Gianluca Monticone

Introduzione al Model Based System Development

System Engineering:
Smart Products

Agenda

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

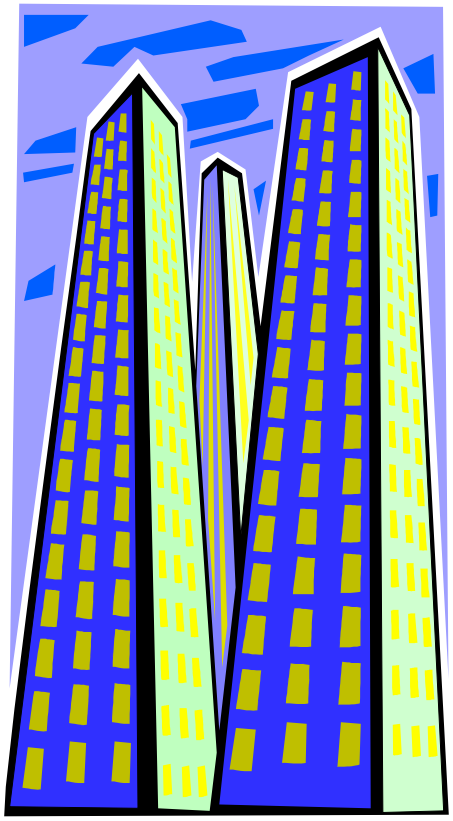
Evolution of Systems Development

- 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

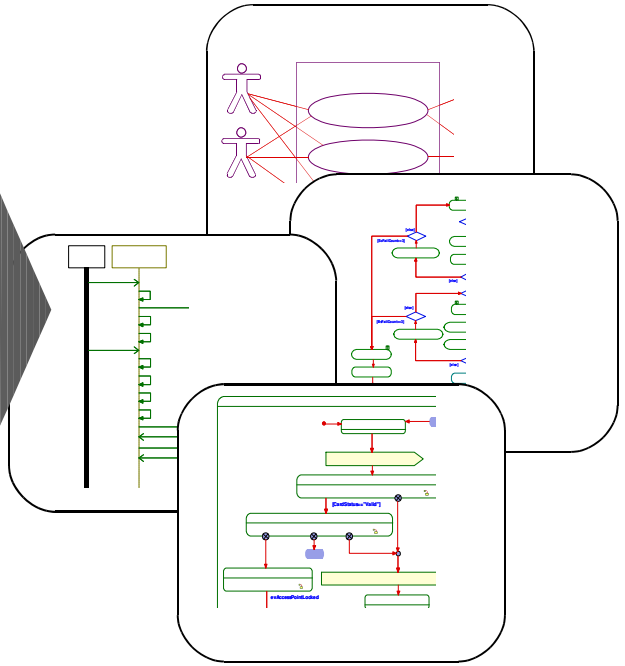
Well, maybe you shouldn't'

Modern Approaches for Describing Systems Are Evolving

To Better Manage Complexity and Reduce Time-to-market

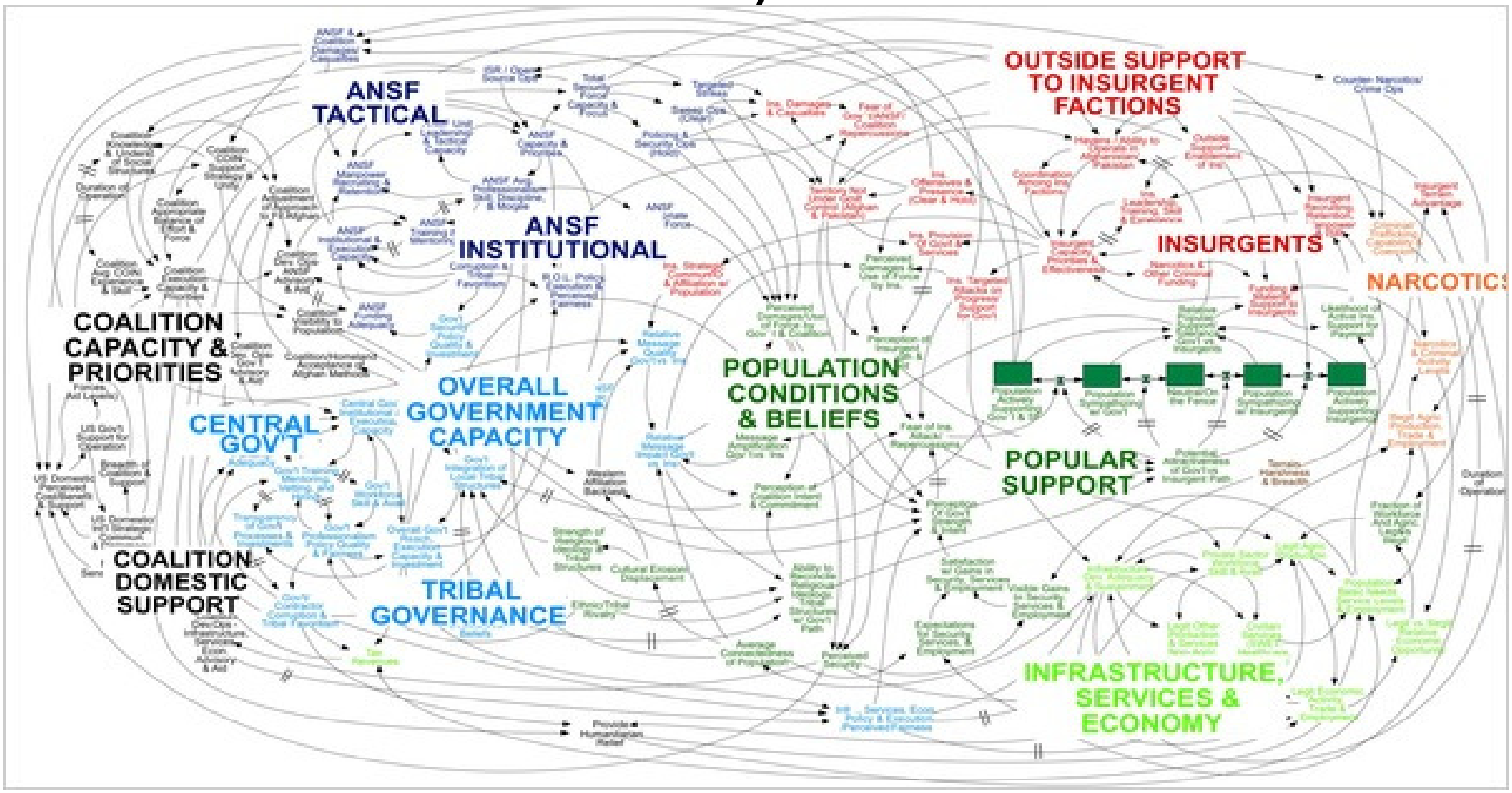
Past

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

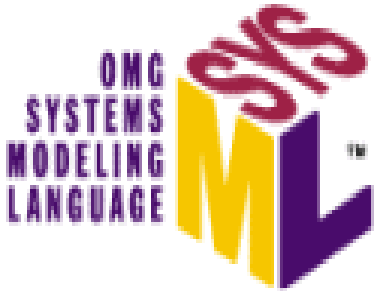
- Model Based System Development
- **UML/SysML**
- IBM Rational Rhapsody

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

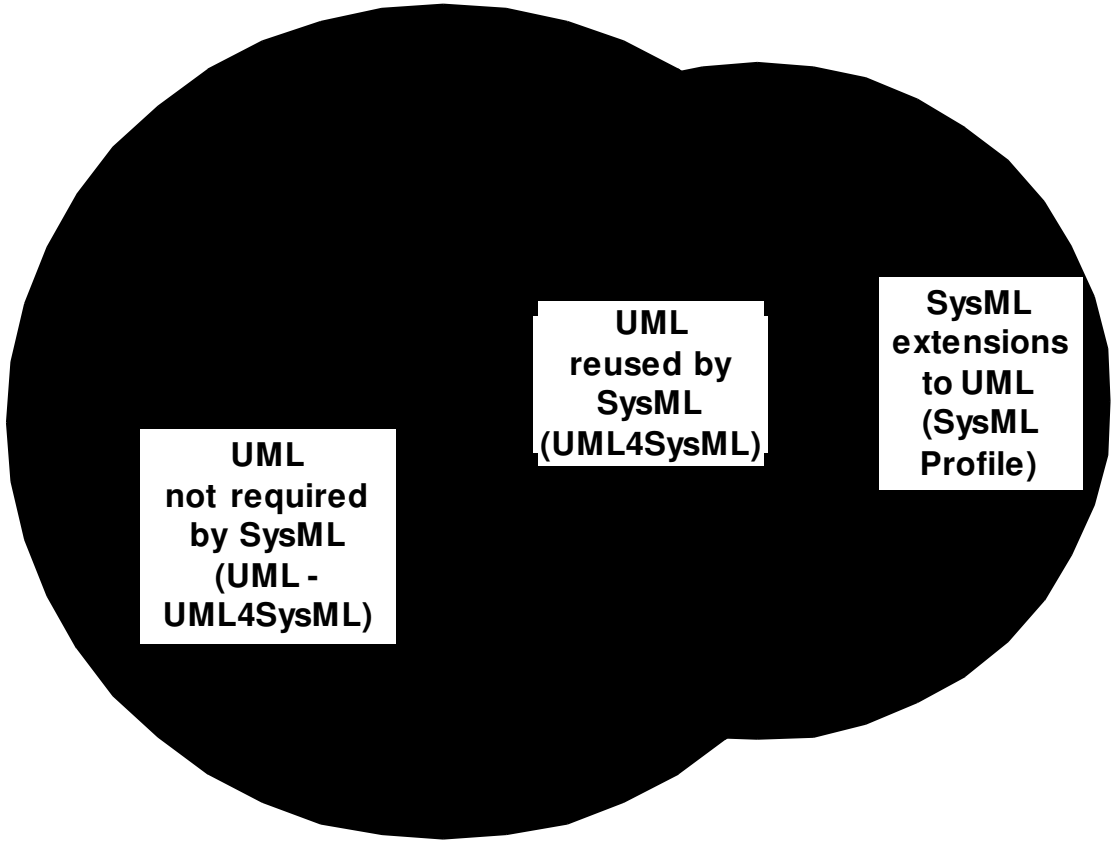
SysML

- 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

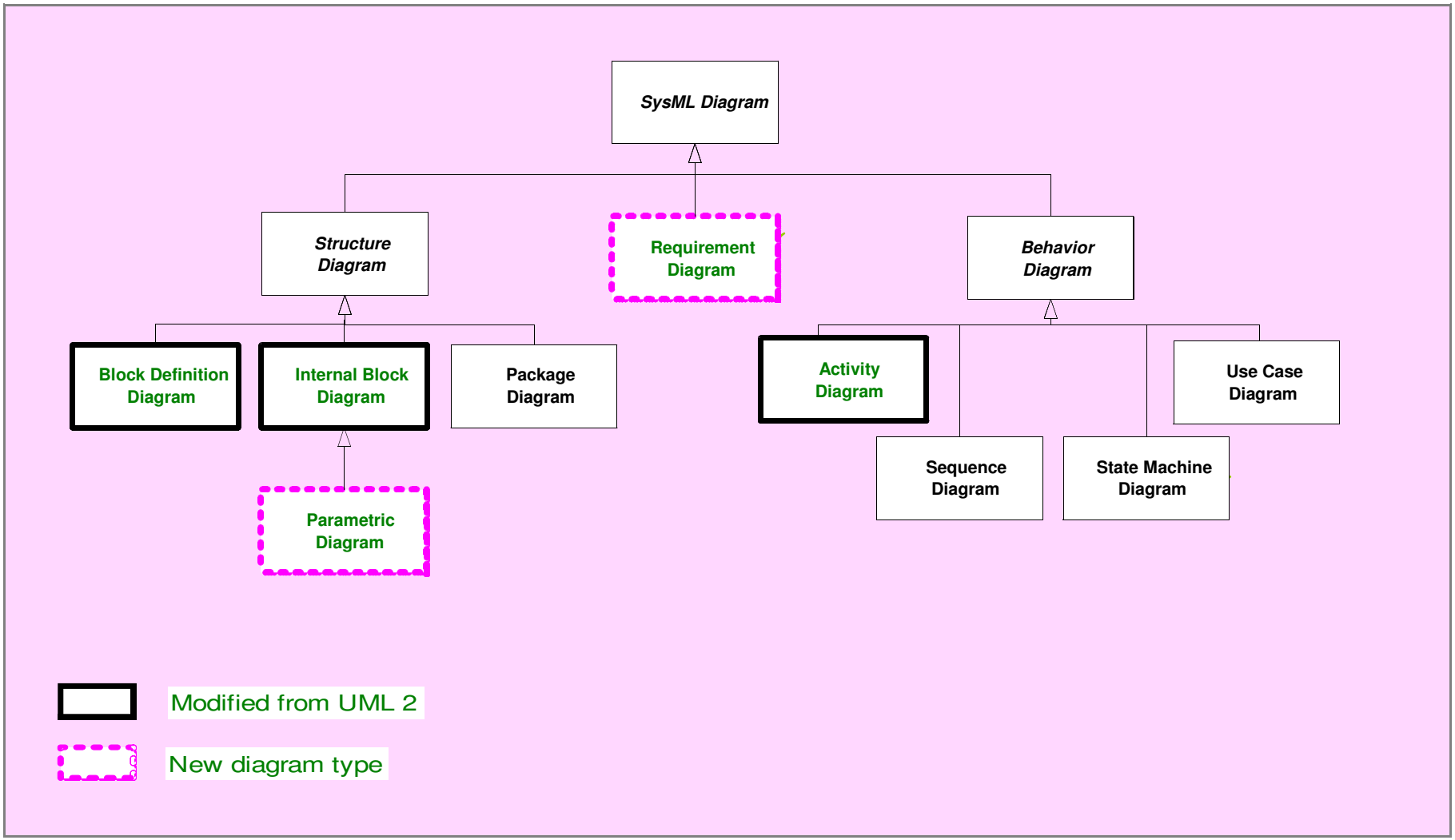


SysML is Critical Enabler for Model Driven SE

Relationship Between SysML and UML



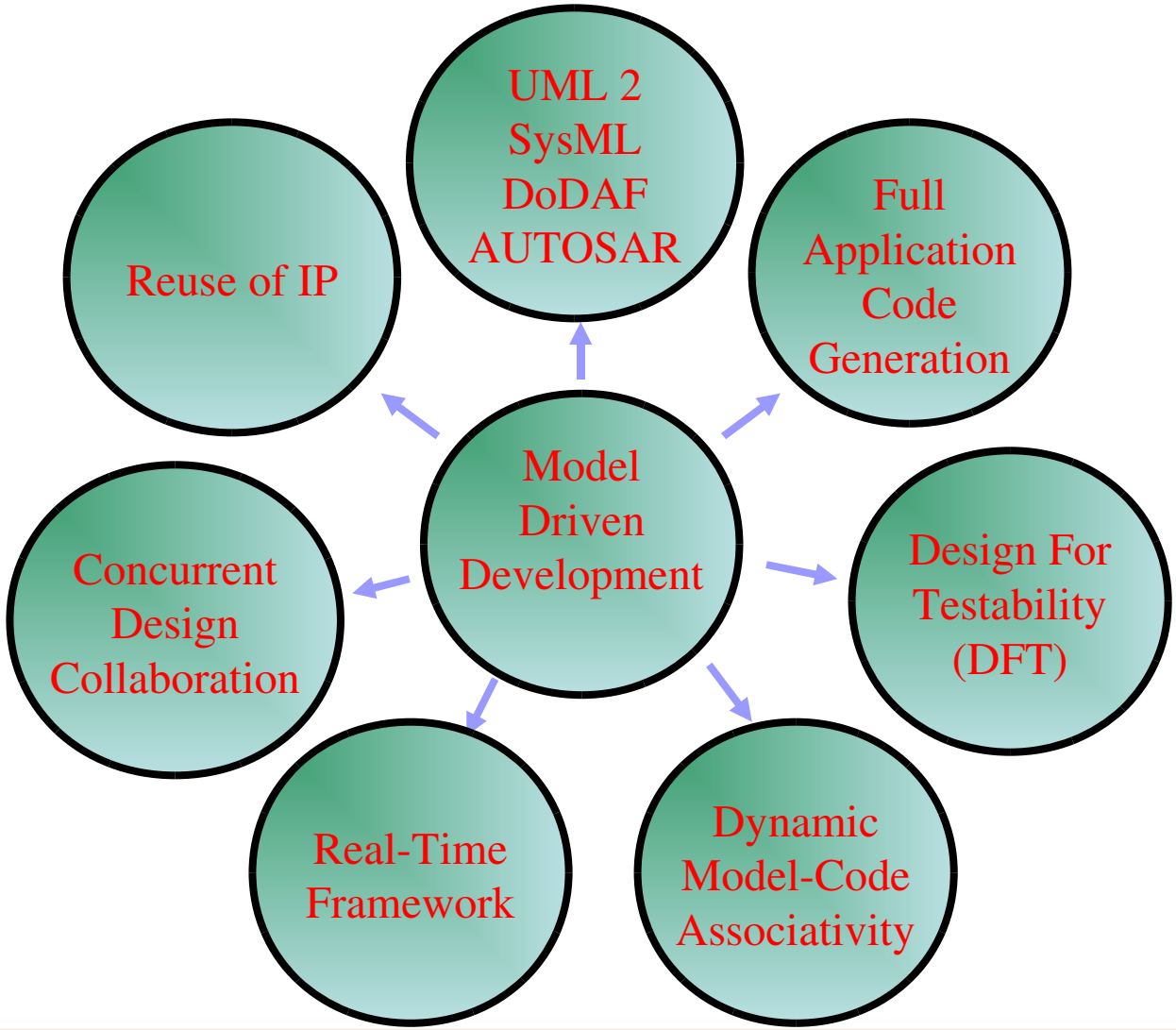
SysML Diagram Taxonomy



Agenda

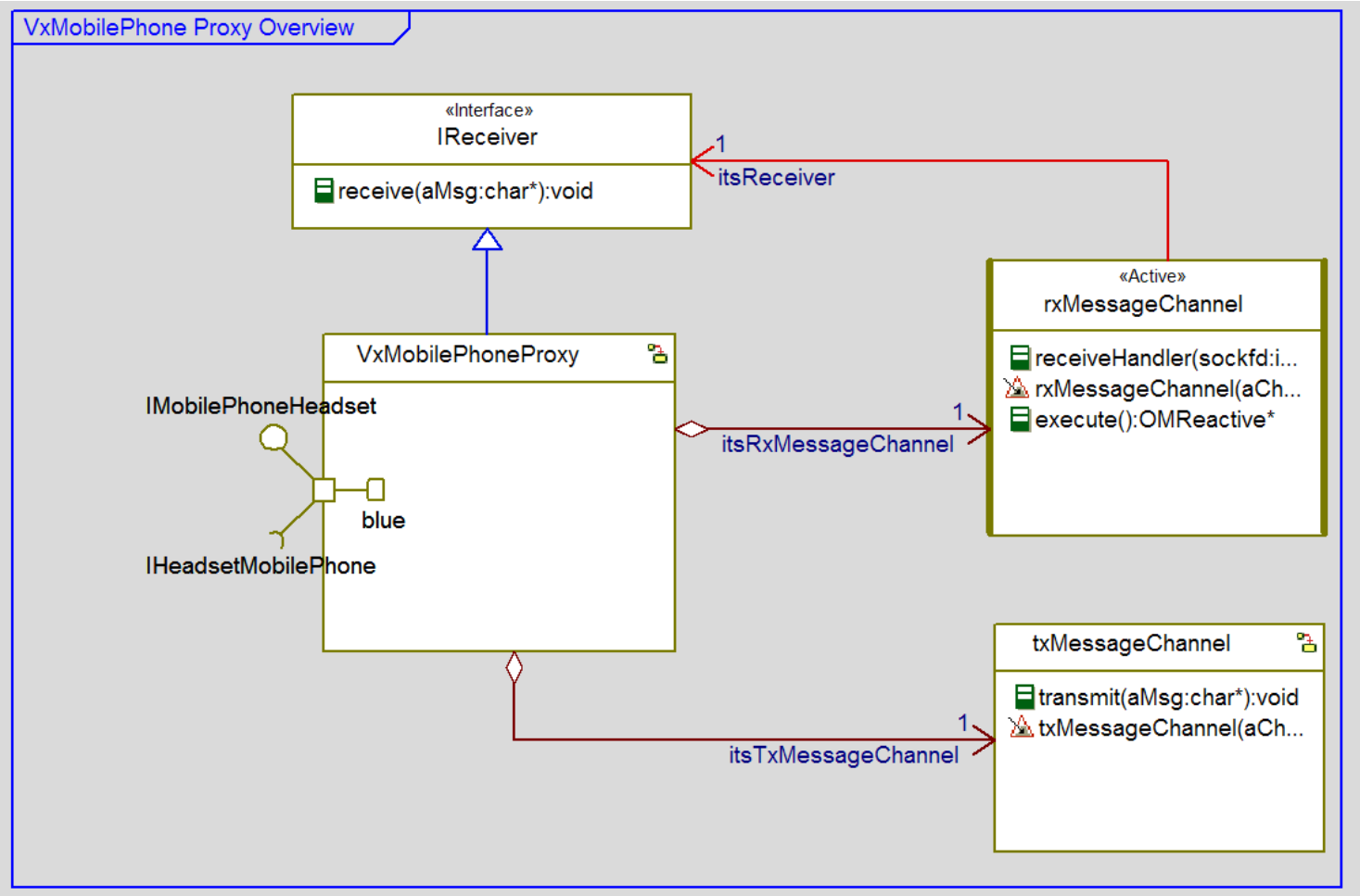
- Model Based System Development
- UML/SysML
- **IBM Rational Rhapsody**

IBM Rational Rhapsody



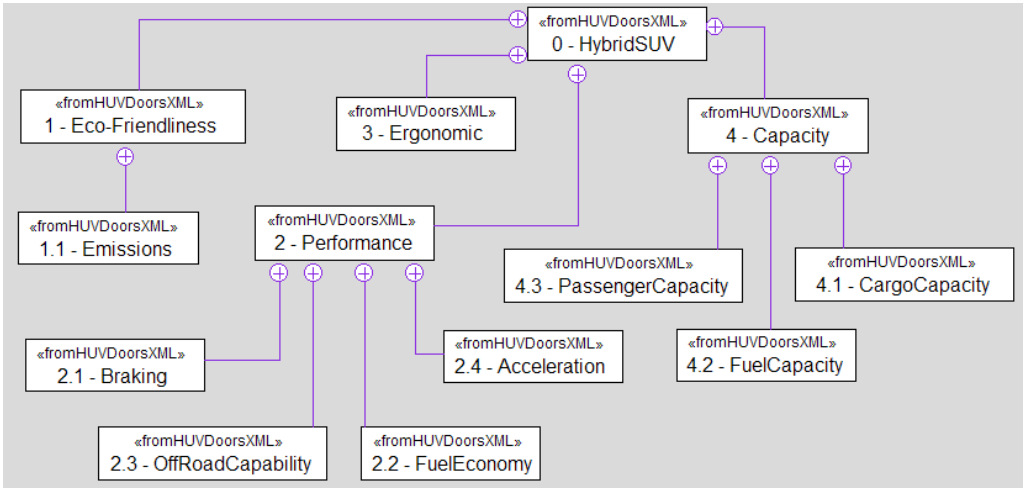
UML 2

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



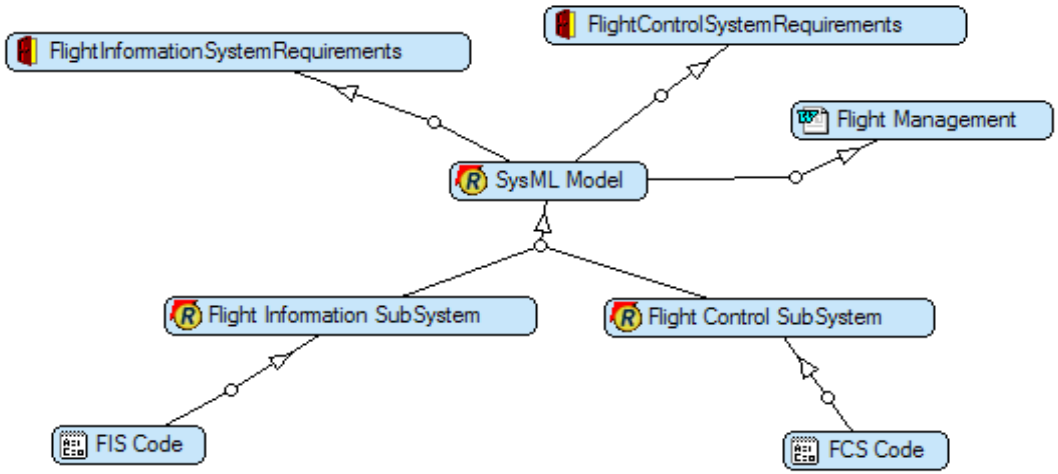
SysML

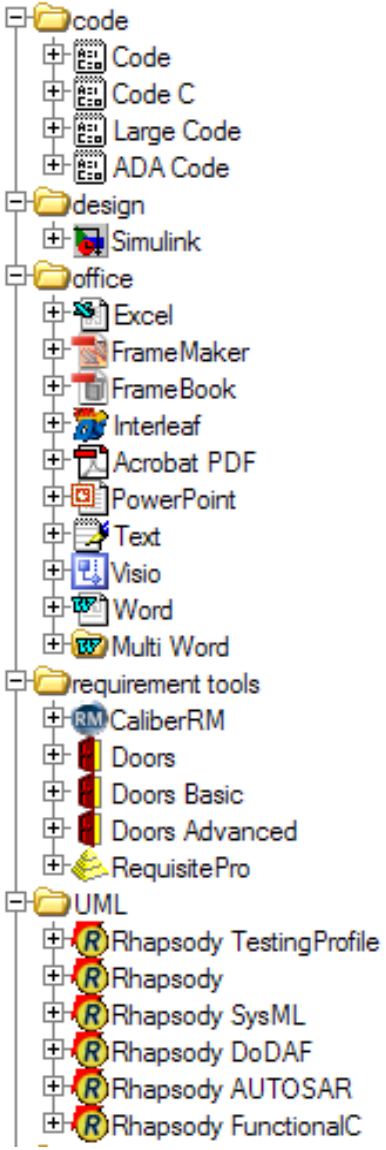
- SysML is a domain customization of UML 2 for systems engineers
  - Supports the standard proposal in its latest form (V1.0)
- Support for SysML views
  - Requirements: Requirements diagram; Use case diagram
  - Structure: Block Definition diagram; Internal Block diagram
  - Behavior: Statechart; Activity 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



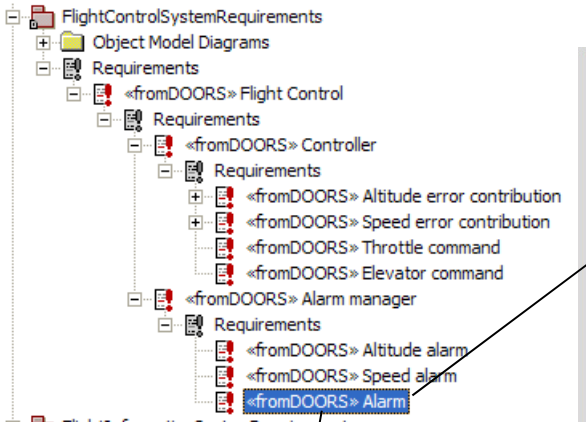
Requirements Modelling

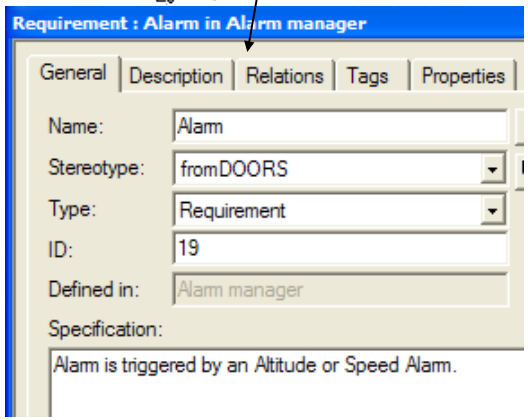
- Requirements Capture
- Requirements Traceability
  - Create traceability links from model to requirements
  - Automatic traceability documentation
- Requirements Analysis
  - Requirement Coverage Analysis
  - Change Impact analysis
  - Automatic report generation

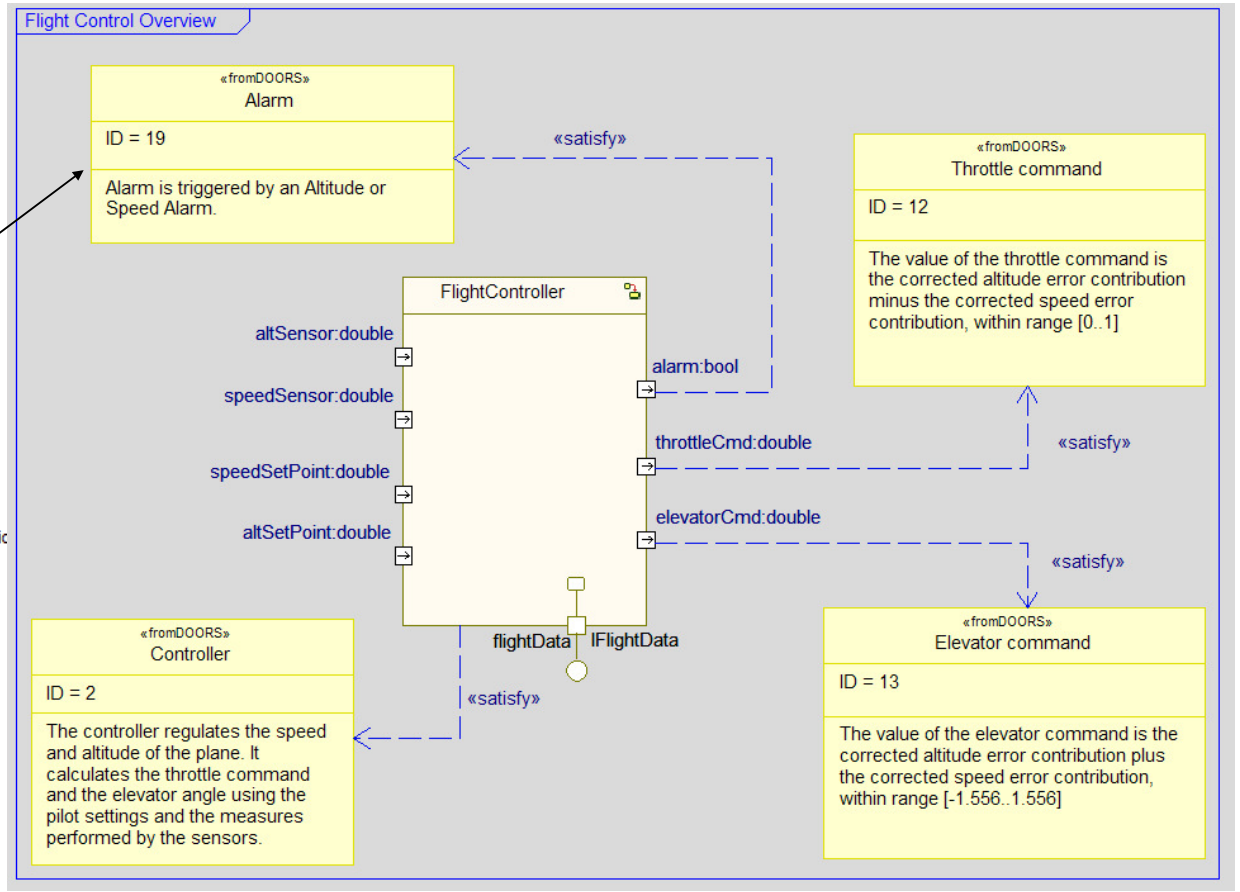




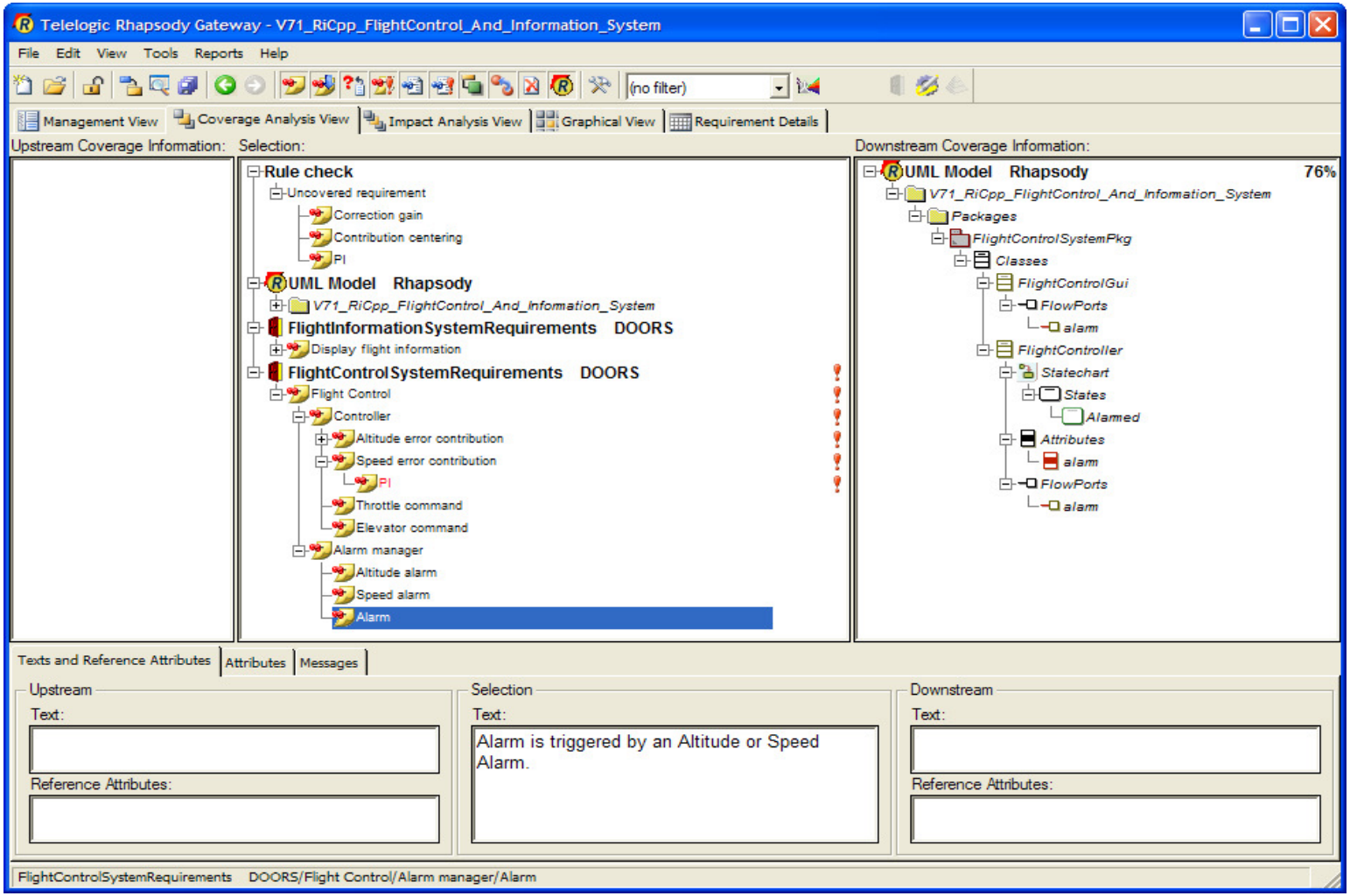
Requirements Capture and Trace



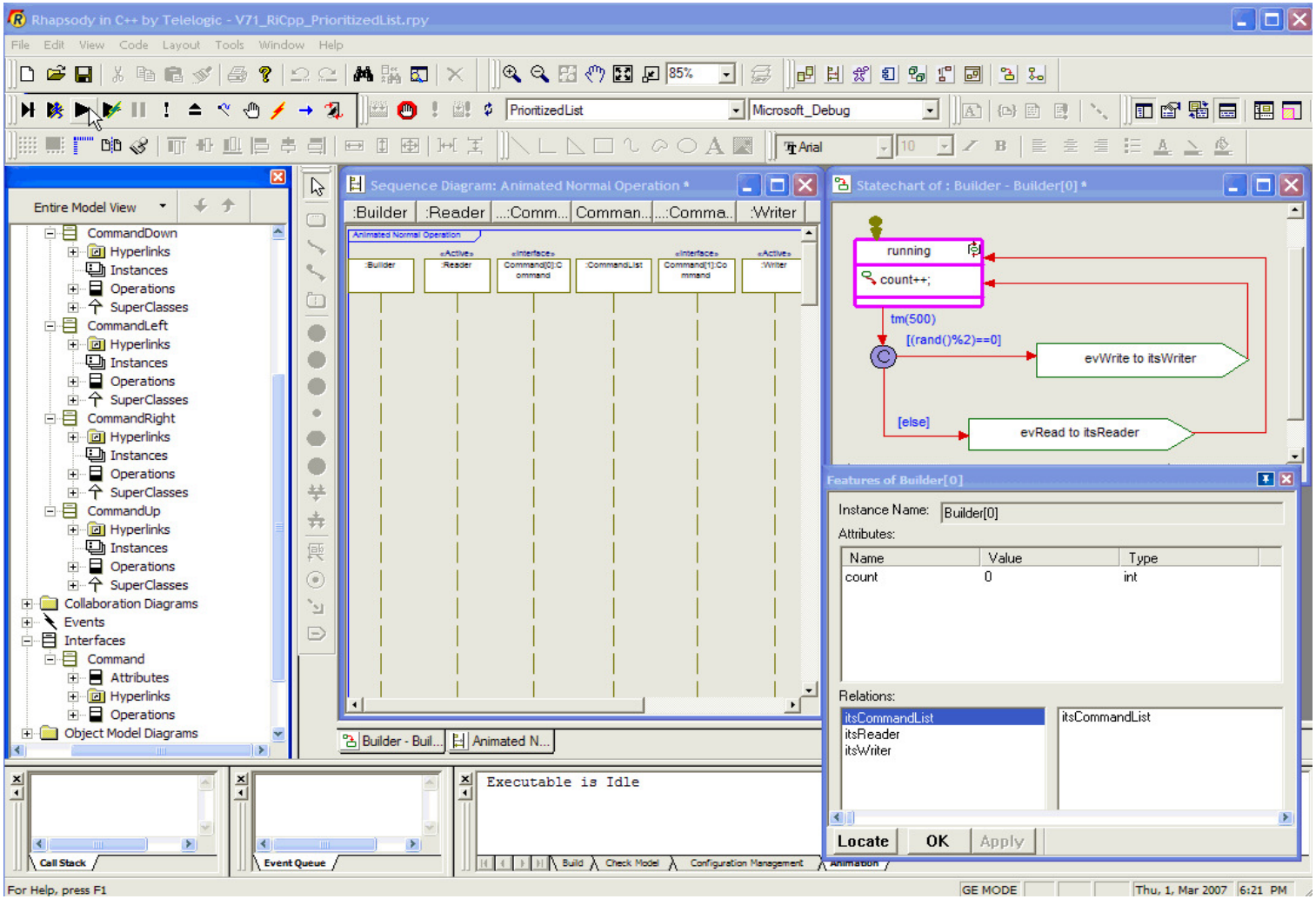




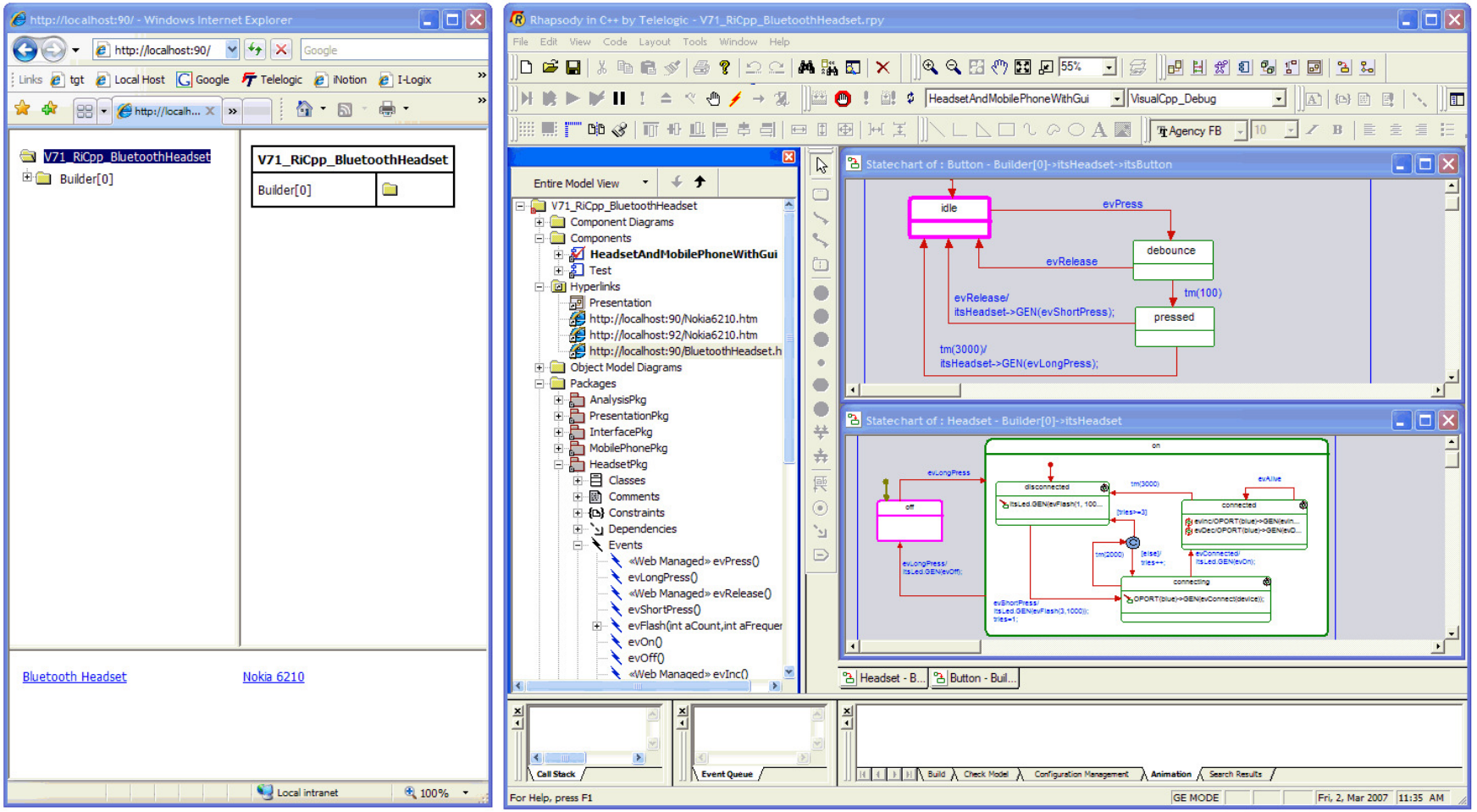
Requirements Coverage Analysis



DFT : Executable Models on Host & Target



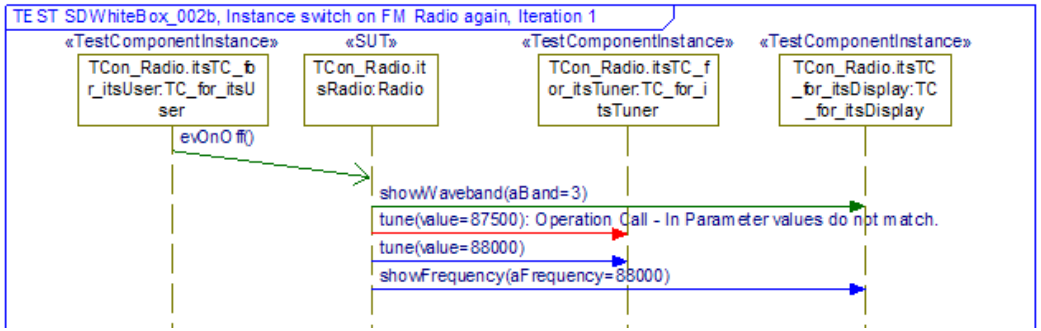
DFT : Rapid HTML Gui's



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

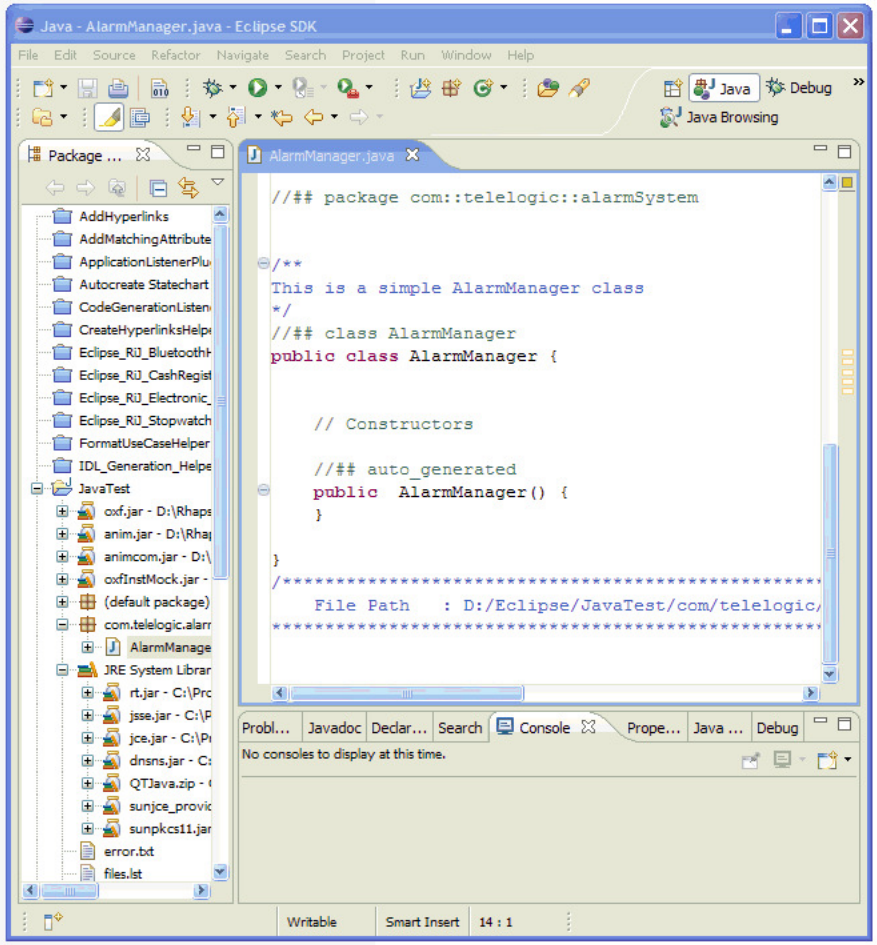


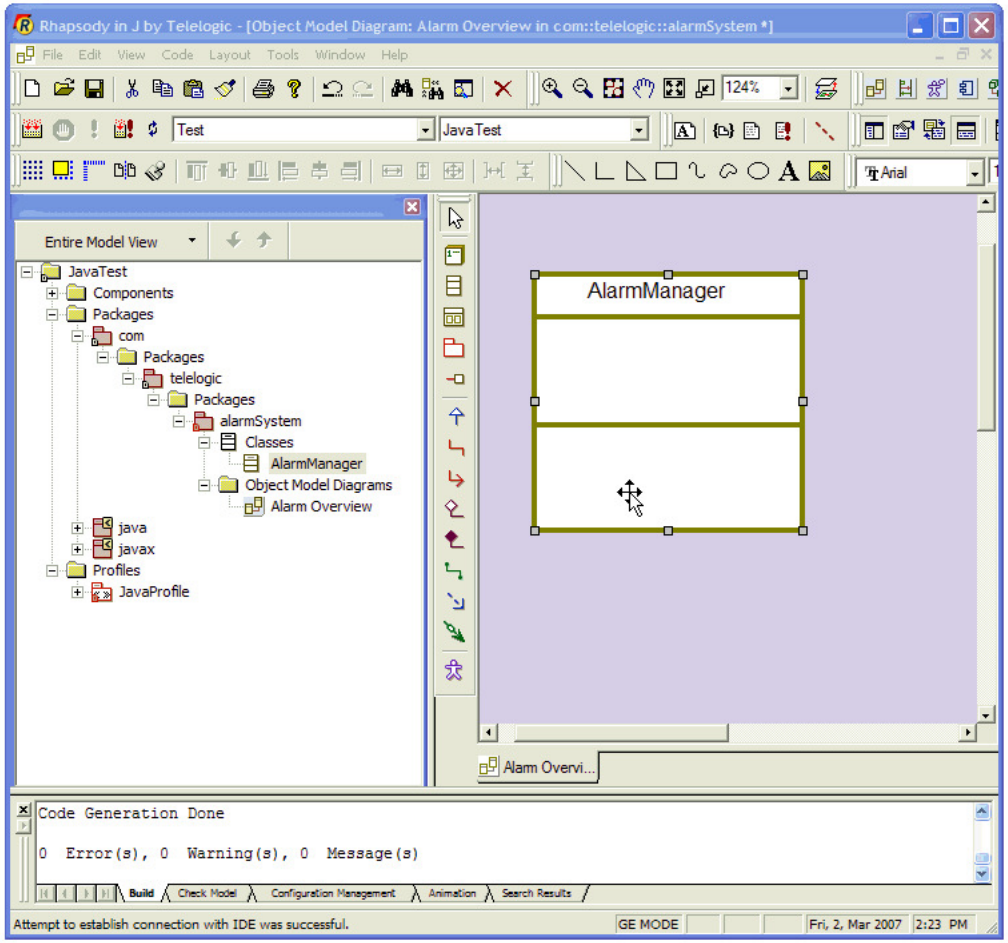
Full Application Code Generation

- 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





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