RUP for Successful J2EE Projects

Peter Eeles
peter.eeles@uk.ibm.com
Agenda

- An Overview of RUP
  - Rational Unified Process
- An Overview of RCJD
  - RUP Configuration for Java Developers
- An Overview of applying RCJD
  - To the development of an Online Auction application
Building J2EE Applications with the Rational Unified Process

Peter Eeles
Kelli Houston
Wojtek Kozaczynski

Forewords by John Crupi and Philippe Kruchten
The RUP “Hump Chart”
RUP Key Concepts

Phase divided into Iteration considers Discipline described by Workflow Detail participates in Role performs Activity modifies responsible for Document Model Element Artifact references
Support is provided for a browser-based process.

Process configurations can be created.

Process content can be defined.

Process plugins are available from DeveloperWorks (e.g. MS®.net, J2EE, systems engineering).
Reduce Scrap/Rework: Avoid Sequential Process

Symptoms of conventional waterfall process

- Late design breakage
- 40% effort on integration and test

![Diagram showing project schedule and development progress](image)
Reduce Scrap/Rework: Use an Iterative Process

Prototypes ➔ Architecture ➔ Functional Releases ➔ Product Release

100%

Modern Project Profile

Waterfall Project Profile

Development Progress (% coded)

Project Schedule

Rational software
Attack Significant Risks Early

Risk exploration period

Risk resolution period

Controlled risk management period

- Waterfall
- Iterative

Risk Reduction

Time

Risk
RCJD

- A configuration of RUP
- Targeted at developers using the J2EE platform
- A subset of RUP
  - No Business Modeling or Deployment disciplines
- An extension of RUP
  - User-Experience Modeling
  - J2EE-specific guidance
RCJD Disciplines

Disciplines
- Inception
- Elaboration
- Construction
- Transition

Phases
- Requirements
- Analysis & Design
- Implementation
- Test
- Configuration & Change Mgmt
- Project Management
- Environment

Iterations
- Initial
- Elab #1
- Elab #2
- Const #1
- Const #2
- Const #N
- Tran #1
- Tran #2
RCJD Key Artifacts

- **Requirements**
  - Glossary
  - Supplementary Specification
  - Use-Case Model

- **Analysis and Design**
  - Data Model
  - Deployment Model
  - Design Model
  - Software Architecture Document
  - User-Experience Model

- **Implementation**
  - Implementation Model
What is a model?

“A complete description of a system from a particular perspective” [RUP]
Why do we model?

- To manage complexity
- To clearly understand different concerns
- To detect errors and omissions early in the lifecycle
- To examine the relative merits of different options
- To communicate with stakeholders
- To drive implementation
- To understand the impact of change
- To ensure that resources are deployed efficiently
What is Model-Driven Development?
Why Model-Driven Development?

- To speed up project delivery
  - In the absence of heroics, it is the fastest way to get from A to B
- To improve the quality of the delivered system
  - Each model can act as a “quality gate”
- To weave together multiple stakeholder views
  - Stakeholders have different, yet related, concerns
- To improve team working
  - Models form the basis of a “contract” between project roles
- To be more efficient
  - Reusing models (and model elements) throughout the lifecycle
PearlCircle

- An Online Auction application

![PearlCircle Online Auction application](image)
Requirements Artifacts

System Analyst
- Glossary
- Stakeholder Requests
- Storyboard
- Supplementary Specifications
- Use-Case Model

Requirements Specifier
- Software Requirement
- Software Requirements Specification
- Use Case

Software Architect
- Software Architecture Document
## Requirements – Glossary

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auction</td>
<td>A sale in which an item is sold to the highest bidder.</td>
</tr>
<tr>
<td>Auction Information</td>
<td>Information about an auction that includes start time and duration of the auction, product information (title, description, image), starting price (minimum initial bid price), minimum bid increment and auction category (the category in which the auction is listed).</td>
</tr>
<tr>
<td>Credit Card Information</td>
<td>Information about a credit card that includes the credit card number, billing address and card expiration date.</td>
</tr>
<tr>
<td>Pending Payment Notice</td>
<td>Information about a payment that the user still owes the system.</td>
</tr>
<tr>
<td>User Information</td>
<td>Information about a user that includes the user name, password and email address.</td>
</tr>
<tr>
<td>Category</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Usability</td>
<td>The system shall provide online help that can be downloaded from the auction site and installed locally, or accessed via a browser.</td>
</tr>
<tr>
<td>Availability</td>
<td>The system shall be continuously available (often referred to as 7 x 24 operation). Backup and maintenance operations shall not require system shutdown.</td>
</tr>
<tr>
<td>Performance</td>
<td>The response time for any query shall be less than 3 seconds when measured on a 100Mb local-area connection. The response time for all transactions, such as creation of an auction, shall be less than 5 seconds from the transaction data is submitted to the time transaction results are reported to the user.</td>
</tr>
<tr>
<td>Supportability</td>
<td>All errors shall be time-stamped and logged in the system error file. Exception messages should identify the system element that threw the caught exceptions.</td>
</tr>
<tr>
<td>Development and Deployment Environment</td>
<td>The application should be developed and deployed on the J2EE platform.</td>
</tr>
</tbody>
</table>
Analysis and Design Artifacts

Software Architect
- Architectural Proof-of-Concept
- Deployment Model
- Design Model
- Event
- Interface
- Signal

User-Interface Designer
- Navigation Map
- User-Experience Element
- User-Experience Model
- User-Experience Navigation Map
- User-Experience Storyboard
- User-Interface Prototype

Designer
- Analysis Class
- Design Class
- Design Subsystem
- Enterprise Java Bean (EJB)
- Use-Case Realization

Database Designer
- Data Model
Multi-tier architecture employed
Key abstractions

```uml
diagram inheritance
  «entity» Auction
  + endDateTime
  + minimumBidIncrement
  + startingPrice
  + startDateTime

  «entity» Category
  + description
  + name

  «entity» Auction Item
  + title
  + description

  «entity» Bid
  + postedBid
  + auctionPaymentCard
  + + amount
  + cancelExplanation
  + dateTime

  «entity» Credit Card
  + cardNumber
  + cardType
  + expirationDate
  + nameOnCard

  «entity» Pending Payment
  + amount

  «entity» User Account
  + address
  + city
  + country
  + email
  + password
  + state
  + zip
  + username

relationship
  1:1 Auction to Auction Item
  1:* Auction to User Account
  1:* Bid to User Account
  1:* Credit Card to User Account
  0..1 Auction Payment Card to Credit Card
  0..1 User Account to User Account
```

Rational software
Key screens (and input forms)

- Sign in
  - Sign in status msg

- Sign in form
  - Password
  - User name

- Item detail
  - Auction end date
  - Auction start day
  - Auction start time
  - Bid status message
  - Category name
  - Description
  - Highest bid
  - Image available
  - Item title
  - Minimum bid increment
  - Number of bids
  - Seller's name
  - Starting price
## Analysis mechanisms

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Verifies that the user has the credentials to access the system</td>
</tr>
<tr>
<td>Authorization</td>
<td>Makes sure that a user requesting specific system services is authorized to access and use those services</td>
</tr>
<tr>
<td>Messaging</td>
<td>Sends email messages to the system users</td>
</tr>
<tr>
<td>Persistency</td>
<td>Stores system state</td>
</tr>
<tr>
<td>Presentation Request Processing</td>
<td>Handles user requests to the system made over the web interface</td>
</tr>
<tr>
<td>System Parameter Management</td>
<td>Handles external parameters</td>
</tr>
</tbody>
</table>
Use-Case Realization (Basic flow)

1: // enter bid info (bidInfo)
2: // create bid
3: // create bid (bidInfo)
4: // has pending payment
5: // create bid (bidInfo)
6: // validate bid
7: /* create */
8: // get email address
9: // send email

The Buyer enters a bid on the auction item
The Buyer selects the option to create a bid
The form passes the request to the controller
The controller determines if the Buyer has any pending payments outstanding (since the Buyer may also have been a Seller in another auction)
The controller passes the request on to the appropriate auction object
The auction validates and creates an associated bid
The Buyer’s email address is retrieved
An email is sent to the Buyer notifying them that their bid has been successful
Use-Case Realization (Participants)

- **Auction**
  - endDateTime
  - minimumBidIncrement
  - startingPrice
  - startDateTime
  - mark as closed()
  - get payment info()
  - get highest bid()
  - get seller()
  - get details()
  - validate bid()
  - cancel bid([in] bid,[in] explanation)
  - create bid([in] bidInfo)

- **Bid**
  - amount
  - cancelExplanation
  - date
  - get bid amount()
  - get buyer()
  - set cancel explanation([in] explanation)

- **User Account**
  - address
  - city
  - country
  - email
  - password
  - state
  - zip
  - username
  - get email address()
  - has pending payment()
  - get open bids()
**User-Experience Storyboard (Screen flows)**

1. **Buyer**: The Buyer indicates that he/she would like to place a bid on a displayed item.
2. **Place Bid**: The system displays the place bid form. The Buyer enters the bid information and submits.
3. **Submit Bid**: If the bid is valid, the system displays the place bid results screen with the bid information displayed.
4. **Navigate**: If the bid is not valid, the system returns to the place bid screen and displays a message.
5. **Navigate**: 5: navigate to ( )
6. **Display Invalid Bid**: 6: display invalid bid msg ( )
User-Experience Storyboard (Participants)

- auction closed
- item detail
  - auction end date
  - auction start day
  - auction start time
  - bid status message
  - category name
  - description
  - highest bid
  - image available
  - item title
  - minimum bid increment
  - number of bids
  - seller's name
  - starting price
  - display auction closed msg
  - display bid cancelled msg
  - navigate to
  - place bid
  - display image

- place bid
- bid canceled
- input form
  - bid information form
    - bid amount
    - cancel bid
    - reset
    - submit bid

- invalid bid submitted
- valid bid submitted
- place bid results
  - bid amount
  - bidder's name
  - email account
  - item name
  - navigate to
### Design (and Implementation) Mechanisms

<table>
<thead>
<tr>
<th>Analysis Mechanism</th>
<th>Design Mechanism</th>
<th>Implementation Mechanism</th>
</tr>
</thead>
<tbody>
<tr>
<td>Authentication</td>
<td>Based on password and user ID</td>
<td>The application authenticates the user ID and password against those stored with the relevant user account</td>
</tr>
<tr>
<td>Authorization</td>
<td>Security role-based authorization</td>
<td>J2EE-provided security mechanism</td>
</tr>
<tr>
<td></td>
<td>Use case-based authorization</td>
<td>The application verifies that a specific user can perform a use case</td>
</tr>
<tr>
<td>Messaging</td>
<td>Mail-based</td>
<td>Java Mail API</td>
</tr>
<tr>
<td>Persistency (user session)</td>
<td>Container-managed session state</td>
<td>HTTP Session managed by the web container</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Stateful Session EJB managed by the EJB container</td>
</tr>
<tr>
<td>Persistency (application)</td>
<td>Container Managed Persistence (CMP)</td>
<td>J2EE-provided mechanism</td>
</tr>
<tr>
<td></td>
<td>Bean-Managed Persistence (BMP)</td>
<td>JDBC API</td>
</tr>
<tr>
<td>Presentation Request</td>
<td>Front Controller J2EE design pattern</td>
<td>The controller implemented as a Servlet</td>
</tr>
<tr>
<td>Processing</td>
<td></td>
<td>The controller implemented as a JSP</td>
</tr>
<tr>
<td>System Parameter Management</td>
<td>Parameters are stored externally and read by the system</td>
<td>Parameters are stored in an XML file and internalized once when the application starts up</td>
</tr>
</tbody>
</table>
J2EE Patterns

- **Presentation Layer**
  - Composite View
  - Front Controller
  - Service to Worker
  - View Helper

- **Business Layer**
  - Business Delegate
  - Service Locator
  - Session Façade
  - Value Object
Artifact Granularities

- **Design Subsystem**
  - Used to model an “Enterprise Component”
- **Enterprise Java Bean**
  - For representing EJBs
- **Design Class**
  - Used to model JSPs, servlets and Java classes
Analysis and Design – Design Subsystem

- Enterprise Components
E.g. Auction Manager elements
- SessionFaçade and BusinessEntity EJBs
- Represented as a UML 1.4 component
Analysis and Design – Design Class

- E.g. Auction Manager SessionFacade elements
Create Auction use-case realization

1: handle (request, response)

2: handleNewAuction (request, response)

3: getCategory (uid)

4: getPendingPaymentBalance (userUid)

5: configureTemplate (request, title, page)

6: handle (request, response)

7: handleConfirmAuction (request, response)

8: getCreditCard (userUniqueId)

9: configureTemplate (request, title, page)
Analysis and Design - Data Model

Auction Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>AUCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ «Column» CATEGORYUID : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» CREDITCARDACCOUNT : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» CREDITCARDEXPIRATION : BIGINT</td>
<td></td>
</tr>
<tr>
<td>+ «Column» CREDITCARDNAME : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» CREDITCARDTYPE : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» ENDTIME : BIGINT</td>
<td></td>
</tr>
<tr>
<td>+ «Column» ITEMUID : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» MINIMUMBIDINCREMENT : REAL</td>
<td></td>
</tr>
<tr>
<td>+ «Column» SELLERUID : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» STARTINGPRICE : REAL</td>
<td></td>
</tr>
<tr>
<td>+ «Column» STARTTIME : BIGINT</td>
<td></td>
</tr>
<tr>
<td>+ «Column» STATUS : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» TIMEZONEUID : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>- «Column» UID : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Primary Key» AUCTIONPK ( )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>BID</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ «Column» AMOUNT : REAL</td>
<td></td>
</tr>
<tr>
<td>+ «Column» AUCTIONUID : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» BUYERUID : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» CANCELEXPLANATION : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» DATE : BIGINT</td>
<td></td>
</tr>
<tr>
<td>+ «Column» STATUS : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>- «Column» UID : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Primary Key» BIDPK ( )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>CATEGORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ «Column» DESCRIPTION : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» NAME : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>- «Column» UID : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Primary Key» CATEGORYPK ( )</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table</th>
<th>ITEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ «Column» DESCRIPTION : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» NAME : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>- «Column» UID : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Column» URL : VARCHAR(250)</td>
<td></td>
</tr>
<tr>
<td>+ «Primary Key» ITEMPK ( )</td>
<td></td>
</tr>
</tbody>
</table>
Analysis and Design – Design Model

- Process view

```
<process> Web Browser

<table>
<thead>
<tr>
<th>BrowseAuctionCatalog Dispatcher</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuctionManagerDelegate</td>
</tr>
<tr>
<td>&lt;ServerPage&gt; browse_auction_catalog.jsp</td>
</tr>
</tbody>
</table>

<process> Web Server

<process> Application Server

<EJBRemoteInterface> AuctionManager
```
Analysis and Design – Deployment Model

Diagram:
- Workstation
- Web Server
- EJB Server
- User Accounts Server: EJB Server
  - Memory: 1 Gb
  - Disk: 20 Gb
Implementation Artifacts

- Integrator
  - Build
  - Integration Build Plan
- J2EE Application
- J2EE Module

- Implementer
  - Developer Test
  - Implementation Element
- Software Architect
  - Implementation Model
  - Software Architecture Document
- Elements deployed in an EJB container
  - EJB elements, Java classes
  - Organized within Java packages
- Java elements deployed in a web container
  - Servlets, Java classes
  - Organized within Java packages
- Other elements deployed in a web container
  - JSPs, image files, HTML pages
  - Organized within directories relative to a “virtual directory”
- Deployment elements
  - J2EE application (EAR)
  - J2EE modules (WAR, EJB-JAR)
  - Deployment descriptors
- Virtual Directory Elements (web container)
- Java Code Elements (web container)
Implementation – Implementation Model

- Java Code Elements (EJB container)
Deployment Elements

- Model Explorer
  - (PearlCircle) Deployment Support
    - Implementation View: Deployment Support Elements
      - Module Dependencies
      - PearlCircle Application Module
      - PearlCircle Business Module
      - PearlCircle Presentation Module
    - «EAR» pearlcircle.ear
    - «EJB-JAR» pearlcirclebusiness.jar
    - «WAR» pearlcirclepresentation.war

- Model Diagram
  - «EAR» pearlcircle.ear
  - «WAR» pearlcirclepresentation.war
  - «EJB-JAR» pearlcirclebusiness.jar
The “RUP Configuration for Java Developers” (RCJD) is
- A subset of RUP
- An extension of RUP

Specifically aimed at developers using the J2EE platform
Questions

Building J2EE Applications with the Rational Unified Process

Peter Eeles
Kelli Houston
Wojtek Kozaczynski

Forewords by John Crupi and Philippe Kruchten

Rational Software
Thank You