Embedded (RSD: Rational System Developer, Rose Technical Developer)
Agenda

- Model-Driven Development
  - Challenges of Architecting Software
  - Rational Systems Developer v6.0.1
  - Summary
The Challenges of Software Development

- Ever-increasing complexity in the operational environments
  - C/C++, Java, .Net
  - Web, Handhelds, disconnected
  - Legacy integration, modernizing

- Ever-expanding choices to make on development solutions
  - Programming Languages, scripting Languages
  - IDE’s, testing tools

- Ever-changing nature how software gets created
  - Globally development teams
  - Outsourcing
  - Compliance and Regulations

Modeling: The Key to Managing Software Complexity

- Modeling is the standard approach in engineering to:
  - Manage Complexity
  - Mitigate Risk
- Software development is the same as every other kind of engineering in this respect
The Evolution of Software Development

Problem: delivering the next level of expectation

Solution: applying the next level of automation

Power

Time

Machine Code - Binary Development

Assembler - Textual Development

Pseudocode - 3GL Textual Modeling

3GL Textual Development

Visual Modeling

Model-Driven Development
What is Model-Driven Development (MDD)?

- The encapsulation of business logic and industry best practices into models
- The use of these models for application development, code generation, testing, and maintenance

- Modeling helps you work at higher levels of *abstraction*
- Higher levels of abstraction lead to higher *productivity*
Models....

- Abstract representations of
  - Business logic
  - Architecture
  - Design
  - Reusable assets
- Allow multiple viewpoints to be expressed independently
- Enable
  - Manipulation
  - Reasoning
  - Logical consistency checks
  - Automation
- More cost efficient
What is Model-Driven Development?

_Different Stakeholders, Different Models_

- **Business Model**
  - Visualization of business processes
- **Use Case Model**
  - Visualization of functional requirements
- **Analysis Model**
  - “What” the system must do to realize the functional requirements
- **User Experience Model**
  - Visualization of user interaction with the system
- **Design Model**
  - “How” the system will realize the functional requirements
- **Data Model**
  - Visualization of persistent storage
- **Implementation Model**
  - Visualization of the code
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
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)
Embedded Software Characteristics

- Timeliness
- Event Driven
- Concurrency
- Dynamic Structure
- Distribution
- Dependability
Importance of Tools in Systems Development

- Complexity
  - Scale causes productivity drop ("Diseconomies of scale")
- Time to market
- Competitive advantage
  - Software IS the product
- User Expectation
  - NOT a computer
- Bottom line
  - Defects can be crippling
  - Adding staff not the solution (Brooks’ law)
Example: Defense Industry Software Evolution

FROM
Small (KLOC)
Monolithic
Closed
Proprietary
Stand-alone
Single product

TO
Large (MLOC)
Architected
Open
Standard
Networked
Product Line

1980s 2000
Agenda

- Model-Driven Development
- Challenges of Architecting Software
- Rational Systems Developer v6.0.1
- Summary
Challenges of Architecting Systems Software

- Communicating a Solution
  - Even experienced teams need clear and consistent communication
  - Programming languages implement architectures -- they do not describe them
  - Architecture decisions are only useful if they can be followed
  - Development requires collaboration between different disciplines - mechanical, electrical, electronic, software, manufacturing, etc
  - Product may depend on key external supplier(s)
  - Suppliers and internal development teams are geographically distributed

“How can our decisions be shared?”
Challenges of Architecting Systems Software

- How to discover bad practices?
  - Often new projects begin with existing frameworks
  - Projects intend to leverage the value of existing code, but often lock-in problems as well

- How to enforce standards?
  - Developers stray from architecture decisions that they do not understand
  - Creating code to implement an architecture is tedious
  - Requirements are sometimes unclear when coding and construction begin
  - Product architecture is not completely understood early in the development process
  - Artifacts contain different technical languages
  - Artifacts are stored in multiple data repositories

“We did not find our tangles until late in the project – but they were there from the beginning.”

“The architecture was solid, but it was not implemented as planned!”
Challenge: Communicating Architecture

**Challenge**
- Overcome the problem of communicating solution to a team

**Resolution**
- Provide technologies to enable effective communication of a design
  - UML 2 Notation
- Integrate the architecture artifacts into the development environment
- Provide Global Access to artifacts internally and for suppliers
Challenge: How to discover complexity?

- **Challenge**
  - How can teams learn about the architecture of existing implementations or frameworks?

- **Resolution**
  - Provide an automated means for discovering architecture
    - Look for Anti-Patterns
    - Visualize complexity
Challenge : Enforcing Architecture Standards

**Challenge**
- How to enforce architecture decisions for a project team?

**Resolution**
- Generate source-code based on architecture decisions
- Create and enforce rules which support the architecture
  - Process Guidance
  - Code rules
  - Visually compare model changes
  - Link requirements to design
The IBM Rational Software Development Platform
A complete, open, modular, and proven solution
The IBM Rational Software Development Platform

**Eclipse-based software development**

- **Analyst**
  - WebSphere Business Integration Modeler & Monitor
  - Rational Software Modeler

- **Architect**
  - Rational Software Architect

- **Developer**
  - Rational Application Developer
  - Rational Web Developer

- **Tester**
  - Rational Functional & Manual Tester
  - Rational Performance Tester

- **Deployment Manager**
  - Tivoli Configuration Manager
  - Tivoli Monitoring

- **Project Manager**

- **Executive**

- **Customer Extensions**

- **ECLIPSE**

- **Rational Team Unifying Platform**

- **Rational Portfolio Manager**

- **3rd Party ISV Tools**
Supporting device development

**Deployment Platforms**
- VxWorks
- J2ME
- Windows CE
- Palm OS
- Linux
- OSE
- More than 100 Platforms

**Rational Team Unifying Platform**

**Rational Portfolio Manager**

**System Engineer**
- Rational Systems Developer
- Rational Rose Technical Developer

**Architect**
- Rational Systems Developer
- Rational Rose Technical Developer

**Developer**
- WebSphere Studio Device Developer
- Micro Environment Toolkit
- Embedded Voice Toolkit
- WindRiver Tornado
- Green Hills MULTI
- Microsoft eMbedded
- Others...
- Rational Systems Developer
- Rational Test RealTime
- Rational PurifyPlus

**Tester**
- Rational Test RealTime
- Rational Manual Tester

**Project Manager**

**Executive**
Governing the Lifecycle of Systems Development

- **Rational RequisitePro**
  - Business goals and system requirements

- **Rational Portfolio Manager**
  - Project investment, resourcing and scheduling

- **Rational Systems Developer**
  - System Architecture and Software Design and implementation

- **WebSphere Business Modeler**
  - Process modeling

- **Rational Manual Tester**
  - System Architecture and Software Design and implementation

- **Rational Test RealTime**
  - Run-time analysis and automated test

- **Rational Unified Process/Method Composer**
  - Methodology and process guidance for Systems (RUP SE)

- **Rational ClearCase, Rational ClearQuest**
  - Change management for all types of assets, from electronic designs to software to test assets
Agenda

- Model-Driven Development
- Challenges of Architecting Software
- Rational Systems Developer v6.0.1
- Summary
What is Rational Systems Developer?

- UML-based, model-driven development, design & construction tool for C/C++, J2SE and CORBA IDL implementation
- Serves as a framework for enabling business partner value-add development
IBM Rational Systems Developer Product Overview

“Architectural Discovery”
- Automatic anti-pattern and pattern detection
- Architectural discovery, analysis, metrics, and stability reporting
- Implementation level architectural rules

“Modeler”
- UML 2.0 Diagrams for Class, Communication, Component, Composite Structure, Deployment, Activity, Sequence, State, and Use Case
- OCL Support
- Automatic diagram generation
- Pattern content
- Extensive open API
- Java-based “scripting” for extensibility
- HTML and XML based data extraction and reporting
- Extensive printing (Windows Only)
- RAS tools
- Rose/XDE Model Import
- Traceability Analysis
- Visual Compare/Merge

“Lifecycle Integrations”
- ClearCase
- ClearQuest
- Requisite Pro
- SoDA
- RUP

“UML Language Transforms”
- Pattern/Transform authoring framework and services
- UML-to-code transforms for Java and C++
- Selective language to UML harvesting
- UML-to-CORBA IDL transformations

“C/C++ Development Tools”
- C/C++ editors and build management
- Compiler and debugger integration
- UML code editors for C/C++

“Java Development Tools”
- Advanced Java tooling - editors, projects, refactoring, etc.
- UML code editors for Java
- Code Review

Operating Environments
- Eclipse V3.0.3
- Windows
- Linux

ClearCase LT
Java Development Tools
Software Modeler
UML Language Transforms
Java Structural Analysis
Lifecycle Integrations
C/C++ Development Tools
Operating Environments
Getting Started: User Assistance

- New User Assistance model to enable users of all skill levels
- Leverages Product Tours to assist with the discoverability of capabilities
- Tutorial Gallery leverages tutorials as learning aids
  - “Watch and Learn”
  - “Play and Learn”
  - “Do and Learn”
- Samples gallery provides completed assets for reference purposes
  - Showcase
  - Application
  - Technology
- All user assistance can be launched from a Welcome perspective
Key Feature: C/C++ Development Environment

- Perspective for C/C++ Development
- C/C++ project hierarchical tree view
- UML class diagram visualization of C/C++ classes and structs
- C/C++ editor with syntax highlighting, code completion, and advanced search
Key Feature: C/C++ Development Environment

- Editing and Navigation
  - C/C++ Syntax Highlighting, Outline View
  - C++ Class Browser (Hierarchy View)
  - C/C++ Search
  - C/C++ Content Assist
- Project Import
  - Automated assistance in setting up CDT for search and content assist.
- UML C/C++ Code Editor
- Debug
  - GDB Integrated
  - Extensible Debug Interface
- Build
  - Standard Make for projects with existing build infrastructure
  - Managed Build
    - Automatic makefile generation
    - GNU tools supported out of box
    - Managed build is extensible, build tools can be plugged-in and build tools options selectable
- Meets Internationalization and Accessibility requirements
- Extensibility
  - Provides extension points for managed build, debuggers, …
**Key Feature: Modeling assistance**

- Simplify the capture of UML models during Analysis and Design
- Make modeling more accessible to a broader audience
- New custom views improve the editing experience

**Diagram links as first-class citizens**

“In diagram editing” to accelerate modeling.

New “Diagram Navigator” view provides a diagram filtered view of the models and workspace

Inheritance view

New properties view
Key Feature: Patterns

- Applying Patterns is very simple
  - Evolution of pattern experience based on lessons learned
- Pattern-authoring provides greater flexibility using Open API
- All Gang of Four design patterns provided
- Additional patterns provided via RAS repository on IBM developerWorks

Pattern rendered as a collaboration to support incremental application
Support "drag-and-drop" of pattern parameters into the collaboration
New patterns view providing outline and description of the pattern
Key Feature: Transformations

- Transformations are optimal for “batch” style computationally intensive operations
  - Model-to-model
  - Model-to-code

- Out-of-the-box code transforms
  - UML-to-Java/J2SE
  - UML-to-C++
  - UML-to-CORBA IDL
  - Plus sample model-to-model transforms

- Transformations may be updated via RAS repository hosted on IBM developerWorks
Key Feature: Architectural Analysis, Discovery, and Control

- Architecture discovery for J2SE
  - High-level software visualization

- Application architecture is reflected in the running code
  - Analyzing code can help assess its maintainability

- Govern the architecture with the assistance of rules
  - Template-based rule authoring

- Anti-pattern and pattern detection
  - Detection of cyclic dependencies, hubs, breakable, etc.
  - Wizard assisted automated quick-fix

Automatic generation of “topic” diagrams based on the results of the code analysis

Patterns and anti-patterns are rendered in the diagram editor. Navigation to detailed code is supported.

Code review pane providing a report of detected issues.

“Details View” providing an overview and avoidance guidance for anti-patterns.
Key Feature: Visualize Java Method Bodies

- Facilitates understanding and application’s behavior by providing visualization of detailed code
- Diagrams can be integrated in Javadoc reports

Integrated with the Java Package view

Select method to be visualized using UML

Leverages UML 2.0 sequence diagram constructs for loops, conditionals, etc...

“Topic” diagram for method is automatically updated/refreshed when method is updated

Alternate abstract view of method behavior
**UML Enhancements:**
JavaDoc with Embedded UML Diagrams

- Produce enriched JavaDoc
  - UML diagrams right on the pages
  - Completely integrated with hyperlinks
**UML Enhancements: Interaction Modeling**

- Interactions are expressed more effectively using UML 2.0 constructs
  - Support specification of test scenarios
    - Loop, alt, opt
    - Interaction fragment references
- Interactions can be rendered as either sequence or communication diagrams
- Sequence diagram editing improvements
  - Ordering and reordering

- Create class operations "on-the-fly" while doing sequence diagram edits.
- Support for UML 2.0 fragments including OPT, ALT, and LOOP
- Fragment collapse/expansion to facilitate working with large sequence diagrams
UML Enhancements: Browse Diagrams

Enables users to understand and discover models and applications without having to create or maintain diagrams.

Web Browser-like controls to navigate workspace or model.
**Team: RequisitePro Integration**

- Open and browse multiple RequisitePro projects
  - See requirements, packages, and views
- Associate requirements with model elements via drag and drop
- Create model elements from requirements
- Customizable synchronization

Requirements Explorer for viewing requirements in Eclipse.

Associate requirements and model elements using Drag-and-Drop.

View requirements traceability from the perspective of either “trace-to” or “trace-from”
Team: Process Guidance

- Integration with Rational Unified Process
- Tool Mentors provide guidance for activities
- User customizable views with user defined content
Team: ClearQuest Integration

- Easy access to queries, charts, & reports
- Console, SQL Query & Properties views
- Hierarchical result set view shows parent-child relationships
- View record forms, charts and reports
Team: ClearCase Integration

ClearCase Navigator view with integrated UCM activities

ClearCase Details view shows selected version information

Display version history, view & update config spec, display search results
Team: Model Compare & Merge

- Model differences & conflicts
- Description of selected difference or conflict
- Choose view type
- Diagram view of selected difference or conflict for contributor
Agenda

- Model-Driven Development
- Challenges of Architecting Software
  - Rational Rose Technical Developer
- Summary
MDA in Rational Rose RealTime

Develop at the model level with IBM Rational Rose RealTime
Analyze at the Model Level

- Requirements Analysis

*Develop at the model level with IBM Rational Rose RealTime*
Design at the Model Level

- Requirements Analysis
- Architecture and Design

Develop at the model level with IBM Rational Rose RealTime
Implement at the Model Level

- Requirements Analysis
- Architecture and Design
- Implementation
  - Classes
  - Structure
    - Behavior

*Develop at the model level with IBM Rational Rose RealTime*
Build at the Model Level

- Requirements Analysis
- Architecture and Design
- Implementation
- Generate/Build/Link
  - Classes
  - Behavior
  - Structure

**MDA transformation of model**

*Develop at the model level with IBM Rational Rose RealTime*
Deploy at the Model Level

- Requirements Analysis
- Architecture and Design
- Implementation
- Generate/Build/Link
- Deployment

Develop at the model level with IBM Rational Rose RealTime
Debug at the Model Level

- Requirements Analysis
- Architecture and Design
- Implementation
- Generate/Build/Link
- Deployment
- Debugging
  - Model level
  - Start, stop, single step
  - Inject, trace messages
  - Variable watch/modify
  - Break
  - Integrate with source debugger

Develop at the model level with IBM Rational Rose RealTime
Test at the Model Level

- Requirements Analysis
- Architecture and Design
- Implementation
- Generate/Build/Link
- Deployment
- Debugging
- Testing

Develop at the model level with IBM Rational Rose RealTime
Analyze at the Model Level

- Requirements Analysis
- Architecture and Design
- Implementation
- Generate/Build/Link
- Deployment
- Debugging
- Testing
- Run-time Analysis
  - Through Rational Test RealTime integration
  - Coverage
  - Performance
  - Memory errors
  - Model or code level

Develop at the model level with IBM Rational Rose RealTime
Natural Embedded Notation and Concepts

Specify behavior with UML hierarchical state machines

Use natural embedded concepts with Rational Rose RealTime
Natural Embedded Notation and Concepts

- Specify behavior with UML hierarchical state machines
- Simplify and optimize concurrency with active objects
  - Two-way encapsulation through port interfaces
  - Simplify concurrent development
    - Avoid expensive thread context switches
    - Avoid mutexes
    - Reduce concurrency errors

Use natural embedded concepts with Rational Rose RealTime
Natural Embedded Notation and Concepts

- Specify behavior with UML hierarchical state machines
- Simplify and optimize concurrency with active objects
- Specify architecture with structured classes
  - UML 2.0 structuring concepts comes from Rose RealTime
    - Structure diagrams, ports, connectors,…
  - IBM Rational has over a dozen years of experience applying structuring concepts

Use natural embedded concepts with Rational Rose RealTime
Apply a Uniform Solution to Host and Your Target

- Test and debug graphically on target just as you do on the host
  - Host is just another target, not a “simulation”

- Works on your target environment
  - C, C++, Java, and CORBA IDL
  - All leading RTOSes (or no RTOS at all)
  - Any 8- to 64-bit processor
  - Any application size
    - Up to multi-million-line
    - Down to footprints of a few Kbytes

- Many platforms supported out of the box

    Other platforms supported through TargetRTS Porting Wizard

Work on host and target with Rational Rose RealTime
Connexis for Distributed UML Designs

- Enables deployment and visualization of distributed applications
- Supports patterns for creating high-availability applications
- Provides the distributed communication infrastructure

Use middleware for distribution with Rational Rose RealTime
Build at the Model Level

- Unequalled code generation:
  - Reduced implementation time
    - Drawing is much faster than coding
  - Increased quality from MDA
    - Gives decreased rework and churn

- **Result**: reduced time to market

Develop at the model level with **IBM Rational Rose RealTime**
Debug at the Model Level

- Generate the details, debug the logic

Model \[\rightarrow\] Generate/Compile \[\rightarrow\] Control/Observation
Debug at the Model Level

- Observe object creation, watchpoints
Debug at the Model Level

- Step through model, observe changes
Debug at the Model Level

- Observe object destruction upon termination
Debug at the Source Level

- Compile, control, visualize and debug from Rational Rose RealTime
- Examples: MULTI, Tornado, Visual Studio
  - Compilation errors mapped back to UML design elements
  - Target download and control
  - UML model debugging
  - Integrated 3GL debugging (e.g., mapped to state transitions)
RTOS Target Support

- All popular RTOS targets are supported “out of the box”
- All other 8- to 64-bit targets (with or without RTOS) are supported through wizard-assisted porting technology
- Your environment is supported
Agenda

- Model-Driven Development
- Challenges of Architecting Software
- Rational Systems Developer v6.0.1

Summary
Summarizing Key Features in Rational Systems Developer

- **Architecture Support**
  - Java, J2SE, C++
  - UML2 Modeling
  - Architecture Discovery via Application Analysis
  - Patterns and Transformations

- **Team Environment**
  - Enhanced Compare / Merge
  - Integrated RequisitePro Views
  - Process Advisor
  - ClearCaseLT included
    - CC and CQ fully integrated

- **Open Platform**
  - Based on Eclipse 3 Shell
  - Broad support for WebSphere tools
  - Testing and Team tools work together
Summarizing

- Model visually
  - To understand complex systems
- Develop at the model level
  - To deliver better applications faster
- Use natural embedded notation and concepts
  - Hierarchical state machines for event-driven object behavior
  - Structures for architectures
  - Active objects to simplify concurrency
- Apply a uniform solution on host and target
  - To simplify cross-platform development
  - To adapt to your target
- Use middleware to simplify distribution
Business Partners

- **EmbeddedPlus**
  - DOORKeeper - DOORs integration for Rational Systems Developer/Rational Software Modeler
  - SYSMT – plug-in to Rational Systems Developer/Rational Software Modeler that provides support for SysML extension of UML 2.0 and model simulation and execution

- **WindRiver**
  - WindRiver Workbench – leading Eclipse-based IDE for C/C++/Java development for RTOS

- **Pathfinder**
  - PathMate - High-performance, configurable, MDA code generation for embedded platforms based on Rational Systems Developer/Rational Software Modeler UML 2.0 Models

- **Coverity**
  - Prevent - Static analysis tools for C++

- **Galorath**
  - SEER-SEM - SEER estimation from Rational Systems Developer/Rational Software Modeler use-case models

- **Intel (not yet committed)**
  - C++ Compiler for Linux – high-optimization compiler for Intel platforms integrated to Rational Systems Developer

* - all partners will validate to RFRS
For More Information…

- Contacting your IBM representative
  - [http://www.ibm.com/contact/us](http://www.ibm.com/contact/us)
  - Call 1 800 728 1212

Technical Resources on IBM developerWorks
- Technical library of whitepapers, utilities, betas
- Downloadable demos
- Discussion forums
Questions
Thank You