OUTLINE

● Introduction
● Implementation Phase
  ● Phase 2 Design
  ● Process
  ● Implementation Overview
  ● Phase 3 Implementation Design
  ● Issues
  ● Features
  ● Demo
● Project Plan
● Test Plan
● Conclusions
● References
Introduction

- KSU Student Portal
  - Web Application
  - Data Bound
  - Heavyweight Approach
- Phase 3: Project Implementation
- Bottom-Up Approach
  - Data Entities
- Top-Down Approach
  - Facade Methods
Phase 2 Design

Use the Information provided by the Design and proceed with the system implementation.

More Details will be added to the application.
Process

- Phase 1:
  1. Requirements Analysis
  2. Use Case Diagram
  3. Use Case Specification
  4. E-R Diagram
Process

- Phase 2:
  1. Use Case Diagram Revision
  2. Sequence & Collaboration Diagrams
  3. Class Diagram
  4. E-R Revision
  5. UML Data Model
  6. Relational Schema
  7. UML Design Revision
  8. Formal Specification
Process

- Phase 3:
  1. Implement Relational Schema in Database
  2. Create Entity Beans From Tables
  3. Implement Relations Between the Entity Beans
  4. Write initial EJB-QL Queries
  5. Create Session Beans
  6. Create Model Facade
  7. Publish Methods
  8. Test Methods
Implementation Overview

- There are more Session Beans than Control Classes from Design
- Load Balance
- Loosy Coupled
- Session Beans perform the business logic accessing to the Entity Beans
- Model Facade is the entry point to the system.
  - Model Controller
  - No Business logic, it relies on the EJBs
  - Session Facade, Business Delegate
Implementation Overview

- Entity Beans
  - AppUser
  - Blog, Blog Entry
  - Message
  - Article, Event, Link
  - Profile, Filter
  - Course, Language, Country, State
Implementation Overview

- **Session Beans:**
  - `AppUserSessionBean`
  - `BlogSessionBean`
  - `MessageSessionBean`
  - `ArticleSessionBean`, `EventSessionBean`, `LinkSessionBean`
  - `FilterSessionBean`
  - `UtilSessionBean`

- **Model Facade**
Issues

- Object Relationships
  - OO Schema = Relational Schema
- Lazy vs. Eager Fetching
- Handling Null Values (Filter)
- EJB-QL (Avoid Using SQL => Portability)
Features

- Distribute System
- Layers Pattern
- Database Independent: ORM + EJB-QL
- Safe (No Arrays) and clean code.
- New Technologies: EJB 3.0, Java 5
- Portable
- Pure OO
- Security: Encrypted passwords, code analyzers
- View: MVC, JSF + ADF Components, JavaScript, CSS, Multiple Languages…
Demo
Project Plan

- **What has been done**
  - Requirements Analysis: 100%
  - System Design: 100%
  - Implementation Design: 70%
  - Coding:
    - Database Layer: 100%
    - Model Layer: 85%
    - View Layer: 2%
    - Client layer: 0%
Project Plan

- What remains to be done
  - **May**: Finish Model, Start View
  - **June**: View (JSP pages)
  - **July**: Testing, Documentation, Enhance View
Test Plan

- System was build with testing in mind.
- Safe environment.
- Test Plan:
  - Unit Testing: Junit
  - Integration Tests
  - Other Tools (FindBugs)
Conclusions

**Goals**

- Mimic a real world enterprise application.
- Full software Cycle.
- Test New Technologies.
- Learn J2EE Platform.
- Solve problems using Javadoc and specifications.
Conclusions

- **EJB 3.0**
  - Easier to use than EJB 2.0.
  - Perform the most tedious work for you (network, security issues).
  - Standard.
  - Portable.
  - Immature, lack of documentation
  - Different interpretations of the specification.
  - Still difficult to use.
Conclusions

- JSF
  - Drag & Drop UI Design for Web Applications.
  - Easy to use.
  - MVC pattern.
  - Validation, Internationalization
  - Immature
  - MVC not as pure as with Struts
References

- www.uml.org
- www.rational.com
- Applying UML and Patterns - An Introduction to Object-Oriented Analysis and Design and the Unified Process, Craig Larman
- Java Server Faces (2004), Hans Bergsten