UML/SysML Architecture Design with Rhapsody and a Framework for Integrating Code, Simulink, Statemate, SDL Suite, etc.

We call it an Architectural Design and Integration Framework

By

Rick Boldt
VP, Product Management

Innovation 2008
TELELOGIC USER GROUP CONFERENCE
Software, Systems and Electronics Engineers Need a Better Architectural Design Environment

Today’s systems present a range of challenges:

• Complexity
  – Multi-team and multi-company relationships create parallel and distributed development often using different tools and methods
  – Electronics and software content is simultaneously intricate and important
  – Diverse sets of different algorithms with unique challenges
    • Closed-loop control, protocol development, logical and state based behavior

• A need for integration and testing that can become costly and time consuming

• Increased need to manage requirement changes during the development cycle

• Desire to reuse existing and future IP

✓ This creates the need for:
  – Better architectural design
  – Framework to integrated code and multiple tools beyond today’s algorithm-focused solutions
  – Environment for effectively managing change
Rhapsody as Architecture Design and Integration Framework

• SysML & UML were designed with architecture in mind
  – Validated across numerous industries
  – Separate views for functional, physical & software architectures
  – Requirements capture & elicitation

• Rhapsody can integrate rich set of external components:
  – Code: C, C++, Java or Ada
  – Tools: Simulink, Statemate, SDL Suite

• Primary benefits are:
  – Integration of various components that exist in the design environment into coherent, persistent design
  – Re-use of existing IP’s in new architectures/platforms
  – Ease the creation, and maintenance of the design
  – Provide standard support to requirements traceability, change management and other aspects of the engineering process
  – Enable co-execution to ensure correct interfacing and interaction
What’s wrong with code and algorithm modeling?

• Answer:
  – Necessary but insufficient
  – 1-dimensional
  – Every other view must be inferred
    • High level structure
    • Interaction/interfacing
    • Dynamic behavior interrelations
  – With large systems, code and algorithm models are unmanageable!

• Why? - Because to understand complex systems you need to understand
  – How pieces of different scale and abstraction work together
  – How different aspects (structural, behavioral) of the systems work and interact
But Why Architecture Modeling???

• Assume you got a great bonus at work…
• You and your wife want to build a new home…
• You meet with one architect…
• Two months later he comes back with a 647 pages document:
  – … indented by 7 meters from the west border of the premises, there is a left corner of the house
  – … The entrance door is indented by another 3.57 meters
  – … 2.30 meters wide and 2.20 meters high, left-hand hinge, opening to the inside
  – … If you come in, there are two light switches and a socket on your right, at a height of 1.30 meters
  – …
• Is it right? Are the requirements correct? Accurate? Consistent? How can you tell?
• Your are NOT happy… 😞 😞 😞
Yes... Architectural Modeling!

• Then you call another architect... And two weeks later he comes with this:

• You are very happy! 😊😊😊

• The second architect used **Modeling** to show different **Views** of the house!
  – Structural, Floor layout, Electrical, Plumbing, Heating
What do we need from Architectural Models?

• Visualization
  – Architectural models use 2+ dimensions to display structural and behavioral aspects
  – Architectural models use abstraction to view the system at different levels of scale from
    • System (very large)
    • Subsystem (large)
    • Component (medium)
    • Class (small)
    • Operation (code) (very small)
  – Graphical notations lend themselves to recursive application, allowing any number of levels of abstraction to be used.

• Understanding, Communication, Consistency, Provability
Use Rhapsody to define the functional, physical and software architecture, plug the algorithms into the architecture, and link to the requirements, test cases and external code.
Use Rhapsody to define the functional, physical and software architecture, plug the algorithms into the architecture, and link to the requirements, test cases and external code.
Visualizing Requirements

Requirements Diagram

- Req. 4.2
  ID = 4.2
  The mobile shall be able to receive streaming video at 384 kbps

- Req. 3.2
  ID = 3.2
  The mobile shall be able to receive short messages when the mobile is registered

- Req. 4.1
  ID = 4.1
  The mobile shall be able to send data at the rate of 364 kbps

- Req. 4.0
  ID = 4.0
  The mobile shall be able to receive data calls at the rate of 128 kbps

- Req. 5.6
  ID = 5.6
  The mobile shall be able to receive a maximum of 350 characters in a short message

- Req. 6.2
  ID = 6.2
  The optimal size of message the mobile can send in a text message is 350 characters

Dependencies:
- Req. 4.1
  Derivatives: Req. 4.2

Analysis:
- DataCall
- MHI

ConnectionManager

Verifying

Confirmations

Innovation 2008 Telelogic User Group Conference

© Telelogic AB
Requirements Traceability

Requirements Diagram

Requirement link

Requirement

Requirements Diagram
Requirements Coverage Analysis
Impact Analysis

Impact Analysis on selected high level requirement 36
Use Rhapsody to define the functional, physical and software architecture, plug the algorithms into the architecture, and link to the requirements, test cases and external code.
Structural Design
Architectural Design and Integration Framework

Use Rhapsody to define the functional, physical and software architecture, plug the algorithms into the architecture, and link to the requirements, test cases and external code.
Visualize Existing and Third Party Code

• Visualizing code clarifies the structure and architecture of the application.
• Streamlined reverse engineering and visualization workflow
• Automatic generation of diagrams from code for documentation
Example Model Integration

- Code and interface definition are imported into Rhapsody, where the complete application is compiled and tested with a click of a button
  - Ensures interfaces are complete
  - Enables interaction between components to be validated
Complete System Design with Rhapsody, Simulink, SDL Suite and Code Blocks

- Provides an integrated multi-domain solution for a multi-domain problem
- Use the proper Domain Specific Languages (DSL) and best tools for the job
- Code and interface definition are imported into Rhapsody, where the complete application is compiled and tested with a click of a button
  - Ensures interfaces are complete
  - Enables interaction between components to be validated
Use Rhapsody to define the functional, physical and software architecture, plug the algorithms into the architecture, and link to the requirements, test cases and external code.
Design and Test Processes Fully Integrated

- Common Browser
- Requirements linked to test cases
- Easy navigation between Design and Test artifacts;
- Design and Test - Always in sync
- Automatically generated test execution reports
Creating a Graphical Test Architecture

- A Test Architecture is a model of a code-based TestBench
  - Automatically generated
  - Updates as model gets updated

- Captures Architecture only
  - No test cases yet
  - Can be compiled!
Test Cases Behaviors as Sequence Diagrams

- Use Sequence Diagrams as graphical test scripts to capture the *required behavior*
- Record and ReUse Animated Sequence Diagrams as test cases for *regression testing* purposes
- Bring in existing test case, use Flowcharts or Statecharts in addition to Sequence Diagrams
Architectural Design and Integration Framework Summary

• Allows engineers to use the right tool, including hand coding, for the job
  – Design physical (electrical), functional and software architectures in Rhapsody
  – Capture algorithms in Simulink, Statemate, SDL Suite or C, C++, Java or Ada code

• Combined model can be executed and tested on the host to ensure proper behavior and integration of the complete system earlier in the process

• Easily trace changes throughout the design chain

• Enables easier reuse of existing IP

Provides a best in class integrated domain specific model based design and development environment
Thank You and Questions

Rick Boldt
VP Product Management,
Rhapsody & Statemate
Automotive Industry Marketing