面向对象程序的分析与设计
Object-Oriented Analysis and Design

Lecture 1

Prof. S. Xu

Contents

2

Introduction to This Course

Instructor

- **Current Position**
  - Professor and Dept. Chair, Algoma University, Canada
  - 中国科学院合肥物质科学学院特聘研究员
- **Education Backgound**
  - Ph.D., Wayne State University, USA
    - MSc, University of Windsor, Canada
    - BSc., University of Manitoba, Canada
  - Ph.D., University of Liege, Belgium
    - MSc. CAGS, Beijing, China
    - BSc. Peking University, China

4

Course Information

- Contact Info:
  - Dr. Simon Xu (徐绍春)
  - scxu@hotmail.com

5

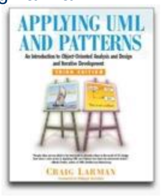
Course Information

- Course Description
  This course builds on knowledge of object-oriented programming, data structures etc. Students are introduced to object-oriented software analysis and design techniques that are currently used in industry.

6

Course Information

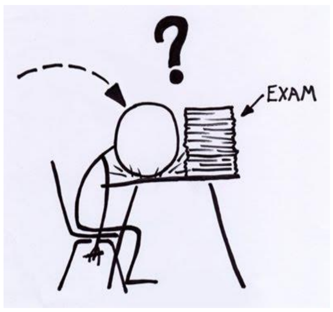
- Textbook
  - Applying UML and Patterns : An Introduction to Object-Oriented Analysis and Design and Iterative Development, Third Edition by Craig Larman
- Class Time and Place
  - Tuesday and Wednesday (11-13)
  - Room 130



7

Course Information

- Evaluation
  - Group Project
  - Attendance
  - Exam??



8

Course Information

- Language
  - English
- UML knowledge
- Software Engineering Contents

9

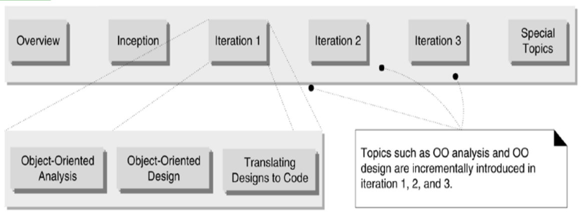
- So you know how to program, at least in Java…
- You also know some useful data structures….
- Programming is fun, isn't it ?
- Do you know how to develop quality software?
- What do you think?
  - Owning a hammer doesn't make one an architect.

10

- If you are given the task of developing …., how would you start?
- Not an easy job…?
- The material in this course offers you:
  - A systematic method to do OOAD, in the context of Software engineering process (the Unified Process , abbr. UP)
    - Before you do any programming, which is more or less trivial
- Hopefully, you will at least know how to develop quality software, which potentially to land you a job as system analyst or alike along your career path, unless you prefer a lifelong job as programmer.

11

Organization of the book and the course

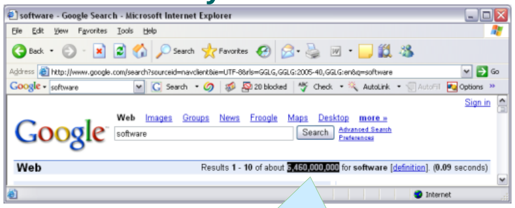

Overview
Inception
Iteration 1
Iteration 2
Iteration 3
Special Topics
Object-Oriented Analysis
Object-Oriented Design
Translating Designs to Code
Topics such as OO analysis and OO design are incrementally introduced in iteration 1, 2, and 3.

- This course is about
  - OOAD
  - Frequently used UML notation
  - Common Design Patterns
  - Iterative software development process

12

Software and Software Engineering

Software Today:



software - Google Search - Microsoft Internet Explorer

http://www.google.com/search?sourceid=navclient&ie=UTF-8&rls=GGLG,GGLG:2005-40,GGLG:en&q=software

Results 1 - 10 of about 6,460,000,000 for software [definition]. (0.09 seconds)

6,460,000,000

Technology Pervades our World

Technology Pervades our World

– GPS/Maps/Gives you directions
– DVD Player
– Microprocessor in keys – anti-theft
– Rearview camera
– etc.

The Six Million Dollar Van?

Spaceship Demopoulos

– Voice Control
– BMW i3
– etc

The Six Million Dollar Van?

"Killer Applications" of Technology

- Email
- Online Shopping and Retailing
- Information Sharing (facebook)
- Cell Phones (a lot of products will be replaced)

•Mobile Computing

Detailed Example in China

- Pre- 2007: Increase rate: 40% per year
- 2007: Income from Software in China – 584 (billion)
- 2008: Income from Software in China – 757 (billion)
- 2009: 900 (billion)

Introduction

- What is Software?

What is Software?

consider software as a product
  – like a car or a toaster
- meets some need
- built to a design
- has a value

==> Creating software is “like” manufacturing a product

What is Software?

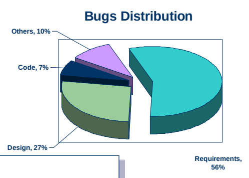
BUT Software is different, how?

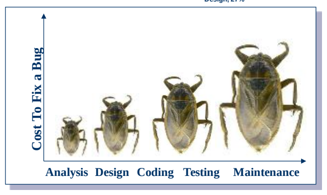
What is Software?

- What has been changed in terms of software?

Software Errors

- People do mistakes due to:
  – Miscommunication between customer and programmers
  – Unrealistic Time Limits
  – Growing Software Complexity
  – Changing Requirements





Infamous Software Error Case Studies

1. The Y2K (Year 2000) Bug
   - Back to 70’s the computers had very little memory
   - To save memory, dates were shortened
   - Some old programs were still used in 80’s even though programmers had retired

Infamous Software Error Case Studies

2. European Space Agency Ariane 5
   - Track control system failure results in self destruction
3. Denver Airport
   - Late delivery of software for the baggage system delays the opening of the airport by 16 months

Practical disasters

- US study (2005): 181 billion US$ spend per year for failing software development projects
- Different reasons:

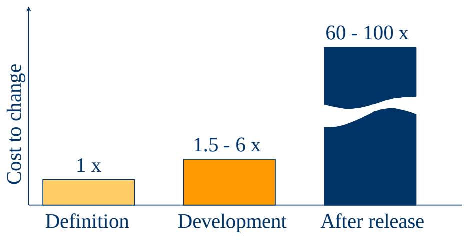
Management myths

- State-of-the-art tools are the solution (CASE)
  - A fool with a tool is still a fool
- Getting behind schedule resolved by hiring additional programmers
  - “adding people to a late software project makes it later”
  - Picking up stroberry

Customer myths

- A general statement of objectives is sufficient to begin writing programs - we can fill in details later.
  - Thorough communication between customer and developer needed
- Changes can be easily accommodated because software is flexible
  - changes happen as a fact of life
  - late changes are expensive

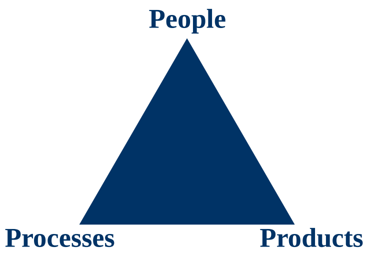
The impact of change



Practitioner’s myths

- Once we write a program and get it to work, our job is done
  - 50-70% of all effort *after* first delivery
- Until I get the program “running”, I really have no way in assessing its quality
  - inspections & reviews
- The only deliverable for a successful project is the working program
  - documentation (users, maintenance)

Three P’s



Life of a Software Engineer

- Life of a Software Engineer

More About OO Analysis and Design

Introduction

- OOAD (and all software design) is strongly related to the prerequisite activity of requirements analysis, which often includes writing use cases.
- Given many possible activities from requirements through to implementation, how should a developer or team proceed? Requirements analysis and OOA/D needs to be presented and practiced in the context of some development process.

Employment Application Review System

EARS is an intranet-based Employment Application Review System for the Department of Math and Computer Science in Algoma University. The system is designed so that department faculty members can review applicants and collaborate asynchronously in order to find the best applicant for a given job opening. This system reduces the overhead of the process and lightens the workload for the search chairperson.

The scope of this project will be to provide a system that allows to:

1. log-in EARS system
2. manage system users (add new accounts)
3. add a new faculty search (committee chair, members, position, search starting date and ending date, add new committee members)
4. List and review applications (view profile, post comments on applicants, change applicants’ statues, perform a faculty review, assign faculty review)
5. set account’s settings (email, name, title, password)

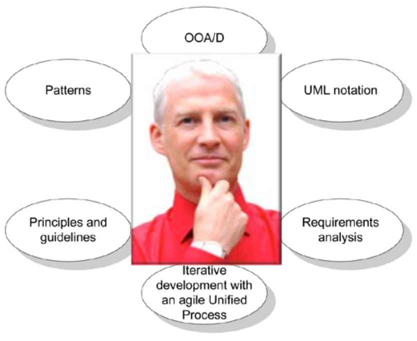
Different users might need various functionalities.

35

Introduction

- An agile (light, flexible) approach to the well-known Unified Process (UP) is used as the sample iterative development process within which these topics are introduced.
- So what’s UP?
- See http://en.wikipedia.org/wiki/Unified_Process for details.

36



OOA/D
Patterns
UML notation
Principles and guidelines
Requirements analysis
Iterative development with an agile Unified Process

37

A few more notes….

- This course is *not* pure about UML.
- Thinking in Objects is more important than knowing UML.
- This course is to explore how to *apply* the UML in the service of doing OOAD, and covers frequently used UML.

38

Analysis and Design

- Analysis: “build (do) the right thing”
  - Analysis focuses on user requirements (functional or non-functional) (What)
  - ***Investigate the problem, rather than a solution.***
- Design: “build (do) the thing right”
  - Design focuses on how to provide the required functionality (How)
  - ***Emphasize a conceptual solution, which will ultimately lead to concrete implementation.***

39

Analysis

- Usually performed by a systems analyst
  - This is a person whose job it is to find out what an organization (or person) needs in terms of a software system
- Analysis looks at the software as a black box of functionality
  - An analyst will never care about the internals of the system, merely how a user would interact with it from the outside

40

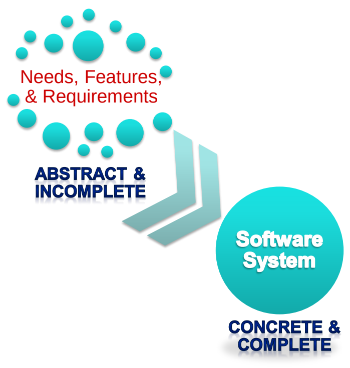
Design

- Usually performed by an architect or designer
  - An architect looks at overall system structure (at a high level)
    - For example, an architect on a distributed system might decide how the software components will be distributed
    - An architect also looks at modularity, and non-functional requirements (such as performance and scalability)
  - A designer looks at system structure (at lower levels)
    - A designer will look at the classes that make up a module, and how they will interact to perform some function of the system

41

Why is Software Design Important?

- Size
- Complexity
- Constraints
- Performance
- Communication



Design Patterns

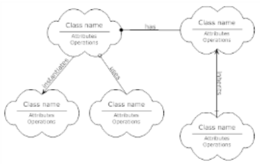
- Patterns are themes that recur in many types of software systems
- Often, the solution for the problem in one context can also be used in another context
  - Thus, the designs (and even implementations) can often be reused from other software systems

WE SHOUTED OUT ABOUT SOME NEW INVENTION & THEN THAT THING HIT HIM!
DeepFat '09

43

Object-Oriented Analysis and Design

- Object-oriented Analysis
  - finding and describing the objects or concepts in the problem domain.
- Object-oriented Design
  - defining software objects and how they collaborate to fulfill the requirements



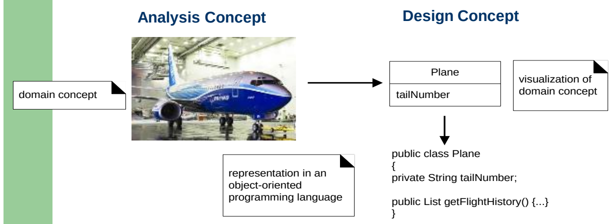
44

Key Questions for Object-Oriented Design

1. What classes do we get from the application domain?
2. How should responsibilities be allocated to classes?
3. What classes should do what?
4. How should objects collaborate?

Guided by design patterns

Analysis and Design Concepts –Flight information system


Analysis Concept
Design Concept
domain concept
Plane
tailNumber
visualization of domain concept
representation in an object-oriented programming language

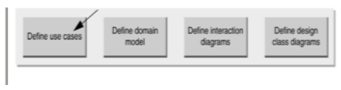
```
public class Plane
{
private String tailNumber;

public List getFlightHistory() {...}
}
```

A short Example

- We've talked about iterative development, UP, requirement analysis, UML, OOAD, etc.
- Let's put them into the perspective of OOAD using this small example.
  - A dice game:
    - software simulates a player rolling two dice. If the total is seven, they win; otherwise, they lose.

47

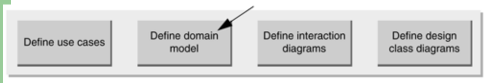
A short Example



- Requirements analysis may include stories or scenarios of how people use the application; these can be written as use cases.
- Use cases are not an object-oriented artifact, they are simply written stories. However, they are a popular tool in requirements analysis

48

A short Example

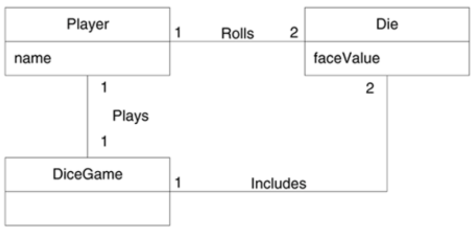


- Object-oriented analysis is concerned with creating a description of the domain from the perspective of objects. There is an identification of the concepts, attributes, and associations that are considered noteworthy.
- The result can be expressed in a ***domain model*** that shows the noteworthy domain concepts or objects

49

A short Example

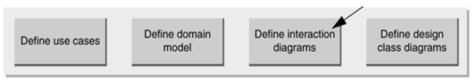
Figure 1.3. Partial domain model of the dice game.



Note that a domain model is not a description of software objects; it is a visualization of the concepts or mental models of a real-world domain. Thus, it has also been called a *conceptual object model*.

50

A short Example

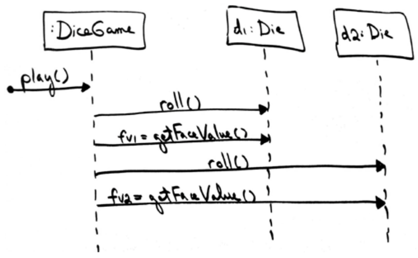


- Object-oriented design is concerned with defining software objects - their responsibilities and collaborations. A common notation to illustrate these collaborations is the ***sequence diagram*** (a kind of UML interaction diagram). It shows the flow of messages between software objects, and thus the invocation of methods.

51

A short Example

Figure 1.4. Sequence diagram illustrating messages between software objects.



52

A short Example

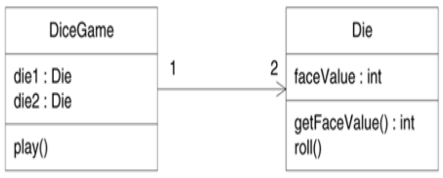
Define use cases
Define domain model
Define interaction diagrams
Define design class diagrams

- In addition to a dynamic view of collaborating objects shown in interaction diagrams, a static view of the class definitions is usefully shown with a ***design class diagram.*** This illustrates the attributes and methods of the classes.

53

A short Example

Figure 1.5. Partial design class diagram.



How does it differ from the domain model?

54

- The dice game is a simple problem, presented to focus on a few steps and artifacts in analysis and design. To keep the introduction simple, not all the illustrated UML notation was explained.

55

- This course is not primarily a UML notation course, but one that explores the larger picture of *applying the UML*, *patterns*, and *an iterative process* in the context of OOA/D and related *requirements analysis*.
- OOA/D is normally preceded by requirements analysis. Therefore, the initial lectures introduce the important topics of use cases and requirements analysis, which are then followed by topics on OOA/D and more UML details.

56

The Challenge and Opportunity: Internet Islands

Software Opportunities??

- Google
- eBay
- Expedia
- Facebook
- LinkedIn
- Amazon.com
- YouTube
- MyCQ
- *Match.com*
- EHARMONY.COM
- Paybal
- GroupOn
- Priceline.com
- Craigslist.com

Software Opportunities??

- Google – Baidu
- Ebay – Taobao
- Expedia – Ctrip
- Facebook – 新浪博客
- LinkedIn –Renren
- Amazon.com -- 当当图书
- YouTube-Youku
- MyCQ – qq
- *Match.com -世纪佳缘*
- EHARMONY.COM- 百合网
- Paybal – 支付宝
- GroupOn – 团宝网
- Priceline.com -- ???
- Craigslist.com -- ???