

IBM Software Group

# Effective Model Driven Development for Complex A&D Systems

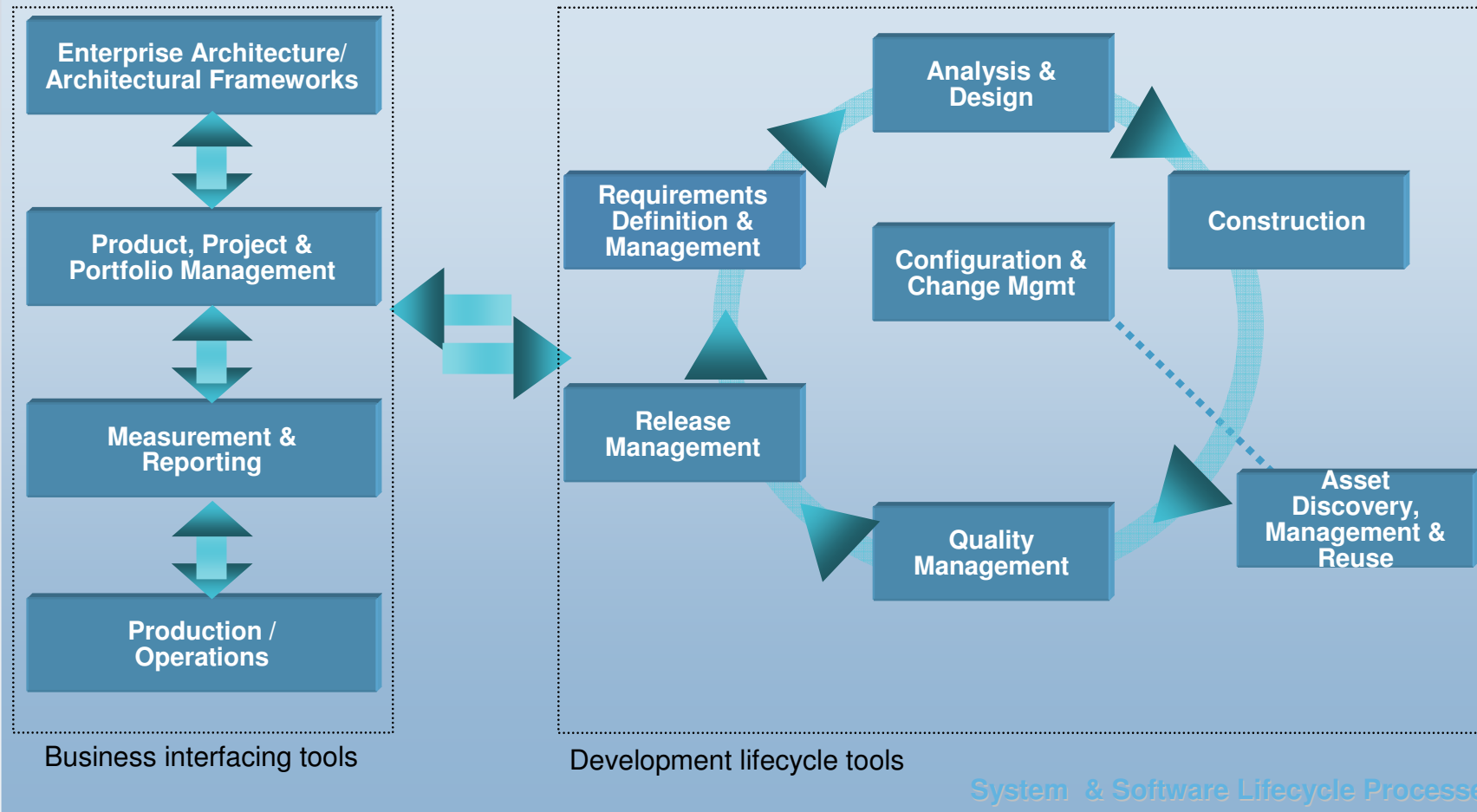
Rational software

Go to IBM

# Portfolio Overview

**Rational.** software

**Telelogic**  
An IBM Company



**Lotus.** software

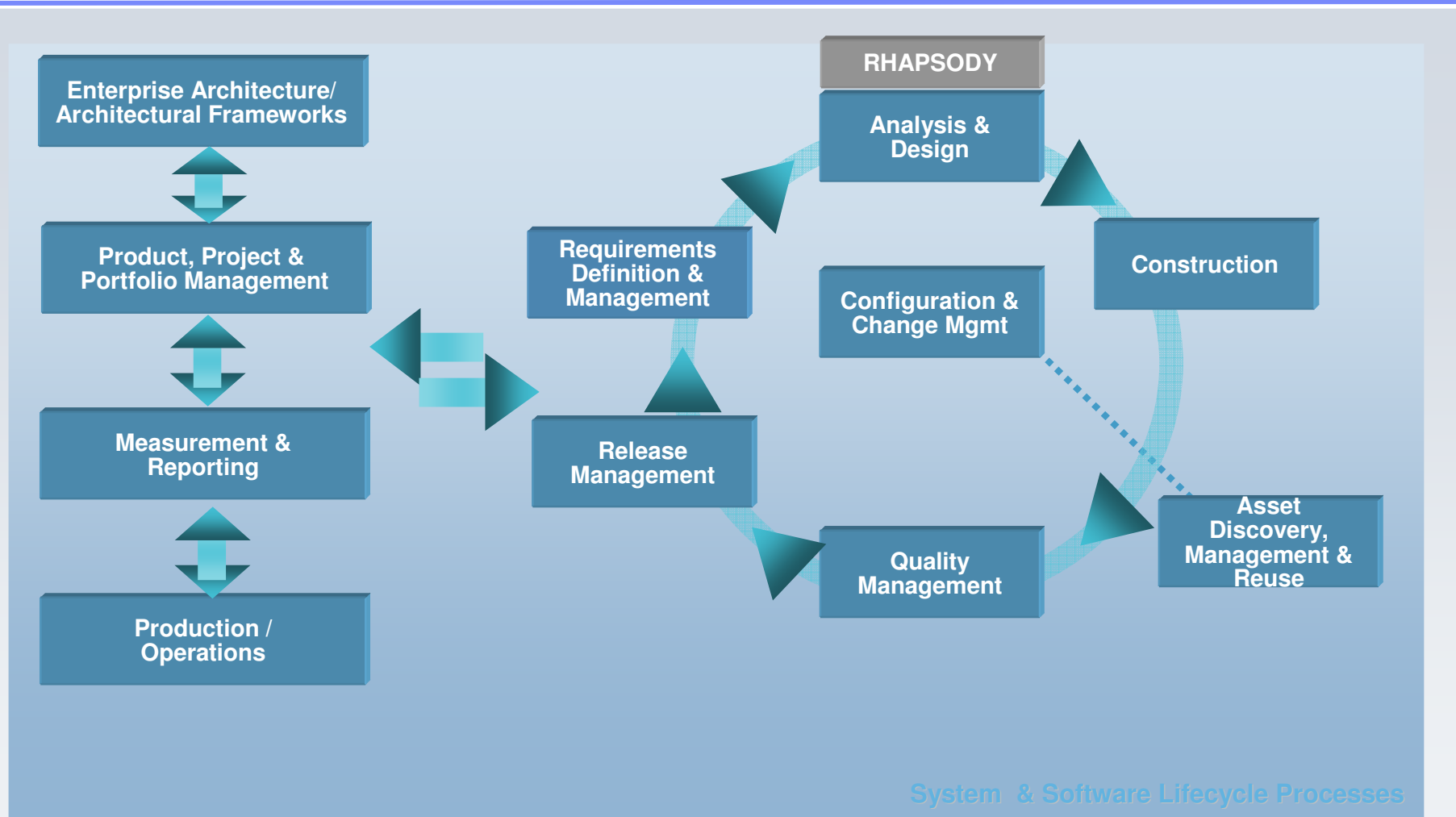
**Information Management** software

**Tivoli.** software

**WebSphere** software

# Modelado y diseño de sistemas

## Telelogic Rhapsody



System & Software Lifecycle Processes

## Embedded SW Development Pains

- **Time to market pressures**
- **Cost of quality is too high (recalls, penalties, reputation).**
- **Design errors found too late in the process when they are most costly to fix.**
- **Testing productivity is low.**
- **Collaboration within large distributed teams is becoming increasingly difficult.**
- **Ineffective communication across the systems architecture, software, mechanical, and electrical disciplines.**
- **Difficulty managing increasing design complexity.**
- **Managing development of multiple products and product variants.**
- **Inability to reuse existing IP.**



## Solution -- Telelogic Rhapsody

Telelogic Rhapsody® is the industry-leading UML® 2.1 and OMG SysML™-based Model Driven Development™ (MDD™) environment for technical, real-time or embedded systems and software engineering.

### ■ Features

- ▶ High level modeling, design level debugging, automatic production code generation, and target deployment
- ▶ Rational Rose Import.
- ▶ Reuse of software assets, whether source code or model based through code visualization and reverse engineering.
- ▶ Model simulation/validation and host based testing early in the development lifecycle improving quality and reducing cost.
- ▶ Integrated requirements modeling, traceability and analysis.
- ▶ Integration with leading CM tools including IBM Rational ClearCase.
- ▶ Small and large team collaboration through 3 way model diff&merge.
- ▶ Model-Driven Testing increasing testing productivity

### ■ Award Winning

- ▶ Two Best in Show awards at the Embedded Systems Conferences from the VDC;
- ▶ The SD Times 100 in multiple years, taking top honors in the Modeling category
- ▶ The Model Driven Development Focus of the Embedded Development Arena award
- ▶ The Embedded Award for Software at Embedded World 2007 for the Rhapsody AUTOSAR Pack.
- ▶ Endorsed by Embedded Market Forecasters as the tool of choice for C developers.



## Slide 5

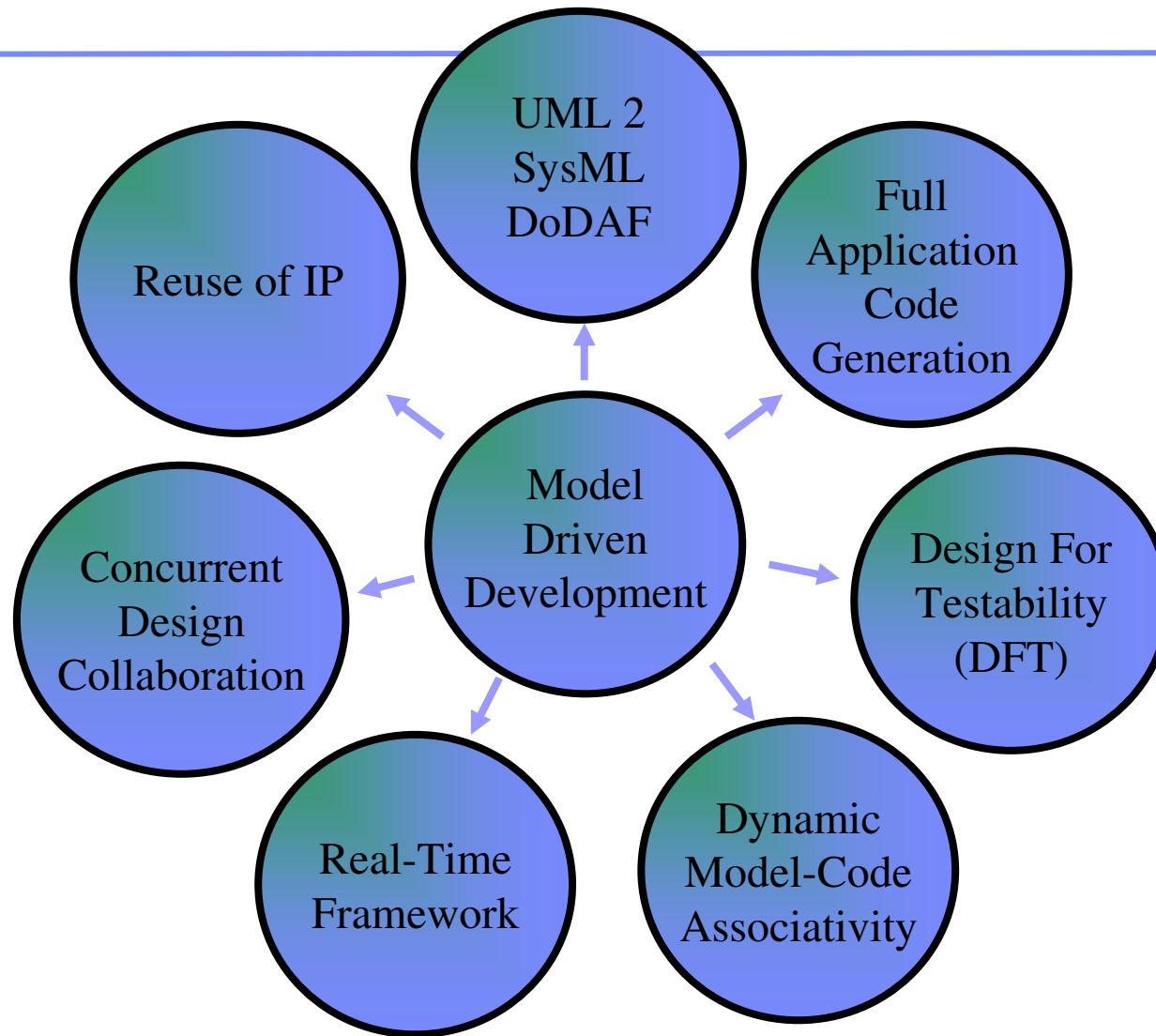
---

**iU1**

We are now the market leader per the last VDC report.

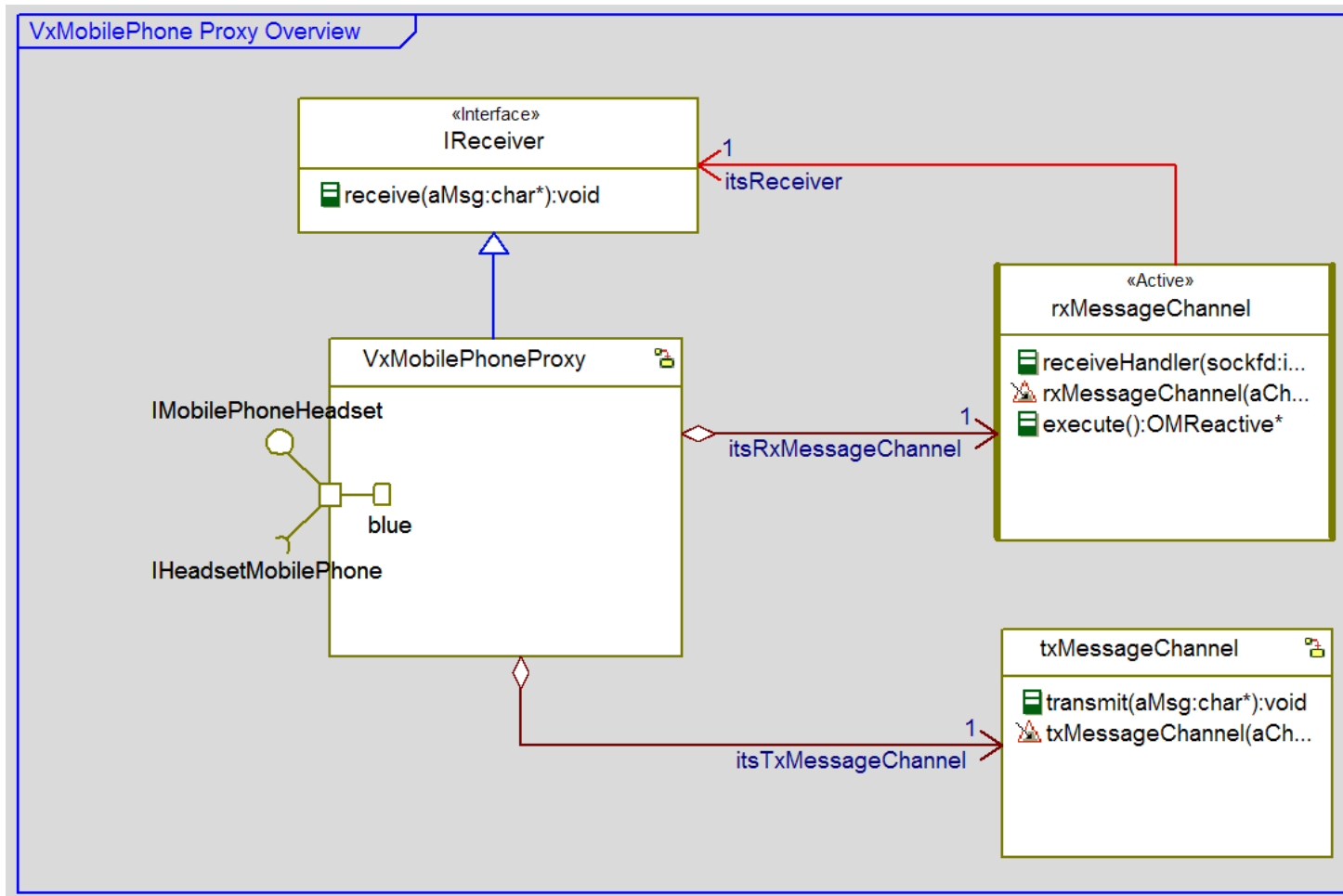
I-logix Authorized user; 29/04/2008

# Model Driven Development



# UML 2

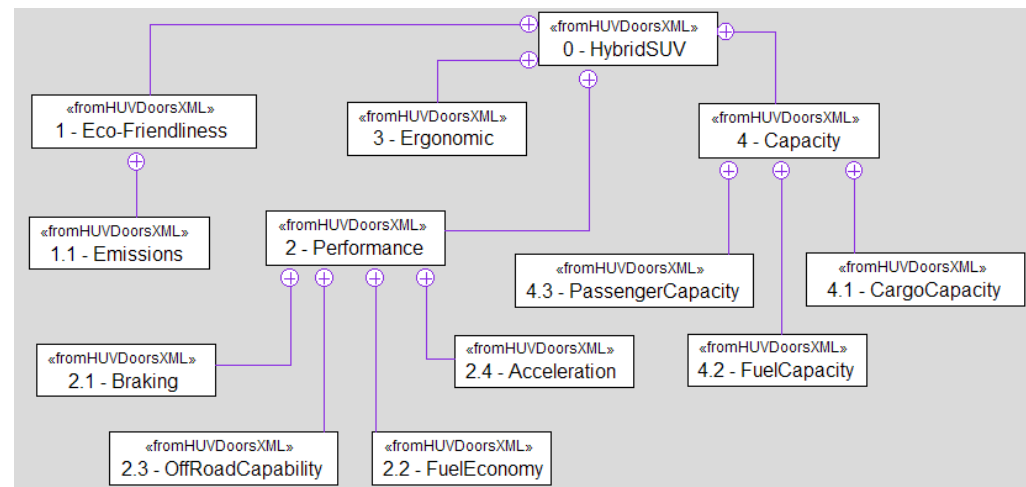
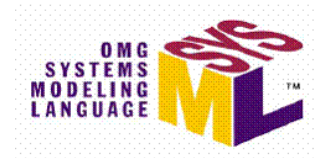
- Rhapsody is the leading UML 2 compliant solution for embedded systems





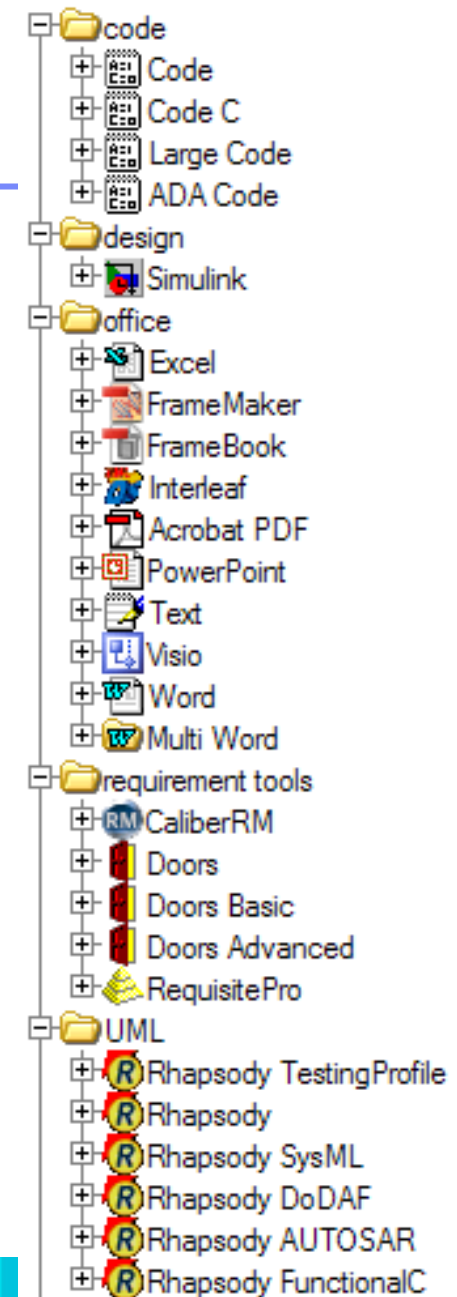
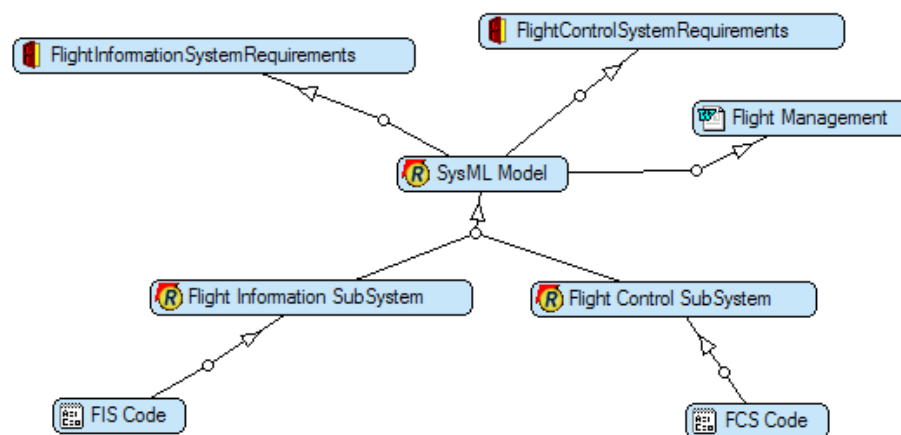
# SysML

- **SysML is a domain customization of UML 2 for systems engineers**
  - ▶ Supports the standard proposal in its latest form (V1.0)
- **Support for SysML views**
  - ▶ Requirements: [Requirements diagram](#); Use case diagram
  - ▶ Structure: [Block Definition diagram](#); [Internal Block diagram](#)
  - ▶ Behavior: Statechart; Activity diagram; Sequence diagram
  - ▶ Constraints: [Parametric diagram](#)
- ***Uniquely Integrated Requirements and Design modeling environment***
- **More than just modeling...**
  - ▶ Simulation of SysML models
  - ▶ System testing for SysML

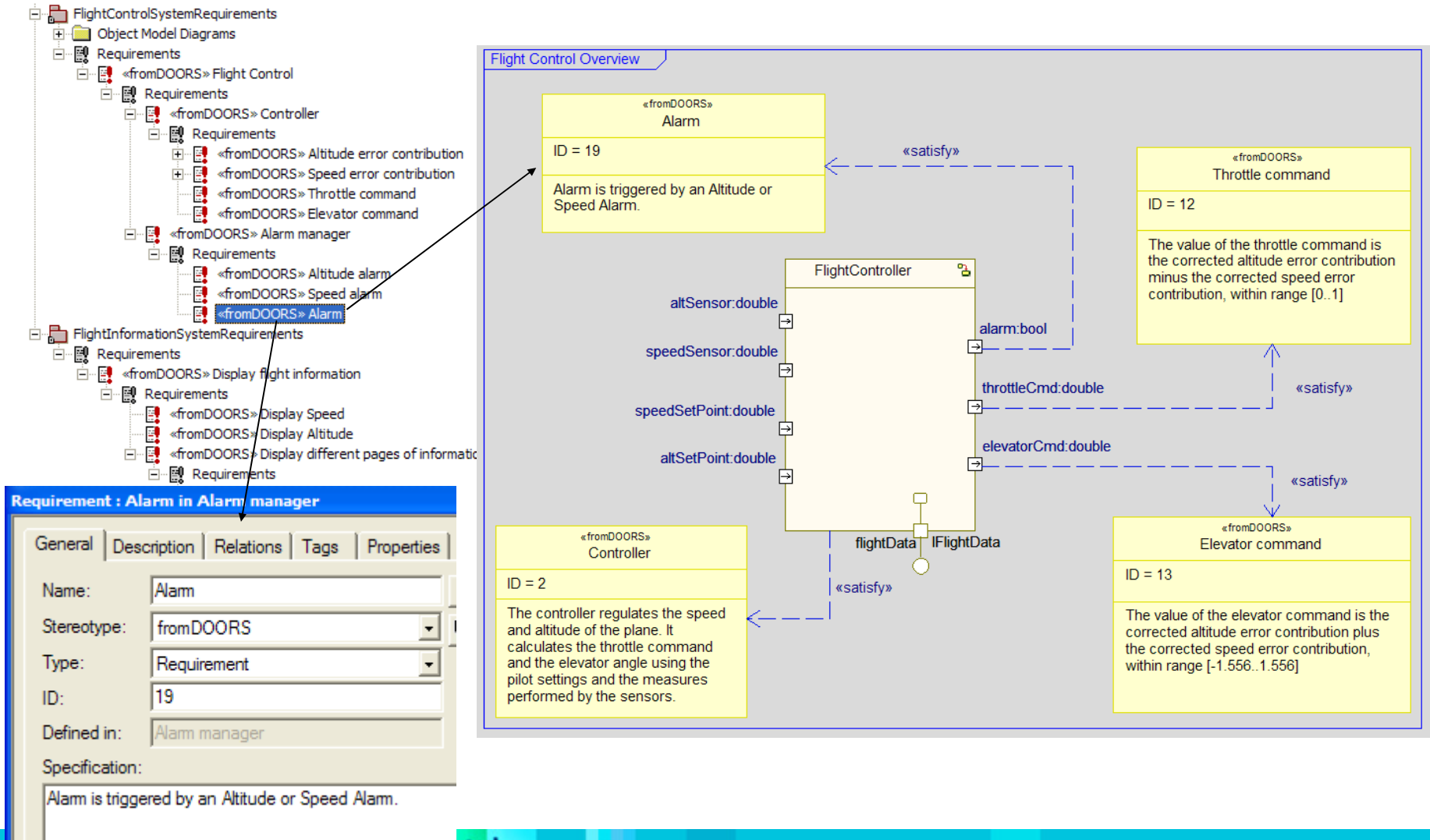


# Requirements Modelling

- **Requirements Capture**
- **Requirements Traceability**
  - ▶ Create traceability links from model to requirements
  - ▶ Automatic traceability documentation
- **Requirements Analysis**
  - ▶ Requirement Coverage Analysis
  - ▶ Change Impact analysis
  - ▶ Automatic report generation

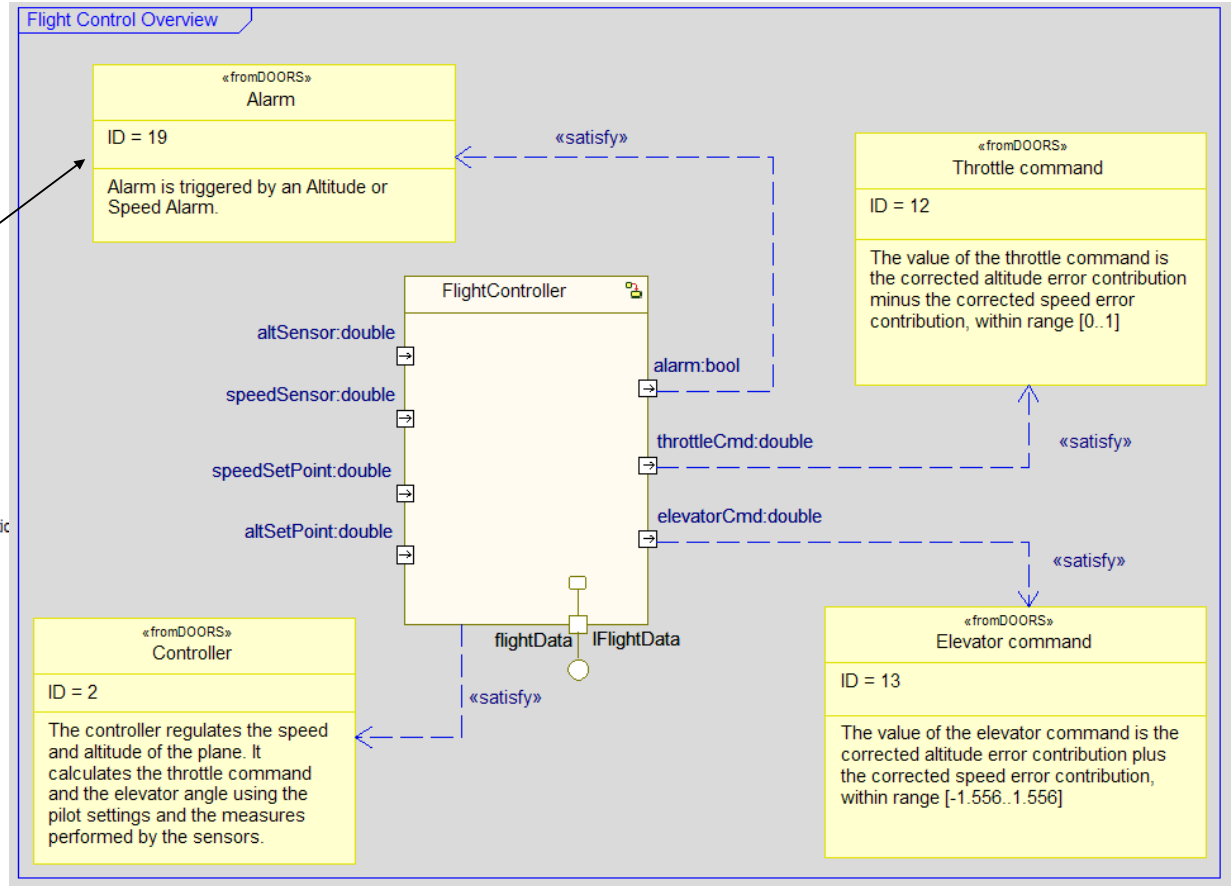


# Requirements Capture and Trace



**Requirement: Alarm in Alarm manager**

General	Description	Relations	Tags	Properties
Name:	Alarm			
Stereotype:	fromDOORS			
Type:	Requirement			
ID:	19			
Defined in:	Alarm manager			
Specification:	Alarm is triggered by an Altitude or Speed Alarm.			



# Requirements Coverage Analysis

Telelogic Rhapsody Gateway - V71\_RiCpp\_FlightControl\_And\_Information\_System

File Edit View Tools Reports Help

Management View Coverage Analysis View Impact Analysis View Graphical View Requirement Details

Upstream Coverage Information: Selection:

- Rule check
  - Uncovered requirement
    - Correction gain
    - Contribution centering
    - P1
- UML Model Rhapsody
  - V71\_RiCpp\_FlightControl\_And\_Information\_System
  - FlightInformationSystemRequirements DOORS
    - Display flight information
  - FlightControlSystemRequirements DOORS
    - Flight Control
      - Controller
        - Altitude error contribution
        - Speed error contribution
          - P1
        - Throttle command
        - Elevator command
      - Alarm manager
        - Altitude alarm
        - Speed alarm
        - Alarm

Downstream Coverage Information:

- UML Model Rhapsody 76%
  - V71\_RiCpp\_FlightControl\_And\_Information\_System
    - Packages
      - FlightControlSystemPkg
        - Classes
          - FlightControlGui
            - FlowPorts
              - alarm
          - FlightController
            - Statechart
              - States
                - Alarmed
              - Attributes
                - alarm
              - FlowPorts
                - alarm

Texts and Reference Attributes | Attributes | Messages

Upstream  
Text:  
Reference Attributes:

Selection  
Text:  
Alarm is triggered by an Altitude or Speed Alarm.

Downstream  
Text:  
Reference Attributes:

FlightControlSystemRequirements DOORS/Flight Control/Alarm manager/Alarm

# DFT : Executable Models on Host & Target

The screenshot displays the Rational Rhapsody IDE interface. The main window shows a sequence diagram titled "Sequence Diagram: Animated Normal Operation \*". The diagram includes lifelines for Builder, Reader, Command, and Writer. The statechart window, titled "Statechart of : Builder - Builder(0) \*", shows a state named "running" with a self-loop labeled "count++". A timer event "tm(500)" leads to a condition "[rand()%2]==0". If true, it triggers "evWrite to itsWriter"; otherwise, it triggers "evRead to itsReader".

Below the statechart, the "Features of Builder(0)" dialog is open, showing the instance name "Builder(0)" and a table of attributes:

Name	Value	Type
count	0	int

The dialog also shows relations for "itsCommandList", "itsReader", and "itsWriter". At the bottom, the "Call Stack" and "Event Queue" windows are visible, along with the status bar showing "GE MODE" and the date "Thu, 1, Mar 2007 6:21 PM".

# DFT : Rapid HTML Gui's

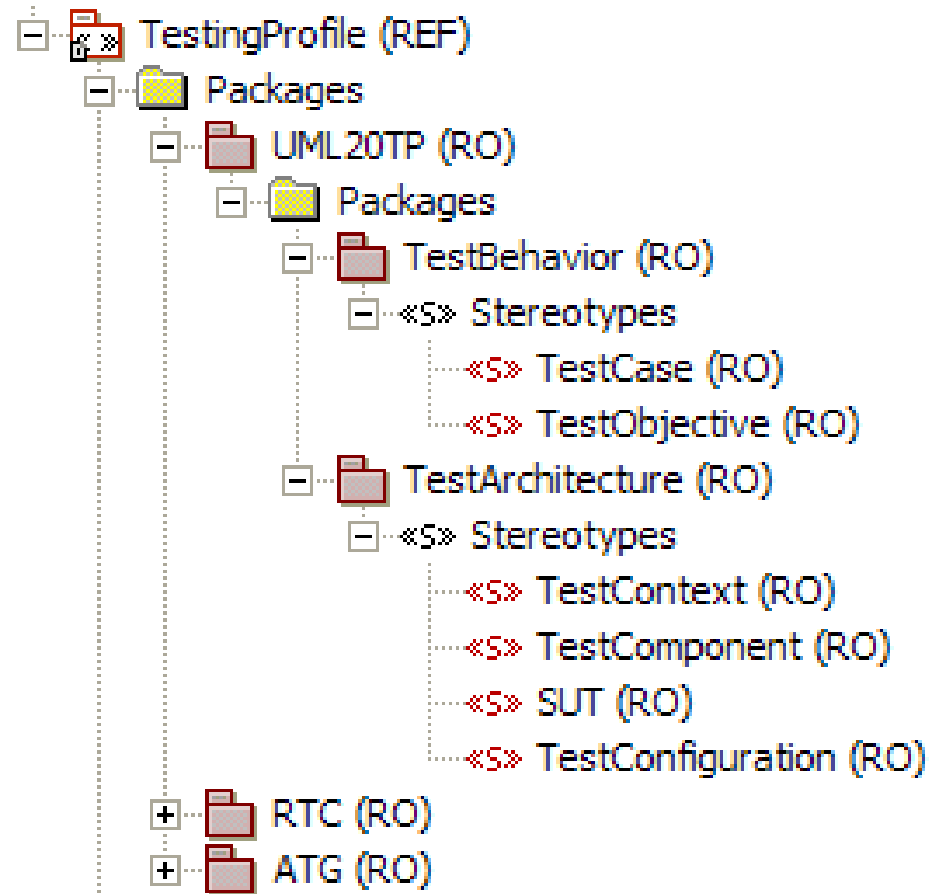
The screenshot displays the Rational software development environment. On the left, a Windows Internet Explorer browser shows a local web page titled 'V71\_RiCpp\_BluetoothHeadset' with a 'Builder[0]' component. The main workspace is divided into several panels:

- Entire Model View:** A tree view showing the project structure, including packages like 'AnalysisPkg', 'PresentationPkg', 'InterfacePkg', 'MobilePhonePkg', and 'HeadsetPkg'. It also lists various events such as 'evPress()', 'evLongPress()', and 'evRelease()'.
- Statechart of : Button - Builder[0]->itsHeadset->itsButton:** A state machine diagram with states 'idle', 'debounce', and 'pressed'. Transitions include 'evPress', 'evRelease', and 'tm(100)'. Actions include 'itsHeadset->GEN(evShortPress)' and 'itsHeadset->GEN(evLongPress)'.
- Statechart of : Headset - Builder[0]->itsHeadset:** A state machine diagram with states 'off', 'disconnected', 'connecting', and 'connected'. Transitions include 'evLongPress', 'evConnect', and 'evLive'. Actions include 'itsLed GEN(evOn)' and 'OPORT(blue)->GEN(connectDevice)'.

At the bottom, there are panels for 'Call Stack', 'Event Queue', and a status bar showing 'GE MODE' and the date 'Fri, 2, Mar 2007 11:35 AM'.

# DFT : Model Based Testing

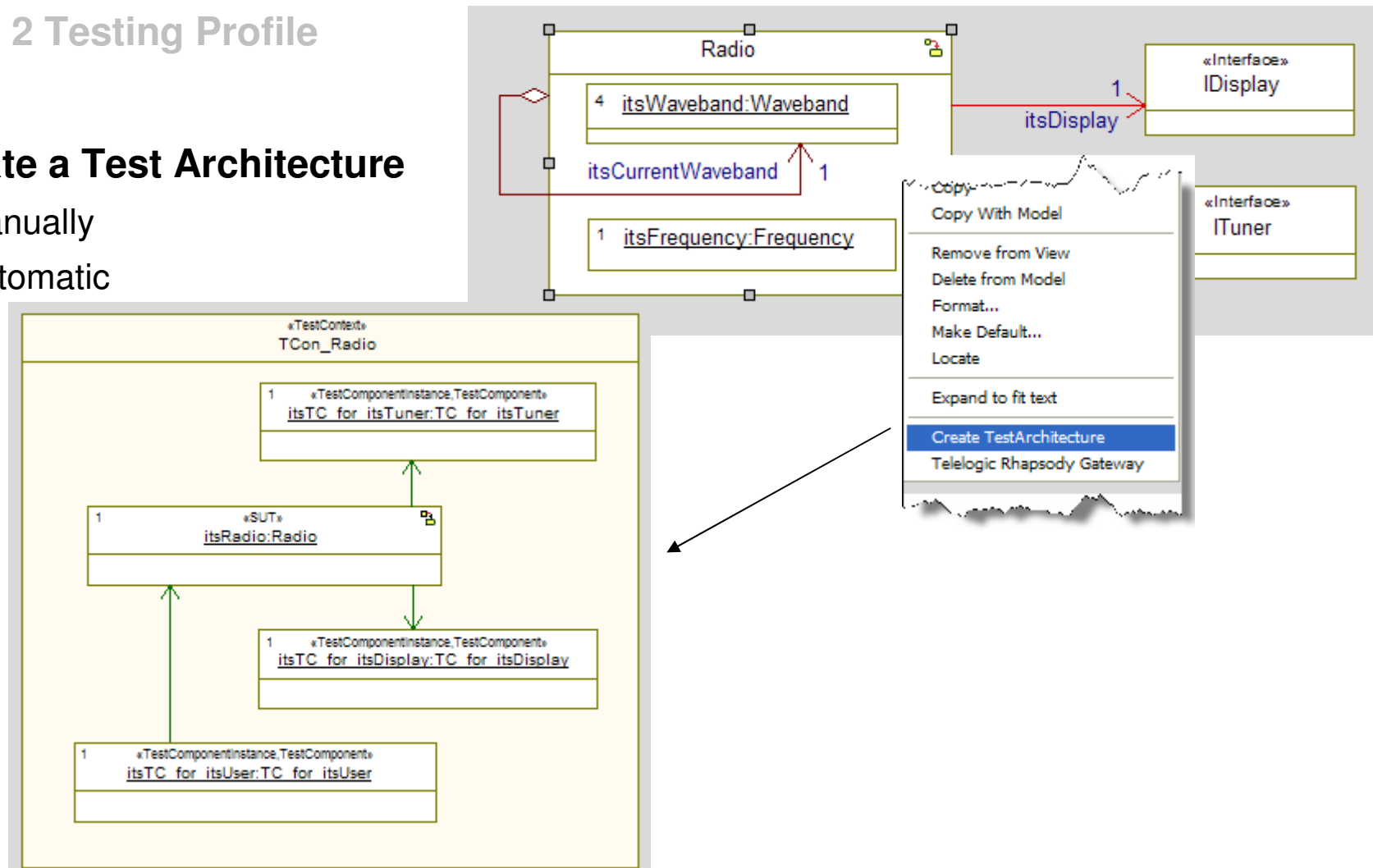
- UML 2 Testing Profile





# DFT : Model Based Testing

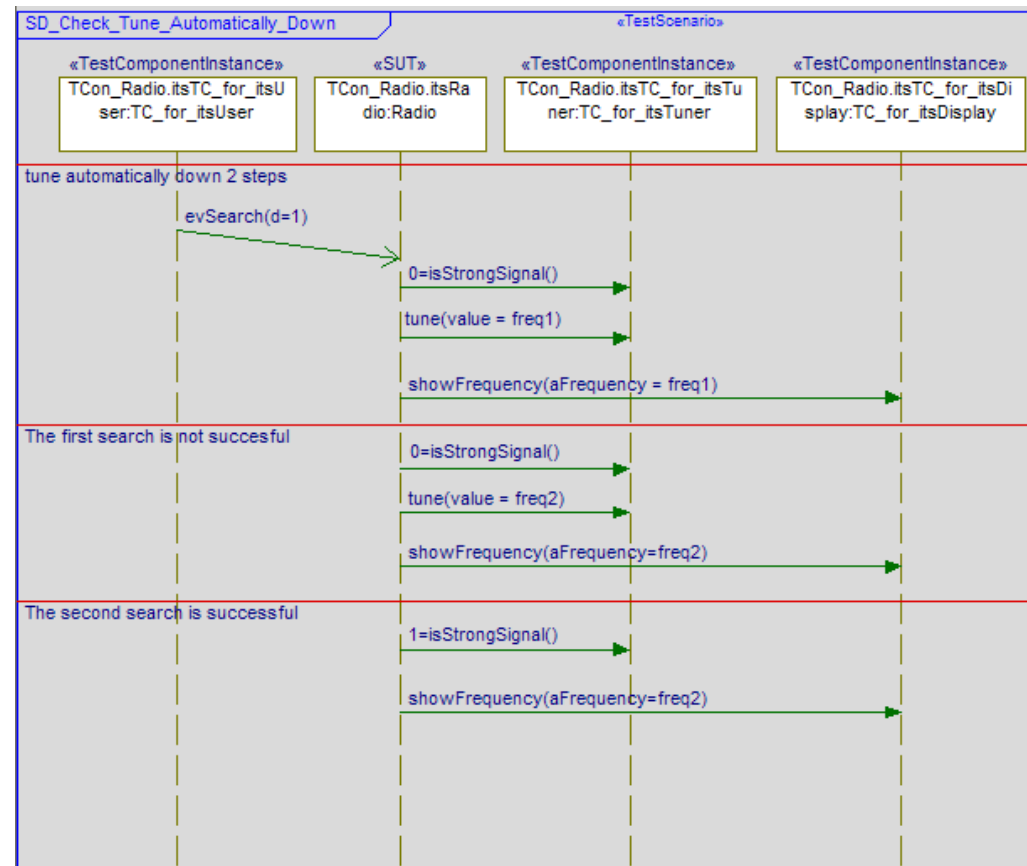
- UML 2 Testing Profile
- Create a Test Architecture
  - ▶ Manually
  - ▶ Automatic





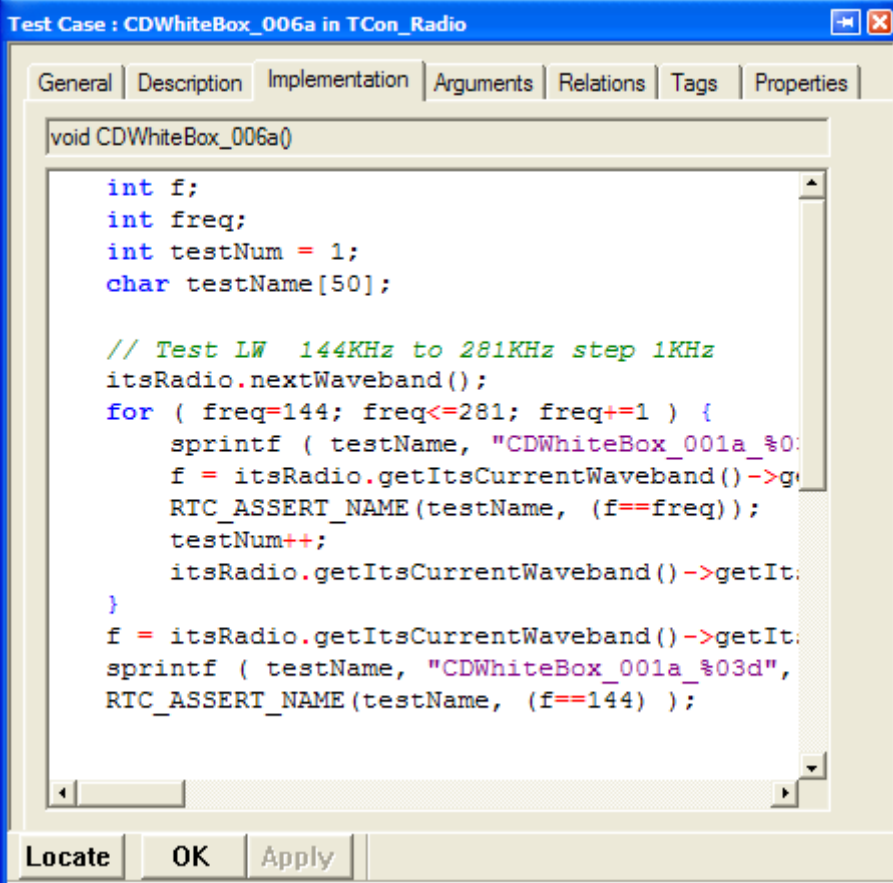
# DFT : Model Based Testing

- UML 2 Testing Profile
- Create a Test Architecture
  - ▶ Manually
  - ▶ Automatic
- Create Test Cases
  - ▶ Test Cases can be written:
    - Via Sequence Diagrams



## DFT : Model Based Testing

- UML 2 Testing Profile
- Create a Test Architecture
  - ▶ Manually
  - ▶ Automatic
- Create Test Cases
  - ▶ Test Cases can be written:
    - Via Sequence Diagrams
    - Manually via code



The screenshot shows a window titled "Test Case : CDWhiteBox\_006a in TCon\_Radio". The window has tabs for "General", "Description", "Implementation", "Arguments", "Relations", "Tags", and "Properties". The "Implementation" tab is selected, displaying the following C code:

```
void CDWhiteBox_006a()

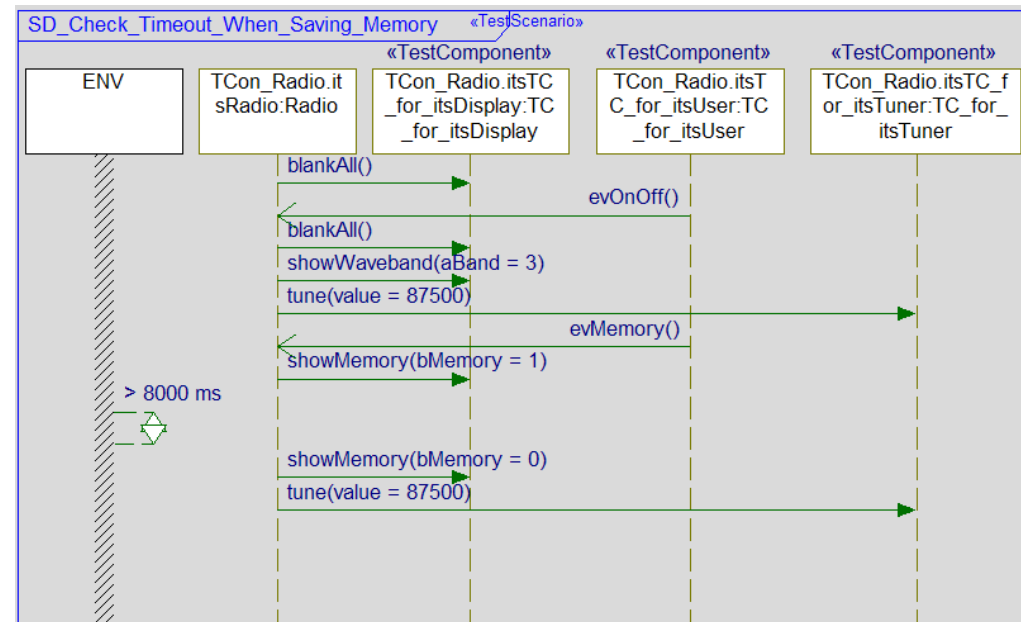
int f;
int freq;
int testNum = 1;
char testName[50];

// Test LW 144KHz to 281KHz step 1KHz
itsRadio.nextWaveband();
for ( freq=144; freq<=281; freq+=1 ) {
    sprintf ( testName, "CDWhiteBox_001a_%03d", testNum );
    f = itsRadio.getItsCurrentWaveband()->getIt();
    RTC_ASSERT_NAME(testName, (f==freq));
    testNum++;
    itsRadio.getItsCurrentWaveband()->getIt();
}
f = itsRadio.getItsCurrentWaveband()->getIt();
sprintf ( testName, "CDWhiteBox_001a_%03d", testNum );
RTC_ASSERT_NAME(testName, (f==144) );
```

At the bottom of the window, there are buttons for "Locate", "OK", and "Apply".

# DFT : Model Based Testing

- UML 2 Testing Profile
  - Create a Test Architecture
    - ▶ Manually
    - ▶ Automatic
  - Create Test Cases
    - ▶ Test Cases can be written:
      - Via Sequence Diagrams
      - Manually via code
      - Automatically via the (ATG)  
Automatic Test Generator



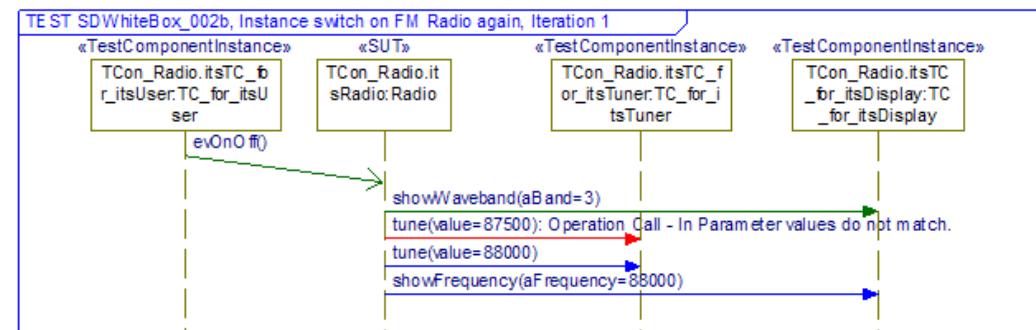
# DFT : Model Based Testing

- UML 2 Testing Profile

- Create a Test Architecture
  - Manually
  - Automatic

- Create Test Cases
  - Test Cases can be written:
    - Via Sequence Diagrams
    - Manually via code
    - Automatically via the (ATG)

Name	Description	Result
SDWhiteBox_001	Check that the Radio can be switched on and off	Passed
SDWhiteBox_002a	Check that when the radio is switched on, that it remembers the waveband and frequency that had previously been selected.	Passed
SDWhiteBox_002b	Check that when the radio is switched on, that it remembers the waveband and frequency that had previously been selected. For this test ensure that the radio is tuned to a different frequency than the default one.	Failed
SDWhiteBox_003	Check that the radio can be tuned forwards and backwards	Passed
SDWhiteBox_004	Check that if the user starts to setup a preset that if they don't complete the setup then after 8 seconds the setup is cancelled. This test uses a test scenario that was generated by the ATG.	Passed
CDWhiteBox_006a	Check that the radio cannot be tuned to a frequency outside of the limits for LW waveband.	Passed
CDWhiteBox_006b	Check that the radio cannot be tuned to a frequency outside of the limits for MW waveband.	Passed
CDWhiteBox_006c	Check that the radio cannot be tuned to a frequency outside of the limits for SW waveband.	Passed
CDWhiteBox_006d	Check that the radio cannot be tuned to a frequency outside of the limits for FM waveband.	Passed
FCWhiteBox_007	Check that each preset can be set to the minimum and maximum frequency for each waveband. Check that these presets are remembered even after the radio has been switched off and then back on.	Passed



- Execute Test Cases
  - The Test Cases can be executed automatically



## Full Application Code Generation

---

- **Rhapsody leverages *all* structural and behavioral model views to produce an executable application**
  - Structure models
  - State charts: event driven behavior
  - Activity graphs: algorithms and process flows
  - Components and artifacts
  
- **Rhapsody generates very clean, readable code, easily debugged through any commercial IDE**
  - Integrated “white-box” Code (C, C++, Java, Ada, IDL) generation
  - MISRA C compliant code generation
  - High productivity; low cost of maintenance
  
- **Rhapsody generates all application construction artifacts to provide an integrated build environment**
  
- **Comprehensive code generation technologies**
  - OO based and / or functional based
  - Stereotype based
  - Rules based : Rules Composer / Rules Player



# Dynamic Model Code Associativity

- Change one view, the others *change automatically*
- Code and Model always in sync

The screenshot displays two Eclipse IDE windows side-by-side, illustrating dynamic model code associativity. The left window, titled "Java - AlarmManager.java - Eclipse SDK", shows the source code for the `AlarmManager` class. The code includes package declarations, a class comment, and a public class definition with a constructor. The right window, titled "Rhapsody in J by Telelogic - [Object Model Diagram: Alarm Overview in com::telelogic::alarmSystem \*]", shows an object model diagram for the `AlarmManager` class. The diagram is a simple rectangular box with a title bar and a central area. The IDE interface includes a Package Explorer on the left, a toolbar at the top, and a status bar at the bottom. A message window at the bottom of the right window displays "Code Generation Done" with "0 Error(s), 0 Warning(s), 0 Message(s)".

```
##### package com::telelogic::alarmSystem
/**
 * This is a simple AlarmManager class
 */
##### class AlarmManager
public class AlarmManager {

    // Constructors

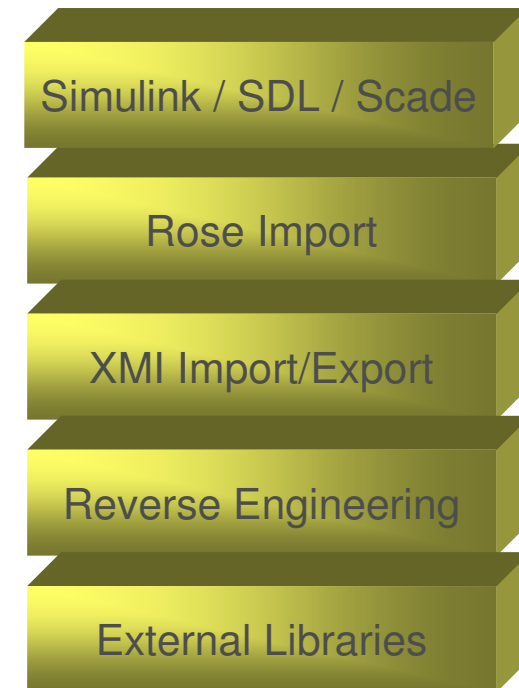
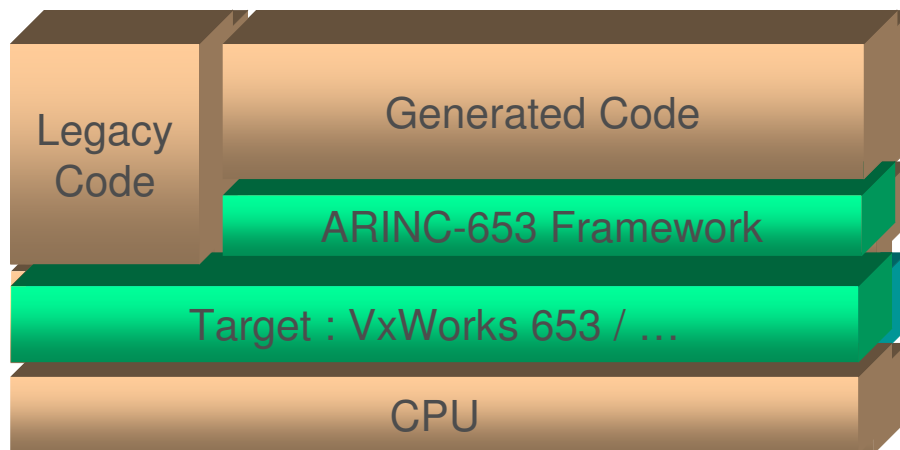
    ##### auto_generated
    public AlarmManager() {
    }

}
#####
File Path : D:/Eclipse/JavaTest/com/telelogic/
#####
```

## Real-Time Framework

### *Rhapsody provides an executable real-time framework*

- **Framework is delivered as a Rhapsody model**
  - ▶ Provides a clear understanding of structure and functionality, enabling fine tuning
  - ▶ Model includes all requirements and design rationale so it's easily understood
  - ▶ Validation suite can be made available through Professional Services
- **Facilitates scaling down for smaller footprint applications**
  - ▶ Pick and choose only the necessary components
  - ▶ Customize components
- **Facilitates certification of the framework**
  - ▶ ex: DO-178B.



## ARINC – 653 Framework

---

- **Use APEX API**
  
- **Framework Certification Package includes:**
  - ▶ Framework and Utilities
  - ▶ Modelling Guidelines
  - ▶ Services Deployment Package
  - ▶ Certification Artifacts and Documents
  
- **Certification package was completed by Verocel in April 2007 for RiC framework**





# Certification Package Artifacts and Documents

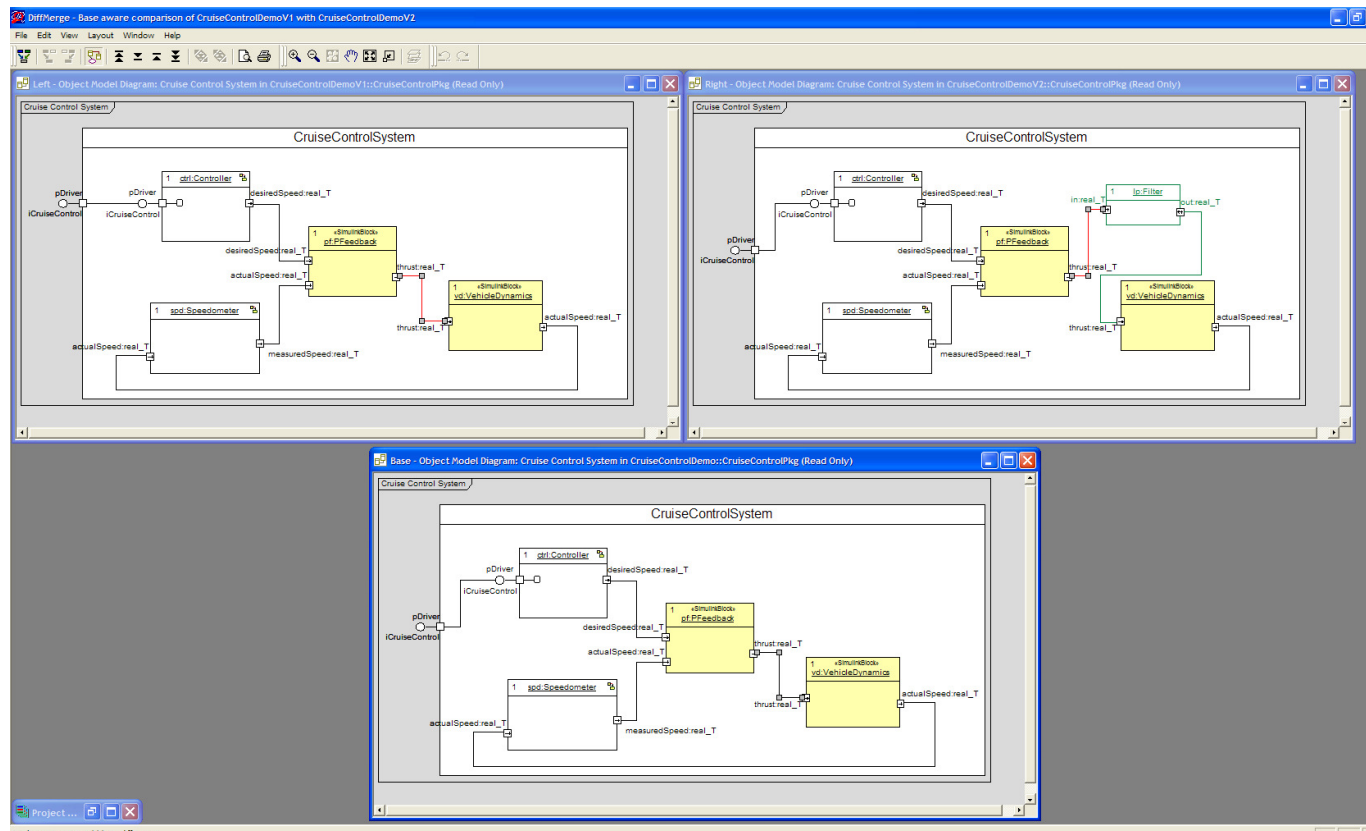
- OXF
  - Requirements
    - Development Requirements Traceability
    - Implementation Requirements Traceability
  - Source Files
    - Traceability
    - PC-Lint Results
  - Design Components
    - Traceability by Source Code
    - Traceability by Function
  - Test Procedures/Results/Reviews
    - Functional Testing
      - Traceability
      - Review Functional Test
      - Summary Functional Test
    - Coverage Analysis
      - Traceability
    - Control Coupling Results
      - OXF Map Check
      - OXF
      - OXF Control Coupling Analysis
  - Documentation

- OXF
  - Requirements
  - Source Files
  - Design Components
  - Test Procedures/Results/Reviews
  - Documentation
    - Process Plans and Standards
      - Software Configuration Management Plan
      - Software Development Plan
      - Software Design Standard
      - Software Plan Addendum
      - Software Quality Assurance Plan
      - Software Requirements Standard
      - Software Verification Plan
    - OXF Project Documents
      - Project Profile
      - Tool Accomplishment Summary
      - Tool Qualification Plan
      - Tool Test Plan
      - Software Design Document
      - Software Requirement Specification
      - High-Level to Low-Level Traceability Document
      - Software Life Cycle Environment Configuration Index
      - Software Configuration Index
      - SCI Tables
      - Requirements Traceability Document
    - OXF White Papers
    - Customer Documents
    - Tool Qualification
    - SQA Audits
    - Problem Reports

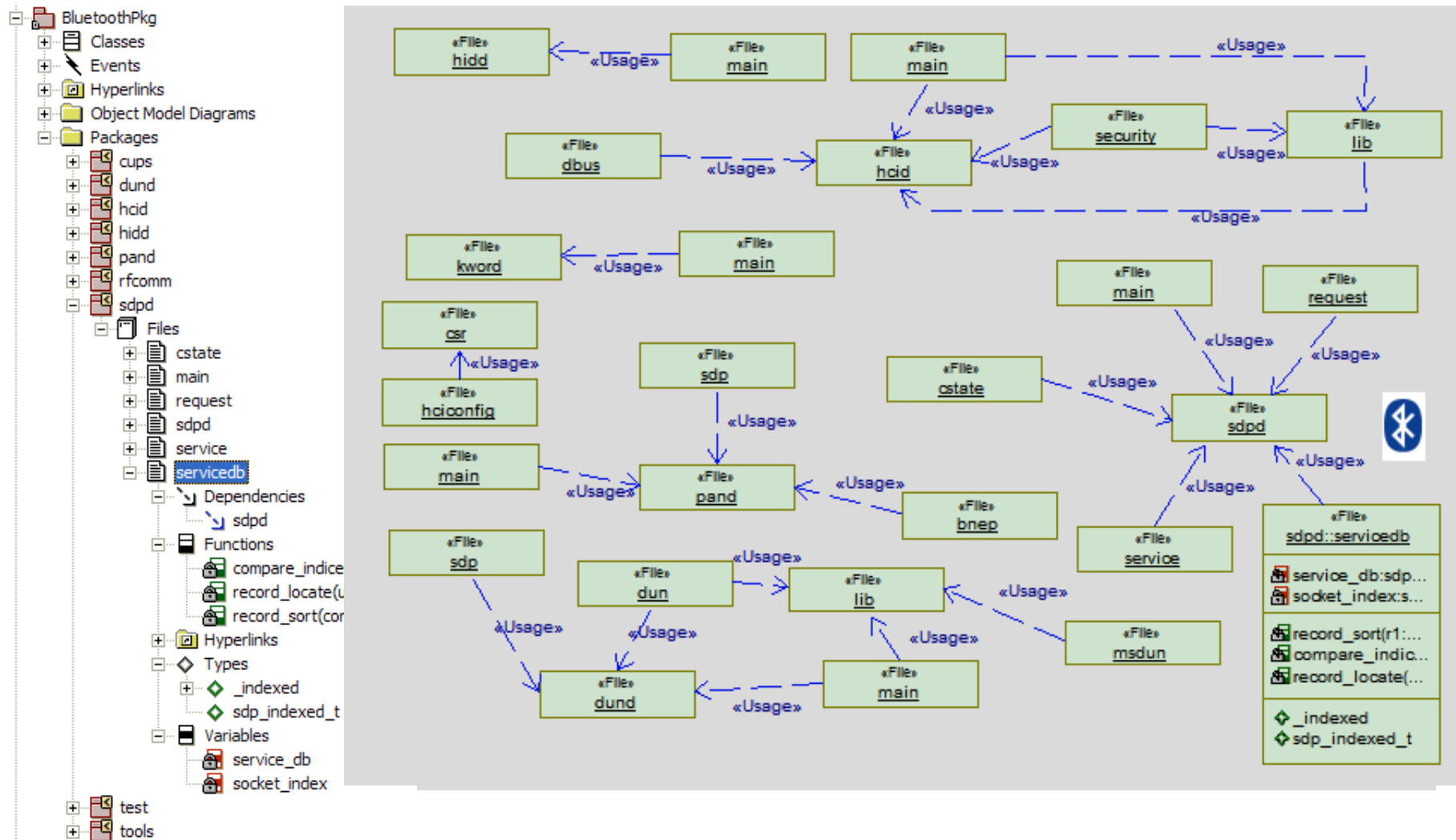


# Concurrent Design Collaboration

- Small and Large Scale Development
- Tight integration with configuration management
- Three way Visual Differencing and Merging



# Reuse of IP : Import Legacy “C” code



# Documentation: Rhapsody ReporterPLUS™

The screenshot displays the Rhapsody ReporterPLUS application window. The title bar reads "TelematicsControlUnitTemplate.tpl - ReporterPLUS". The interface is divided into several sections:

- Model Path:** |Model->[project]->[objectModelDiagrams]
- Project Tree (Left):** A hierarchical view of the project structure, including folders like "deploymentDiagrams", "hyperLinks", "itsDependencies", "objectModelDiagrams", "profiles", "property", "references", "requirements", "sequenceDiagrams", "stereotype", "stereotypes", and "structureDiagrams". Below this is a list of report sections such as "Project Introduction", "Table Of Contents", "Display All Component Diagrams", "Display all Deployment Diagrams", "Display all Structure Diagrams", "Display all Object Model Diagrams", "Display all Sequence Diagrams", "for each package", "Concurrency", "List of Interrupt Routines", "List all Active classes", "Table of all active classes", "iterate over all classes", "Text", "Gateway Rules Violation Report", and "Gateway Traceability Report".
- Properties Table (Right):** A table listing various attributes and their values.
 

Name	Type	Value
fullPathName	String	
fullPathNameIn	String	
isOfMetaclass	Boolean	
isReferenceUnit	Boolean	
isShowDisplayName	Boolean	
isStub	Boolean	
Language	String	
lastModifiedTime	String	
metaClass	String	
name	String	
requirementTraceabilityHandle	Long	
- Report Configuration Panel (Bottom Right):** A panel with tabs for "Text", "Iteration", "Condition", "Sort", "No Data", and "Properties". The "Simple" radio button is selected. It contains three dropdown menus:
  - Attribute:** \$name of [stereotype]
  - Operation:** =
  - Value:** "Active"
 Buttons for "Clear", "Apply", and "Cancel" are located at the bottom of this panel.
- Bottom Panel:** A small table showing event data.
 

Event	evConnected
Event	evInc
Event	evDec



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](#)

© Copyright IBM Corporation 2008. All rights reserved. The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. IBM, the IBM logo, Rational, the Rational logo, and other IBM products and services are trademarks of the International Business Machines Corporation, in the United States, other countries or both. Other company, product, or service names may be trademarks or service marks of others.

