IBM Rational Software

Tehoja arkkitehtuurimallinnuksesta

Sytyke-laivaseminaari 7.-9.9.2005

ON DEMAND BUSINESS

Jouko Poutanen
jouko.poutanen@fi.ibm.com

Overview

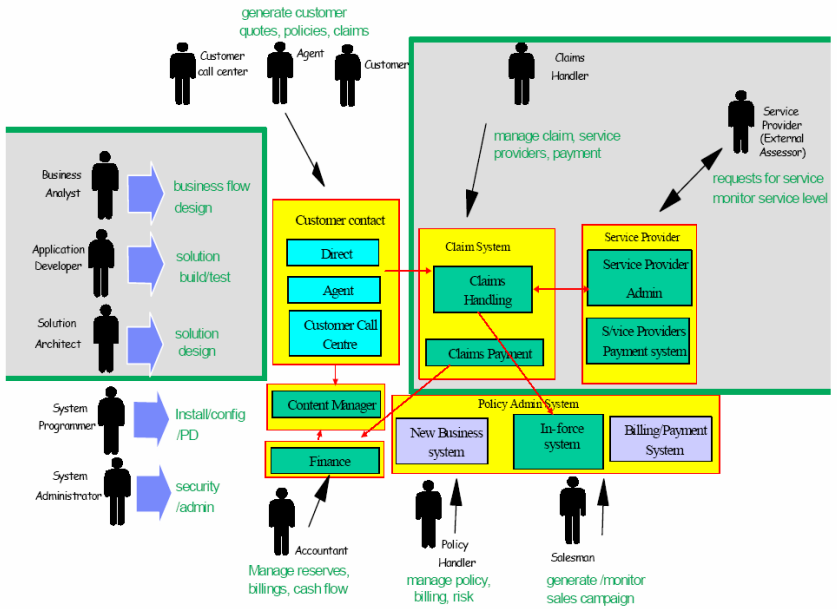
- The Challenges of Enterprise Software Development
- Modeling: The Key to Managing Software Complexity
- What is Model-Driven Development?
- What do we mean by “Practical Model-Driven Development”?
- Practical Model-Driven Development with Rational Software Architect

Overview

- The Challenges of Enterprise Software Development
- Modeling: The Key to Managing Software Complexity
- What is Model-Driven Development?
- What do we mean by “Practical Model-Driven Development”?
- Practical Model-Driven Development with Rational Software Architect

The Challenges of Enterprise Software Development

- Ever-increasing complexity in the operational environments
  - J2SE, J2EE, .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



More Layers,
More Servers,
More Frameworks,
More "Moving Parts",

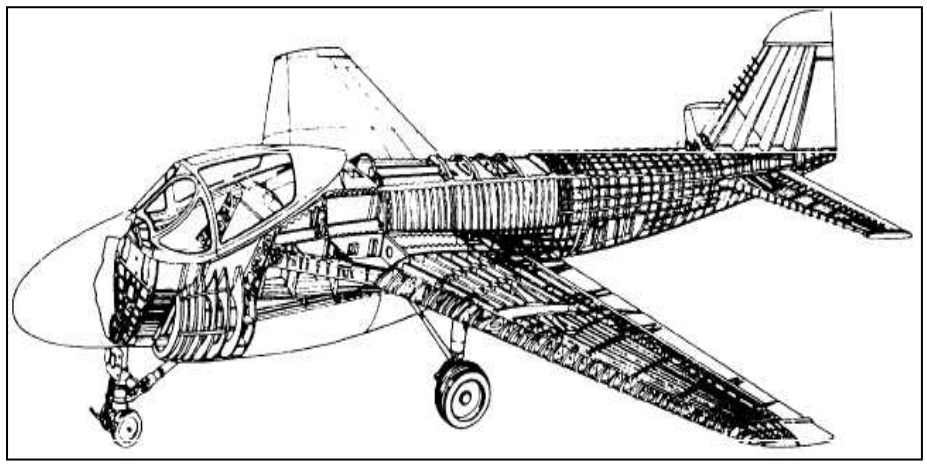
More Complexity

Overview

- The Challenges of Enterprise Software Development
- Modeling: The Key to Managing Software Complexity
- What is Model-Driven Development?
- What do we mean by “Practical Model-Driven Development”?
- Practical Model-Driven Development with Rational Software Architect

Modeling: The Key to Managing Software Complexity

- Consider the Boeing 747 Jumbo Jet
  - More that 6 million parts
  - 170 miles of Wiring
  - Hundreds of Cockpit controls
- To design and build it, Boeing produced more than 75,000 engineering drawings

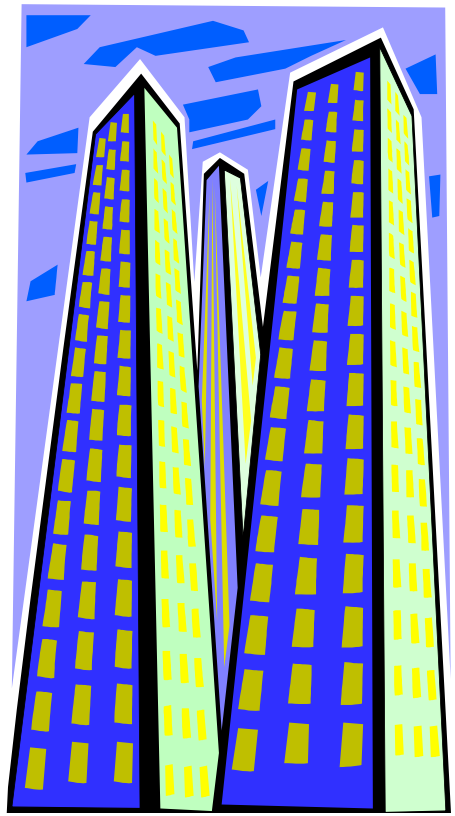


Why? They were building a complex system
And they had to get it right!

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

Maybe you *have to*



But then, maybe you should

Well, maybe you shouldn't'

Modeling: The Key to Managing Software Complexity

- What people are saying today about software modeling*
  - "…ensures that customers get what they ask for"
  - "…allows me to create domain-specific areas of expertise and bring in the right people at the right time"
  - "…allows me to better understand my enterprise at different levels of detail for different stakeholders"
  - **"…higher levels of reuse, reduce overall costs**

Those who have adopted software modeling are finding that it improves technical quality, reduces business costs, and better manages risk.

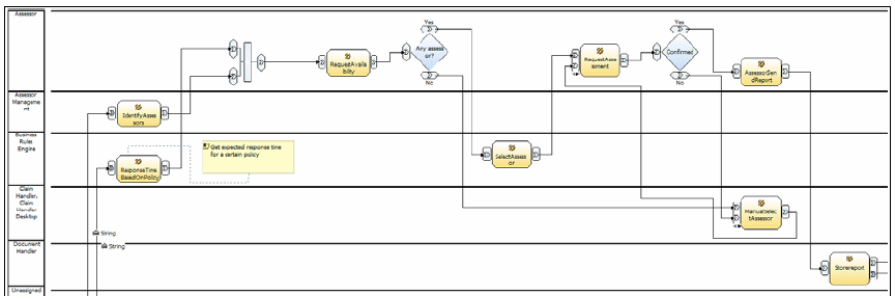
Source: IBM whitepaper, The Value of Modeling
http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/ValueOfModeling.pdf

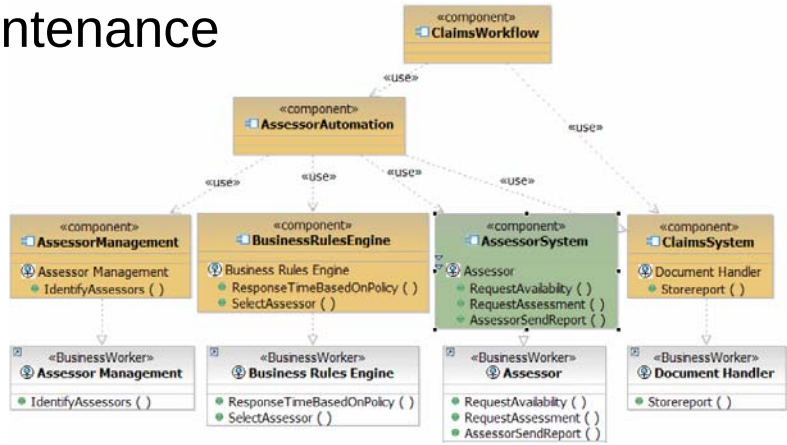
Overview

- The Challenges of Enterprise Software Development
- Modeling: The Key to Managing Software Complexity
- What is Model-Driven Development?
- What do we mean by “Practical Model-Driven Development”?
- Practical Model-Driven Development with Rational Software Architect

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*

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

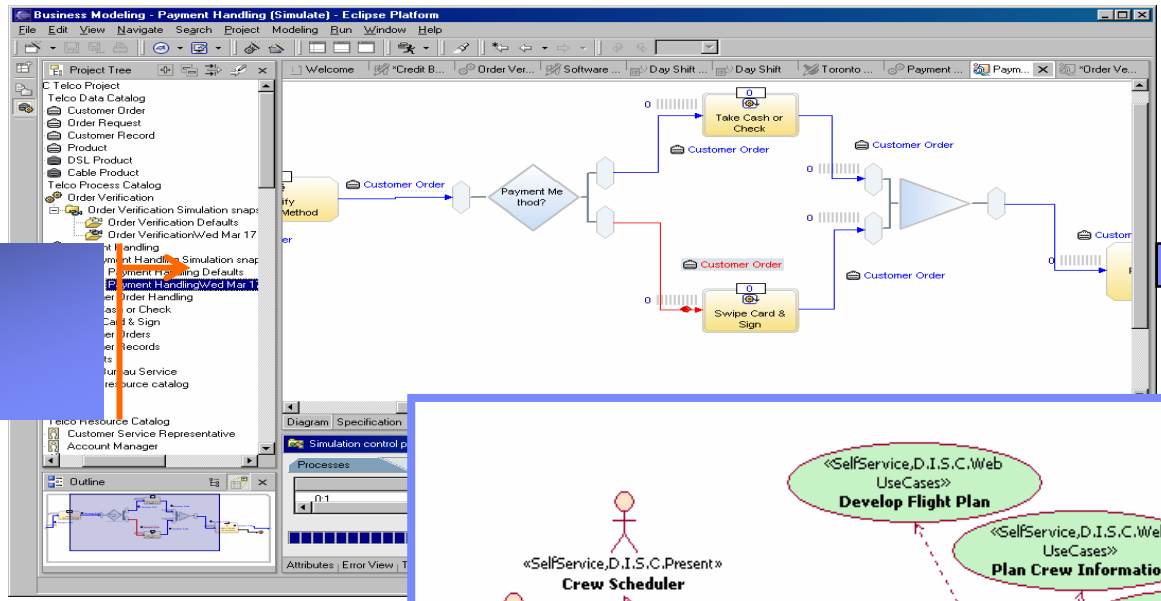
What is Model-Driven Development?

Different Stakeholders, Different Models

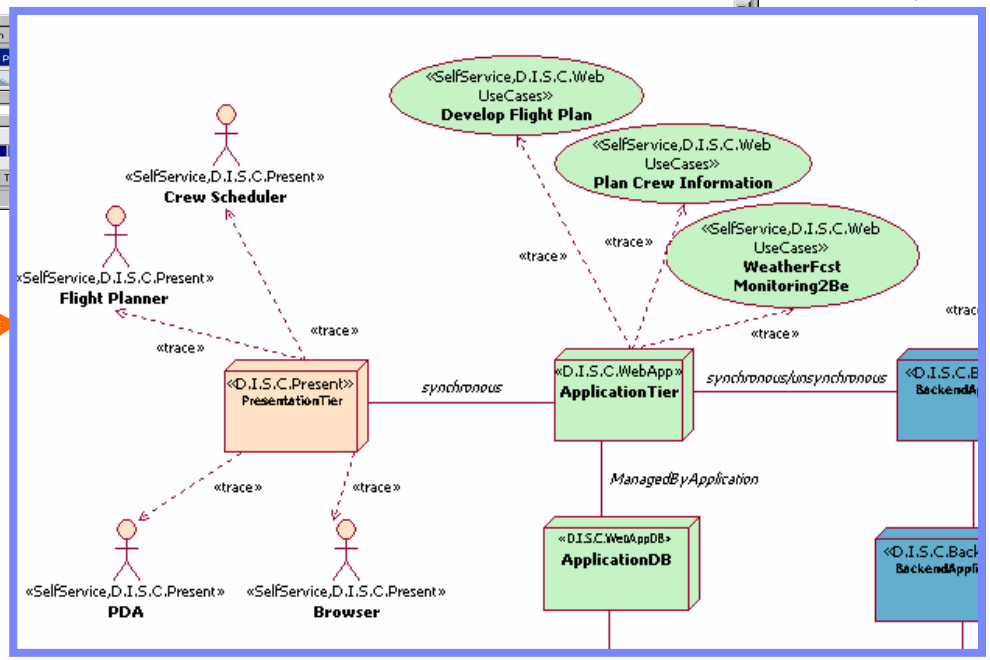
- Business Model
  - Visualization of business processes
- Use Case Model
  - Visualization of functional requirements
- Analysis Model
  - “What” the system must do to realize the functional requirements
- User Experience Model
  - Visualization of user interaction with the system
- Design Model
  -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

Example

Model the business



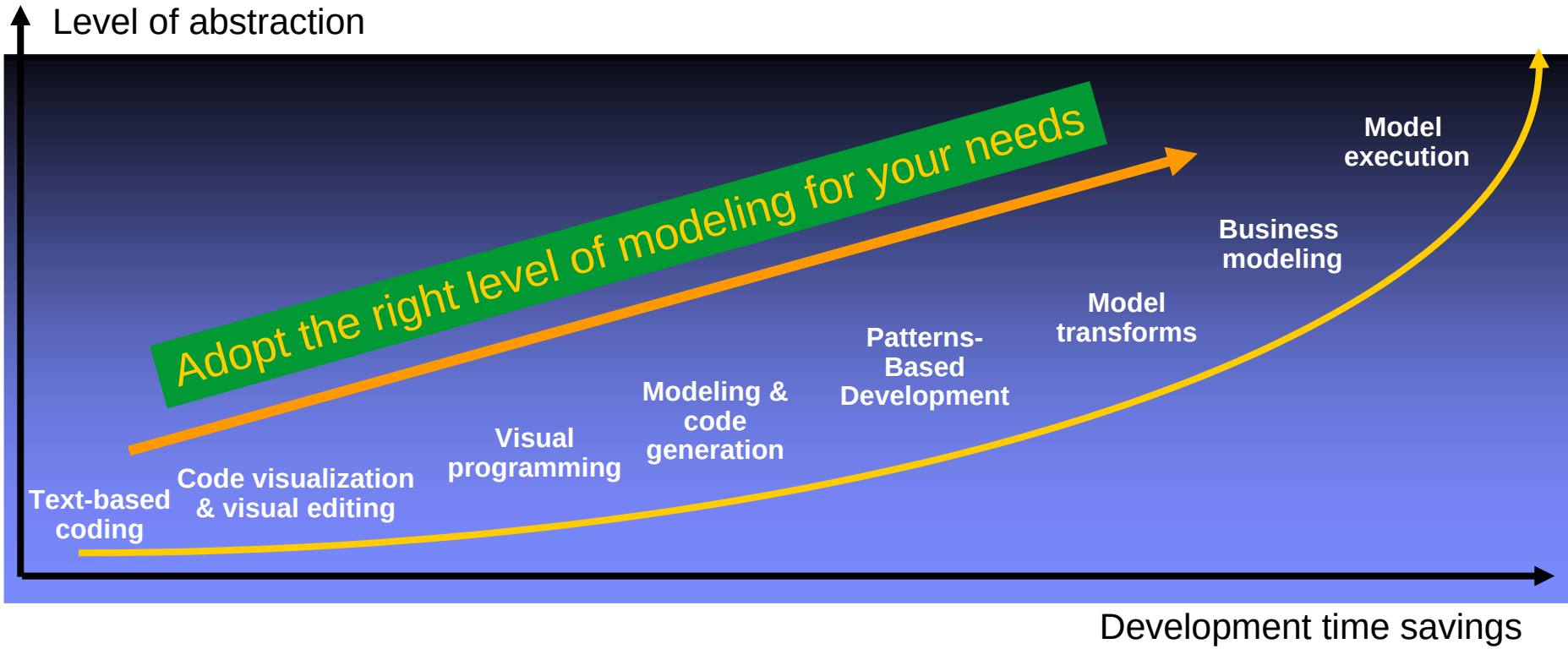
Apply relevant architectural patterns



IBM WebSphere Business Integration Modeler
IBM Rational Software Architect

What is Model-Driven Development?

Model-driven development is **not** “one size fits all”



- You can model at various levels of abstraction and detail
- The key is to choose the level that’s right for your project and team

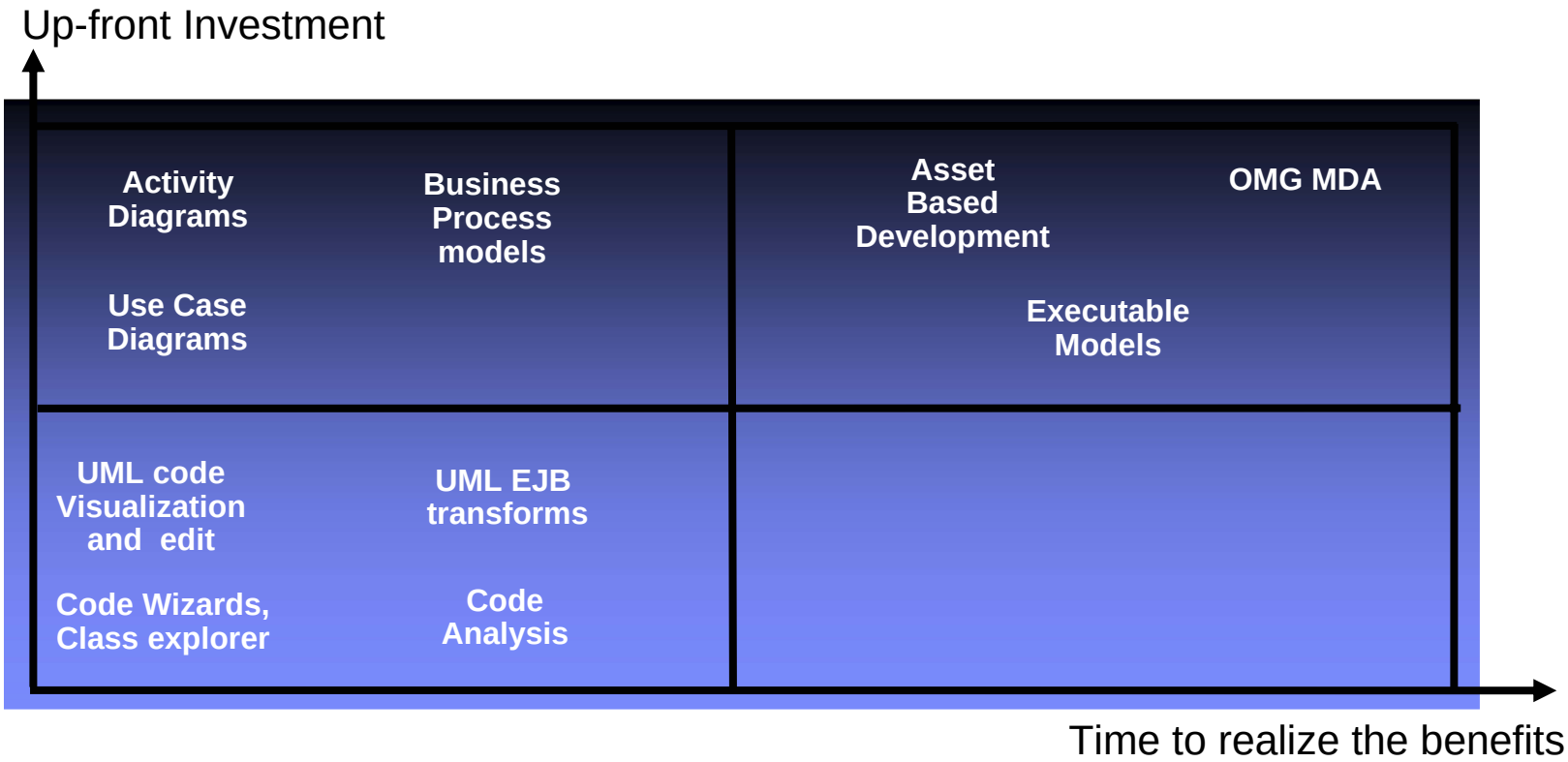
Overview

- The Challenges of Enterprise Software Development
- Modeling: The Key to Managing Software Complexity
- What is Model-Driven Development?
- What do we mean by “Practical Model-Driven Development”?
- Practical Model-Driven Development with Rational Software Architect

What do we mean by "Practical Model-Driven Development"

- Why don't more developers model their software?
  - Is it cultural?
    - "Just give me an editor and a debugger"
    - "We can't afford modeling tools"
    - "Modeling is documentation — I hate doing documentation"
    - "In the past we've tried and failed.
- The key reasons are the perceptions that
  - Modeling is ***hard***
  - Modeling will slow me down!

Practical Model-Driven Development



- To a developer practical model-driven development
  - Requires low up-front investment
  - Results in high short-term benefits

Overview

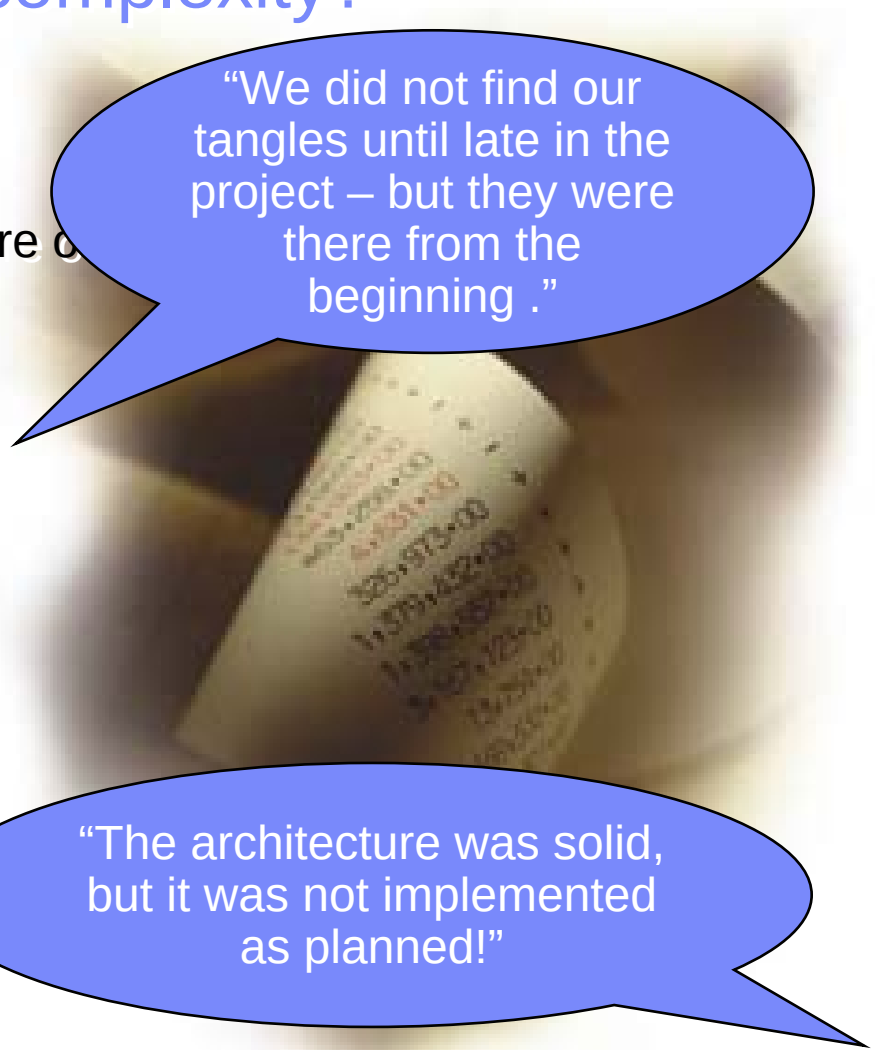
- The Challenges of Enterprise Software Development
- Modeling: The Key to Managing Software Complexity
- What is Model-Driven Development?
- What do we mean by “Practical Model-Driven Development”?
- Practical Model-Driven Development with Rational Software Architect

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

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



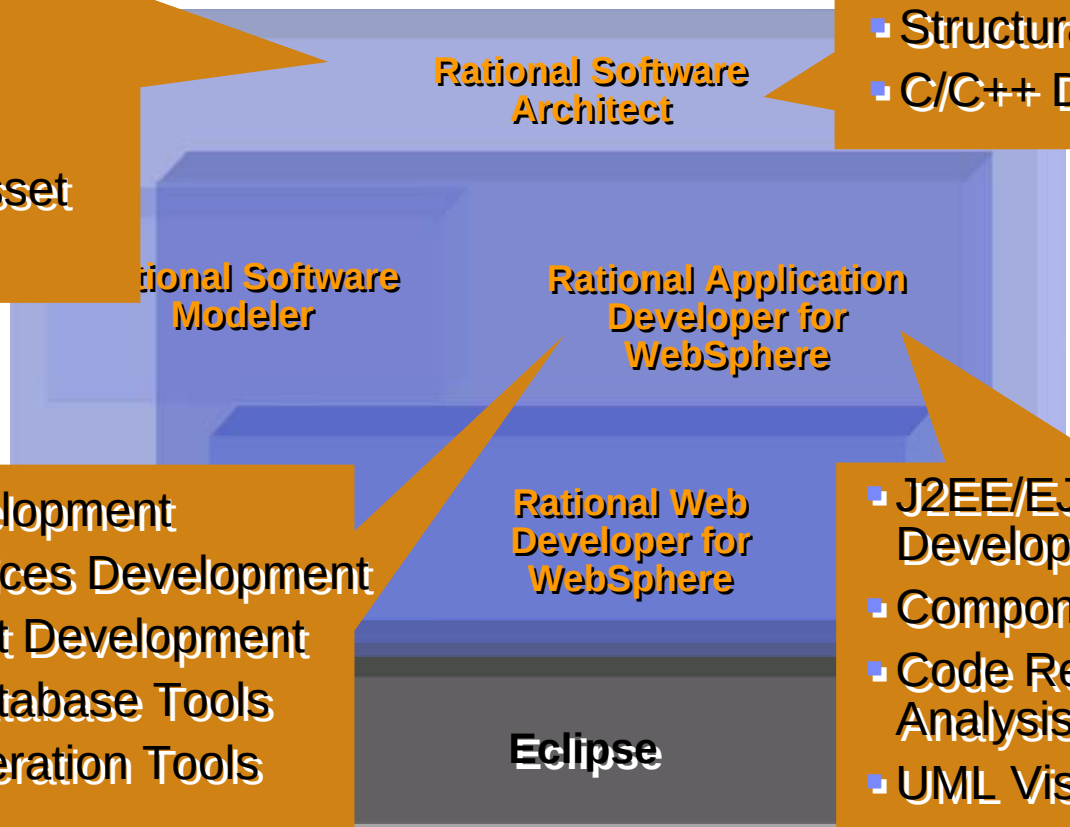
Challenge : Enforcing Architecture Standards

- **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

A Quick Look at the New Rational Design and Construction Tool Set

- UML 2.0 modeling
- Pattern/ Transform Authoring
- Reusable Asset Browser

- UML Language Transforms
- Structural Review & Control
- C/C++ Development Tools

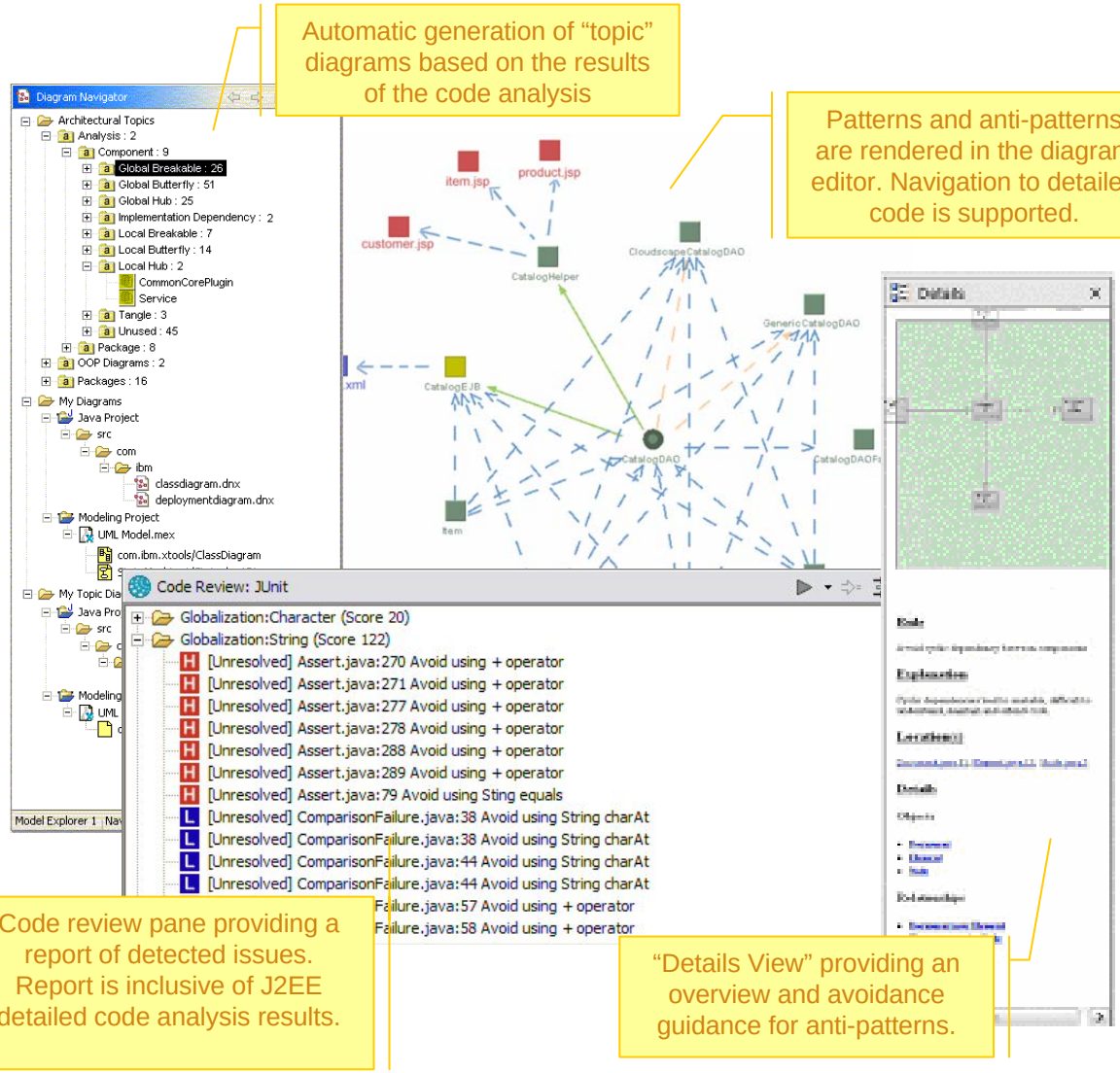


- Web Development
- Web Services Development
- Rich Client Development
- XML & Database Tools
- Java Generation Tools
- Unit Test

- J2EE/EJB & Portal Development
- Component Testing
- Code Review & Runtime Analysis
- UML Visual Editors
- Configuration Management

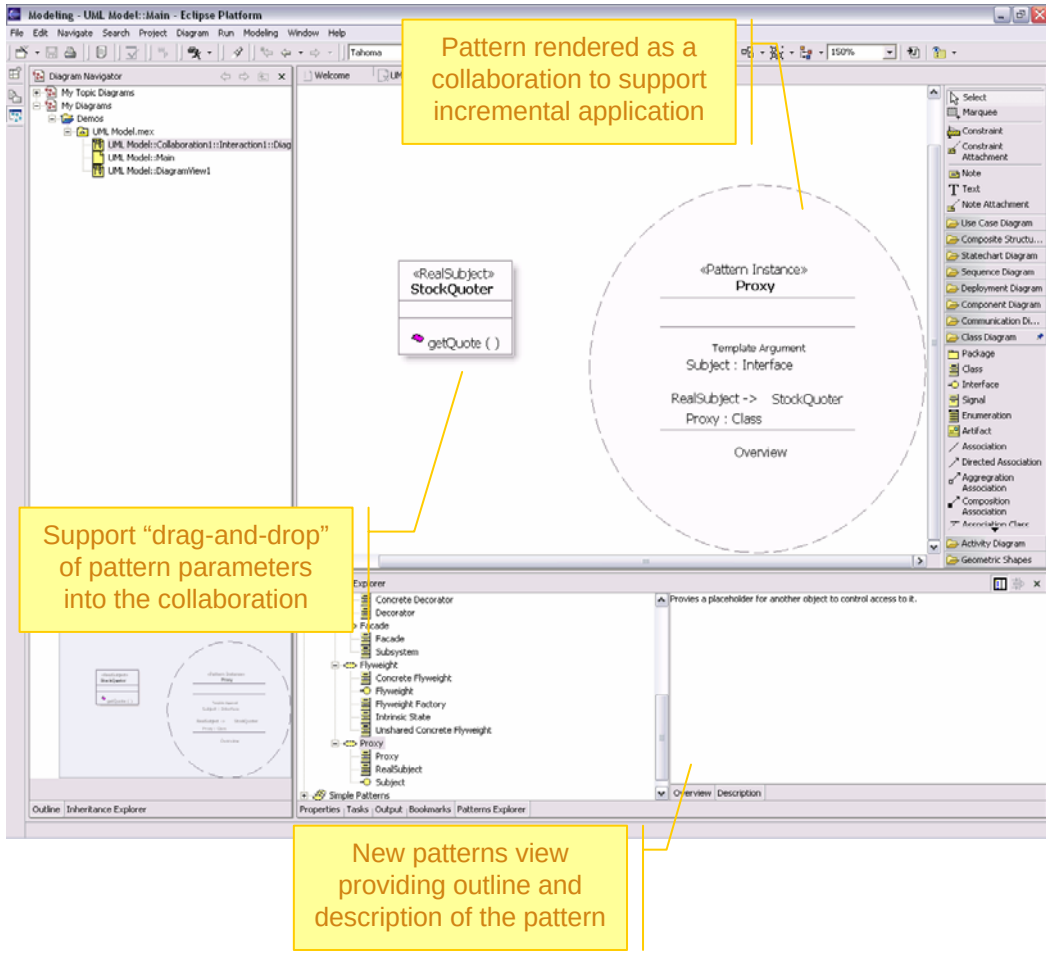
Architectural Analysis, Discovery, and Control

- Architecture discovery for J2EE and 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



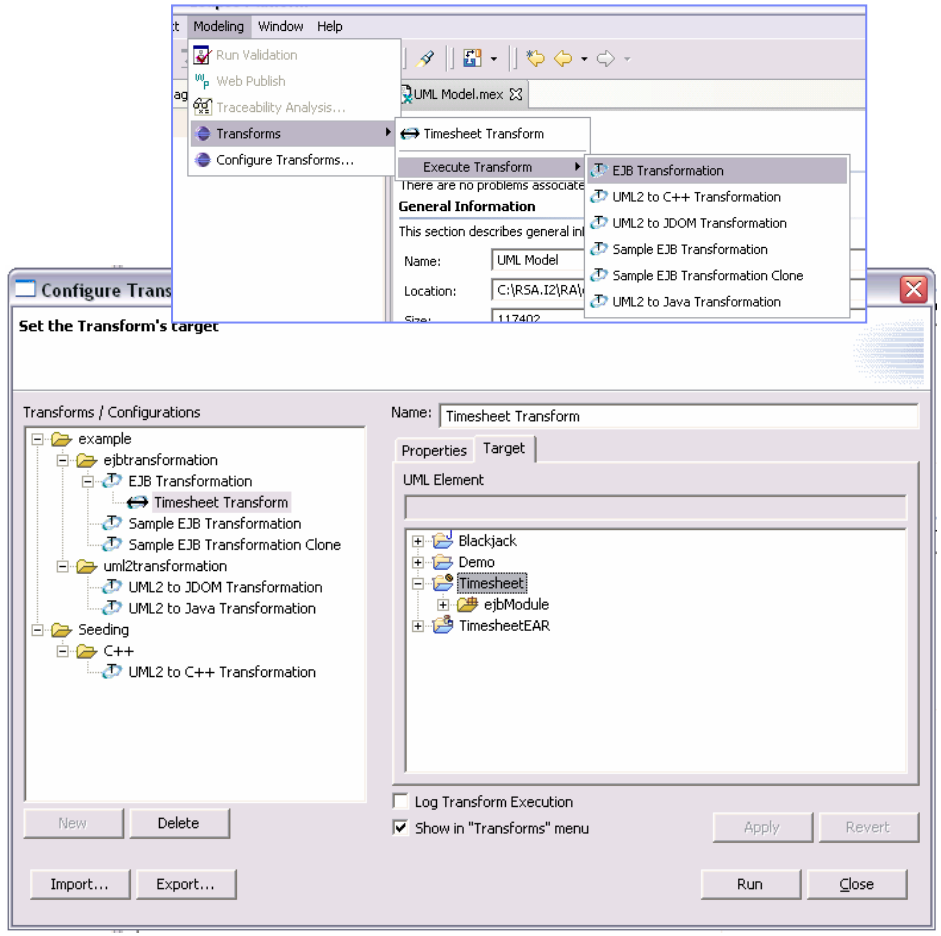
Key Feature: Patterns

- 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



Key Feature: Transformations

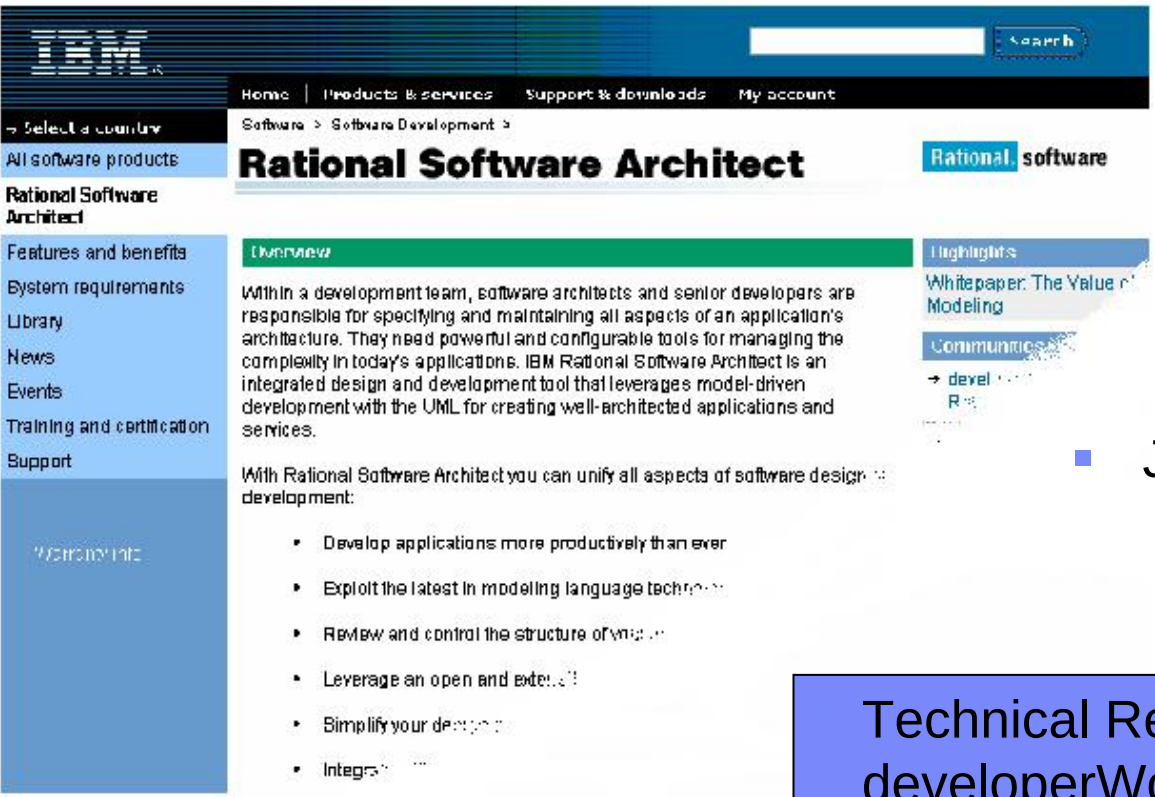
- Transformations are optimal for “batch” style computationally intensive operations
  - Model-to-model
  - Model-to-code
- Out-of-the box code transforms
  - UML-to-J2EE/Java
  - UML-to-C++
  - Plus sample model-to-model transforms
- Transformations may be updated via RAS repository hosted on IBM developerWorks
  - Examples:
    - Web Services transformations
    - XSD transformations



Summary

- Modeling is the key to managing the complexity and risks associated with today's enterprise software systems.
- Model-driven development helps teams work at higher levels of *abstraction,* which leads to higher *productivity*
- "Practical model-driven development" refers to tools and techniques that allow developers to rapidly gain the benefits of model-driven development with little up-front investment.
- Rational Software Architect is IBM's premier design & development tool enabling model-driven development and model-driven architecture. It incorporates all the capabilities in Rational Application Developer for WebSphere for building scaleable Web, Web services, Java, J2EE and portal applications

Lisätietoja…



- Jouko Poutanen
  - **Jouko.poutanen@fi.ibm.com**
  - 040 830 3908

Technical Resources on IBM developerWorks

- www.ibm.com/developerworks/rational
- Technical library of whitepapers, utilities, betas
- Downloadable demos
- Discussion forums


Questions


Thank
You