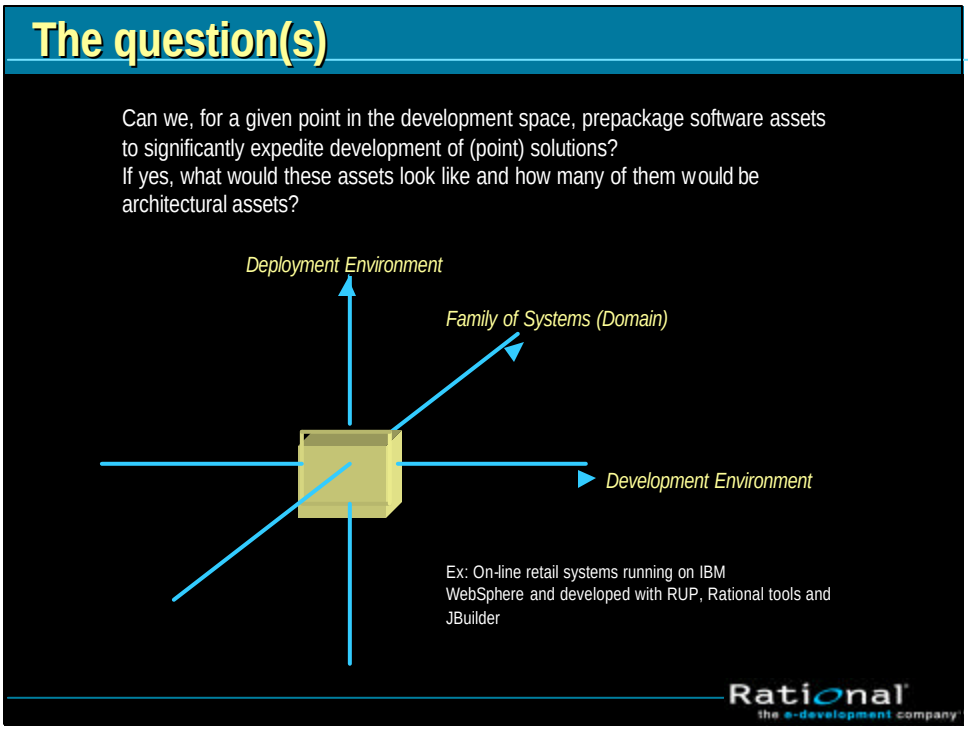

speed
quality
Rational
the e-development company
Can We Sell Application Architectures?
(WICSA 2001 Key-note Presentation)
WICSA2001
SOFTWARE
ARCHITECTURE
Amsterdam, NL
August 28-31, 2001
Royal Netherlands Academy of Arts and Sciences
Amsterdam, The Netherlands
Dr. Wojtek (Voytek) Kozaczynski, Rational Software, USA


The question(s)
Can we, for a given point in the development space, prepackage software assets to significantly expedite development of (point) solutions?
If yes, what would these assets look like and how many of them would be architectural assets?
Deployment Environment
Family of Systems (Domain)
Development Environment
Ex: On-line retail systems running on IBM WebSphere and developed with RUP, Rational tools and JBuilder
Rational
the e-development company

## What is an architectural asset?

- If

  Software architecture encompasses the set of significant decisions about the organization of a software system

  - Selection of the structural elements and their interfaces by which a system is composed
  - Behavior as specified in collaborations among those elements
  - Composition of these structural and behavioral elements into larger subsystems
  - Architectural style that guides this organization

- Then

  Architectural asset is a software artifact (or a group of artifacts), that captures these significant architectural decisions

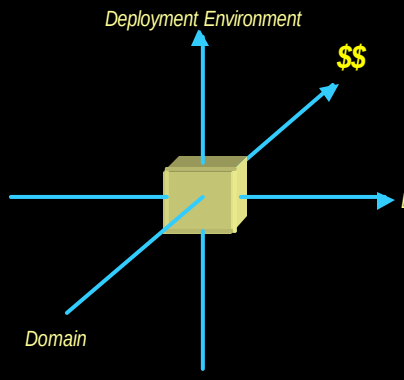
Rational the e-development company

## Outline

- *Enterprise applications, a line in the design* space
- Deployment environments
- Development environments
- Software assets
- Conclusions

Rational the e-development company

## "Money talks"




There should be economic incentives for the asset producers and consumers to prepackage and purchase architectural assets ✍ the domain should have certain characteristics:

- Large software development investments
- High levels of repeatability
- Shortage of skilled developers
- …

Rational the e-development company

## Enterprise Applications

- ✍ Implement all or part of a business process
  - ✍ Contain a model of a "business reality"
- ✍ Large amounts of highly structured information
  - ✍ May have 1000s of data types and associations
  - ✍ May have 1000000s of instances with complex relationships
- ✍ Small amounts of complex computation
  - ✍ User interaction
  - ✍ Constraint enforcement
  - ✍ Process automation, …
- ✍ Often complex, distributed deployment infrastructure
  - ✍ Multiple nodes
  - ✍ Complex interfaces and communication

Rational the e-development company

## ...

- Specific quality of service requirements
  - Security of critical data and processes with widely exposed interfaces
  - Scalability to large numbers of clients and transactions
- Overwhelming logistical complexity
  - Multiple stakeholders
  - Different technologies and development methods
  - Rapid response to constantly changing business conditions
  - Concurrent projects at different stages of development


Rational
the e-development company

## Large investments

- By 2004, the worldwide IT services industry will grow at a compound annual growth rate (CAGR) of 11% to reach almost $584.6 billion

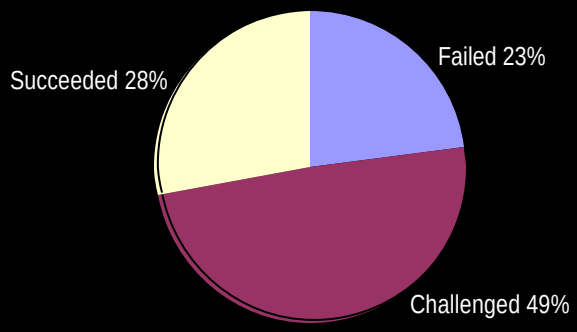
  IDC *Worldwide IT Services Industry Forecast Analysis, 1997-2004.*

  - Systems integration will grow at 13% CAGR; Rising from $59 billion in 1999 to $109 billion by 2004
  - Packaged software support and integration will grow at 15% CAGR; Rising from $35 billion in 1999 to $71 billion in 2004
  - Network infrastructure management will grow at 16% CAGR (interoperability is the major issue)

Rational
the e-development company

## Unimpressive success rates

- According to the Standish Group's Chaos study
  - In 2000 Corporate America started 300,000 new software projects
  - Close to 70% of the projects were either challenged or failed




Standish Group's Extreme Chaos Report, 2001

Rational the e-development company

## Development still difficult

- Ad-hoc development
- Development at low levels of abstraction
  - Developers must cope with broad abstraction gap between requirements and designs
- Reliance on labor-intensive activities
- No economically-significant reuse
- High levels of discovery and one-off implementations

… while enterprise systems can be grouped into application families showing significant levels of architectural similarities

Rational the e-development company

## The Enterprise Applications' Domain Is It!

Rational
the e-development company

## Outline

- *On-line enterprise applications, a line in the design space*
- *Deployment environments*
- Development environments
- Software assets
  - Components
  - Pattern
  - Architectural mechanisms
  - Application frameworks, prepackaged software architectures
- Conclusions

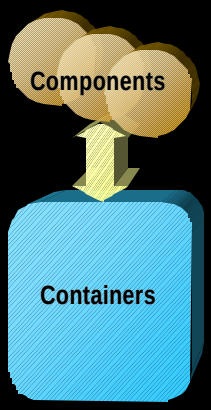
Rational
the e-development company

## Two dominant deployment platforms

- Java 2 platform, Enterprise Edition (J2EE)
  - JSP
  - Servlets
  - EJBs
  - JMS
  - JNDI
  - JDBC
  - JVM
- .NET
  - ASPX
  - .NET Components
  - MSMQ
  - Windows Registry
  - ODBC
  - Common Language Runtime
- The plaftorms share HTML, XML, SOAP, ..

Rational the e-development company

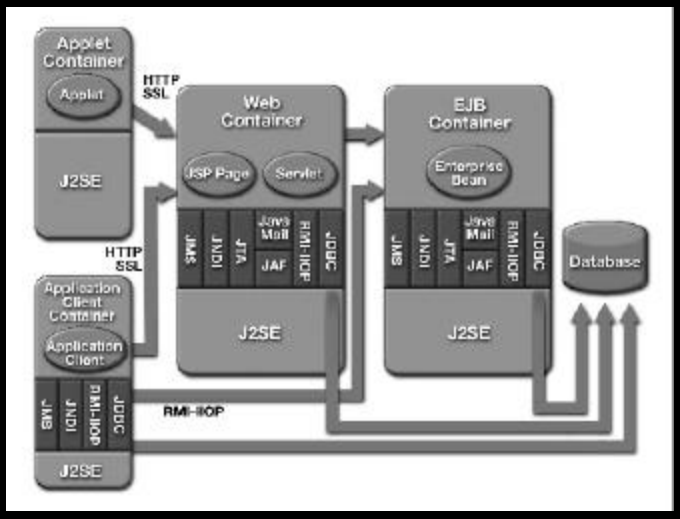
## J2EE; container-based deployment architecture




- Separation of concerns
  - Application developers focus on components
  - Platform developers focus on containers
- Containers mediate clients and components
  - Transactions, resource pooling, persistence
  - Policies specified through configuration not code

Rational the e-development company

## J2EE Container Architecture



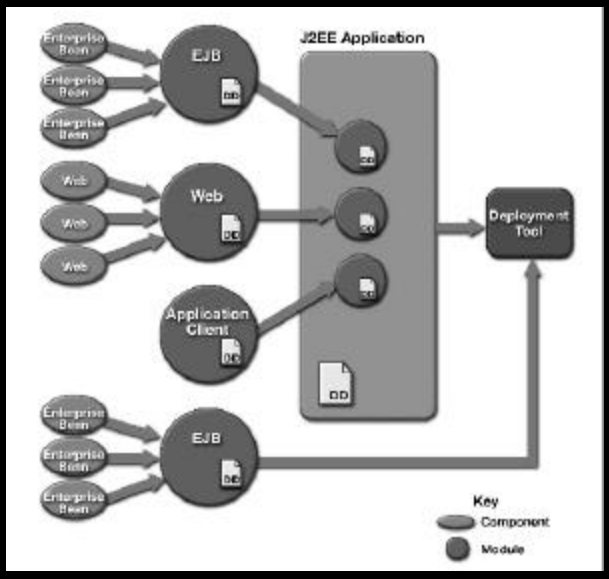

Rational
the e-development company

## Component Types

- Client Components
  - Applets, applications
  - Execute on client virtual machine
  - All other types execute on server virtual machine
- Web Components
  - Servlets, JavaServer Pages
  - Deployed, managed, and executed by web container
  - Respond to requests from HTTP and other protocols
  - Generate web-based application user interface
- Application Server Components
  - Enterprise Beans (Session & Entity)
  - Deployed, managed, and executed by application container
  - Maintain conversational or persistent state in instance variables between method invocations
  - Participate in distributed transactions that span multiple resources
  - Provide services concurrently to large numbers of clients
  - Perform client authentication and authorization to access protected services

Rational
the e-development company

## Packaging Architecture



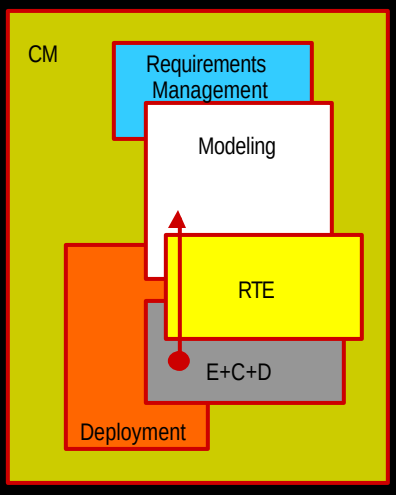

Rational the e-development company

## Outline

- *On-line enterprise applications, a line in the design space*
- *Deployment environments*
- *Development environments*
- Software assets
- Conclusions

Rational the e-development company

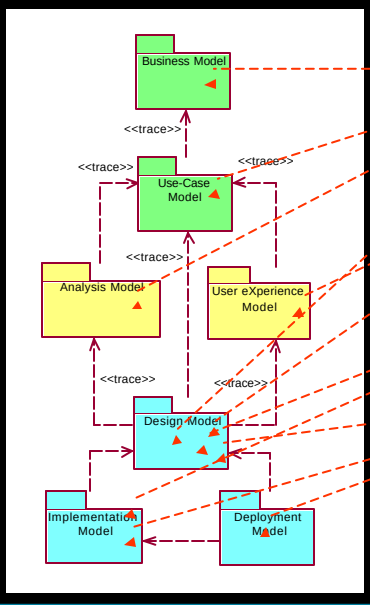
## Development environment = tools

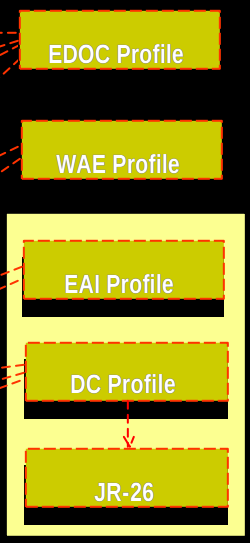



- Software assets depend on tools
- Especially assets represented at higher levels of abstraction
  - Models
  - Requirements

Rational
the e-development company

## Models and UML profiles

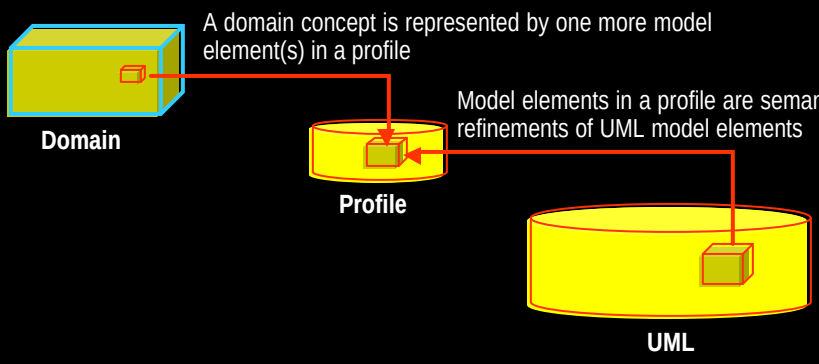



Technology Profiles

Rational
the e-development company

## Profiles tighten UML semantics

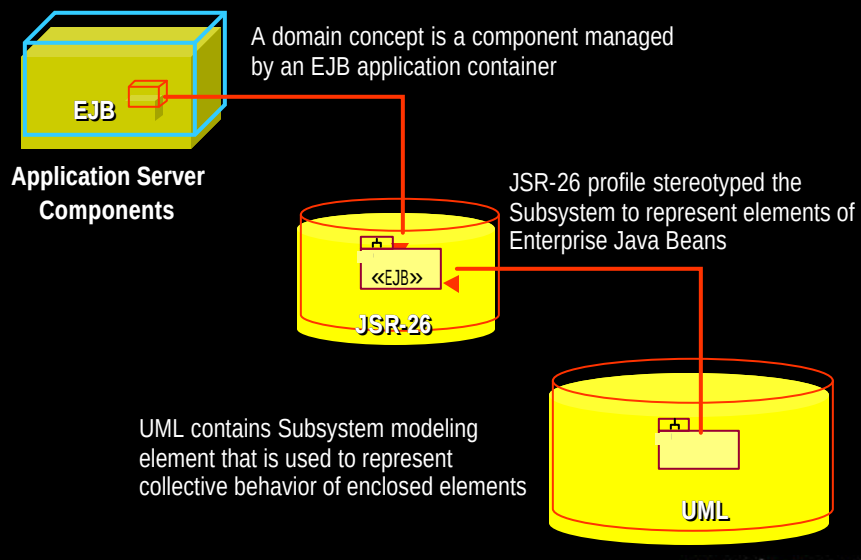



A *profile* contains model elements which have been customized for a specific domain or purpose by extending the metamodel using stereotypes, tagged definitions and constraints. A profile may specify model libraries on which it depends and the metamodel subset that it extends.

*— UML 1.4 Reference Manual*

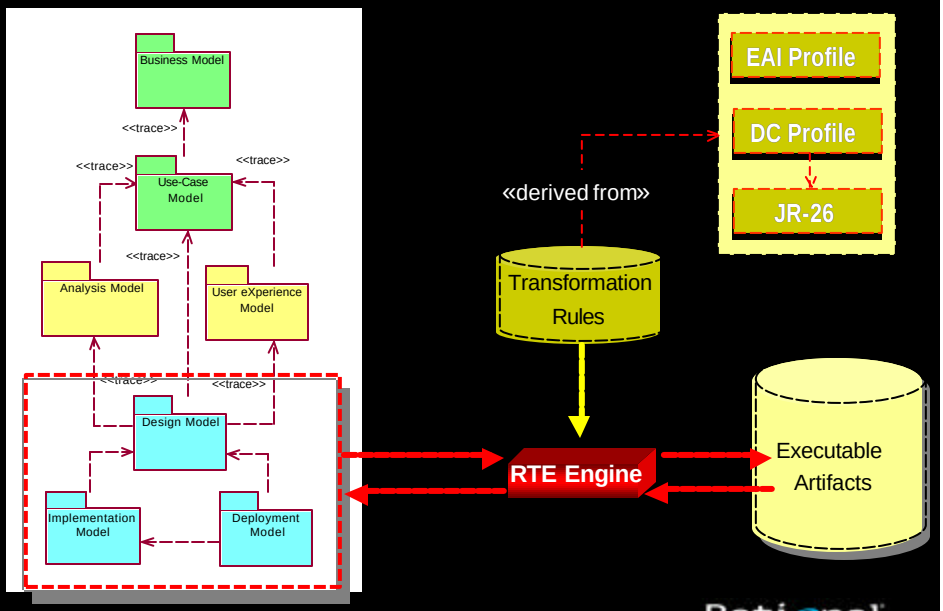
Rational the e-development company

## Example of a profile element




Rational the e-development company

## RTE and UML profiles




## Development environment dependencies

Despite all standardization (UML, Profiles, Java, J2EE, XML, SOAP, …) complex assets including requirements, models and generated artifacts are still very dependent on their development environments

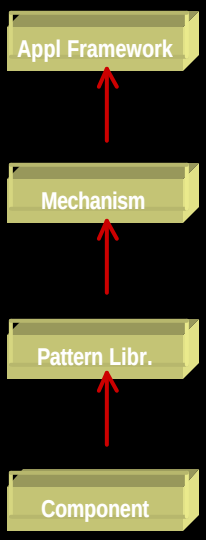
Rational
the e-development company

## Outline

- *On-line enterprise applications, a line in the design space*
- *Deployment environments*
- *Development environments*
- *Software assets*
  - *Components*
  - *Pattern*
  - *Architectural mechanisms*
  - *Application frameworks; prepackaged software architectures*
- Conclusions

Rational the e-development company

## Categories of software assets




- Customizable “starter kit” for an application family (ex: a base for on-line market systems)
- Reusable solution template for a domain-specific functionality or service (ex: user presentation request processing)
- Set of related reusable design templates (ex: GoF patterns, Sun J2EE patterns, .NET patterns)
- A reusable, plugable element of a software solution with well defined interfaces and behavior

Rational the e-development company

## Components

- A component is a self-contained software construct that has a defined use, has a run-time interface, can be autonomously deployed, and is built with foreknowledge of a specific component socket.
- A component socket is a well-defined and well-known run-time interface to a supporting infrastructure into which the component will fit.
- A component is built for composition and collaboration with other components.
- A component socket and the corresponding components are designed for use by a person with a defined set of skills and tools.

*Herzum & Sims*:
Business Component Factory: A Comprehensive Overview of Component-Based Development for the Enterprise.

Rational the e-development company

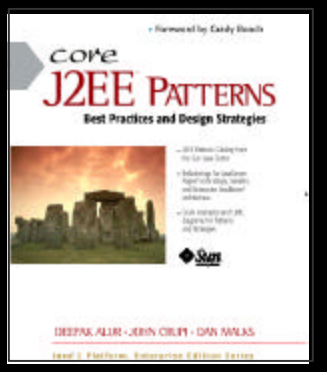
## Patterns

- Expert solutions to recurring problems
  - Codify best practices identified by experience
- Solution descriptions, not solution instances
  - Applied to create specific solution instances
- Constrain structure and behavior of participating elements
  - Variability points (placeholders) permit adaptation and customization
- Can be combined to create solution mechanisms (architecture frameworks)

Rational the e-development company

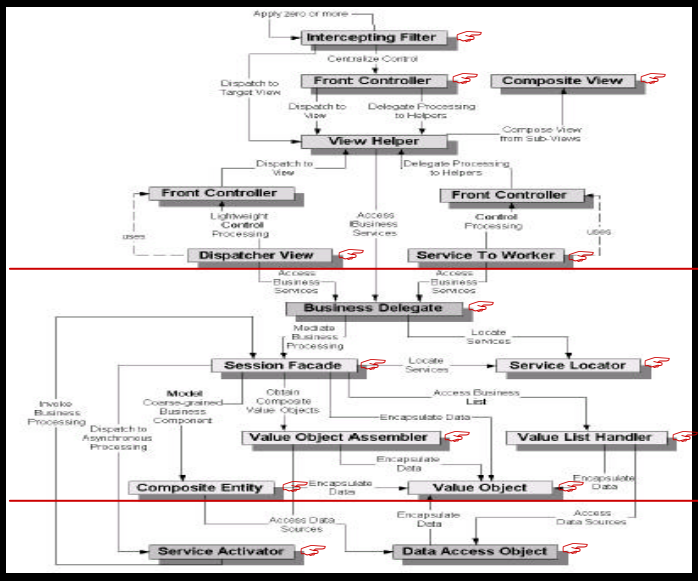
## Types of Patterns

- Can be differentiated by various factors
  - Levels of abstraction
  - Software development life cycle phases
  - Application domains
  - Technology domains
- Often grouped into pattern libraries
  - GoF patterns
  - Sun J2EE patters



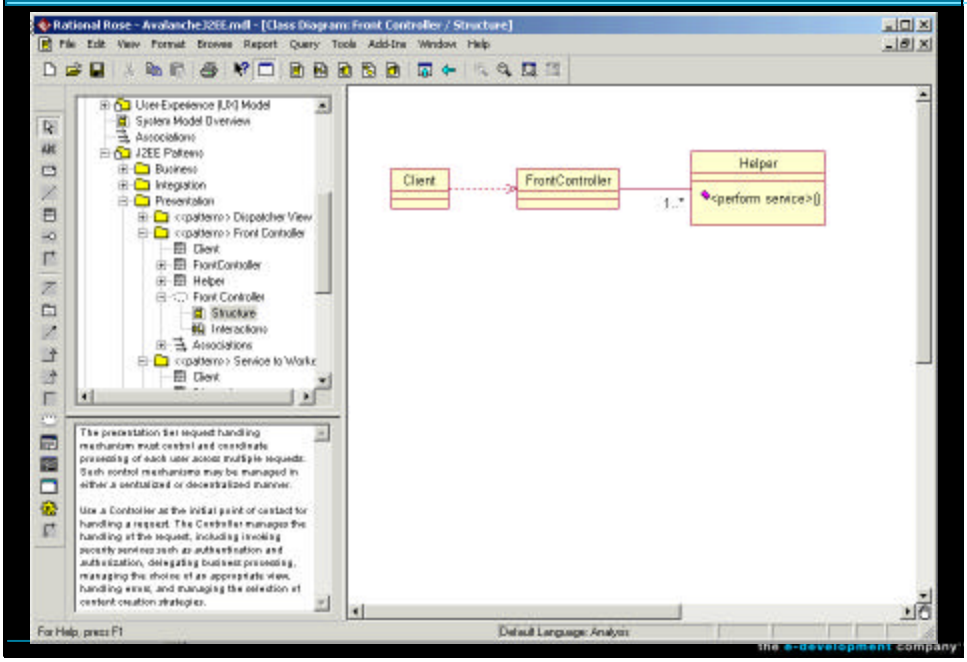

Rational
the e-development company

## The Sun J2EE pattern library

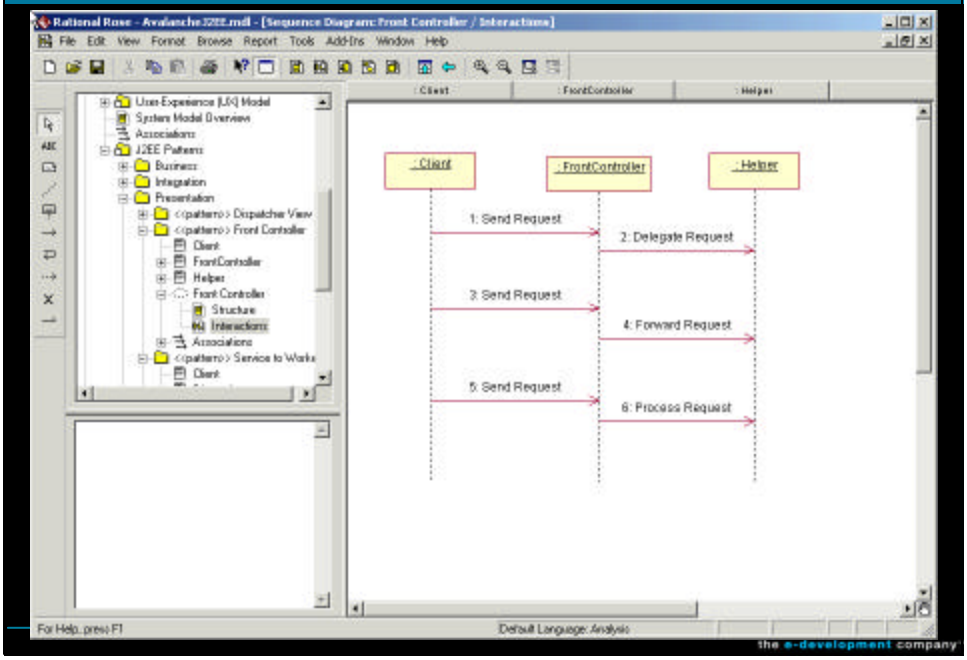



Rational
the e-development company

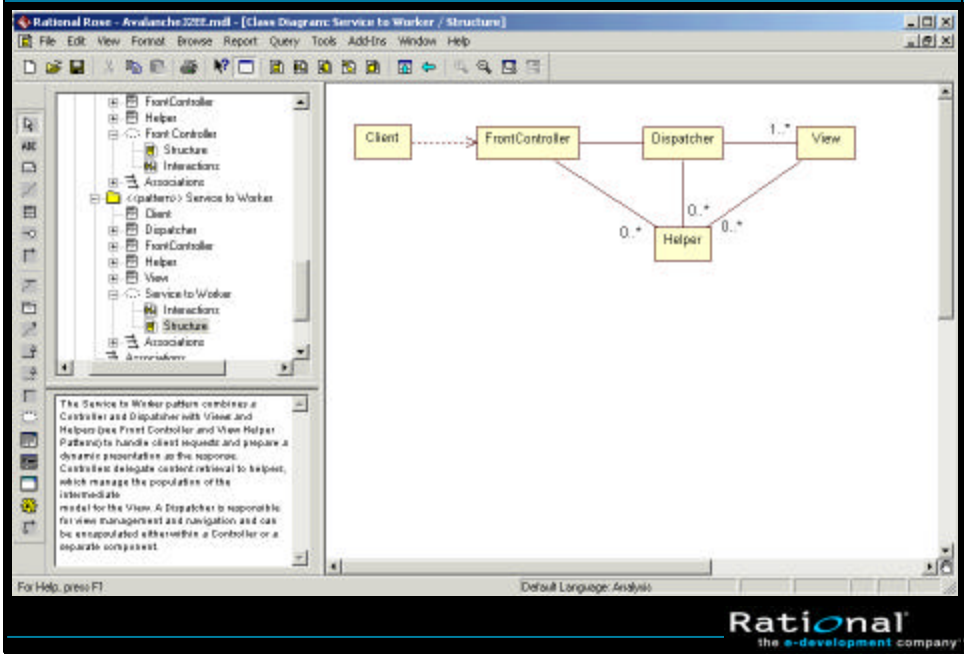
# Front Controller structure




# Front Controller interactions

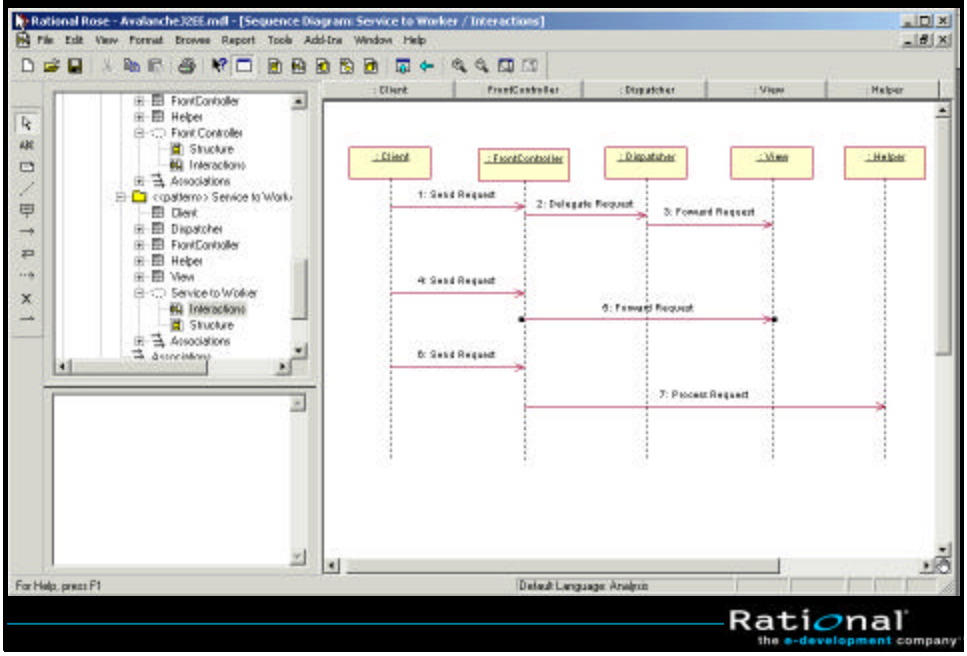


## Service to Worker structure




Rational
the e-development company

## Service to Worker interactions



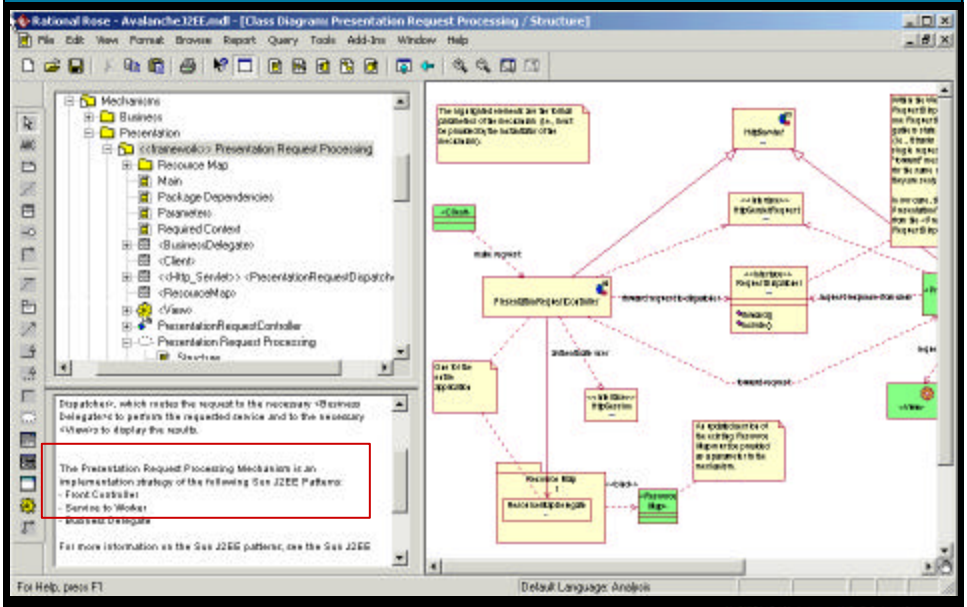

Rational
the e-development company

## Use of J2EE patterns

- The J2EE patterns have great education value
- However, they are not directly applicable
  - Each pattern can be implemented in multiple ways
    - Different implementation strategies
    - Implementation strategy must be decided before a pattern can be used
  - They are not used individually, but in groups
    - A grouping mutually constraints implementation strategies of participating patterns
- What we found directly (re)usable are architecture mechanisms (aka architecture frameworks)
  - Groupings of two or more pattern implementation strategies
  - In UML mechanisms are represented as model templates captured in «framework» package

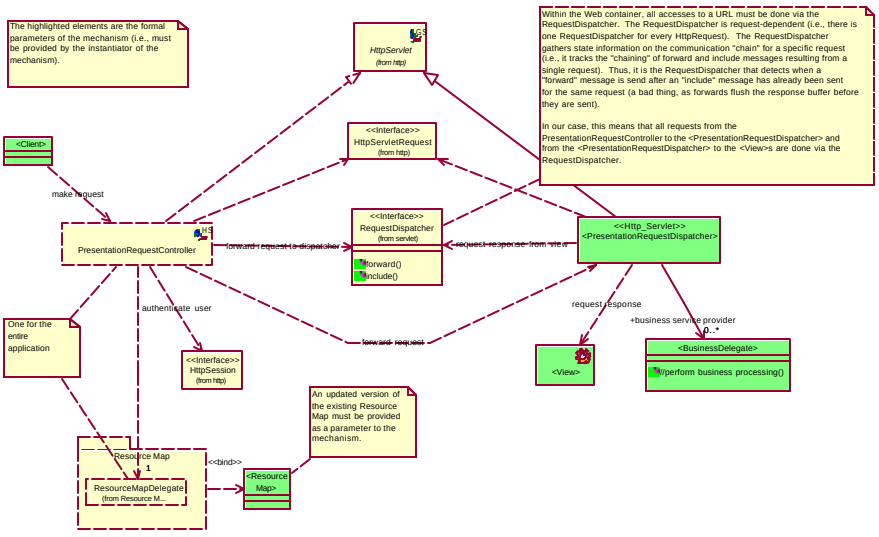
Rational the e-development company

## Presentation request processing mechanism

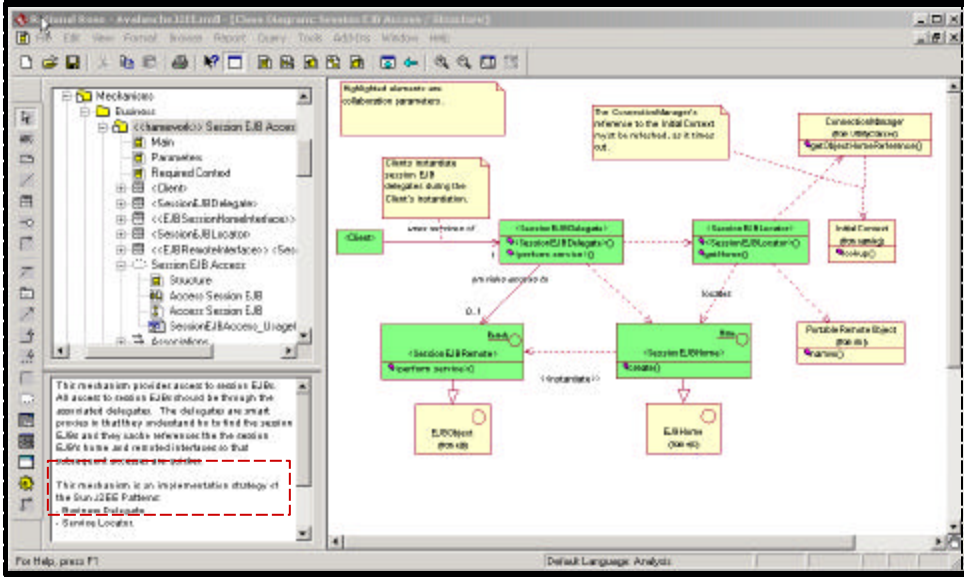



Rational the e-development company

# Mechanism is a model template




# Access to app server services mechanism

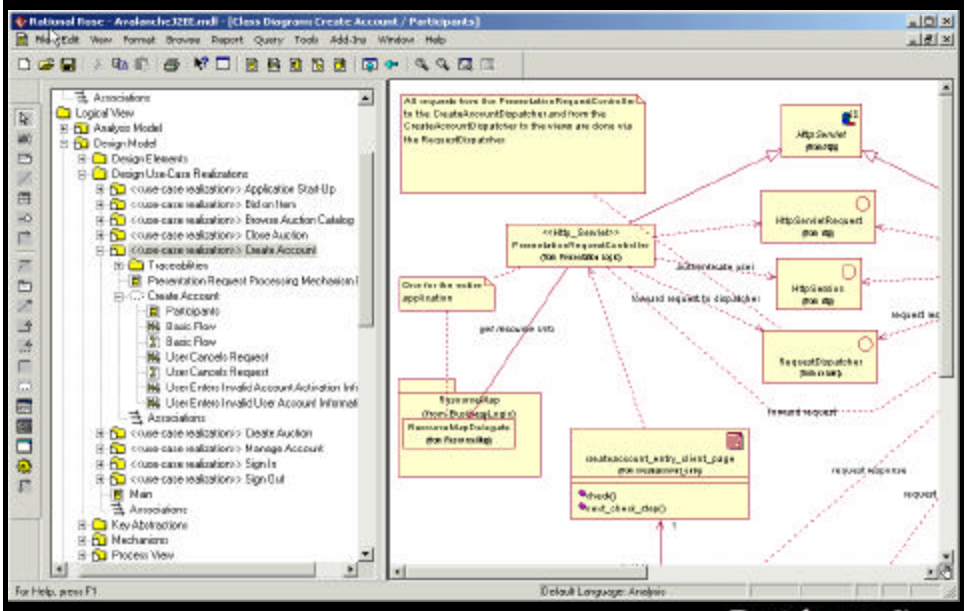


## Key architecture decisions

- There is a single entry point to the application (the front controller)
- The front controller uses the resource map to identify user's request
- The front controller authenticates users
  - Reroutes requests from new users to sign-in use-case dispatcher
- There is one dispatcher per use-case
- Dispatchers delegate business logic to application server components (EJBs) via business delegates
  - Business delegates are a separation layer between presentation and business layers of the system
- Dispatchers delegate generation of user interfaces to views
- Front controller and dispatchers are implemented as servlets
- Views are implemented as JSPs
- Delegates are implemented as Java Beans
- Business services are implemented as Session EJBs
  - Directly or as façades
- Entity Beans use CMP for state persistency

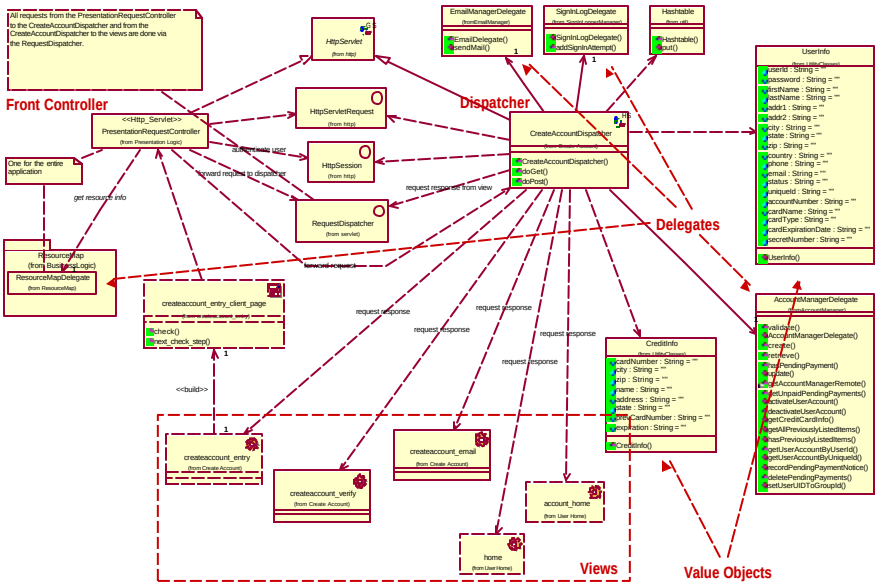
Rational the e-development company

## Consequence; all use-case realizations look the same



Rational the e-development company

## ...




## System structure becomes predictable and repeatable

- System decomposed into a few business components
  - Vertical, large-grain system "chunks" responsible for sets of related uses cases
- Each use-case maps into structurally consistent use case realization
  - One dispatcher
  - Shared delegates and services
  - Shared views
- New use-cases can be added without any impact on existing use case realizations
  - Update to resource map
  - New dispatcher
  - New services or extensions to existing services
    - Changes to existing services should be hidden by business delegates
  - New views or extensions to existing views
    - Changes to existing vies should be hidden behind view parameters

That's architecture to me!

Rational
the e-development company

## We can do better than prepackaging mechanisms

- We can prepackage application frameworks
- Application framework is an implementation of a "solution construction starter kit"
  - Requirements, models, code, deployment descriptors, documentation and more
  - Encapsulates key design decisions = architecture
  - Developers can concentrate on implementation of business use-cases
- Rational On-line Retail Application Framework includes
  - Presentation Request Processing mechanism
  - Session EJB Access mechanism
  - Front Controller
  - Resource Map service (and delegate)
  - Email service
  - Logging service
  - Systems Parameters Management service
  - Partial implementation of the User Account Management business component
    - Create Account use-case realization
    - Sign-in use-case realization

Rational the e-development company

## Outline

- *On-line enterprise applications, a line in the design space*
- *Deployment environments*
- *Development environments*
- *Software assets*
- *Conclusions*

Rational the e-development company

## In conclusion

- Yes, we can prepackage architecture-centric assets for a given point in the development space
- The most articulated form of these assets are application frameworks
- Creation of application frameworks should be driven by business, not technical concerns
  - Large investments in multiple instances of an application family
  - Common, mature deployment environment
  - Powerful, commonly-accepted development environment
  - Agreement on representation of reusable assets
- Look for application frameworks coming up from Rational, Microsoft, IBM and other companies
- Can we make $$ on prepackaging architectures?

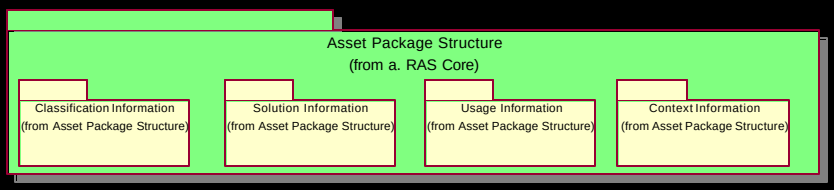
Rational
the e-development company


## Thank You

wojtek@rational.com

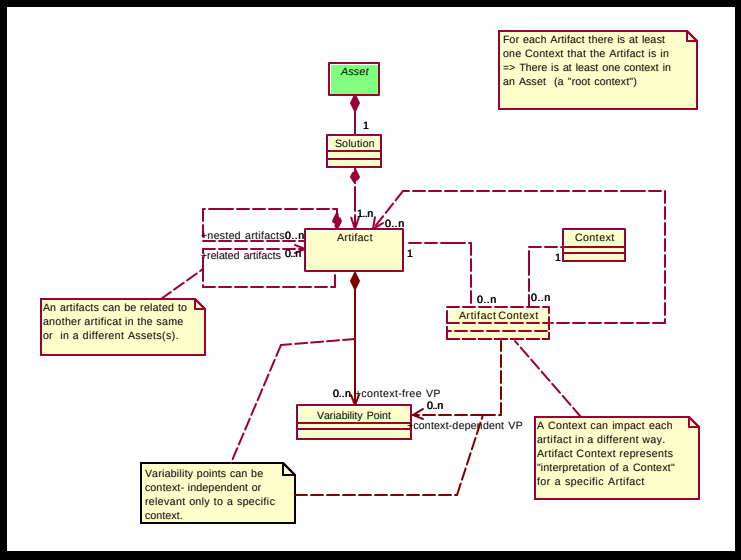
# Asset-Based Development

- Development based on economically-significant reuse of software assets
- Reusable Asset Specification (RAS) defines standard way of packaging assets




Rational
the e-development company

# Structure Of An Asset




Rational
the e-development company

# Application Of An Asset