



**Gianluca Monticone**

Introduzione al Model Based System Development



**System Engineering:  
Smart Products**

# Agenda

- **Model Based System Development**
- UML/SysML
- IBM Rational Rhapsody



# Evolution of Systems Development

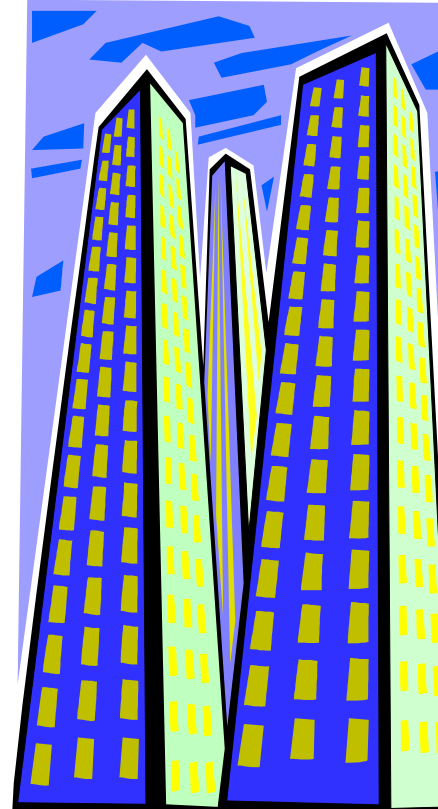
- Business environment changes
  - Device is just one part of a larger solution (e.g., iPod, Tivo, Blackberry)
    - Awareness of role in broader solution
    - Working closely with customers and partners
- Technology changes
  - Complex mixture of technologies
    - Need both device-optimized and enterprise-scale technologies
  - Integration and interoperability are mandatory
    - Increasing standardization (e.g., XML)
  - Increased role of “Systems” modeling (e.g., SysML, SOA, BPM)



# Modeling: The Key to Managing Complexity

- Manage Complexity
  - Ability to abstract detail, drill down for details
- Improve communications
  - Between internal stakeholders/teams
  - With customers
- Reduced ambiguity and errors
  - Formal languages like SysML and UML offer precise notation
- Models can seed detailed design and implementation
- Models can be simulated, documents can not.

Maybe you  
have to



But then, maybe you should

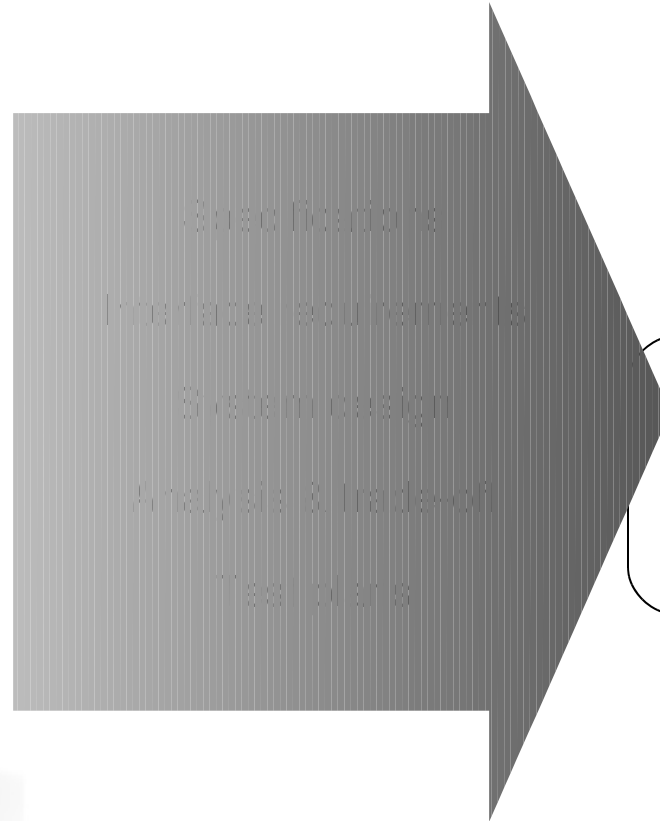
Well, maybe  
you shouldn't



# Modern Approaches for Describing Systems Are Evolving

*To Better Manage Complexity and Reduce Time-to-market*

*Past*



Use Reusable

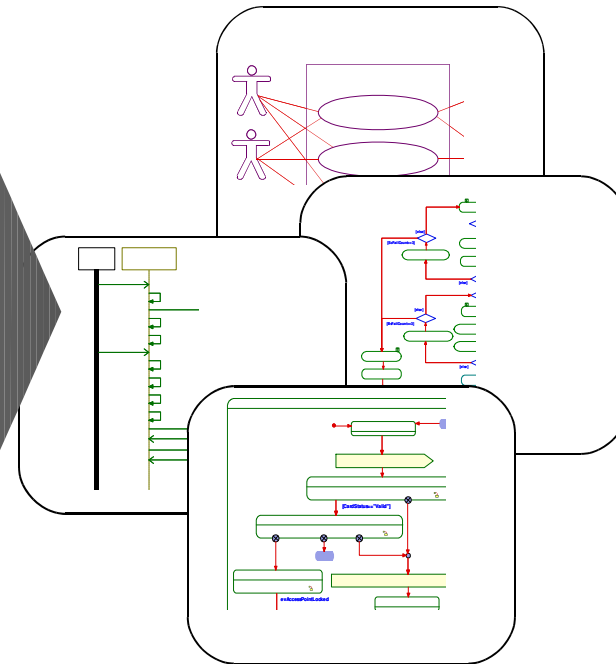
Interface Requirements

System design

Analysis & trade-off

Test plans

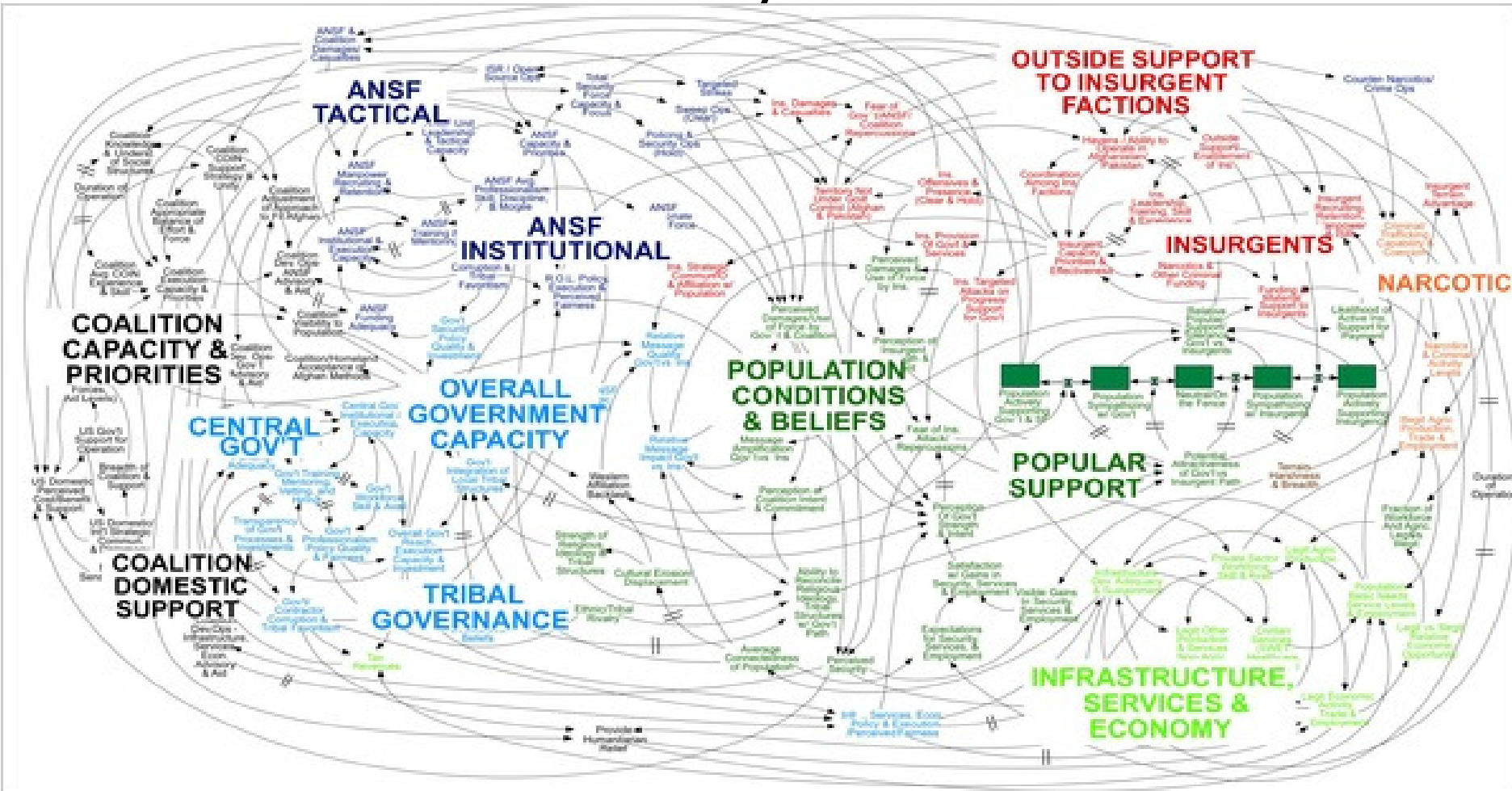
*Present/Future*



*Moving from manual methods to a model-driven approach*



# We Have Met the Enemy and He Is PowerPoint\*



A PowerPoint diagram meant to portray the complexity of American strategy in Afghanistan.

\* Gen.

Stanley A. McChrystal

Jesi, 8 Luglio 2010



# Agenda

- Model Based System Development
- **UML/SysML**
- IBM Rational Rhapsody



# UML – The Language of Model-Driven Development

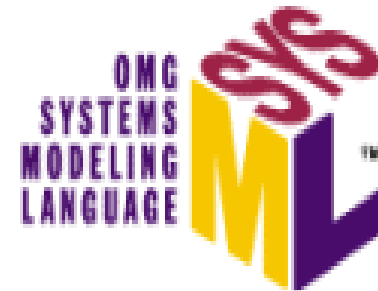
- Model-driven development is aided by a common *language* across all stakeholders
  - Unified Modeling Language (UML) is the standard language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system
  - UML allows software architects, designers and developers to specify, visualize, construct, and document all aspects of a software system





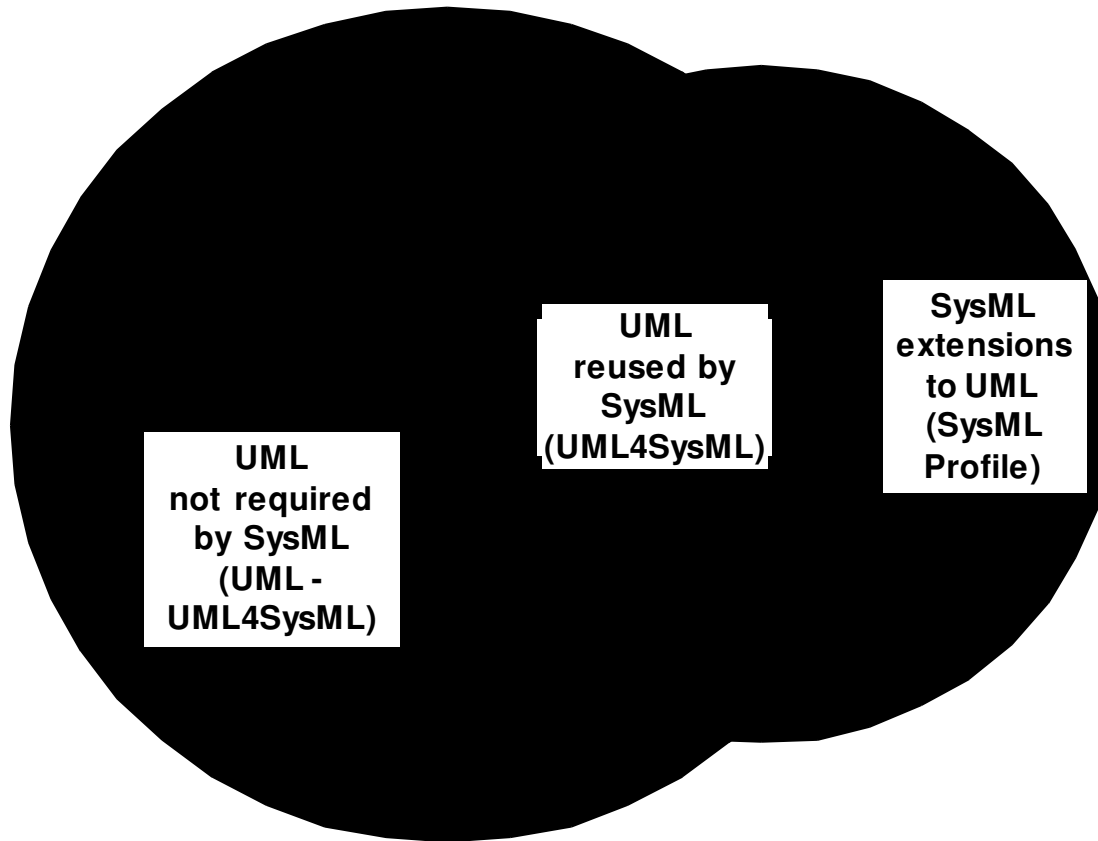
# SysML

- A graphical modelling language in response to the UML for Systems Engineering RFP developed by the OMG, INCOSE, and AP233
  - a UML Profile that represents a subset of UML 2 with extensions
- Supports the specification, analysis, design, verification, and validation of systems that include hardware, software, data, personnel, procedures, and facilities
- Supports model and data interchange via XMI

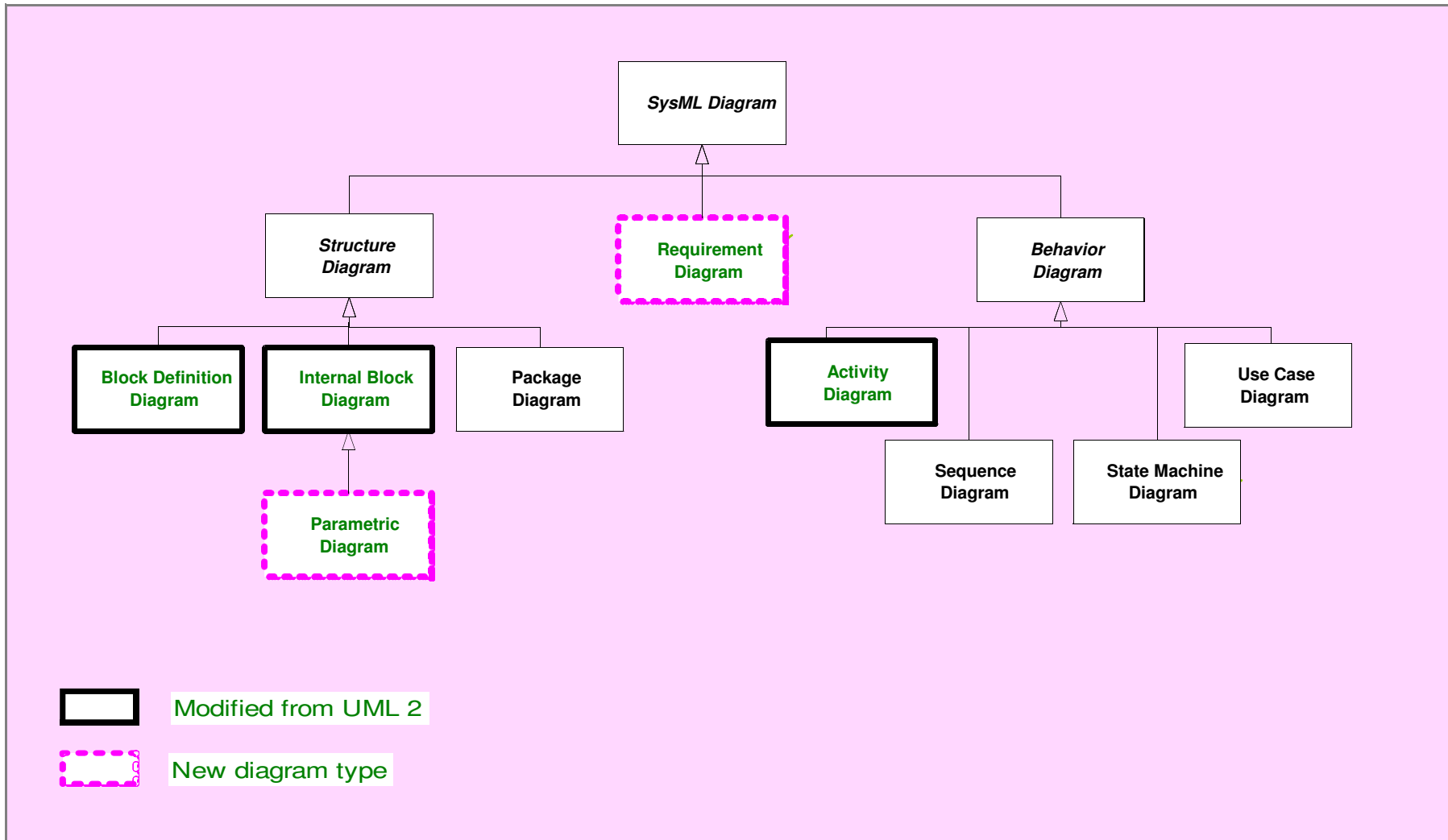


**SysML is Critical Enabler for Model Driven SE**

# Relationship Between SysML and UML



# SysML Diagram Taxonomy

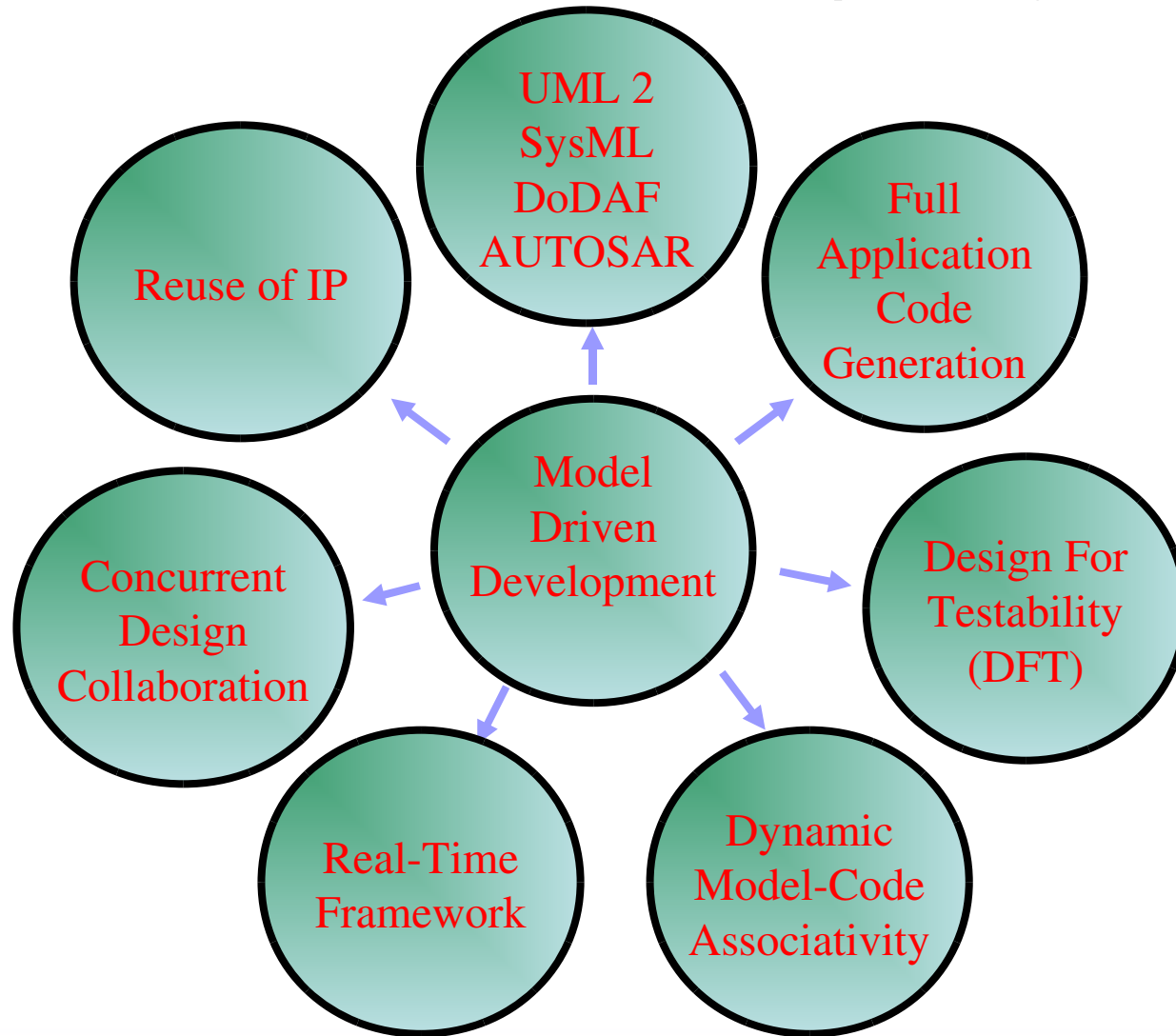


# Agenda

- Model Based System Development
- UML/SysML
- **IBM Rational Rhapsody**

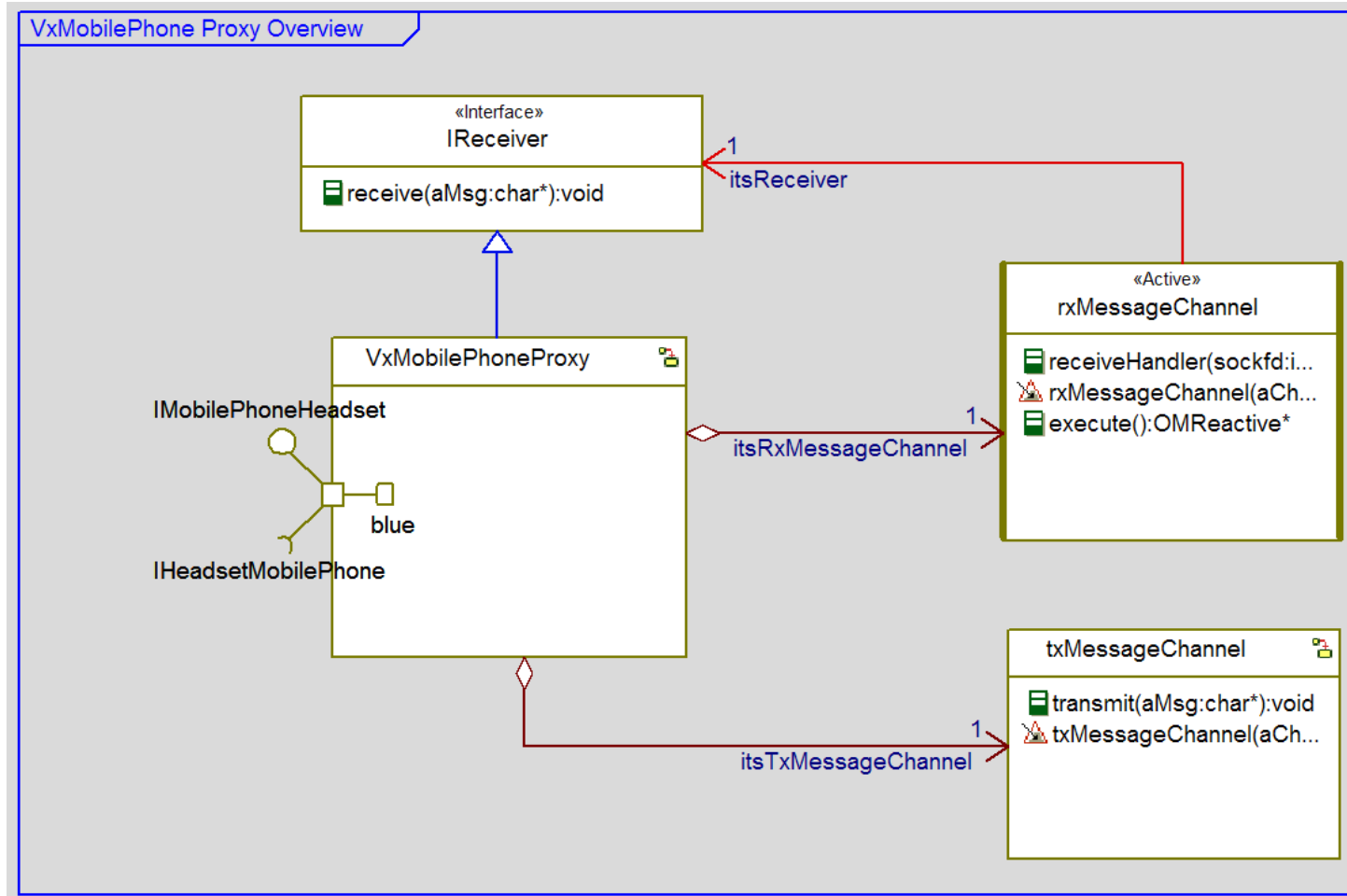


# IBM Rational Rhapsody



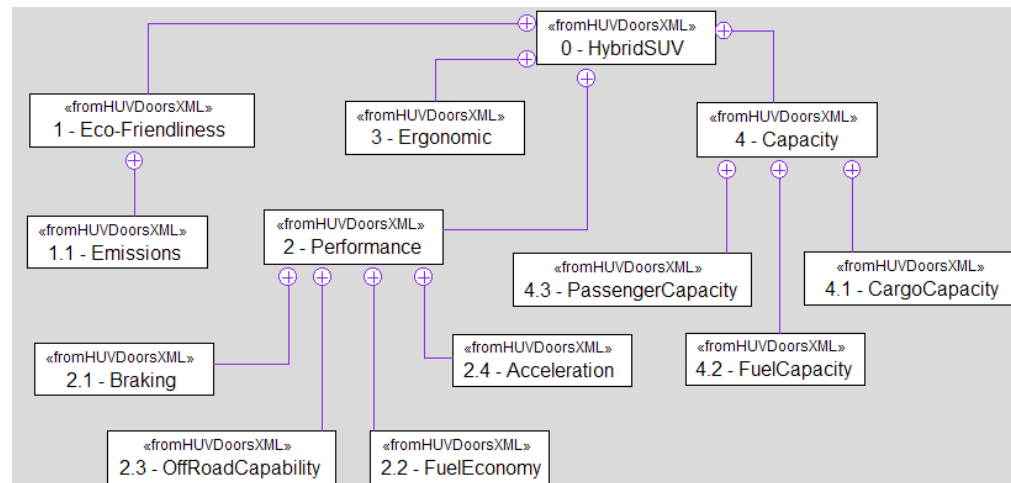
# UML 2

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



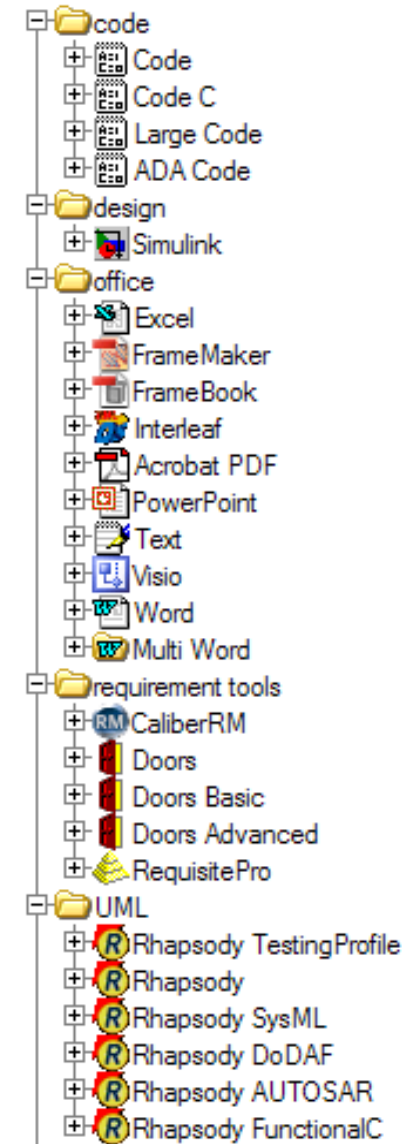
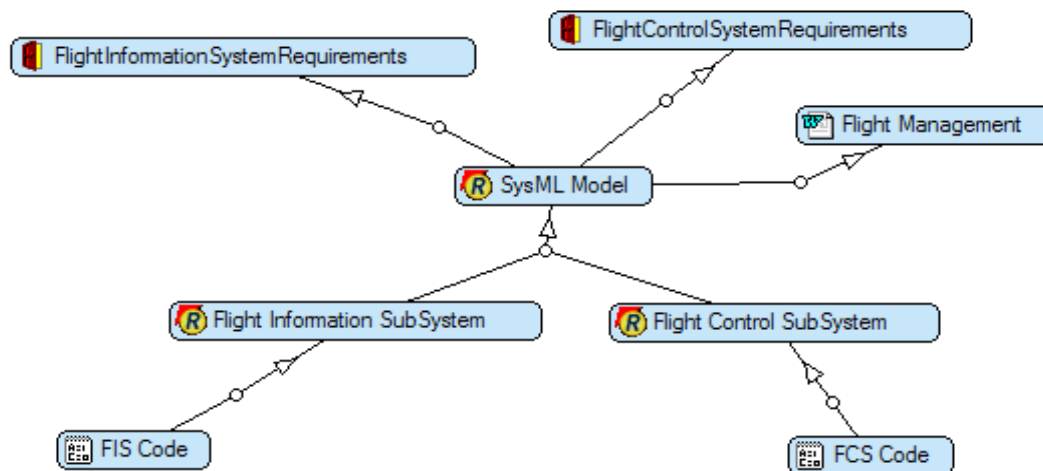
# SysML

- 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



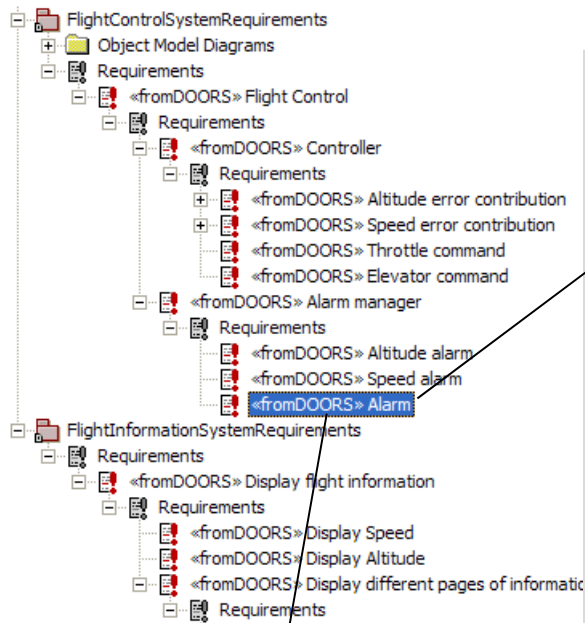
# Requirements Modelling

- 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



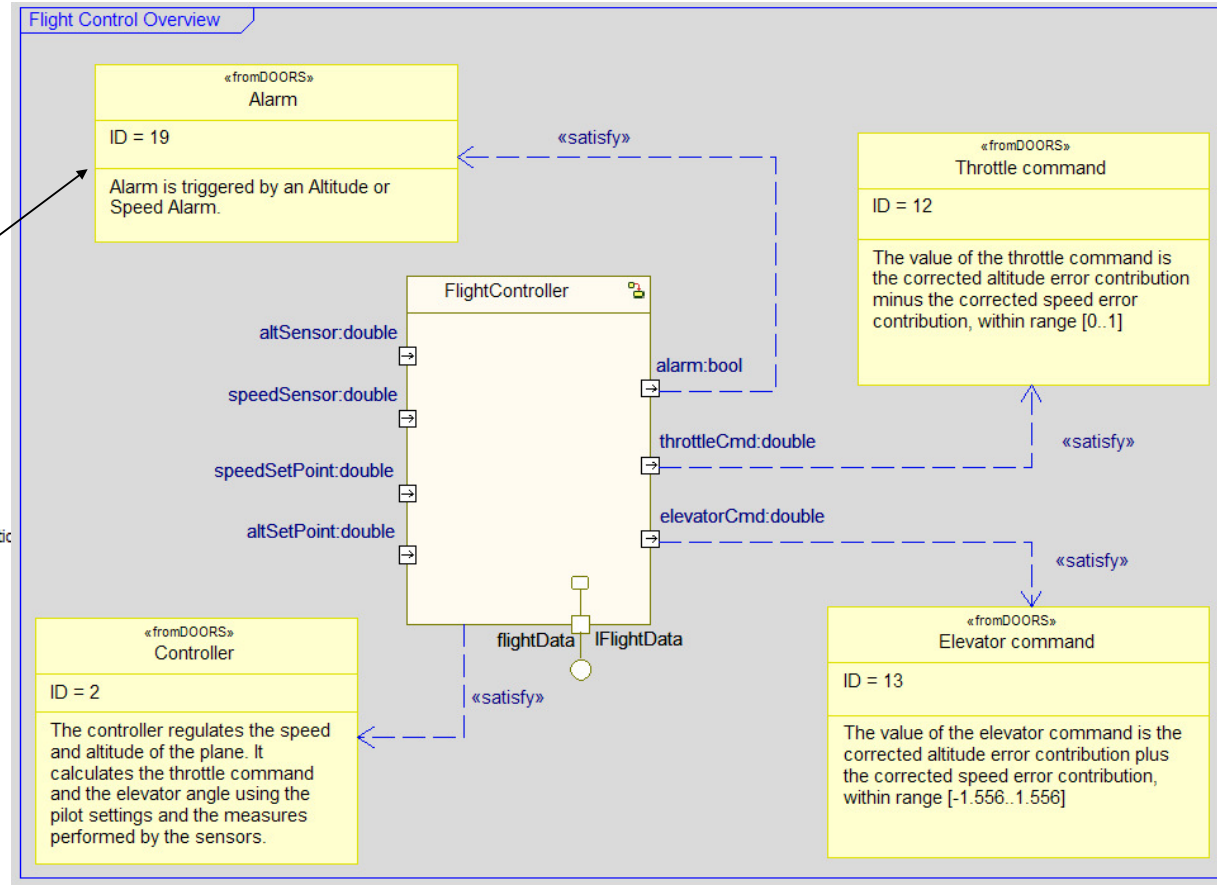


# Requirements Capture and Trace



**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
    - PI
  - 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
              - PI
            - 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
              - alam
          - FlightController
            - Statechart
              - States
                - Alarmed
            - Attributes
              - alam
            - FlowPorts
              - alam

Texts and Reference Attributes | Attributes | Messages

Upstream  
Text:  
Reference Attributes:

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

Downstream  
Text:  
Reference Attributes:

FlightControlSystemRequirements DOORS/Flight Control/Alarm manager/Alarm



# DFT : Executable Models on Host & Target

The screenshot displays the Rhapsody in C++ IDE interface. The main workspace is divided into several panes:

- Entire Model View:** A tree view on the left showing the project structure with folders for CommandDown, CommandLeft, CommandRight, CommandUp, Collaboration Diagrams, Events, Interfaces, and Object Model Diagrams.
- Sequence Diagram: Animated Normal Operation \*:** A UML sequence diagram showing interactions between objects: Builder, Reader, Command(2) Command, CommandList, Command(1) Command, and Writer. Lifelines are shown as vertical dashed lines.
- Statechart of : Builder - Builder(0) \*:** A statechart diagram for the Builder object. It features a state named 'running' with an internal transition 'count++'. A timer event 'tm(500)' leads to a condition '[(rand()%2)==0]'. From this condition, two paths emerge: one labeled 'evWrite to itsWriter' and another labeled 'evRead to itsReader'.
- Features of Builder(0):** A panel showing instance details for Builder(0). It includes a table for attributes and a list for relations.

The attribute table in the Features of Builder(0) panel is as follows:

Name	Value	Type
count	0	int

The relations section lists: itsCommandList, itsReader, and itsWriter.

At the bottom of the IDE, there are panes for 'Call Stack', 'Event Queue', and a status bar showing 'Executable is Idle', 'Build', 'Check Model', 'Configuration Management', and 'Animation'. The system tray at the bottom right indicates 'GE MODE' and the date/time 'Thu, 1, Mar 2007 6:21 PM'.



# DFT : Rapid HTML Gui's

The image displays a development environment with two main windows:

- Windows Internet Explorer:** Shows a local web page titled "V71\_RiCpp\_BluetoothHeadset" with a "Builder[0]" component. The address bar is at "http://localhost:90/".
- Rhapsody in C++:** Shows a statechart editor for "Button - Builder[0]->itsHeadset->itsButton". The statechart includes states like "idle", "debounce", and "pressed", with transitions for events like "evPress" and "evRelease".

Below the statechart editor, there is another statechart for "Headset - Builder[0]->itsHeadset" showing states like "on", "disconnected", "connecting", and "connected".

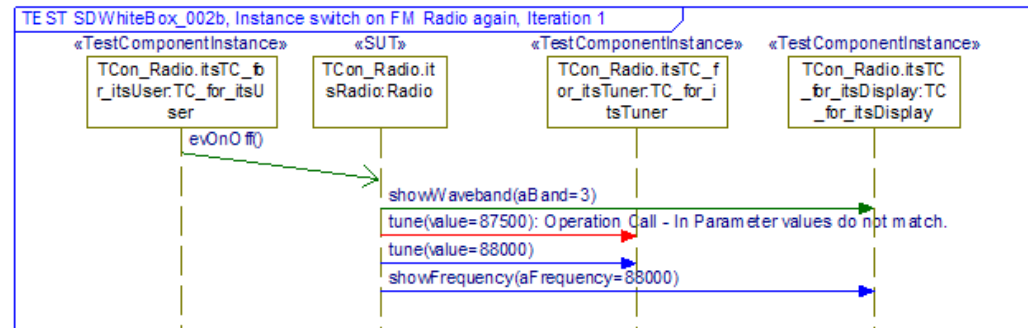
The bottom of the Rhapsody window shows a "Call Stack" and "Event Queue" panel, and a status bar with the text "For Help, press F1" and "GE MODE".



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

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



# 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



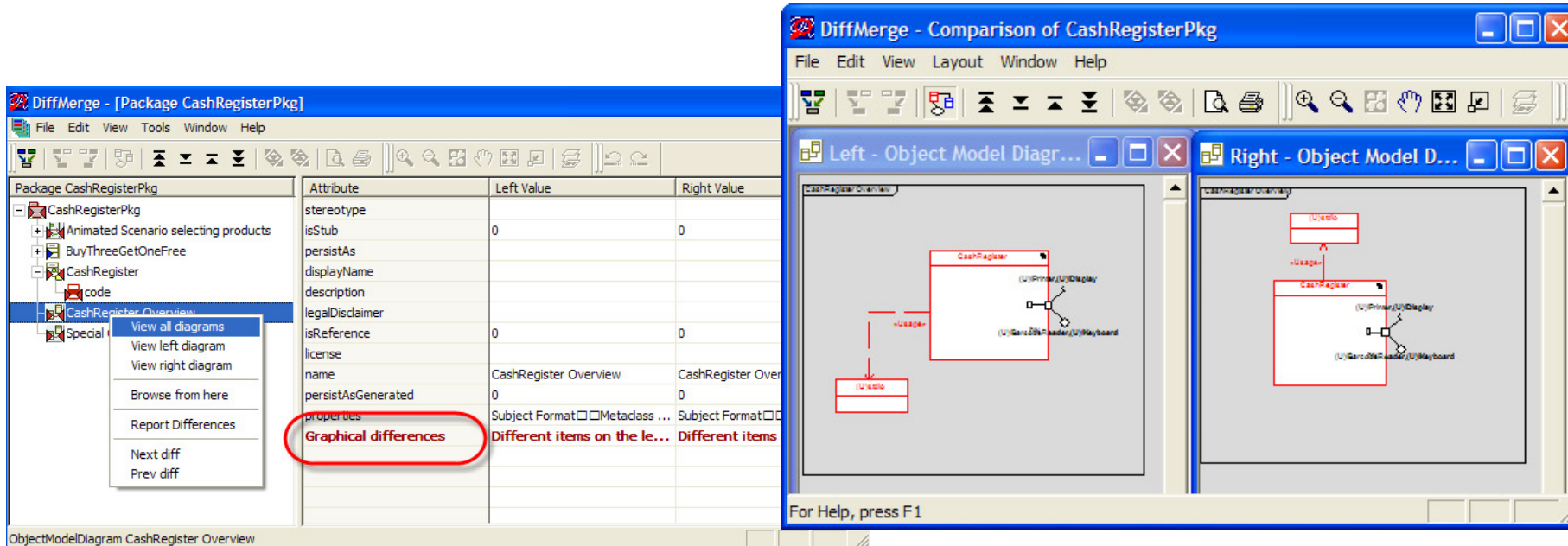
# Dynamic Model Code Associativity

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

The screenshot displays two windows from the Eclipse IDE. 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 (OMD) for the `AlarmManager` class. The diagram is a simple rectangular box with a title bar and a central area. The 'Entire Model View' on the left of the OMD window shows a hierarchical tree structure of the project, including packages and classes. The status bar at the bottom of the OMD window indicates 'Code Generation Done' with '0 Error(s), 0 Warning(s), 0 Message(s)'. The status bar at the bottom of the Eclipse IDE shows 'Attempt to establish connection with IDE was successful.' and the date 'Fri, 2, Mar 2007 2:23 PM'.

# Collaboration

- Tight integration with configuration management tools
  - Telelogic Synergy, Rational ClearCase, Rational Team Concert or any SCC-compliant tool
- Graphically identify differences between versions and evaluate design alternatives
- Quickly accept or automatically and merge changes between multiple versions







© Copyright IBM Corporation 2007. 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, the on-demand business 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.

