RUP for J2EE

Peter Eeles
peter.eeles@uk.ibm.com

Rational software
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
The RUP “Hump Chart”

Disciplines:
- Business Modeling
- Requirements
- Analysis & Design
- Implementation
- Test
- Deployment
- Configuration & Change Mgmt
- Project Management
- Environment

Phases:
- Inception
- Elaboration
- Construction
- Transition

Iterations:
- Initial
- Elab #1
- Elab #2
- Const #1
- Const #2
- Const #N
- Tran #1
- Tran #2
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

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

Phases:
- Inception
- Elaboration
- Construction
- Transition

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
RCJD Models

Use-Case Model → realized by → Design Model

Design Model → refined into → Implementation Model

Implementation Model → deployment defined in → Deployment Model

Refined into

User-Experience Model

Refined into

Data Model
PearlCircle

- An Online Auction application
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

Rational software
## 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>
</tbody>
</table>
The Close Auction Use Case is internally-initiated.
<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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
  - Software Architecture Document

- **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
Analysis (and Design) – Design Model

- Multi-tier architecture employed
Key abstractions
Analysis (and Design) – User-Experience Elements

- Key screens (and input forms)

```plaintext
<<screen>>
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

<<screen>>
sign in
+ sign in status msg

<<input form>>
sign in form
+ password
+ user name
```
## Analysis (and Design) – Software Architecture Document

- **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>
Analysis (and Design) – Design Model

- 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 ( )
10: // Bid

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
Analysis (and Design) – Design Model

- Use-Case Realization (Participants)

```
<boundary>
Place Bid UI
+ // enter bid info ([in] bidInfo)
+ // create bid ()
</boundary>

<boundary>
Email Service Interface
+ // send email ()
</boundary>

<control>
Place Bid Controller
+ // create bid ([in] bidInfo)
</control>

<entity>
Auction
+ endDate
+ minimumBidIncrement
+ startingPrice
+ startTime
+ // mark as closed ()
+ // get payment info ()
+ // get highest bid ()
+ // get seller ()
+ // get details ()
+ // validate bid ()
+ // create bid ([in] bidInfo)

1
- posted bid

<entity>
Bid
+ amount
+ cancelExplanation
+ dateTime
+ get bid amount ()
+ get buyer ()
+ get buyer ()
+ get cancel explanation ([in] explanation)

* - buyer

1

<entity>
User Account
+ address
+ city
+ country
+ email
+ password
+ state
+ zip
+ username
+ // get email address ()
+ // has pending payment ()
+ // get open bids ()

* - seller
```
The Buyer indicates that he/she would like to place a bid on a displayed item.

The system displays the place bid form. The Buyer enters the bid information and submits.

If the bid is valid, the system displays the place bid results screen with the bid information displayed.

If the bid is not valid, the system returns to the place bid screen and displays a message.
Analysis (and Design) – User-Experience Model

- User-Experience Storyboard (Participants)
Auction management
## J2EE Architecture

<table>
<thead>
<tr>
<th>Client Device</th>
<th>HTTP</th>
<th>Presentation Tier</th>
<th>Server</th>
<th>Business Tier</th>
<th>Integration Tier</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML, XML, WML</td>
<td>HTTPS</td>
<td>Web Server</td>
<td>EJB Server</td>
<td>JNDI</td>
<td>JMS</td>
</tr>
<tr>
<td>Applet Container</td>
<td>HTTPS</td>
<td>JSP, Servlet</td>
<td>EJB Container</td>
<td>Directory Service</td>
<td>Message Queue</td>
</tr>
<tr>
<td>Applet</td>
<td>HTTPS</td>
<td>J2EE Services, J2SE Services</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Client Container</td>
<td>HTTPS</td>
<td>Application Client</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application Client</td>
<td>HTTPS</td>
<td>J2EE Services, J2SE Services</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## 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>Persistency (application)</td>
<td>Container Managed Persistence (CMP)</td>
<td>Stateful Session EJB managed by the EJB container</td>
</tr>
<tr>
<td></td>
<td>Bean-Managed Persistence (BMP)</td>
<td>J2EE-provided mechanism</td>
</tr>
<tr>
<td>Presentation Request Processing</td>
<td>Front Controller J2EE design pattern</td>
<td>The controller implemented as a Servlet</td>
</tr>
<tr>
<td></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
J2EE Patterns – Business Layer

Client

Web Container

<HttpServlet>
FrontController

UseCaseDispatcher

CompositeViewTemplate

CompositeViewElement

Enterprise Component

BusinessDelegate

ServiceLocator

EJB Container

EJBSessionBean
SessionFacade

ValueObject

EJBEntityBean

BusinessEntity
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
Analysis and Design – Design Subsystem

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

- E.g. Auction Manager Session Façade 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 (userUniqueUid)
9: configureTemplate (request, title, page)
Analysis and Design - Data Model

Auction Tables

- **Table**
  - **AUCTION**
    - + «Column» CATEGORYUID : VARCHAR(250)
    - + «Column» CREDITCARDACCOUNT : VARCHAR(250)
    - + «Column» CREDITCARDEXPIRATION : BIGINT
    - + «Column» CREDITCARDNAME : VARCHAR(250)
    - + «Column» CREDITCARDTYPE : VARCHAR(250)
    - + «Column» ENDTIME : BIGINT
    - + «Column» ITEMUID : VARCHAR(250)
    - + «Column» MINIMUMBIDINCREMENT : REAL
    - + «Column» SELLERUID : VARCHAR(250)
    - + «Column» STARTINGPRICE : REAL
    - + «Column» STARTTIME : BIGINT
    - + «Column» STATUS : VARCHAR(250)
    - + «Column» TIMEZONEUID : VARCHAR(250)
    - - «Column» UID : VARCHAR(250)
    - + «Primary Key» AUCTIONPK ( )

- **Table**
  - **BID**
    - + «Column» AMOUNT : REAL
    - + «Column» AUCTIONUID : VARCHAR(250)
    - + «Column» BUYERUID : VARCHAR(250)
    - + «Column» CANCELEXPLANATION : VARCHAR(250)
    - + «Column» DATE : BIGINT
    - + «Column» STATUS : VARCHAR(250)
    - - «Column» UID : VARCHAR(250)
    - + «Primary Key» BIDPK ( )

- **Table**
  - **CATEGORY**
    - + «Column» DESCRIPTION : VARCHAR(250)
    - + «Column» NAME : VARCHAR(250)
    - - «Column» UID : VARCHAR(250)
    - + «Primary Key» CATEGORYPK ( )

- **Table**
  - **ITEM**
    - + «Column» DESCRIPTION : VARCHAR(250)
    - + «Column» NAME : VARCHAR(250)
    - - «Column» UID : VARCHAR(250)
    - + «Column» URL : VARCHAR(250)
    - + «Primary Key» ItemPK ( )
Analysis and Design – Design Model

- Process view
Analysis and Design – Deployment Model

- 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
Implementation – J2EE Implementation Element Categories

- **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
Implementation – Implementation Model

- Virtual Directory Elements (web container)
- Java Code Elements (web container)
Implementation – Implementation Model

- Java Code Elements (EJB container)
Implementation – Implementation Model

- Deployment Elements

![Diagram showing deployment elements and the relationship between EAR, EJB-JAR, and WAR files](image)
Summary

- The “RUP Configuration for Java Developers” (RCJD) is
  - A subset of RUP
  - An extension of RUP
- Specifically aimed at developers using the J2EE platform
IBM Software Group

Questions?

Rational software

@business on demand software