Object-Oriented Development - Use-Case Realization Analysis

Prof. Dr. U. Aßmann

Technische Universität Dresden
Institut für Software- und Multimediotechnik
Gruppe Softwaretechnologie
http://www-st.inf.tu-dresden.de
Literature

OO Development – An Example

- Linköping University: PUM software engineering course
- PUM-13 .pum group in the course 2001/02
  - Customer: Nexus, Linköping
- Goal: Course management system for employees of Nexus
  - Web-based
  - Course management
  - Course registration
  - Course exercises
- Chosen method of development: object-oriented, in particular use-case realization analysis
Other Constraints

► Windows environment
  ▪ .NET: MS IE with ASP.net page technology, C#
  ▪ ADO.NET, with nexus internal database

► The system is clearly an interactive system that relies on persistent course data.

► Hence, an early decision can be taken: it is a system with 4-tier (BCED) architectural style
The Domain Model of the Web-Based Course System

- **Pupil**
  - **CourseStatus**
    - beginDate
    - endDate
    - ready
    - resultProcent

- **Education**
  - name
  - description
  - lastChanged

- **Teacher**

- **Course**
  - name
  - description
  - lastChanged
  - changedBy
  - active
  - **Module**
    - name
    - description
    - lastChanged
    - changedBy
    - active

- **Link**
  - name
  - description
  - URL

- **ModuleStatus**
  - endDate
  - ready

- **QuestionStatus**
  - status

- **Question**
  - category
  - text

- **Answer**
  - category
  - text

- **Alternative**
The User in the Domain Model

- Users must occur as actors in the UCDs also
Use Case Analysis
Development From Use Cases

- First a top-level UCD (elaboration), using the terms of the domain model
- Then, refinement of Use Cases to more detailed Use Cases
  - Restructure Use Cases
  - Use-case realization analysis: derive communication diagrams from use cases until you find direct actions of classes
  - Actors become active classes
  - Actions become either classes (reified methods) or methods
Top-Level UCDs

- **User**
  - Login

- **CourseOwner**
  - Create New Course

- **Pupil**
  - Answer Questions

- **CourseEditor**
  - Modify Course
Refinement of Editing Use Cases

Course Owner

Course

Module

New Module

Change Module

Remove Module

Edit Information Link

Create New Course

Edit Information Link

Create Editor

Delete Editor

Transfer Ownership

Course Editor

Question

New Question

Change Question

Remove Question

Edit Information Link

Activate Course

Deactivate Course

Edit Course Info

Edit Information Link
Refinement of Course Studies Web Use Cases
Use Case Realization Analysis
Use-Case Realization Analysis

- Slicing of use cases
  - Grouping a scenario around an actor
- Derive communication diagrams between actors and system parts
  - Transform the use cases into communication diagrams, showing the control flow through the parts of the system
- Also possible: transform the use cases into message sequence charts (sequence diagrams)
Use Case Realization Analysis with “Answer Questions”

1: Submit Answer
2: Transmit Answer
3: Get correct answers
4: Record answers
5: Check if correct
6: [If all correct] Set module status to completed
7: Check if all modules correct
8: [If all modules correct] Set course status to completed
Use Case Realization Analysis with “LoggingIn”

1: Submit username passwd
2: Transmit username passwd
2.1: authenticate user
2.2: userid
3: record a cookie (session identifier)

User

Login Page

User Manager

User Database

User DAO

ASP Session
Use Case Realization Analysis with "CreateNewCourse"

1. Create new course
   1.1: new course
   1.2: new course id
   1.3 [no ids in cache] get 100 new ids
   1.4: return new id
   1.5 create new, empty course
   1.6: return course set
   1.7 get course data
   1.8 display new page

- **CourseOwner**
- **Create NewCourse**
- **Edit Course Page**
- **Course Manager**
- **Course DAO**
- **Course Database**
- **Current Course Set**
Use Case Realization Analysis with “CreateNewCourse”

Course Owner

2: submit course data
2.1 set course data
2.2 setCourse(CourseSet)
2.3 get current userid
2.4 get course owner
2.5 setCourse(CourseSet)
2.5.1 get data
2.6 display new page

Edit Course Page

User Manager

2.4.1 get course owner
2.5.2 set course data

Course DAO

Course Database

Current Course Set

<<entity>>

Course Owner

<<entity>>
Use Case Realization Analysis with “RequestClassStatus”

Teacher

1: Request Class Status
2: get class info
3: get class data
4: get class results
5: get user names
6: get user ids
7: check that user is a teacher

Course Owner

2.6 display new page

Teacher Class View

Class Manager

User Manager

User Database

Classes DAO

Results DAO

User DAO

Teacher

Request Class Status
Use Case Realization Analysis with “ChangeView”

1. Request View Change
2. http get: change view
3. look up template for view
4. return template
5. forward to template
6. include
7. include
8. return new view
9. display new view
Remarks

► In all UCD analyses, for the elements of the domain, *managers* have been introduced
  ▪ user manager
  ▪ course manager
  ▪ module manager
  ▪ question manager

► They belong to the *application logic* (*business logic*) of the application (business logic layer)

► It results a clear 4-layer architecture for all communication diagrams
BCED in all Use Case Realization Analyses

Presentation (Boundary)

Application Logic (Control)

Data Access (Middleware, Entity)

Database

Course Owner

Edit Course Page

2: submit course data

2.6 display new page

User Manager

2.3: get current userid

2.4: get course owner

2.5: setCourse(CourseSet)

Course Manager

2.2: setCourse(CourseSet)

2.1 set course data

Course Database

<<database>>

Current Course Set

<<entity>>

Course Owner

Course DAO

<<entity>>

Edit Course Page

2.5.2: set course data

2.4.1: get course owner

Presentation (Boundary)

Application Logic (Control)

Data Access (Middleware, Entity)

Database

Course Owner

Edit Course Page

2: submit course data

2.6 display new page

User Manager

2.3: get current userid

2.4: get course owner

2.5: setCourse(CourseSet)

Course Manager

2.2: setCourse(CourseSet)

2.1 set course data

Course Database

<<database>>

Current Course Set

<<entity>>

Course Owner

Edit Course Page

2: submit course data

2.6 display new page

User Manager

2.3: get current userid

2.4: get course owner

2.5: setCourse(CourseSet)

Course Manager

2.2: setCourse(CourseSet)

2.1 set course data

Course Database

<<database>>

Current Course Set

<<entity>>
Coarse Grain Structure: Packages and Layers

- Hence, a classical layered 4-tier architecture results in the packaging
  - With acyclic USES relationship
- Package Structure Top Level

![Diagram showing a 4-tier architecture with Package Structure Top Level]
4 Subsystems

- Course Management Web
- Course Studies Web
- Classes Web
- Authentication Web
- Helper subsystem: view management that allows for changing views on pages
First Refinement

Presentation
- Presentation CourseMgmtWeb
- Presentation CourseStudiesWeb
- Presentation ClassWeb
- Presentation AuthenticationWeb
- Presentation ViewMgmt

Application Logic
- Application LogicCourses
- Application LogicClasses
- Application LogicUsers

Data Access
- Data Access: Education
- Data Access: User

Database
- Education Database
- Nexus External Database
Second Refinement Presentation Layer

- Introduction of facade classes (design pattern)
Employ Design Patterns

- Design Patterns are class collaboration structures
  - Facade: abstracting, encapsulating a subsystem
  - Strategy: vary
  - Mediator: mediating between different objects
  - Factory: class to create other objects
  - Model-View-Controller: presentation pattern between the presentation and application logic layer
What Have We Learned

► An interactive web application has usually a BCED 4-tier architecture
► Design starts from the domain model
  ▪ and derives objects for the domain concepts in all layers
    • view objects in the presentation layer
    • managers in the application logic layer
    • entity objects in the middleware layer
    • databases in the database layer
► Use case realization analysis slices through the system
► ... important for the projects...
The End