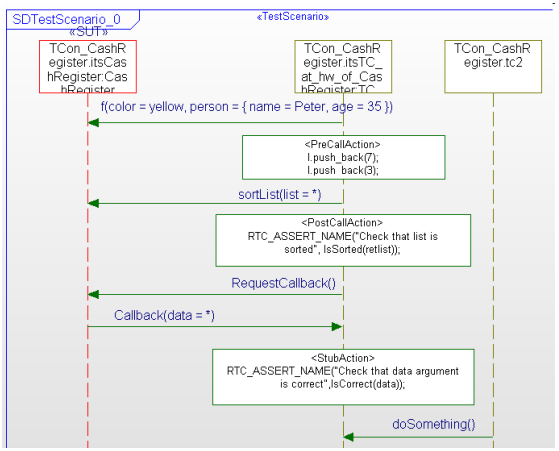
- Development and Test Process
- **Key Enabler for Model Based Test**
- Rational Solution for System and Software Engineering

Key Enablers of Model Based Testing

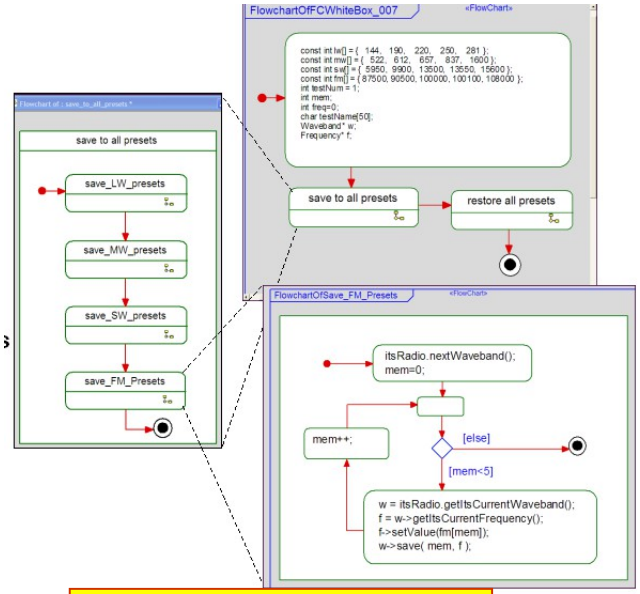


Capture test cases with UML

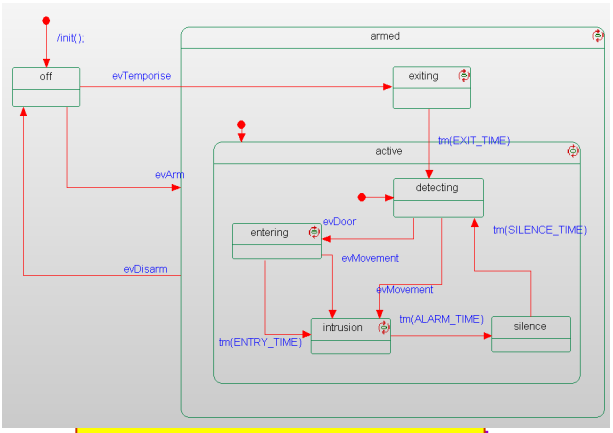
- **Based on OMG's standard UML Testing Profilie (www.omg.org)**
- **Specify test cases visually for better communication across teams**
- **Creating code tests cases or importing Cunit/Cpp unit tests also possible**



Sequence Diagram Test Case

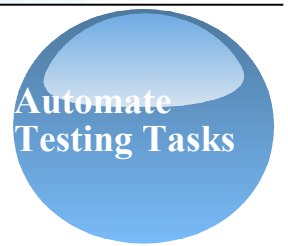


Flowchart Test Cases

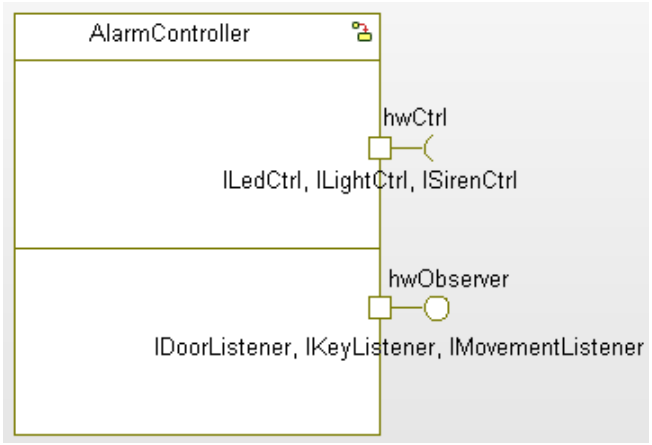


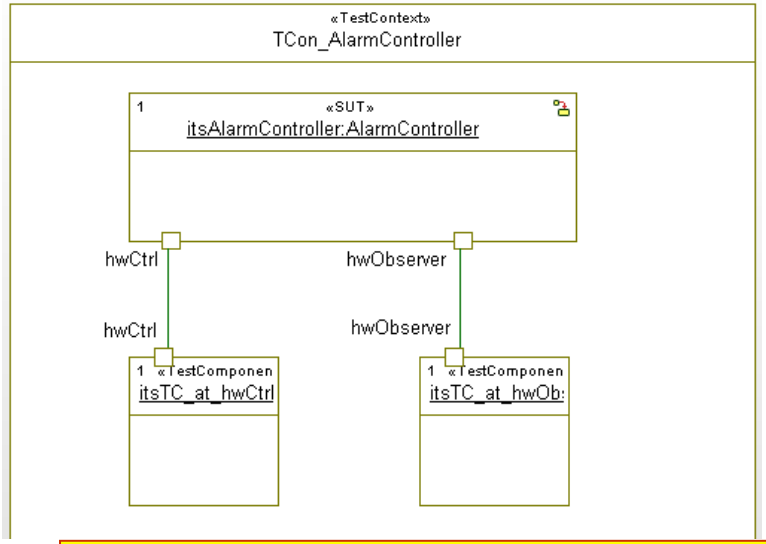
Statechart Test Case

Automate Quality



- **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**



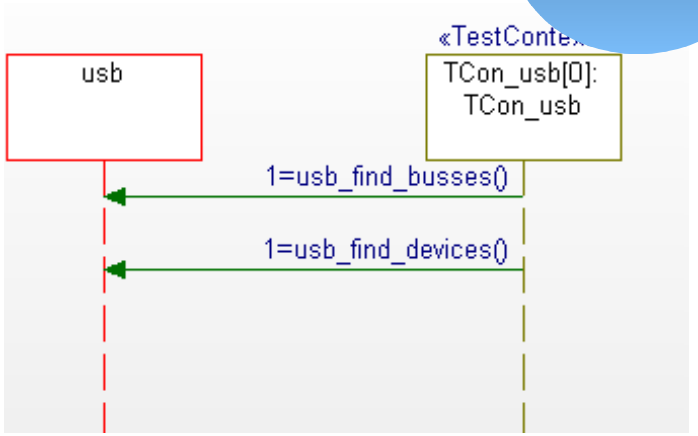


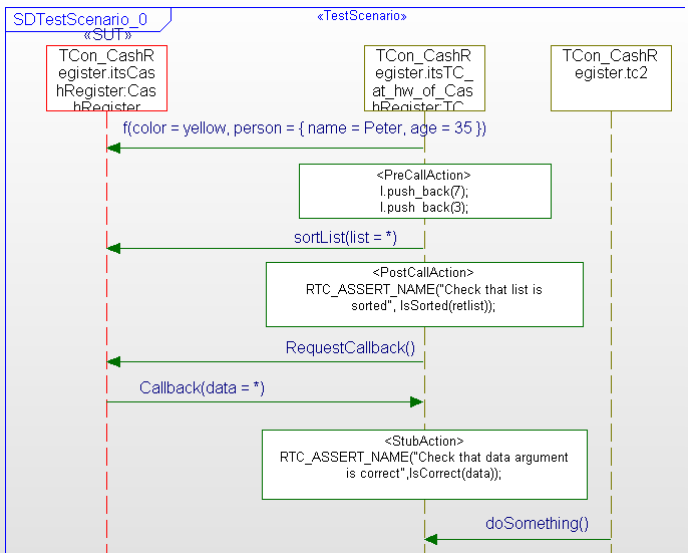
Automatically Created Test Architecture

Model-Based Testing

IBM Rational Rhapsody Test Conductor Add On

- 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

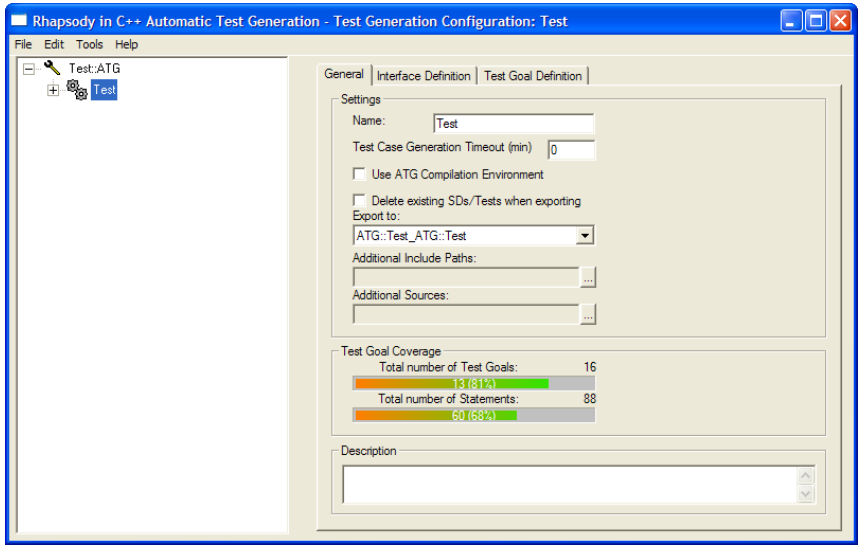


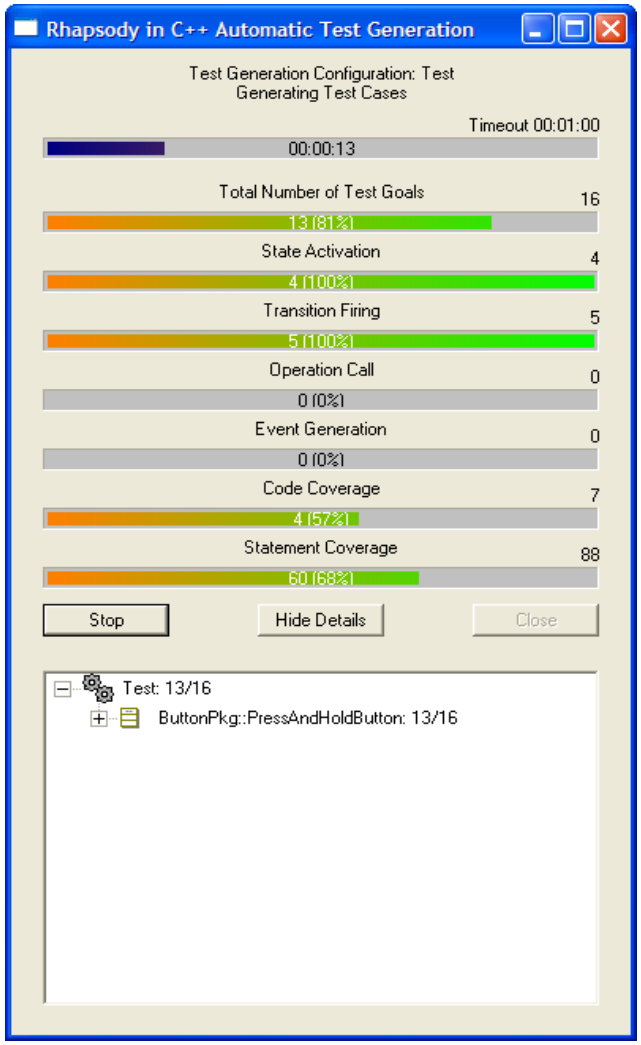


Model Driven Test Generation

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

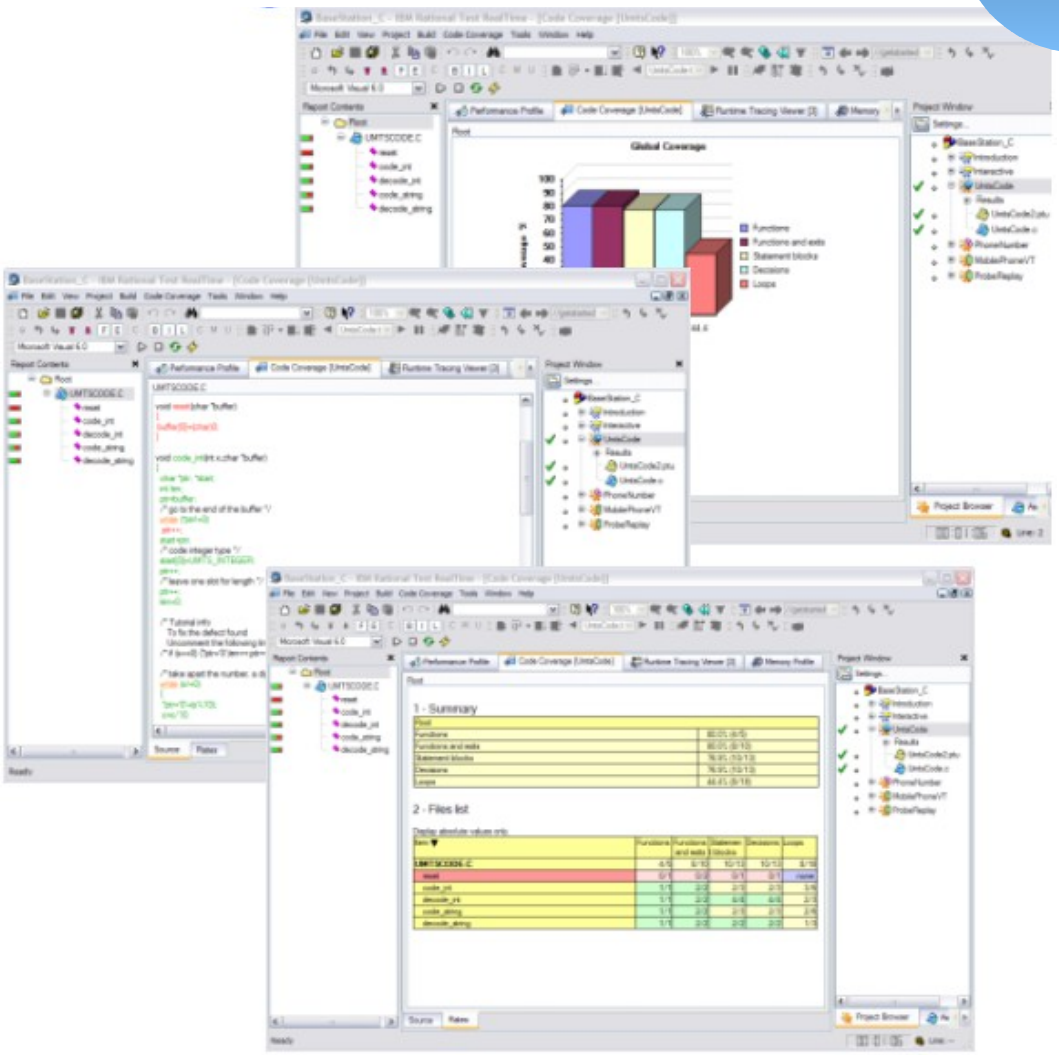
Automate Testing Tasks

Runtime Analysis:

Code Coverage

Performance Analysis

Memory Profiling



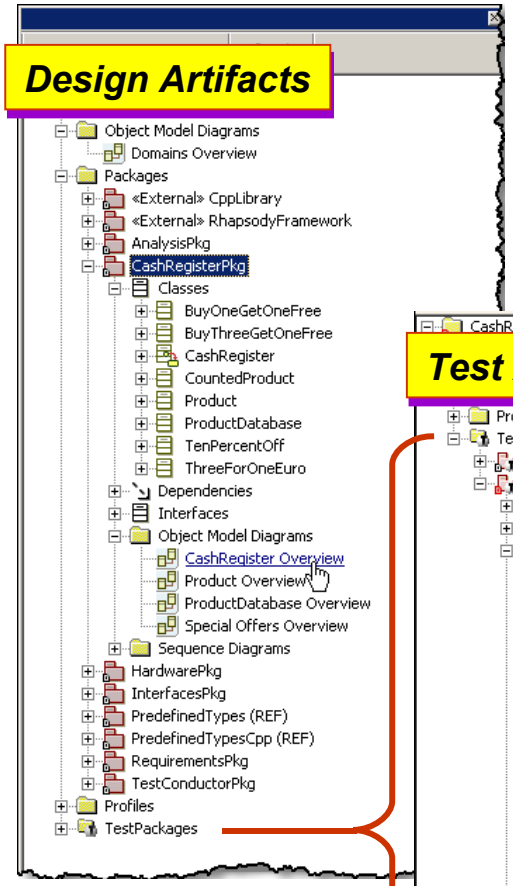
Integrated Design and Test Environment

Manage test cases within Rational Rhapsody

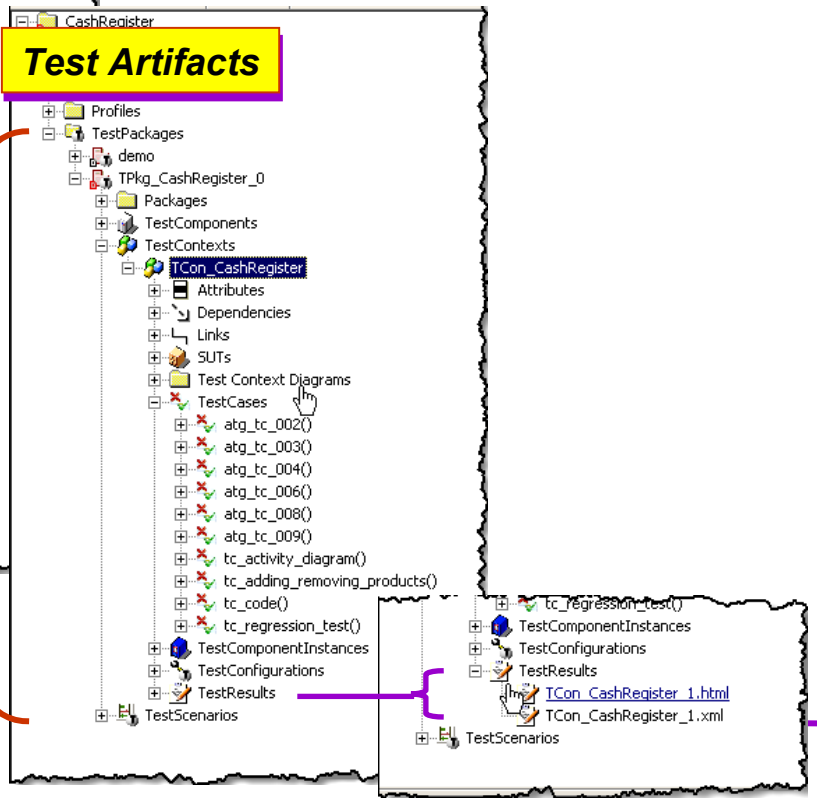
Integrated Design & Test Environment

- **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

Design Artifacts



Test Artifacts



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

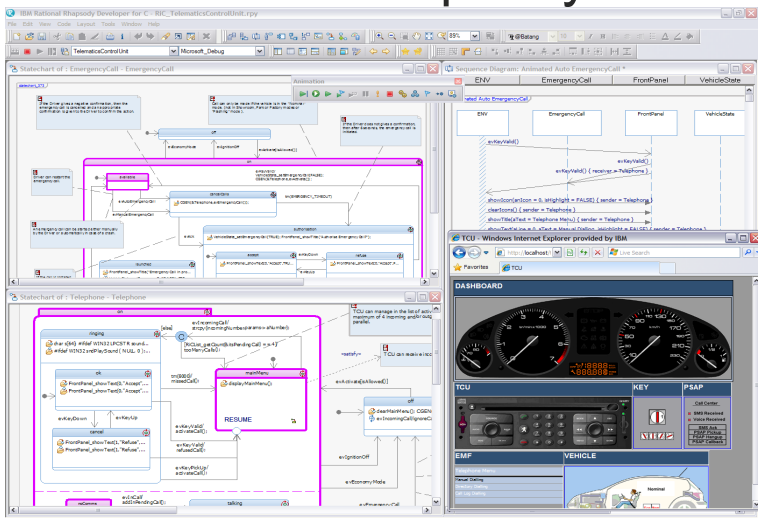
Tested Project	
Project:	CashRegister
Active Component:	TCon_CashRegister_5
Active Configuration:	DefaultConfig

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

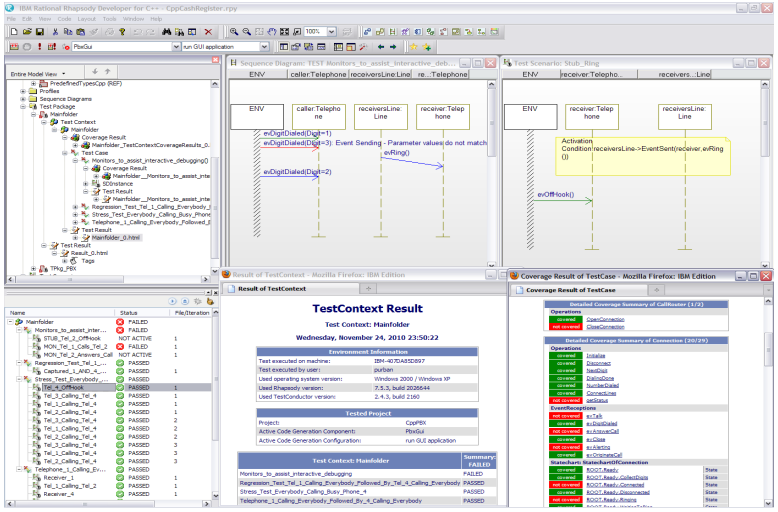
Model-Driven Development Model Based Test

IBM Rational Rhapsody® with TestConductor

Integrated Design & Test Environment



Simulation for model level debugging



Specific, execute and monitor test cases with Rational Rhapsody TestConductor Add On

Capabilities

- Requirements-driven analysis, design and test for systems engineering and embedded / real-time software applications
- Rapid design validation and verification with frequent simulation and testing
- Development and deployment of *complete* C, C++, Java and Ada applications targeting embedded device
- Model based testing with Rhapsody TestConductor Add On visualizes requirement and unit test cases

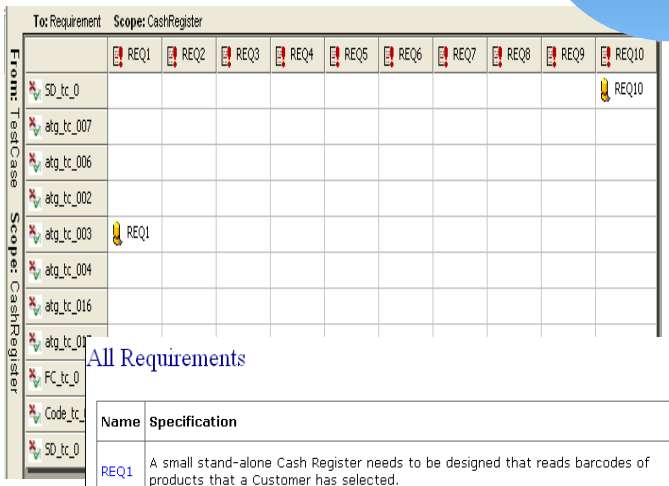
Benefits

- **Build the right product** through non-ambiguous communication and frequent collaboration
- **Eliminate defects early** and increase quality by continually testing the design
- **Reduce development time** by automatically generating applications and automating test execution and reporting
- **Improve traceabilty** cross link requirements, design, code and test cases and help maintain consistency

Requirements to Test Results Coverage

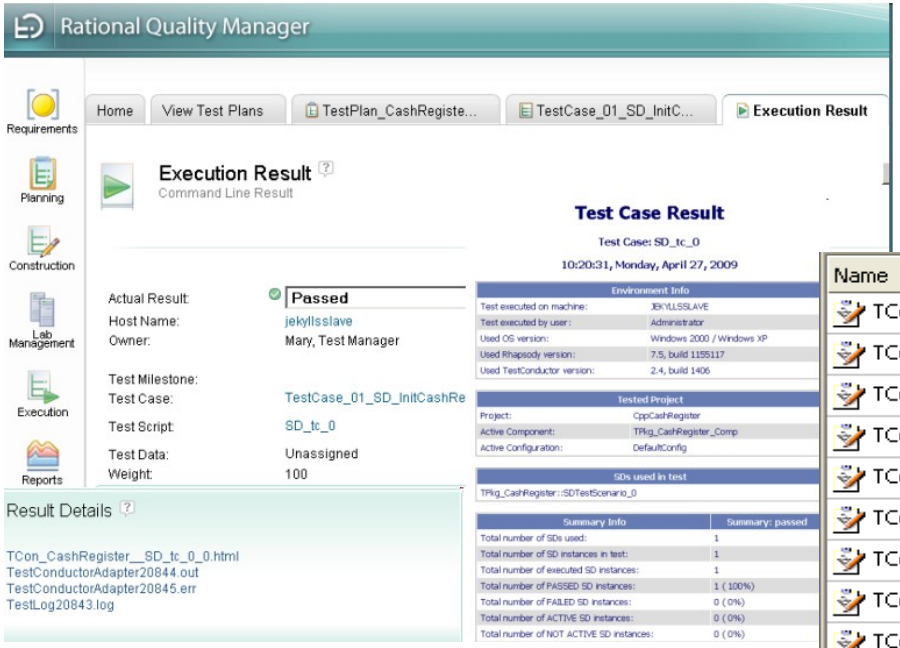
Reporting Testing Results & Coverage

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



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)
	ssible to add special offers 1 Euro".	not covered
		FC_tc_0 (Passed)
	el the last selected product,	not covered
	forms in the future.	not covered
	ucts.	not covered

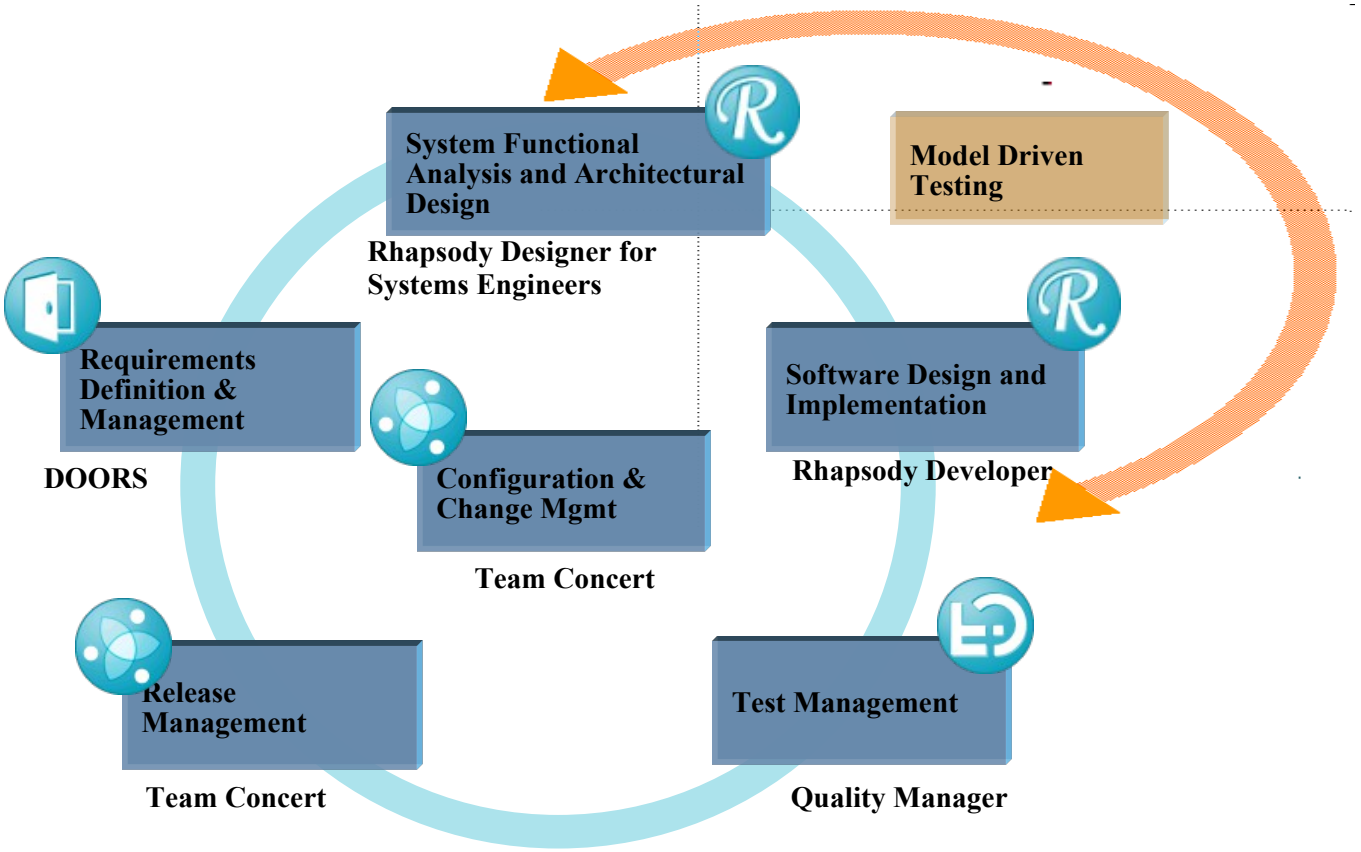


Name	Verdict
TCon_CashRegister__SD_tc_0_4.html	Failed
TCon_CashRegister__atg_tc_007_7.html	Passed
TCon_CashRegister__atg_tc_006_9.html	Passed
TCon_CashRegister__atg_tc_002_9.html	Passed
TCon_CashRegister__atg_tc_003_9.html	Passed
TCon_CashRegister__atg_tc_004_9.html	Passed
TCon_CashRegister__FC_tc_0_0.html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

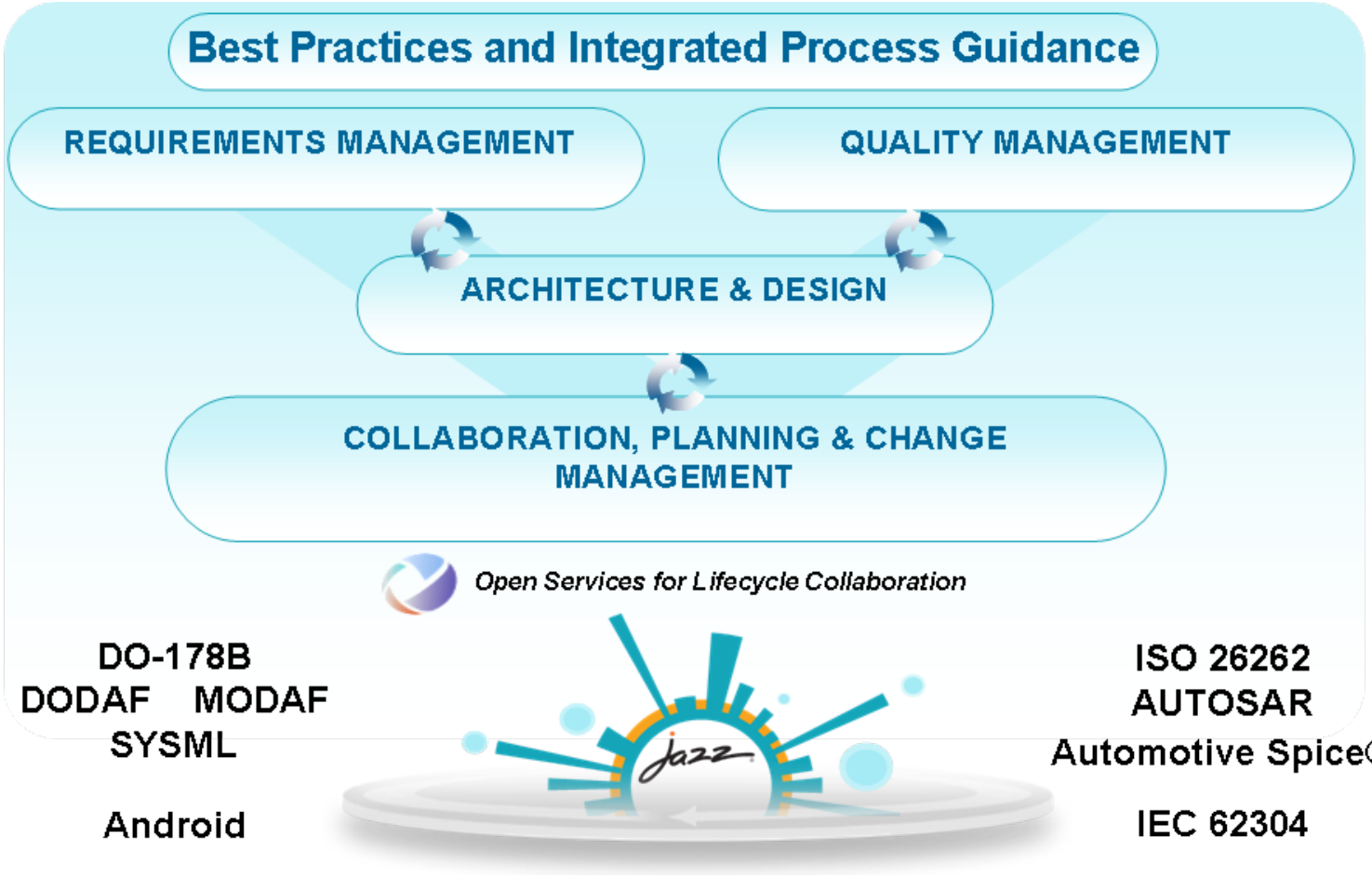
LifeCycle Collaboration & Traceabililty



Unify Lifecycle Disciplines across Systems and Software Engineering

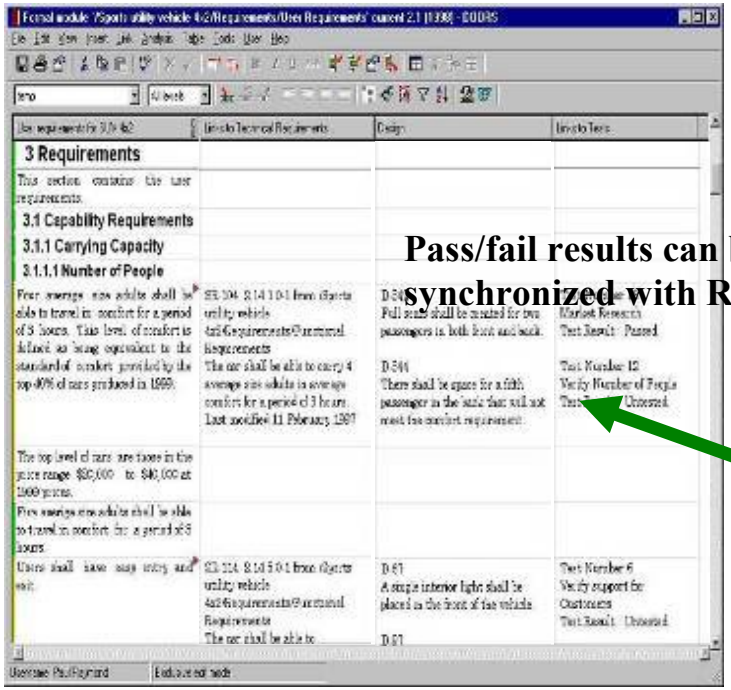
Best Practices, Tools and Services on an open platform

LifeCycle Collaboration & Traceabililty

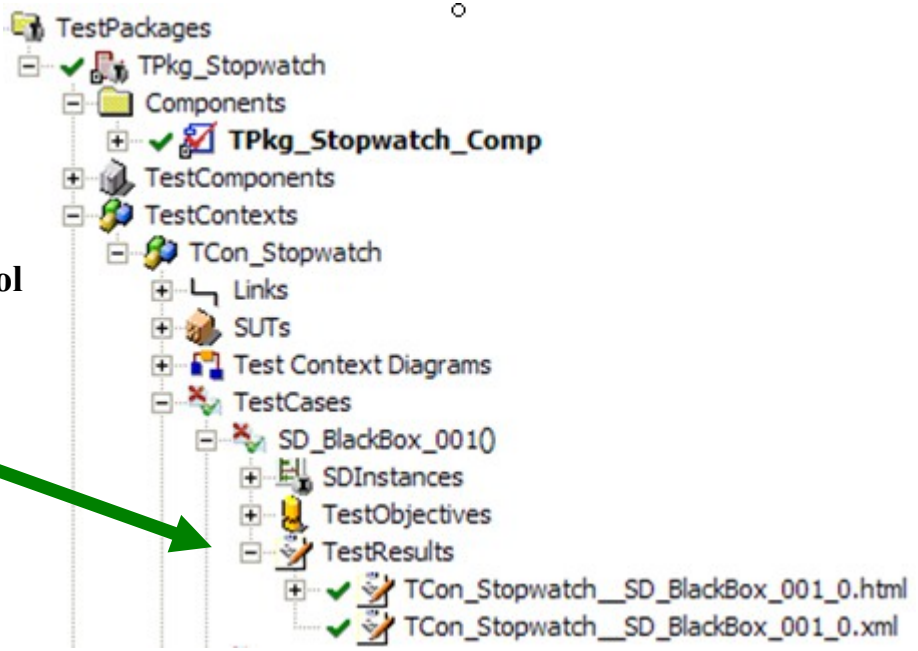


Requirements-driven testing

- **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**



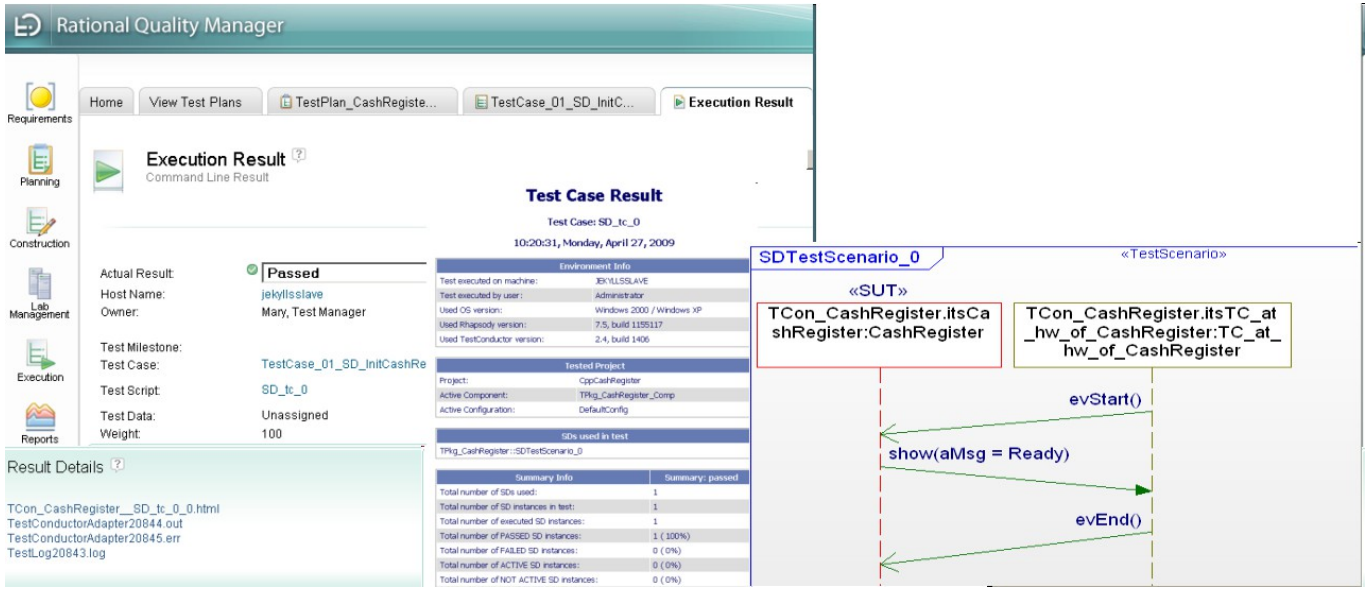
Pass/fail results can be synchronized with RM tool



Rational Rhapsody TestConductor integration with Rational Quality Manager

LifeCycle Collaboration & Traceabililty

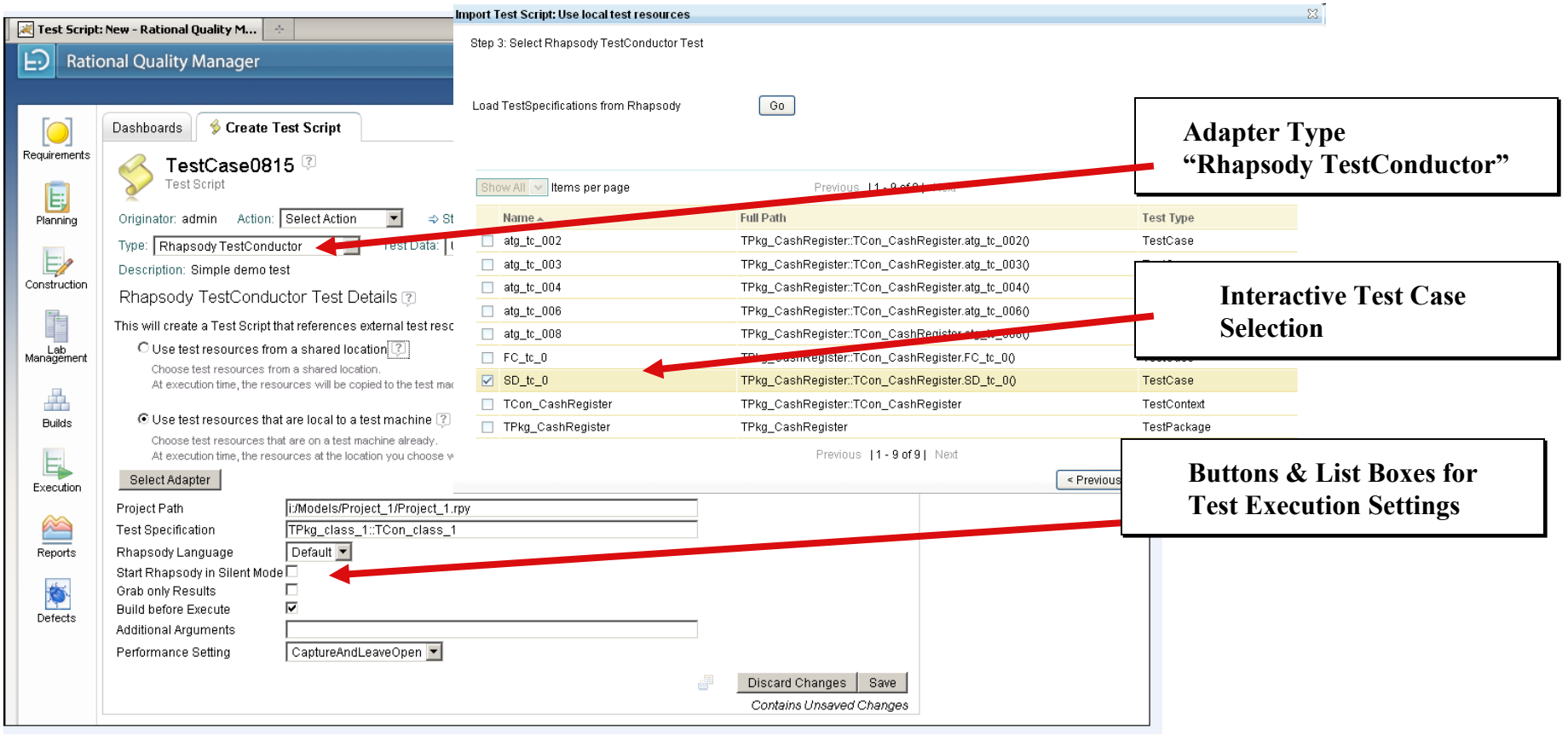
- **Enables full execution control & management of model based Rhapsody TestConductor test cases from RQM**
- **Execution status (passed/failed) and result reports (Execution Results, Coverage Results) accessible through RQM**
- **RQM can utilize TestConductor execution results to continuously provide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



Integration with Rational Quality Manager

LifeCycle Collaboration & Traceability

- 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

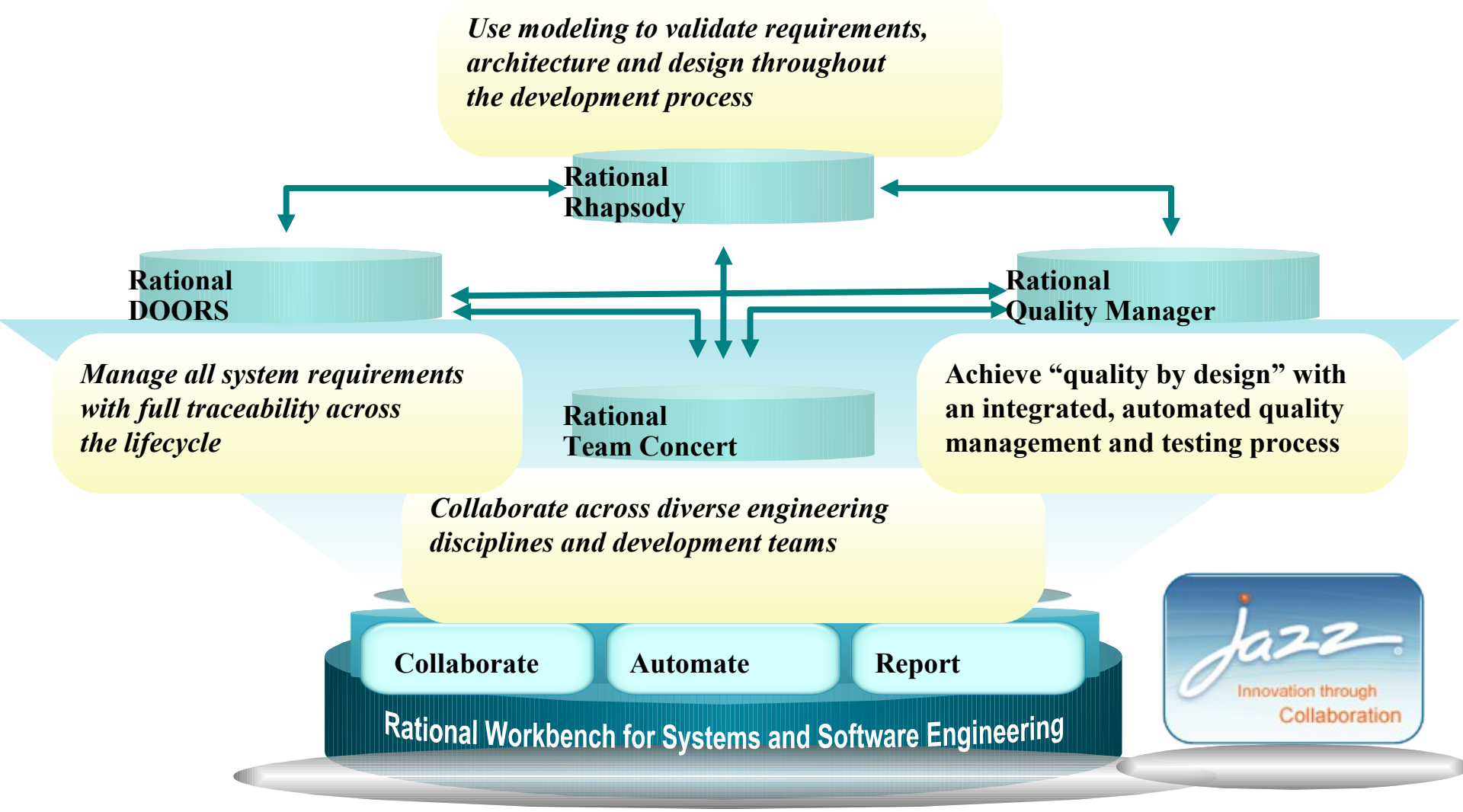


Agenda

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

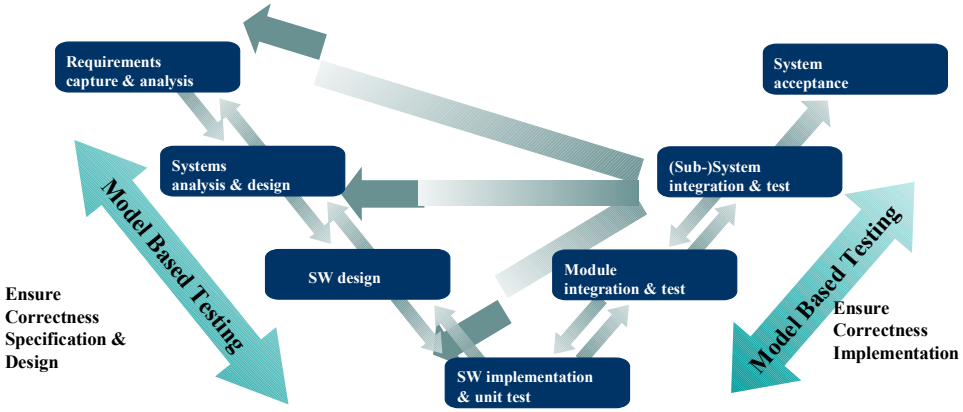
Rational Solutions for Systems and Software Engineering

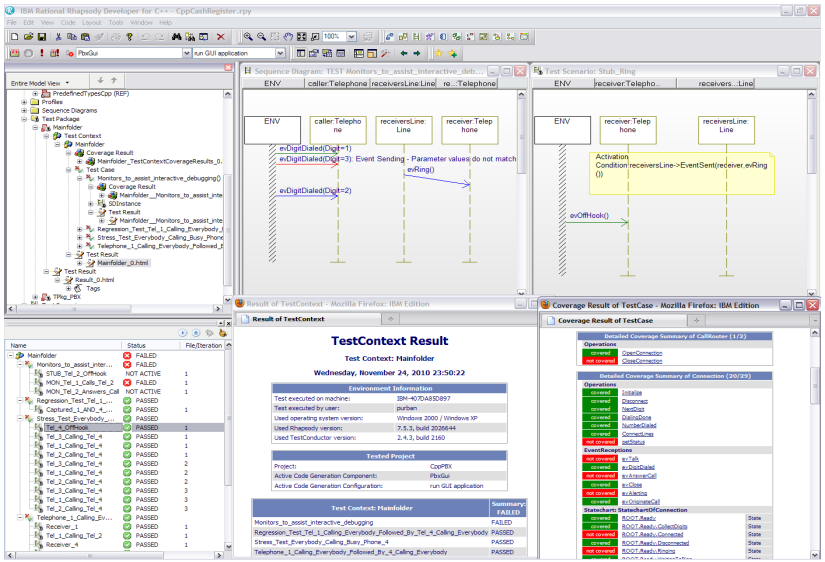
Built on a core product set



Build in Quality from Concept to Launch

- Visual test definition for improved collaboration
- Automated test execution, monitoring and test architecture creation
- Early validation of requirements during systems engineering
- Record simulations often to validate functionality and verify correctness
- Automate unit and regression testing to help improve software quality
- Link to requirements, model, code and test data for traceability
- Automatically create and execute tests from the design model or target platform





Ensure Success with Rational

Rational. software

Process and methodology	Implementation services	Training and mentoring	Technical services
• Process framework workshops	• Adoption quick starts	• Product familiarity	• Measured Capability Improvement Framework (MCIF)
• Rational Harmony family of Best Practices and Processes	• Deployment support	• Product expertise and specialization	• Product optimization and customization
• Process training	• Project architecture workshops	• Technology transfer	• Tool configuration
	• Project management	• Adoption mentoring	
	• Planning support	• e-Learning	
	• Escalation/risk mitigation		

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