Object Oriented Analysis and Design

Session 1:
Course Introduction & Overview of OOAD and UP

Session Outline
1. Course Details
   - Textbook, topics, web site
2. Course Philosophy
3. Homeworks and Grading
4. Overview of Object-Oriented Analysis and Design (OOAD)
5. Overview of the Unified Process (UP)

Course Outline
• Course Outline
• Book Availability
  – Allied Books

We will also cover
Only if time permits
• Web Engineering
  – Content Design
  – Site Design
• Architecture
• Prototyping
• Agile Methods
  – XP, SCRUM, etc.
• Project Management
  – Scheduling & Tracking
  – Risk Management
• Process Maturity Models
  – CMM, CMMI, ISO...

Course Philosophy
1. Team Work
2. Learning by Doing
3. Preparation for Realworld Projects
Overview of OOAD & UP

Important Note
• Difference between
  – OOAD
  – Process
  – UML
  – OOP language

Software is Pervasive…
• System software
• Real-time software
• Business software
• Engineering & scientific software
• Embedded software
• Personal computing software
• Web-based software
• Artificial Intelligence software

Important Questions
• Why does it take so long to finish a software project?
• Why does it cost so much to develop software?
• Why do we deliver software with significant bugs still present?
• Why is it so hard to measure progress during development?

Important Questions [2]
• The answers to these questions require a focus on the software development process
• These issues are getting worse, not better, due to the constant pressures of “internet time” (product life_cycle < 6 months)

The Solution
• A framework which supports constant, evolutionary improvement in product and process
• An explicit model of the development process is the key element missing in past experiences
Software Engineering

- **Process**
  - Life-cycle support for all activities, from conception to retirement
- **Methods**
  - Well-defined, formal techniques for each step in the life-cycle process
- **Tools**
  - Description languages and support software for particular methods

Engineering is:

Creating cost-effective solutions ...
... to practical problems ...
... by applying scientific knowledge ...
... building things ...
... in the service of mankind

“Software Engineering entails making decisions under the constraints of limited time, knowledge, and resources.”

[Source: M. Shaw]

Maturity: Progressive Codification

---

Fundamental Questions

- What is the problem to be solved?
- What are the characteristics of a possible solution?
- How will the solution be designed?
- How will the solution be built?
- How will we test the design and implementation?
- How will we maintain over time?

[Source: M. Shaw]

Course Topics and Skills

- Including:
  - Teaming
  - Project Management

[Source: M. Shaw]

Object Orientation

[Source: Larman, 2002]
Key Steps in OOAD [1]

Use Case: a textual description or “story” describing the system

Example:

Play A Dice Game: “A player picks up and rolls the dice. If the dice face values total seven, they win; otherwise, they lose.”

Key Steps in OOAD [2]

Domain Model: diagram(s) showing domain concepts, attributes, and associations

Example:

[Larman, 2002]

UML Class Diagram

[Iterative & Incremental Development]

Key Steps in OOAD [3]

Interaction Diagram: shows the flow of messages between software objects (method invocation)

Example:

UML Sequence Diagram

[Larman, 2002]

Key Steps in OOAD [4]

Class Model: shows attributes, methods and associations for software (solution) objects (not domain objects)

Example:

UML Class Diagram

[Larman, 2002]

Iterative & Incremental Development

Early iterations are focused on the “Y-axis” of the system. We feedback and adaptation the system converges towards the final target requirements and design.

In late iterations, significant changes in requirements are rare but can occur. Such changes may give an organization a competitive business advantage.
Scheduling & The Unified Process

Disciplines vs. Phases

Artifact Refinement in UP

Layers and the Textbook Case Study

Learning Path and Iterations
A Note on UML Modeling

• All UML modeling exercises must be handed in electronically (included in a Word document)
• Otherwise, the choice of modeling tool is up to you…
  – Payware: Visio
  – Freeware: Poseidon CE (www.gentleware.com)
  – Hand-drawn & scanned to GIF
  – etc.

Inception Questions

• What is the vision and the business case for this project?
• Feasible?
• Buy and/or build?
• Rough estimate of cost: $10K, $100K, $1M, …
• Should we proceed or stop?

Goals of Inception

• “…to do just enough investigation to form a rational, justifiable opinion of the overall purpose and feasibility of the potential new system.” [Larman 2002, p.35]
• Envision the product scope, vision and business case
• Do the stakeholders have basic agreement on the vision of the project? Is it worth investigating seriously?

Inception Artifacts

<table>
<thead>
<tr>
<th>Artifact</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use Case Model</td>
<td>Describes the high-level goals and requirements, the business case, and overall system organization.</td>
</tr>
<tr>
<td>Stereotypes</td>
<td>Use stereotypes to add semantics to the high-level use cases.</td>
</tr>
<tr>
<td>Actor</td>
<td>The external entities that interact with the system.</td>
</tr>
<tr>
<td>Message Sequence Chart</td>
<td>Describes the high-level interactions among the actors and the system.</td>
</tr>
<tr>
<td>Activity Diagram</td>
<td>Describes the high-level sequence of activities involved in the use case.</td>
</tr>
<tr>
<td>Statechart</td>
<td>Describes the dynamic behavior of a particular component.</td>
</tr>
</tbody>
</table>

Surf Report: Inception Steps

• Clarify initial requirements
  – Mapping data to email fields wasn’t clear
  – Identify rough functional requirements
    – Screen scraper, extractor, email formatter, send email routine; timer
  – Feasibility issues
    – How hard is the screen scraping? (examine actual page sources)
    – How reliable are the sources? (examine actual page sources)
  – Business case: Is there a product vision?
    – Reliability of sources; availability of data for other areas
    – Scaling up: consider whether the screen scraping can be extended to other areas of interest.
  – Added value for customers vs. business model
    – (e.g., what is a realistic subscription fee?)

Example: Surf Report

Subject: Weather info for my pager
Date: Thu, 10 Jan 2002 10:47:46 -0800

Here is a summary of the info I would like on my pager. I can read the first 156 characters of any email I receive, so I need to keep it abbreviated (see examples). Here is a list of web pages and info I would like from each page.

http://tbone.biol.sc.edu/tide/tideshow.cgi?site=La+Jolla%2c+Scripps+Pier%2c+California
  Tides for the current day.

http://meteora.ucsd.edu/weather/observations/sio_pier/display/sio_pier.obs
  Wind speed (WS), wind direction (WD), tide height (TH).

http://cdip.ucsd.edu/cdip_htmls/data.shtml
  Hs, Tp, Dp for various buoys.

Example: (Tides,L0:03,2.04 ft,H6:28,6.09 ft,L13:49,-0.84 ft,H20:21,3.61 ft)

I would like to get this sent to me at 5:00am, 9:00am, 1:00pm, 4:00pm.
It’s Not Inception If…
- It takes more than a few weeks
- You attempt to define most requirements
- Estimates are expected to be reliable
- You define a concrete architecture
- No biz case or vision document
- Too little or too much use case modeling

Understanding Requirements
- “Capabilities and conditions to which the system and project must conform” [Jacobson et al., 1999]
- Challenges: (find, communicate, record, manage) the requirements
- Requirements always change, so effective management is critical

Factors that Challenge Projects
37% of the factors related to problems with requirements!

The FURPS+ Model
- Functional — features, capabilities, security
- Usability — human factors, help, documentation
- Reliability — failure frequency, recoverability
- Performance — response times, throughput, accuracy, availability, resource utilization
- Supportability — adaptability, maintainability, internationalization, configurability

The FURPS+ Model [2]
- Implementation — resource limitations, languages/tools, hardware
- Interface — with legacy systems
- Operations — sysop management
- Packaging — delivery, installation
- Legal — licensing, etc.

Other Terminology
- Quality Attributes, or “-ilities”
  - Usability, reliability, supportability, performance (non-functional)
- Functional vs. Non-Functional
  - Behavioral features vs. everything else
Documenting Requirements

- Primarily, in the use case model (functional requirements)
- Also, Supplementary Specifications (other requirements)

Choosing an appropriate architecture often depends on a clear understanding of non-functional requirements.

Questions and Discussion

References

- Craig Larman
- Slides from a similar course at CMU (SW engineering for IT)