IBM Rational Software
Development Conference
2008

WHERE TEAMS ARE R-HEROES


# Being Rational about Complex Systems Development

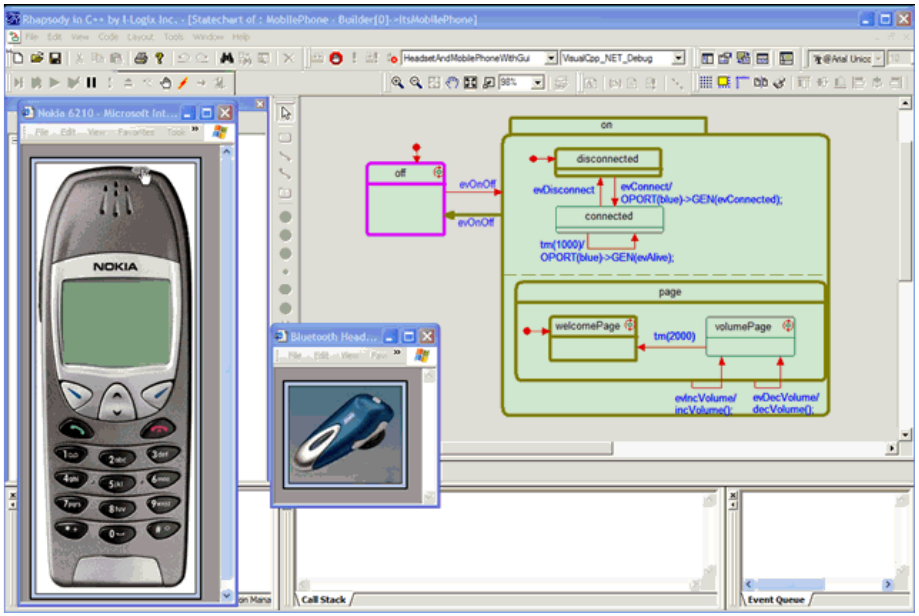
**Francis Choi**
Practice Manager
Modeling
ASEAN & Greater China
francisc@hk1.ibm.com

***SD02***

Rational. software Telelogic An IBM Company

Go to IBM

# Model-Driven Development for Embedded Systems Telelogic Rhapsody



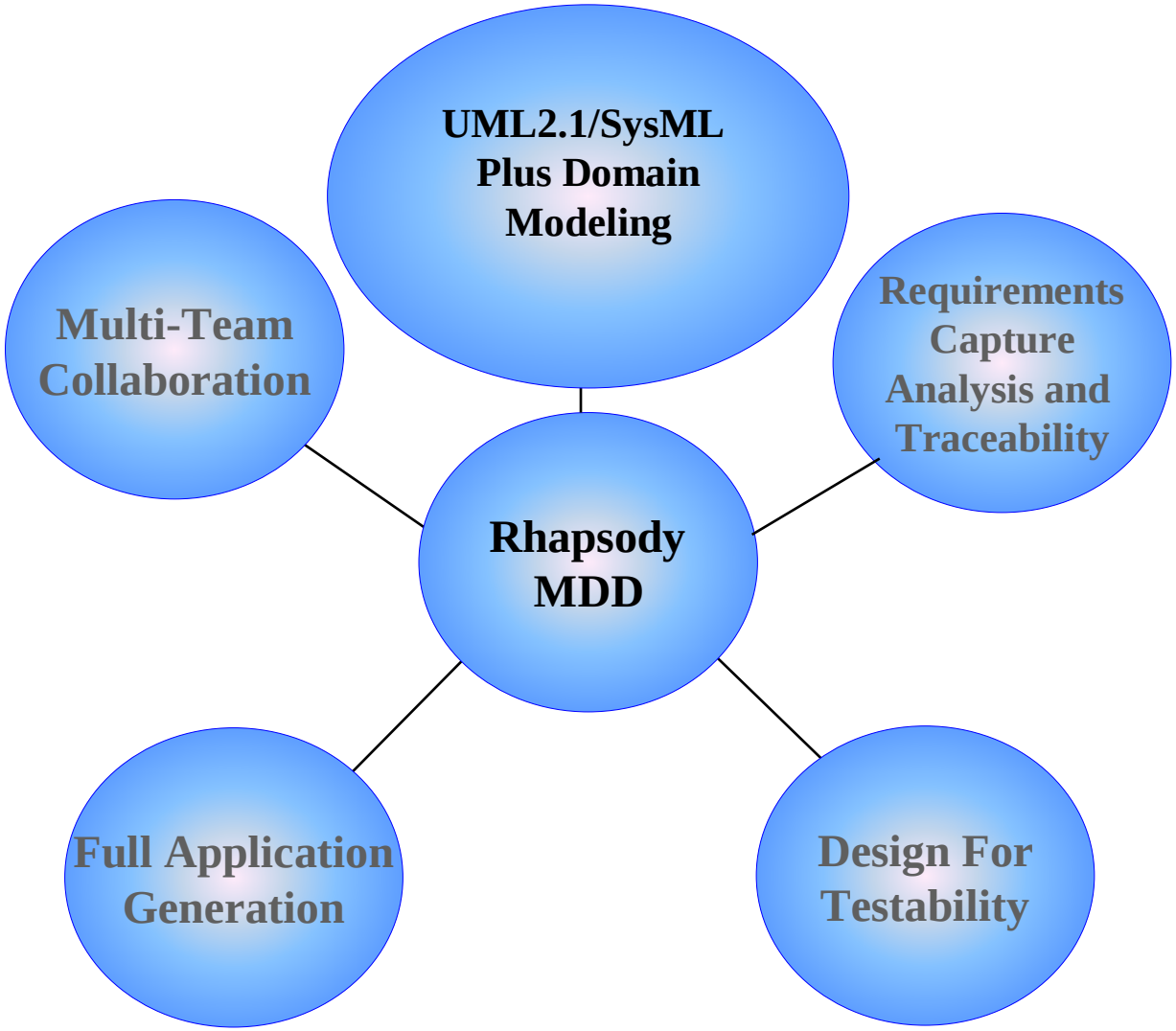

## Capabilities

- Systems design and software development with UML 2.1, SysML, DoDAF and AUTOSAR
- Validate and verify designs with model based simulation and test throughout the process
- Produce complete C, C++, Java and Ada applications; developing in either the code or model while ensuring the two remain in sync

## Benefits

- Optimized communication and collaboration ensures the right product is built
- Eliminate defects early and increase quality by iteratively testing the design as it is built
- Reduce development time by automatically generating  applications and documentation

# Better communication through formal models



# Conceptual Collaboration in Text


## Developer 1: (American)

"Ok. Here's how it works. Thread A will pass event X to thread B and that will change B's state to Running from what it was before which was Init. When B changes to Running it will send back an event Y to A and then wait for 2 second and then go back to Idle. Thread A will have started in Idle also and will go to Run after B sends back event Z which happens after the 2 seconds before going to Idle. All this should happen in less then 5 seconds."

## Developer 2: (Chinese)

"Huh ?" What are you talking about?

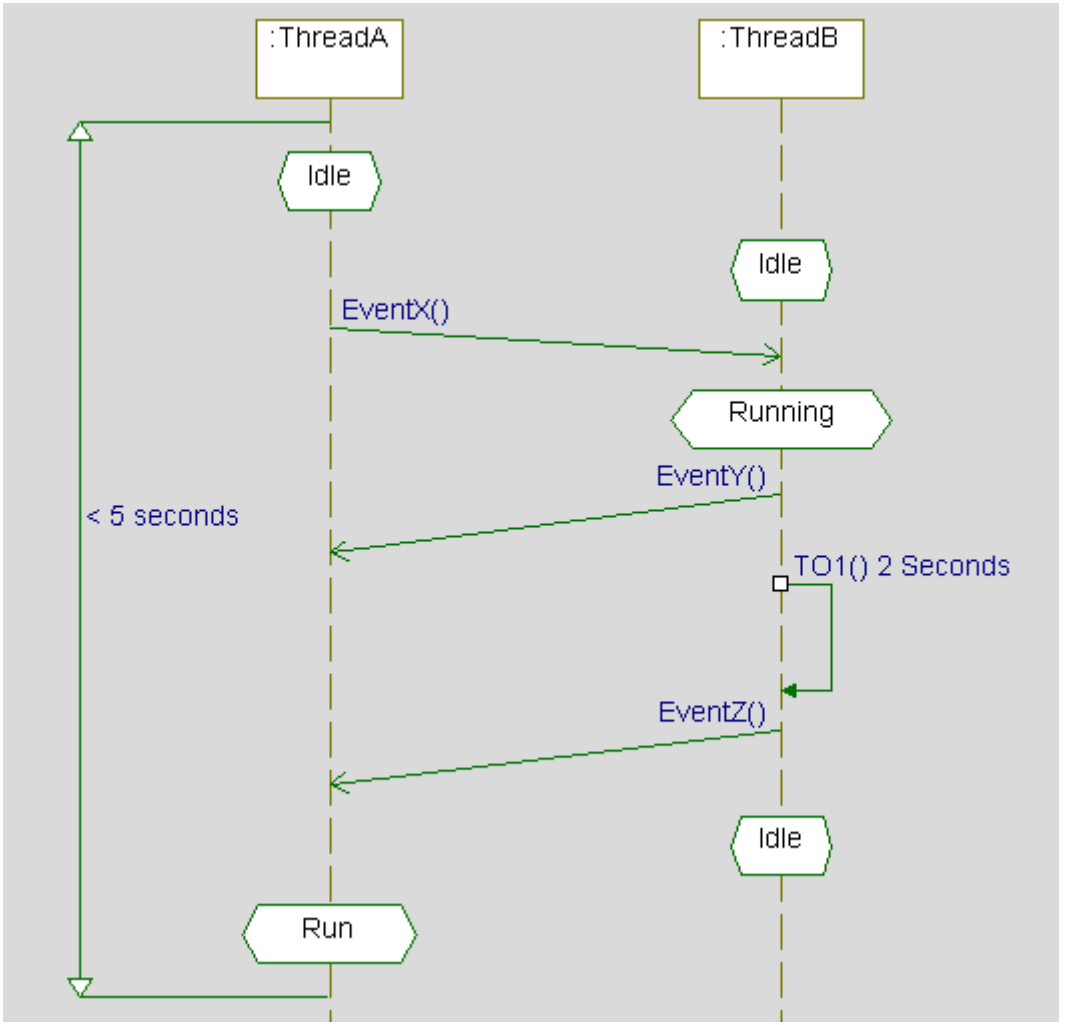
# Conceptual Collaboration in MDD

American

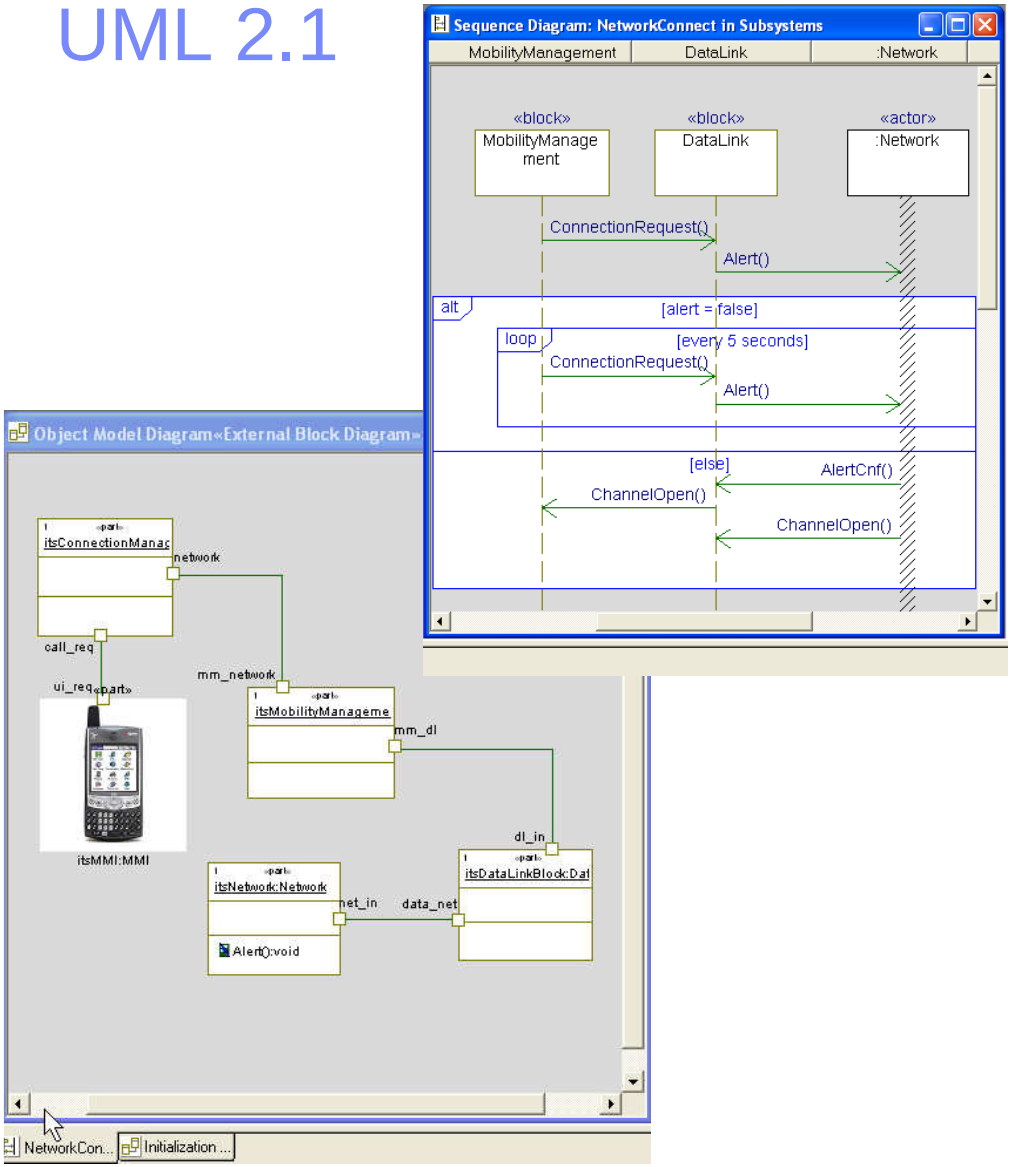
"Here look at this Sequence Diagram."

Chinese:

"Ahhh, now I see!"

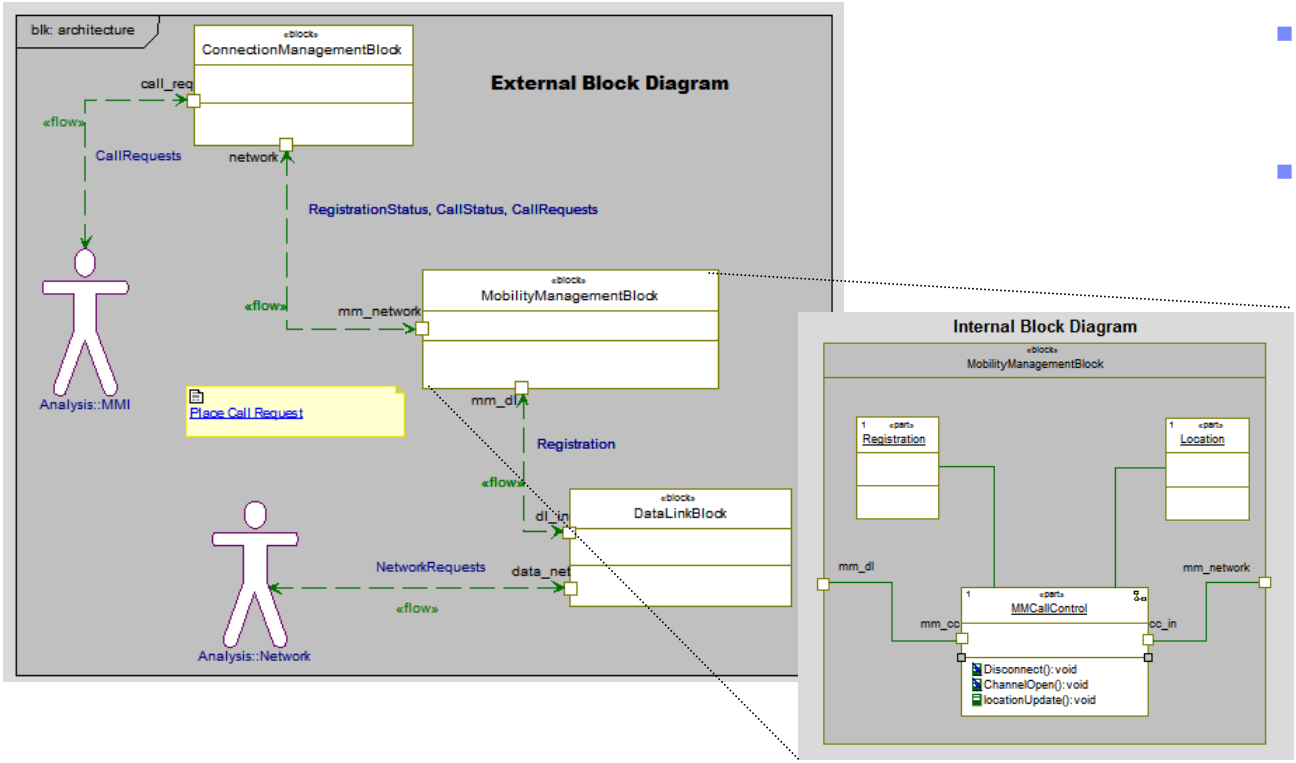
# UML 2.1




## UML Views

- Structure
  - Structure diagram
  - Package diagram
  - Component diagram
  - Object model diagram
    - Class and Object
  - Deployment diagram
- Behavior
  - Statecharts
  - Activity diagrams
  - Use case diagram
- Interaction
  - Sequence diagram

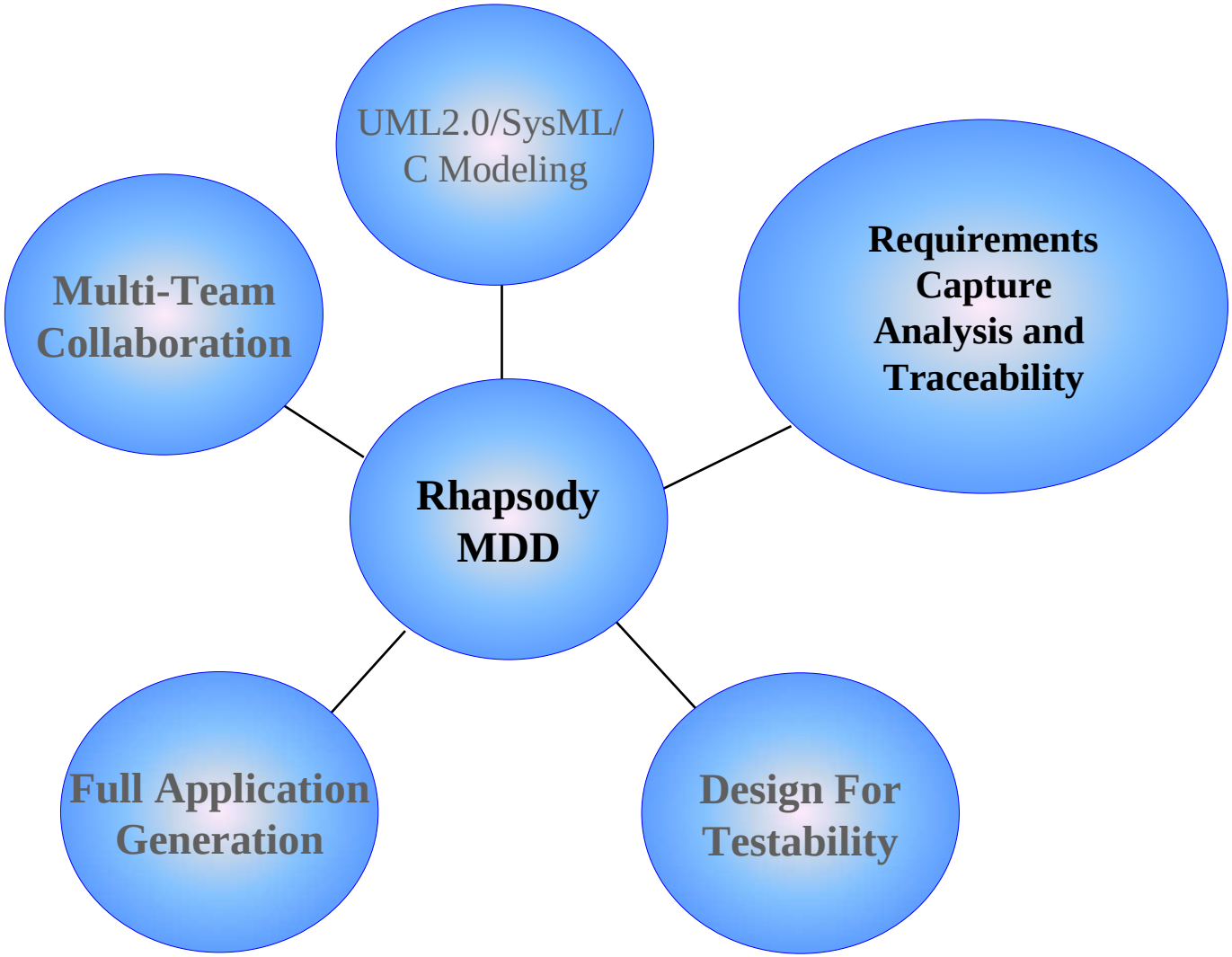
# SysML




- Telelogic has been an leading contributor to SysML right from the very beginning.
  - Member of INCOSE Model Driven Design Workgroup
  - Member of the AP233 Systems Tools Working Group
  - Charter member of SE DSIG
  - Issued a response to the SE DSIG RFI
  - Members of the SysML response to the SE DSIG RFP

- Systems Modeling Language
- SysML views
  - Requirements
    - Requirements diagram
    - Use case diagram
  - Structure
    - External block diagram
    - Internal block diagram
  - Behavior
    - Statechart
    - Activity diagram
    - Sequence diagram
  - Constraints
    - Parametric diagram

# Ensure your design always meets the requirements

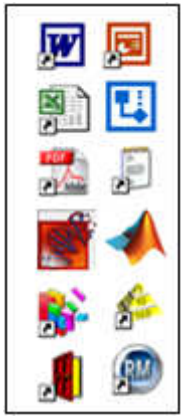


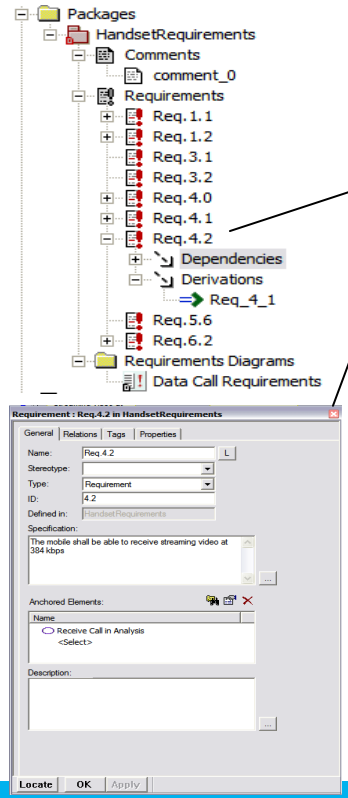
IBM

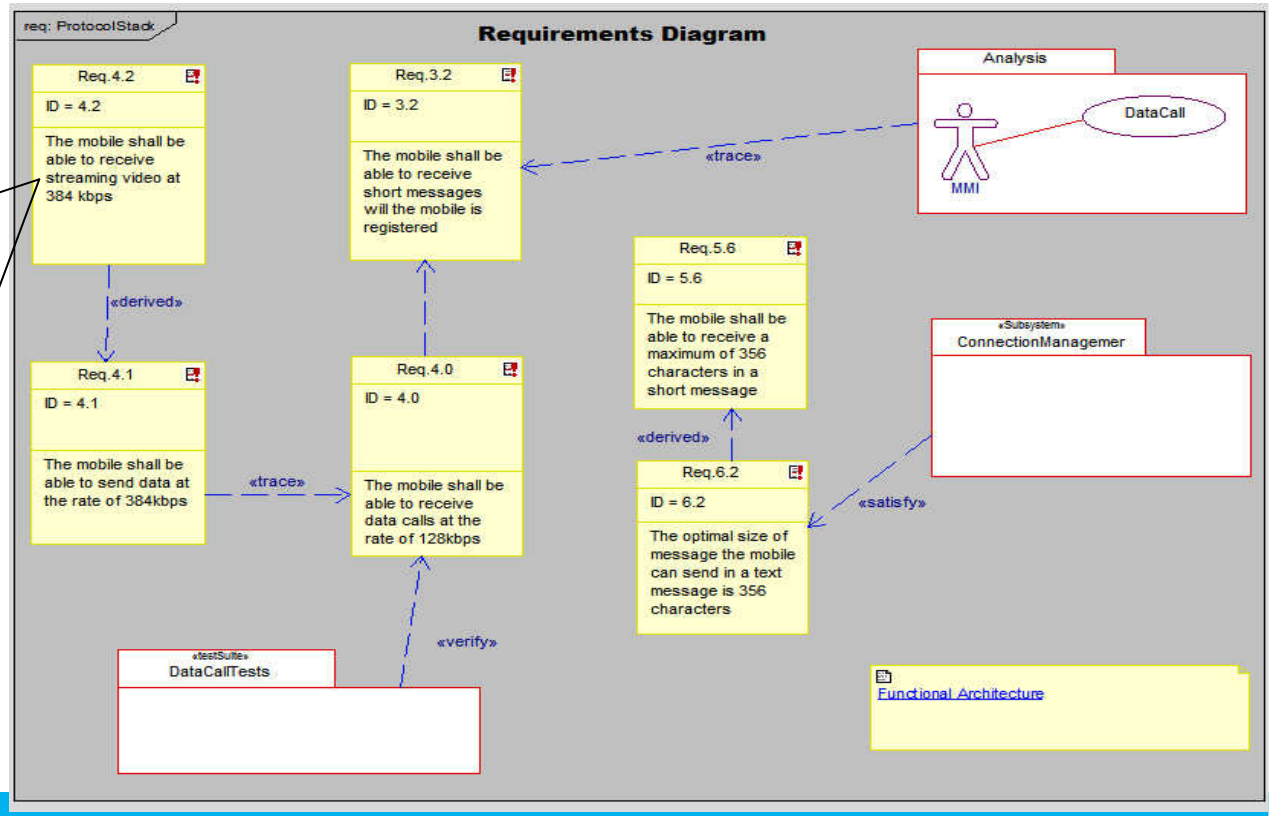
Telelogic
An IBM Company

# Visual requirements capture

- Use Requirements and Use case diagrams to define Requirements.
- Supplement definitions and descriptions with Tags and Constraints
- Describe requirements behavior using Sequence, Activity  and State diagrams
- Include advanced graphics and icons with domain specific modeling

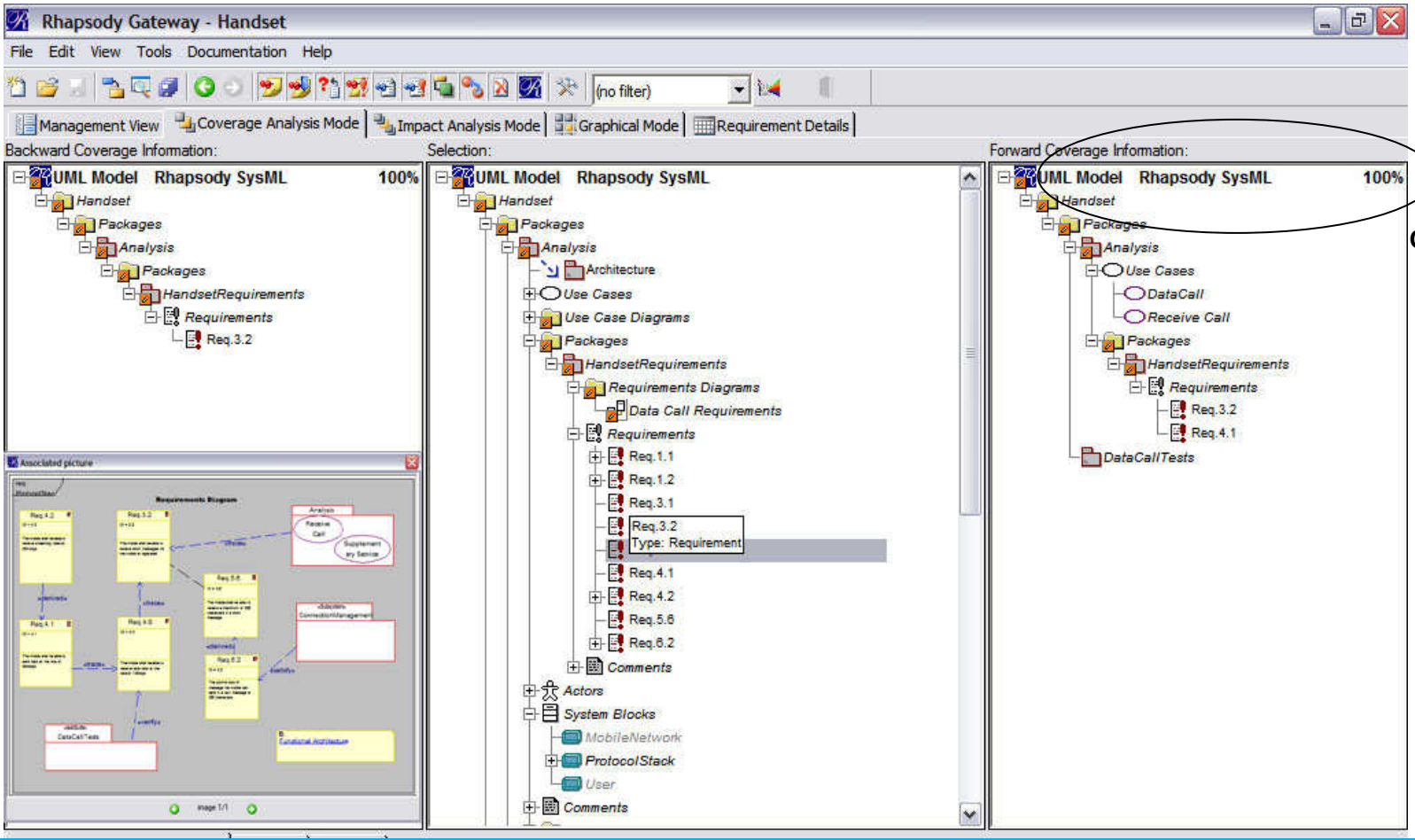




# Requirements coverage analysis

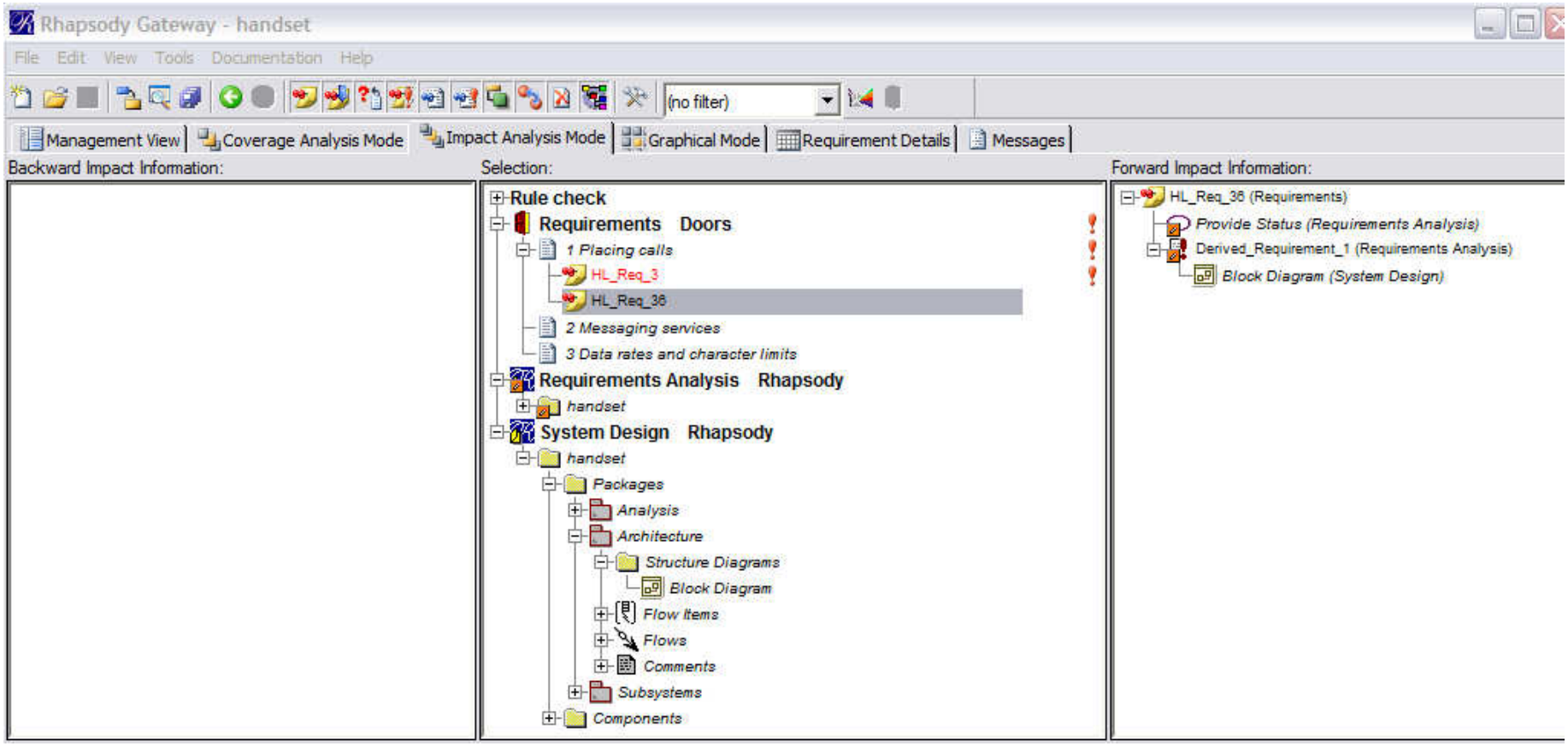
- Identify requirements that have not been addressed in the Rhapsody design
- Find Rhapsody design elements not justified by a requirement



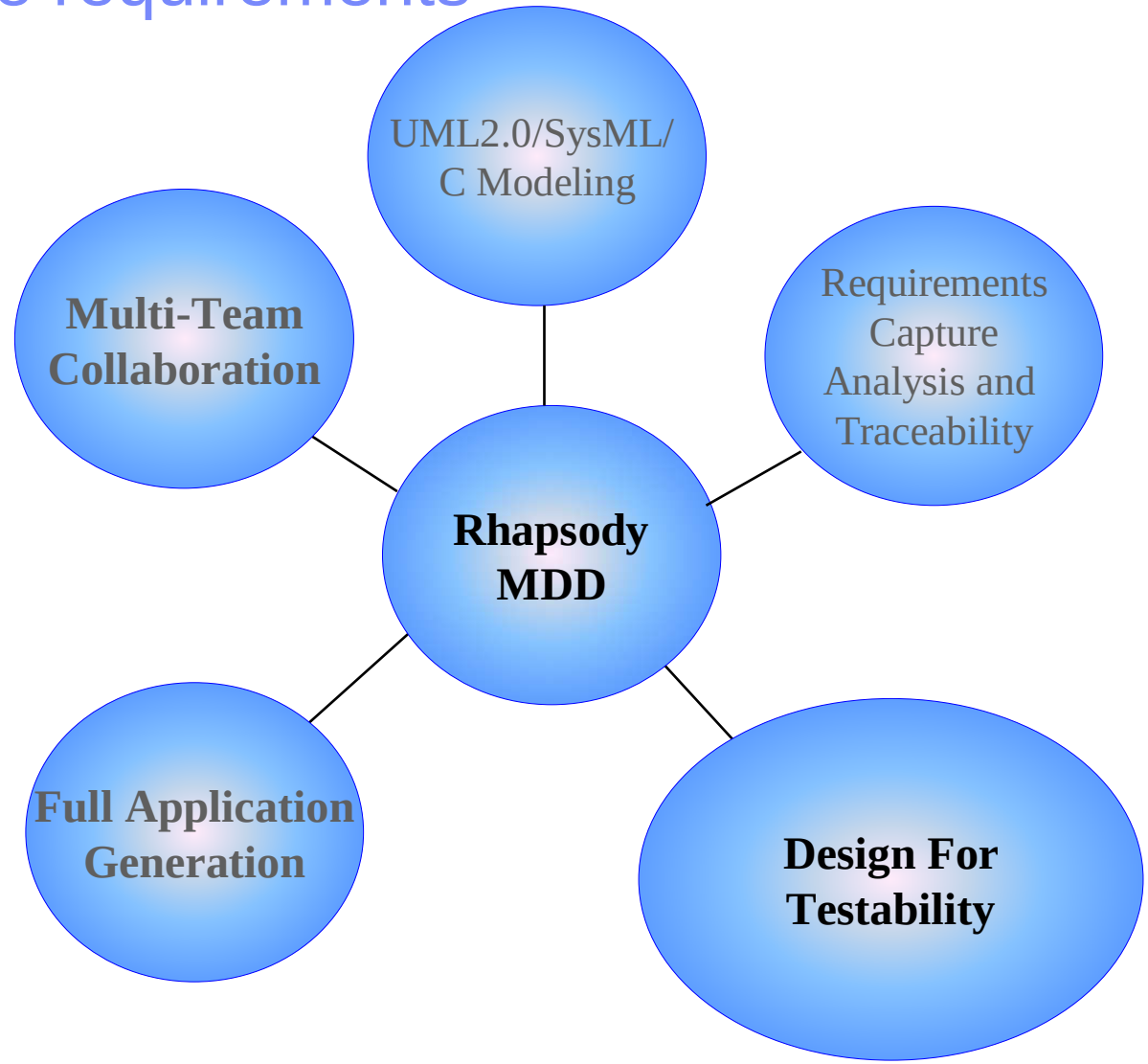

**Coverage**

# Change impact analysis

- Locate elements potentially impacted by a requirement change
- Determine requirements possibly impacted by a design change

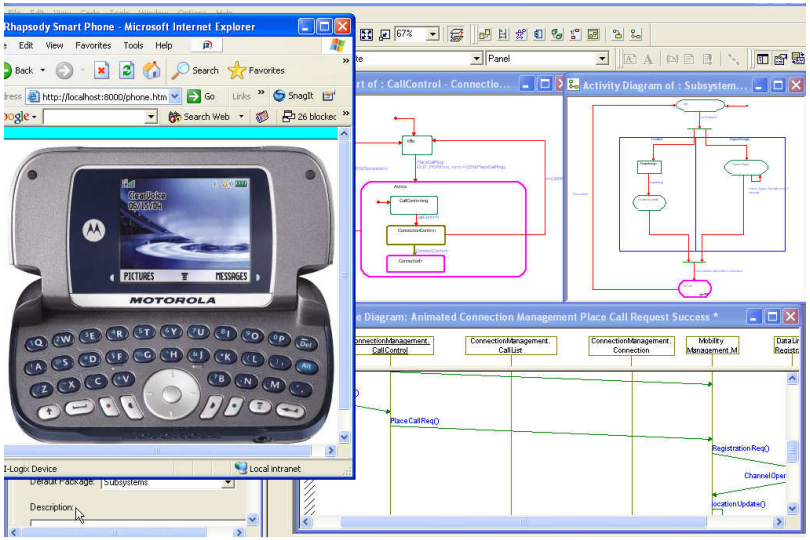


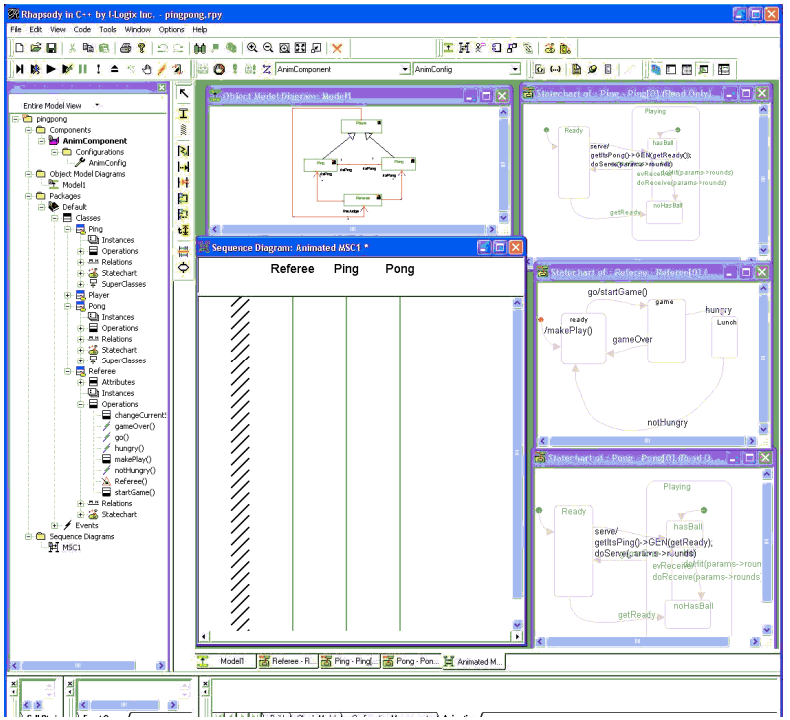
# Eliminate defects early and always validate against the requirements



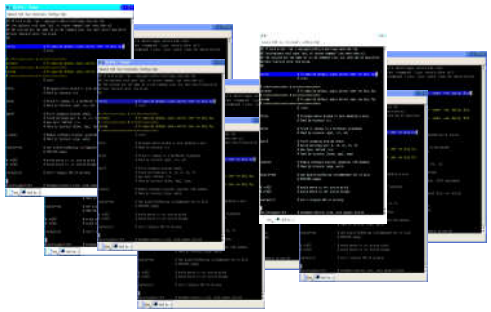
# Simulation, Execution & Animation

- Simulate to verify that model is correct
  - The best way to avoid errors & therefore reduces development cost
  - Rapid Simulation at the design level or even target level debugging
- Virtual prototype / Panel graphics support
  - Ideal communications aid for design reviews and to share information.



# Traditional Code Centric Testing Process



*Coding TestBenches Manually...* → *Coding TestCases Manually...* → *Code on Host...* → *Code on Target...*

***Error Prone;***

***No fun!***

***Difficult!***

***Poor coverage is the norm…***

***Puzzling;***

***Time consuming!***

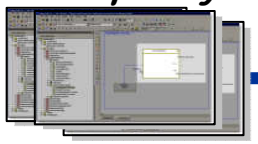
***Frustrating;***

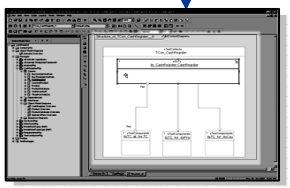
***Expensive!***

# Model Driven Testing Process

*Coding TestBenches Manually...* → *Coding TestCases Manually...* → *Code on Host...* → *Code on Target...*

***Rhapsody Design***

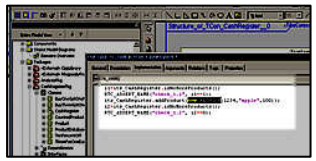




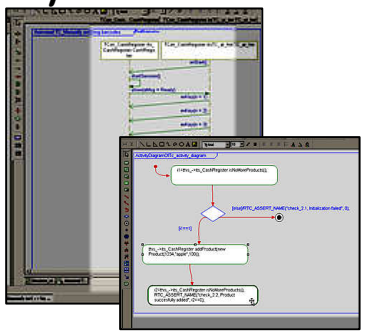
***AutoGen. Graphical TestBench***

***UML Testing Profile***

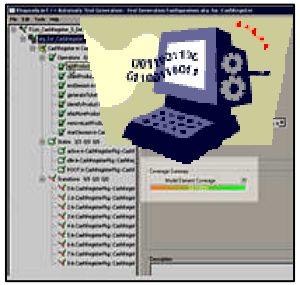
***Code TestCases***



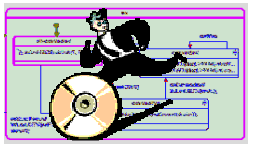
***Graphical TestCases***



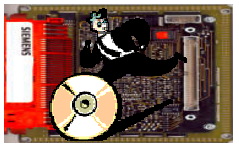
***AutoGen. TestCases***



***Host Based***

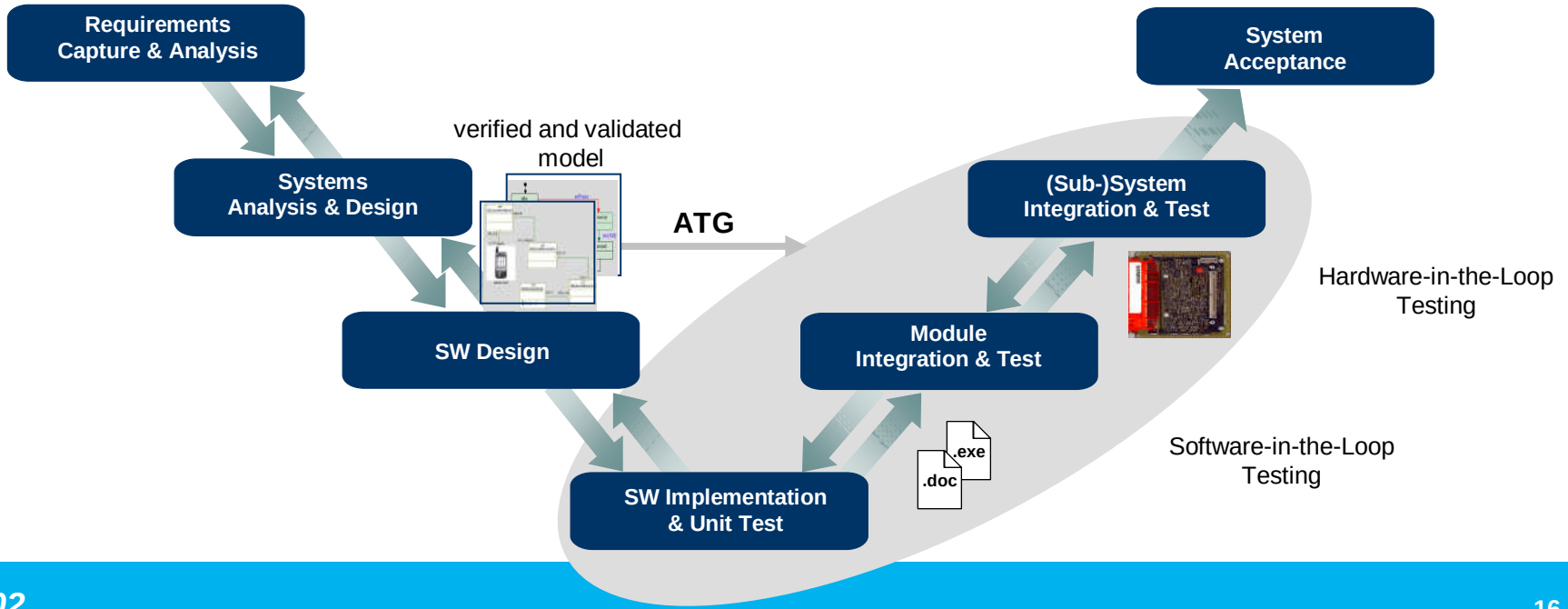


***Target Based***

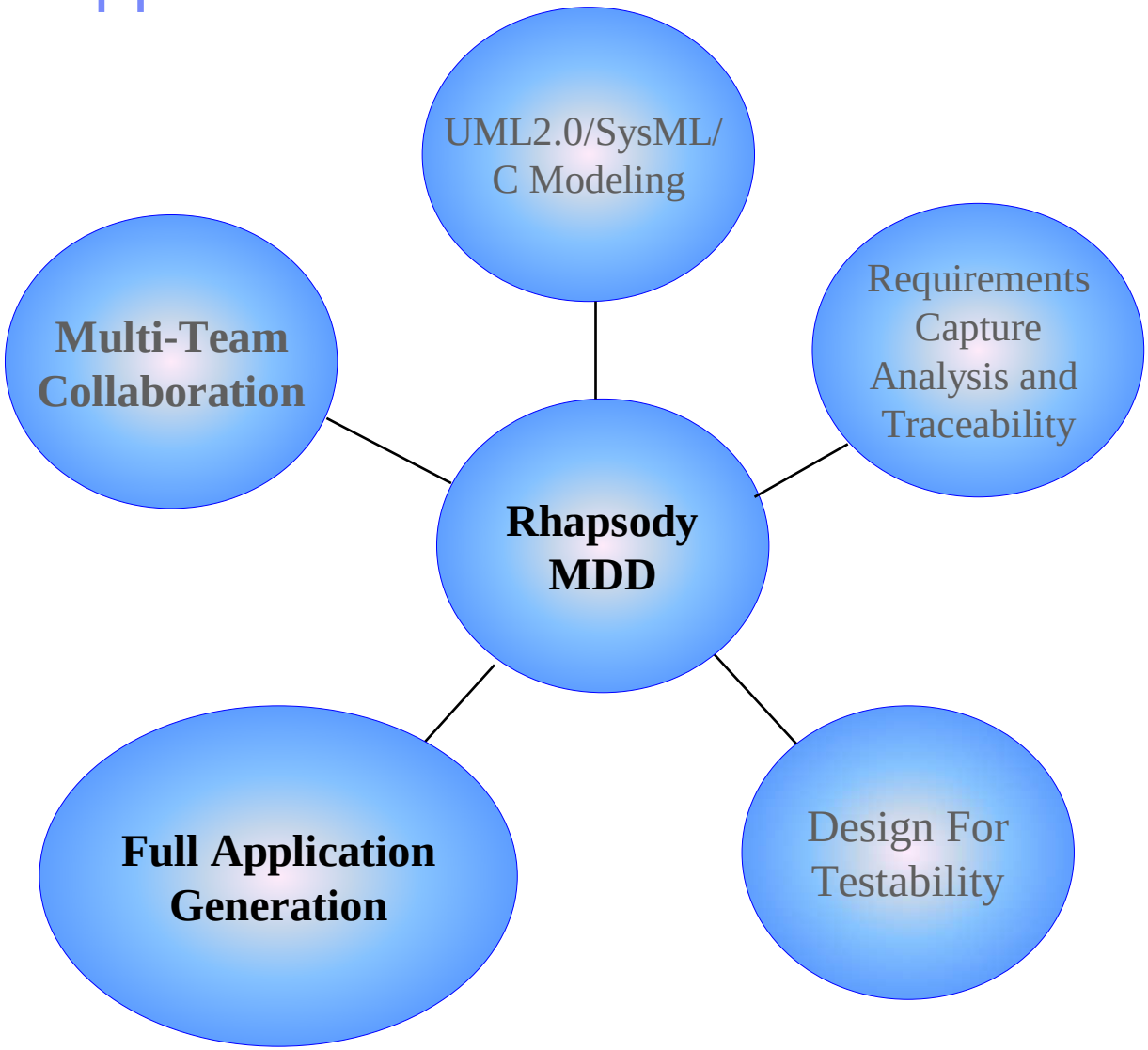


# Automatic test generation

- Automatic Test Generation (ATG) provides model driven test generation
  - Generates test cases with high coverage of the model
  - Consistent with the UML testing profile
- Automatically generates tests from the model
  - Coverage of states, transitions, operations, events
  - Produce all relevant combinations of inputs for MC/DC analysis
- Perform regression testing on the model
- Export to 3rd part tools for testing the implementation



# Meet time to market pressures producing complete applications faster

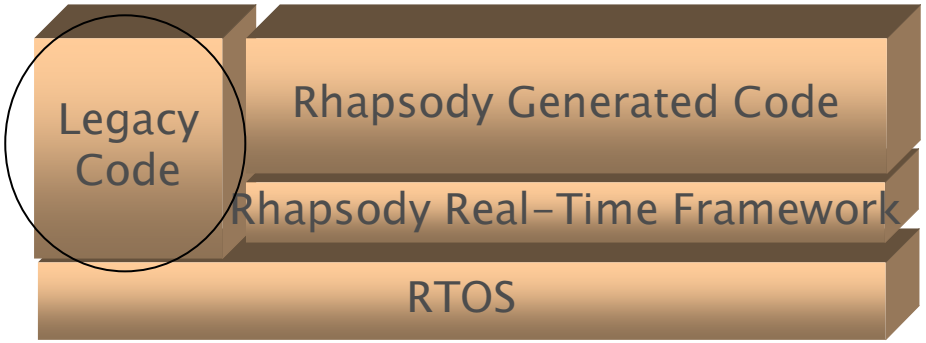


# Full application generation

- Rhapsody leverages all structural and behavioral model views to produce an complete executable applications, not frames!
  - Rhapsody generates very clean, readable C, C++, Java and Ada code, easily debugged through any commercial IDE
  - Rhapsody generates all application construction artifacts to provide an integrated build environment
  - Complete customizable code for all behavioral and structural components in your design using rules based technology
  - Generates certifiable code for DO-178B and MISRA C
- Application Generation is not just about code generation.  Its about
  - Seamless reuse of existing code using
    - Reverse engineering
    - Code visualization
  - Working the way you want provided by unique Dynamic Model Code Associativity (DMCA)
    - Code centric development allows the coder to code while producing models
    - Model centric development allows the modeler to model while producing code
    - Combine both at the same time to enable model/code developement
  - Rapidly deploy your application to any RTOS or systems with no RTOS enabled by the unique Real time framework
    - Rhapsody generates all application construction artifacts to provide an integrated build environment

# Seamless reuse of code (IP)

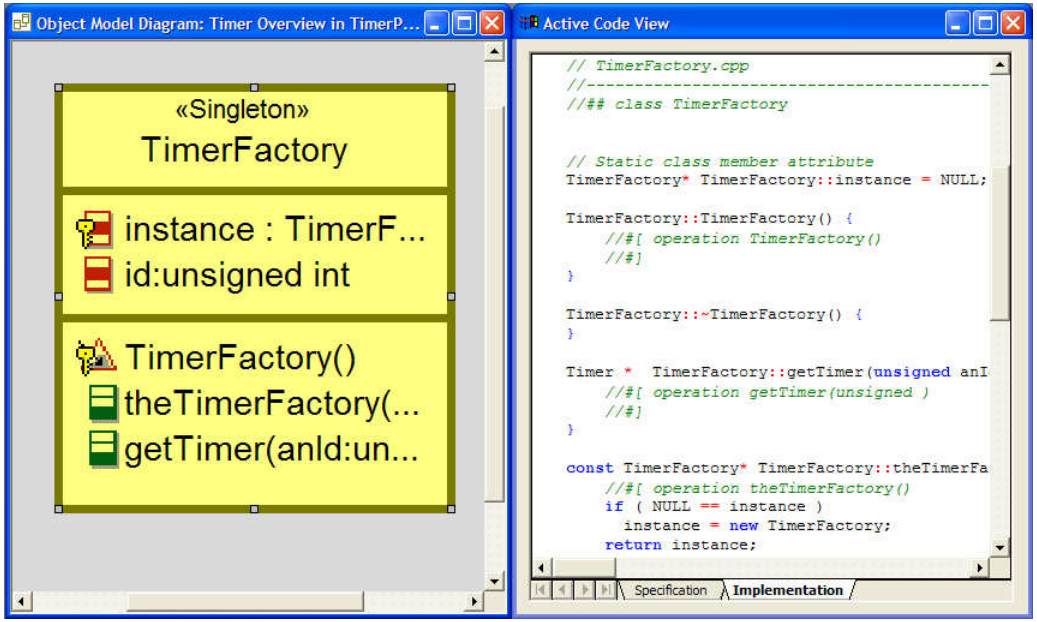
- Rhapsody can incorporate external code to
  - Reuse code from other projects
  - Integrate code developed by a third party
  - Import code generated from another tool
- Such code can be:
  - Viewed externally (code visualization)
    - Provides easy referencing from the model
    - Automatically allows graphical visualization of your IP
    - Seamless workflow between model and hand-written code
  - Automatically reverse engineered to become part of the model



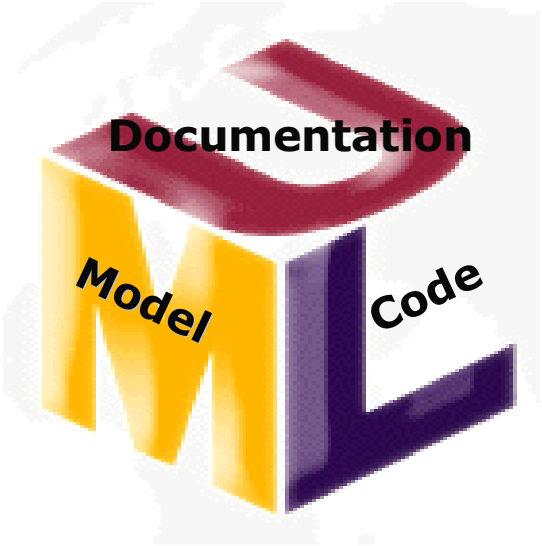
# Flexible developer paradigm

- Reduce learning curve and increase effectiveness by working at code or design level
- Design and Code are always kept synchronized: Dynamic Model Code Associativity (DMCA)
  - Change one view, the others change automatically
  - Critical for real-time embedded software development

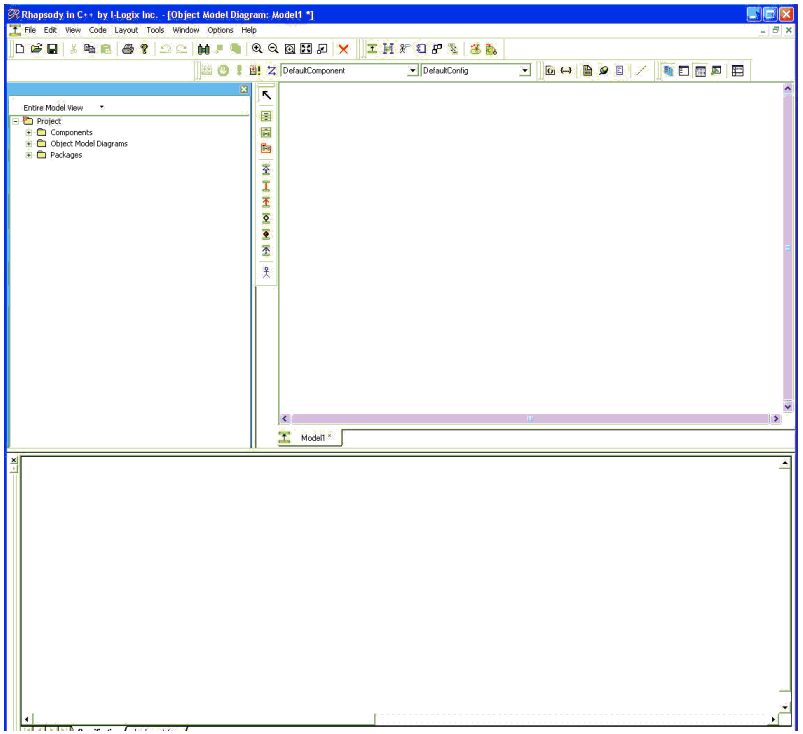
***Rhapsody works the way you do***



# Rhapsody works the way you do

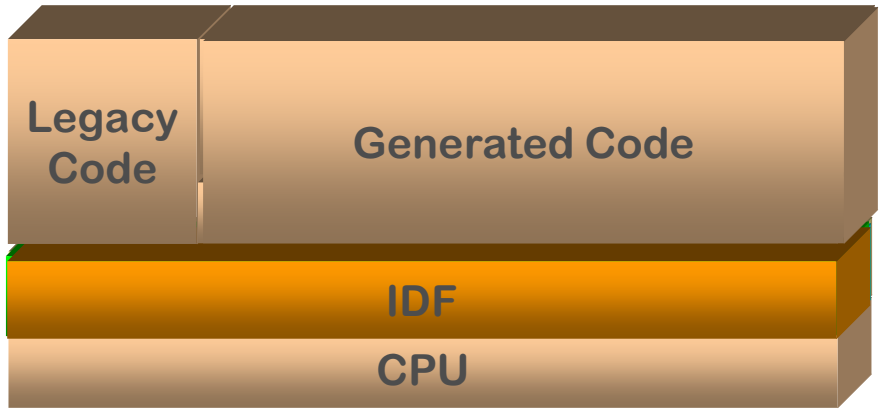



- Dynamic Model Code Associativity (DMCA)

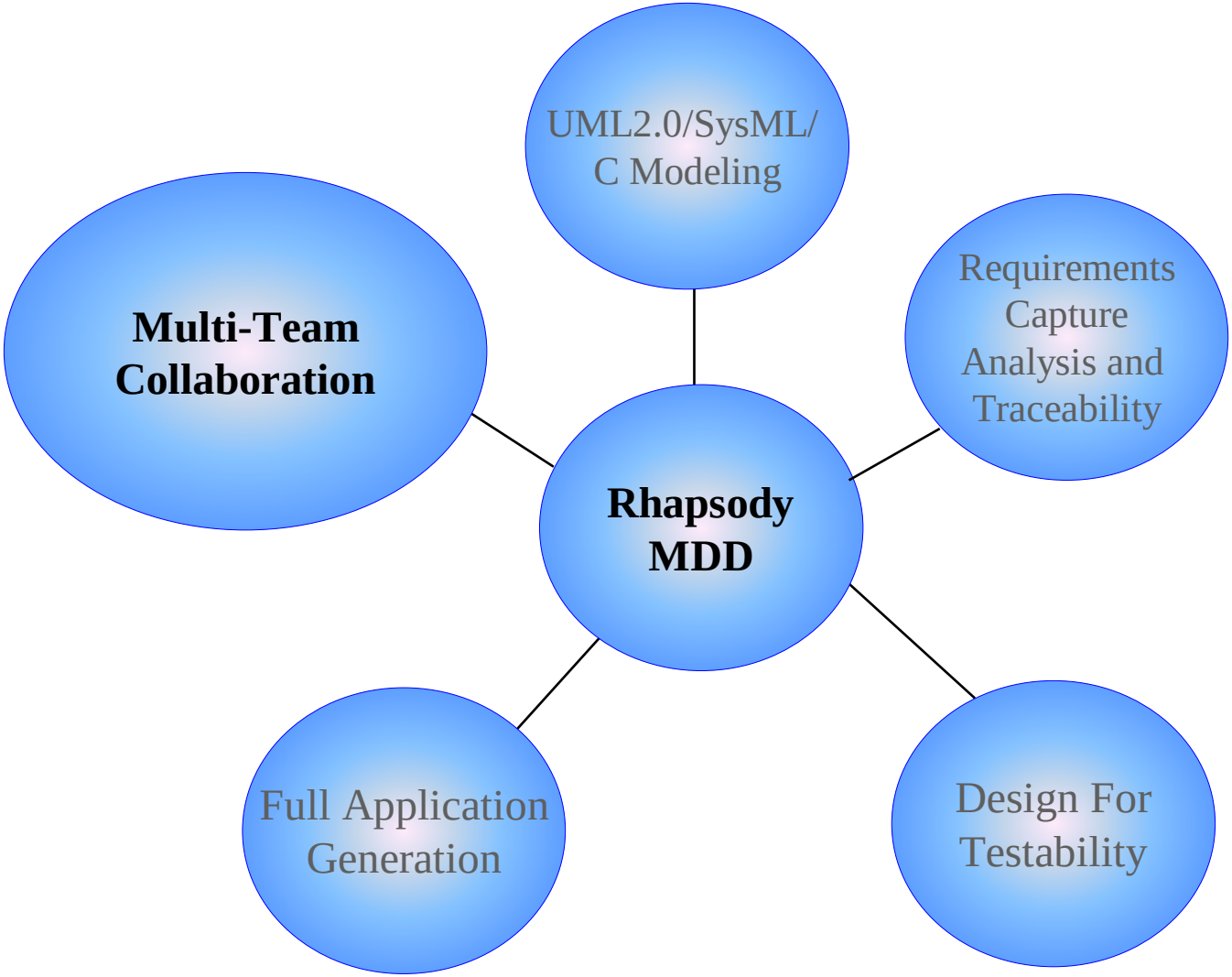


# Rapidly deploy your design onto any target

- Rhapsody Real-Time Framework
  - Provides implementation for real-time semantics
  - Abstracts away the platform APIs (RTOS)
  - Allows rapid retargeting to any RTOS or to systems with no RTOS at all
- Key technology in supporting a Model Driven Architecture (MDA)
  - Provides Platform Independent Models (PIM)
  - Simply select the proper adaptor to create a Platform Specific Model (PSM)



# Enables all size teams to effectively communicate and work together

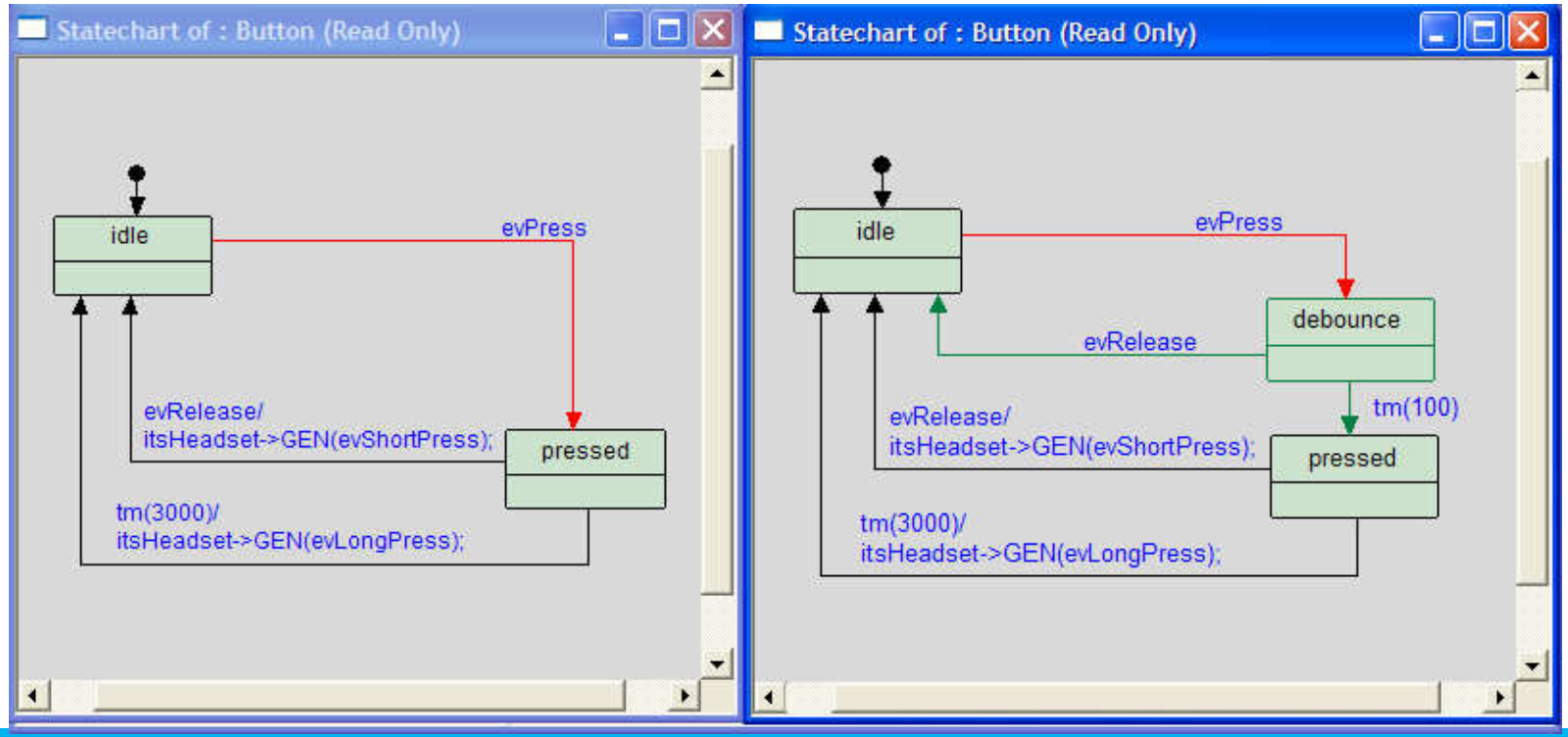


# Multi - team collaboration

- Rhapsody promotes concurrent collaborative large scale engineering
- Create, review, share and modify Rhapsody models within a single project, company or globally
  - Interfaces with all the popular CM tools to ensure project data remains synchronized and under configuration control
    - Check in/out model information to the CM system
  - Built in diff/merge capability to graphically understand changes and evaluate different designs
  - Built in Panel graphics Webify communication tool to help better communications
- Promotes both opportunistic and strategic reuse via modeling and automation

# Configuration management with graphical differencing and merging

- Tight integration with configuration management tools
  - SYNERGY, ClearCase, Source Integrity, PVCS Dimensions & Version Manager
  - Any SCC compliant tool
- Graphically identify differences between versions and evaluate design alternatives
- Quickly accept or automatically and merge changes between multiple versions



# Customizable Automatic Documentation Generation

- Customize reports exactly the way you wish
  - Wizard-based document generation provides incredible flexibility
- Generates documentation in HTML, PowerPoint, Word, Rich Text Format, etc. from a Rhapsody model
  - Compatible with most word processing systems via RTF
  - Creates hyperlinks for fast report navigation
- Wide choice of out-of-the box templates, plus easy customization
- Enables collaboration via sharing of templates and sub-templates

Telelogic
An IBM Company

# Documentation generation

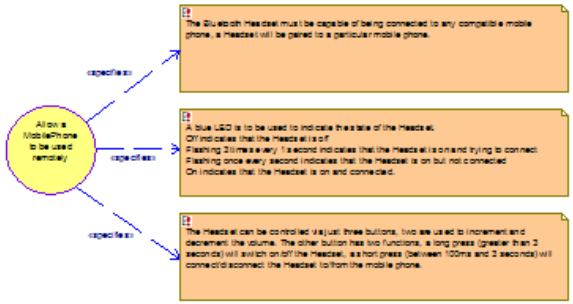
## 2. Package : RequirementsPkg

Package containing requirements that have been imported from DOORS via the Gateway.

| Name | Description | Owner | Status | Anchors |
|---|---|---|---|---|
| HLReq_1 | The Bluetooth Headset must be capable of being connected to any compatible mobile phone, a Headset will be paired to a particular mobile phone. | Andy | PercentComplete10 | Requirement: RhapReq_1_1<br>Requirement: RhapReq_1_2 |
| HLReq_2 | A blue LED is to be used to indicate the state of the Headset.<br>Off indicates that the Headset is off<br>Flashing 3 times every 1 second indicates that the Headset is on and trying to connect<br>Flashing once every second indicates that the Headset is on but not connected<br>On indicates that the Headset is on and connected. | Mark | Complete | Requirement: RhapReq_2_1<br>Requirement: RhapReq_2_2<br>Requirement: RhapReq_2_3 |
| HLReq_3 | The Headset can be controlled via just three buttons, two are used to increment and decrement the volume. The other button has two functions, a long press (greater than 3 seconds) will switch on/off the Headset, a short press (between 100ms and 3 seconds) will connect/disconnect the Headset to/from the mobile phone. | Mark | Complete | Requirement: RhapReq_3_1<br>Requirement: RhapReq_3_2 |

### 2.1 Object Model Diagram : Requirements Taxonomy

Diagram showing all the requirements and how they are connected to use cases.



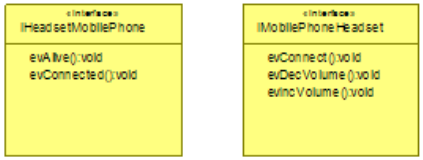

Document generated using "RhapsodyPackageOrientedProjectReport.tpl" Template

## 3. Package : InterfacePkg

Package containing interface classes

### 3.1 Object Model Diagram : Interfaces

From the various scenarios, the following interface classes can be created to describe the communication between the Headset and the MobilePhone




### 3.2 Class IMobilePhoneHeadset <<Interface>>

Interface class describing the operations that must be implemented by the MobilePhone and that will be used by the Headset

### 3.3 Class IHeadsetMobilePhone <<Interface>>

Interface class describing the operations that must be implemented by the Headset and that will be used by the MobilePhone

Document generated using "RhapsodyPackageOrientedProjectReport.tpl" Template

# DEMO
# – Rhapsody –