CS6110

OBJECT ORIENTED ANALYSIS & DESIGN

Module 1

Dr.S.Neelavathy Pari
Assistant Professor (Sr. Grade)
Department of Computer Technology
Anna University, MIT Campus

Module 1

- Introduction to OOAD with OO Basics
- Unified Process
- UML diagrams

ACTIVITIES

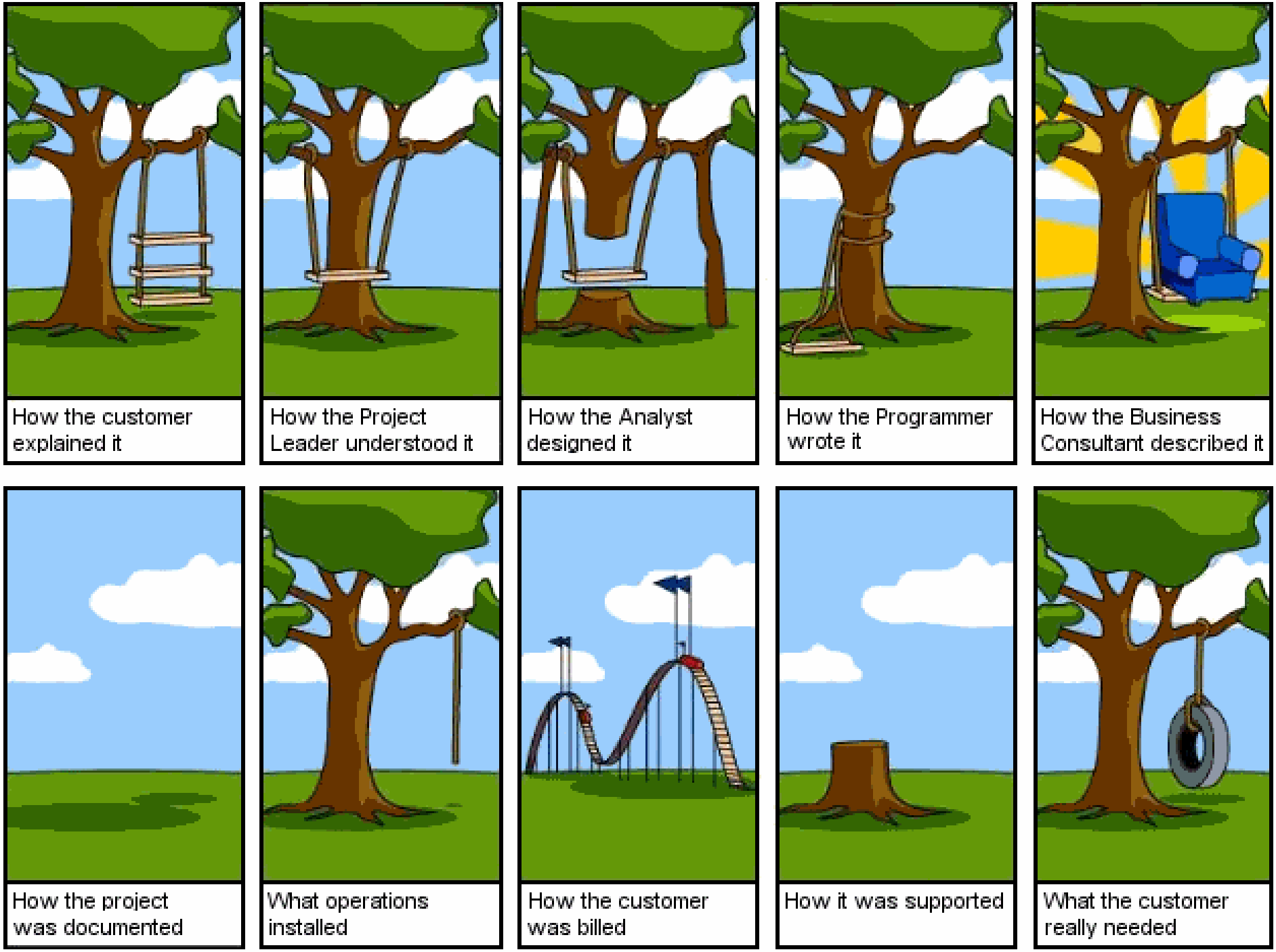
- EL -Identifying a suitable case study to work on for a complete end-end implementation
- EL –Document the Software Requirement Specifications(SRS) for the identified case study

- Practical –Getting familiar with the case tool

Introduction

- We will discuss about
  - iterative development
  - evolutionary development
  - the Unified Process (UP)
  - agile approaches
  - UML

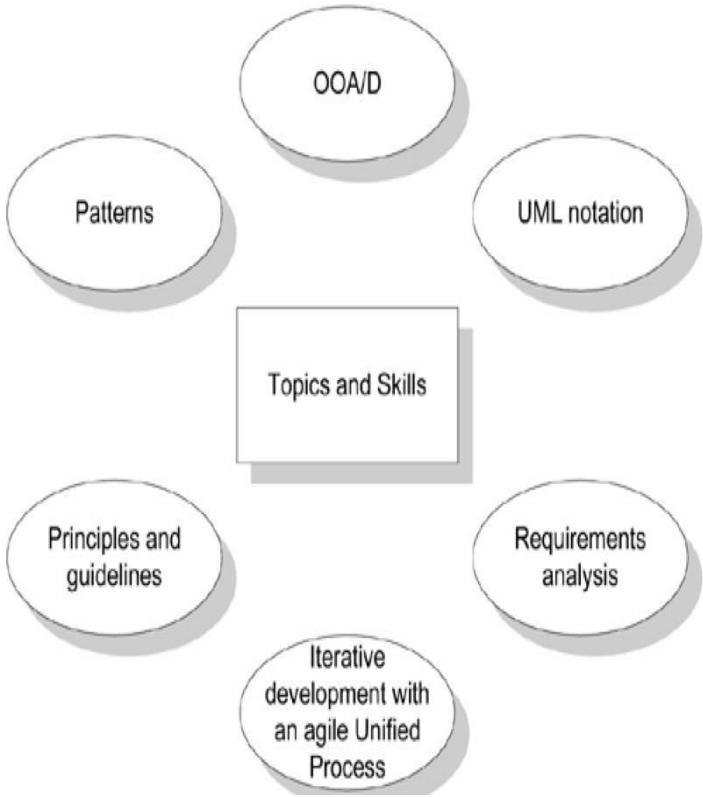
Thinking in Objects


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

What does it mean to have a good object design?

To Learn the skills in object-oriented analysis and design (OOA/D).

- well-designed, robust, maintainable software using OO technologies and languages



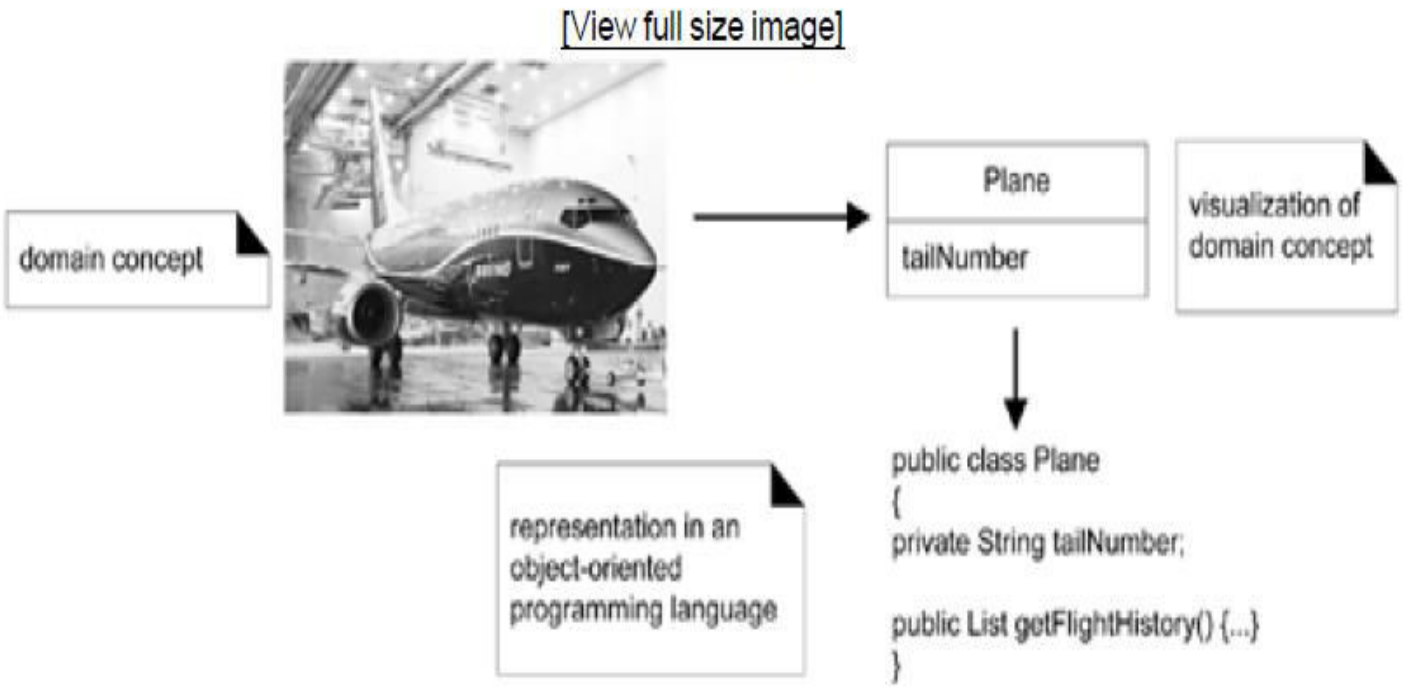
Analysis : Emphasizes an investigation of the problem and requirements rather than solution

Design : Emphasizes a conceptual solution that fulfills the requirements rather than its implementation

OOA : Emphasis on finding and describing the object and concepts in the problem statements

OOD : Emphasis on defining objects and how they collaborate to fulfill the requirements.

Representation of Objects



Example – DICE Game

Simulation - a player rolling two dice , if the total is 7 they win otherwise Lose

Define Use Cases

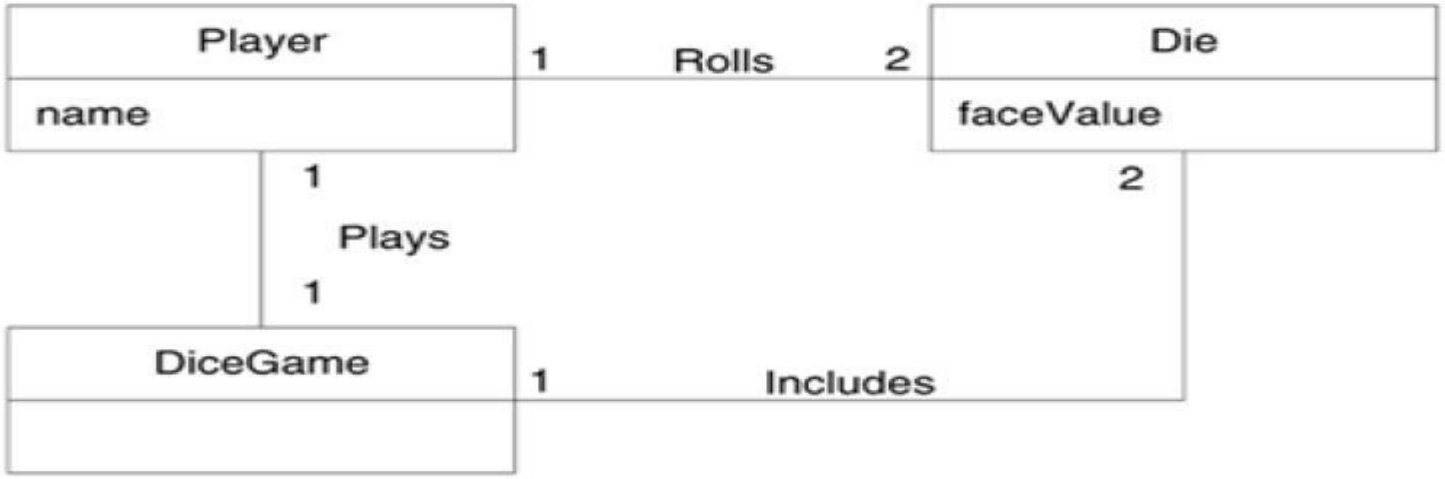
- Scenarios of how people use the application
- Brief version of the play a dice game use case

  – Player request to roll the dice
  – System present the result
  – If the dice face value to be 7 , player wins or lose

Requirement Analysis Phase

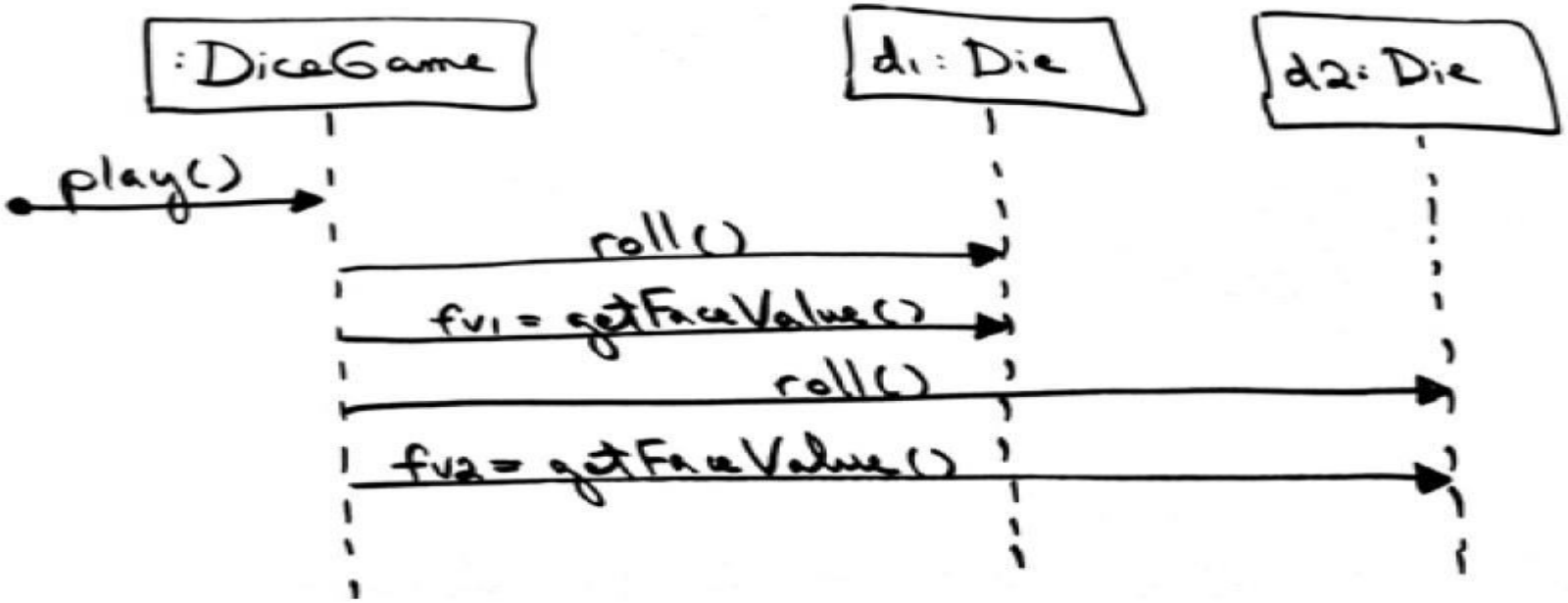
Define a Domain Model

- OOA is concerned with creating a description of the domain from the perspective of objects
- Identification of concepts , attributes, and associations



Interaction Diagrams

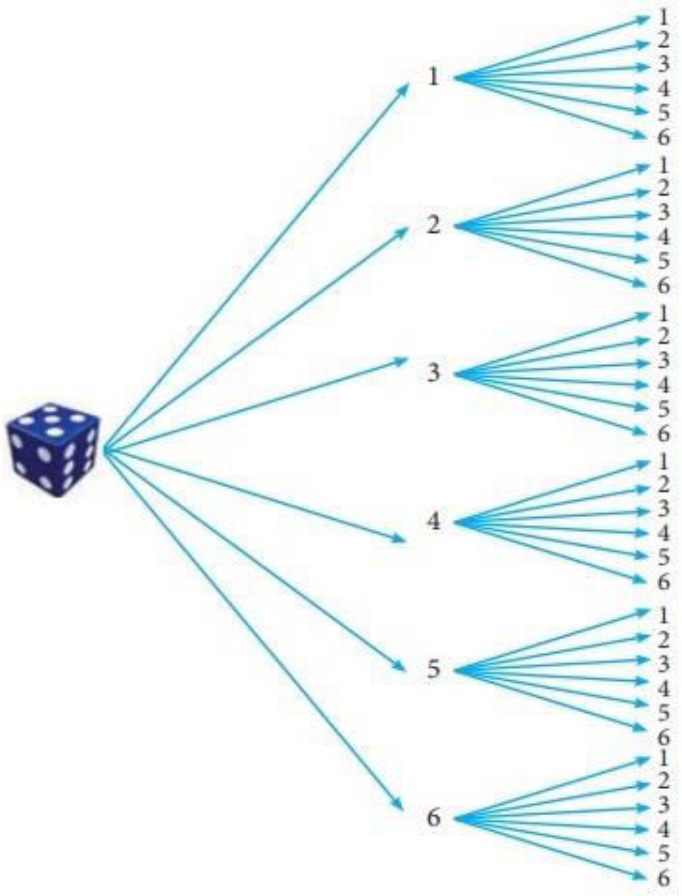
- **Assign Object Responsibilities and Draw Interaction Diagrams**



Class Diagrams

- **Define Design Class Diagrams**

Rolling Dice – Tree Representation



(1,1),(1,2),(1,3),(1,4),(1,5),(1,6)

(2,1),(2,2),(2,3),(2,4),(2,5),(2,6)

(3,1),(3,2),(3,3),(3,4),(3,5),(3,6)

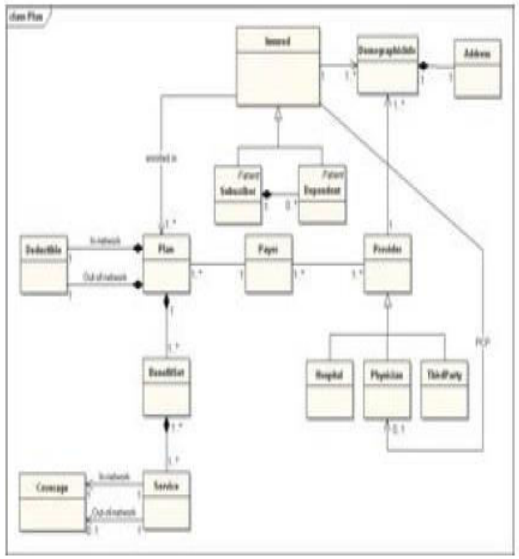
(4,1),(4,2),(4,3),(4,4),(4,5),(4,6)

(5,1),(5,2),(5,3),(5,4),(5,5),(5,6)

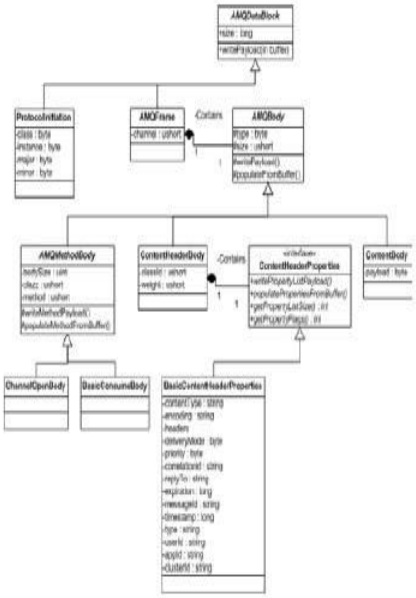
(6,1),(6,2),(6,3),(6,4),(6,5),(6,6)

Example 2

Analyze the system



Model the system



Design the software

Analysis and Design:

- Analysis is the *investigation of the problem* - ***what are we trying to do?***
  - Here is where use cases are created and requirements analysis are done
- Design is a *conceptual solution that meets the requirements* – ***how can we solve the problem***
  - Note: Design is *not implementation*
  - UML diagrams are not code (although some modeling software does allow code generation)

OO Analysis and Design:

- Object-oriented analysis: Investigate the problem, identify and describe the objects (or concepts) in the problem domain
  - Also, define the domain!
- Object-oriented design: Considering the results of the analysis, define the software classes and how they relate to each other

Note :

Not every object in the problem domain corresponds to a class in the design model, and vice versa

Iterative, Evolutionary, and Agile

Iterative Development

- Suppose you were assigned to write a movie script for a company
- They know what they want, in terms of what kind of movie, how long, setting, etc., but they need you to fill in the details. How would you do it?
- You could spend a lot of time talking to them, getting as much information as possible, and then write the script and send it to them and hope you nailed it ...
- Risky: Chances of getting it all right are slim, and if you missed you need to go back and start making changes and edits, which can be complicated if you want the story line to work
  - You could also start with a draft, and send the incomplete version to them for feedback
  - They would understand that this is not finished, but just a draft.
  - They provide feedback, you add more details, and the cycle continues
  - Eventually, you end up with the finished product that is close to what they wanted

- This is how iterative development works……

Unified Process (UP)

A widely adopted process for building, deploying, and possibly maintaining software ,Flexible and open, can adopt practices from other methods such as Scrum

We will concentrate on Agile UP

Iterative and evolutionary development

Development is organized into a series of short, fixed length mini-projects called iterations

Note – emphasizes early programming and testing of partial systems, even before all requirements are finalized!

Direct contrast to sequential methods (i.e. Waterfall), where coding is not started until the requirements phase is finished.

Each iteration enlarges and refines the project, with continual feedback and adaption as the main drivers

Iterative Methodology

- Each iteration is fixed length, called time boxed
- Typical iteration: Spend the first day analyzing the current state of the project, defining goals for the current iteration, and then working on design in small teams
- Then teams work on implementing new code, testing, more design discussions, and conducting daily builds of partial system
- Usually involves demonstrations and evaluations with stakeholders (customers or customer reps)
- Also planning for next iteration – each iteration may involve analysis and design work

Thinking in Objects and UML - 1

- The Unified Modeling Language (UML) is a standard diagramming notation; sometimes referred to as a blueprint.
- It is NOT OOA/OOD or a method
- **Only a notation for capturing objects and the relationships** among objects (dependency; inheritance; realizes; aggregates, . .)
- UML is language-independent
- Analysis and design provide software "**blueprints**" captured in UML.
- Blueprints serve as a tool for thought and as a form of communication with others.

Thinking in Objects and UML – 2

- But it is far more essential to 'think' in terms of **objects** as providing '**services**' and accommodating '**responsibilities**.'

- **Discuss**: What is meant by 'services?' How indicated?
  - How might you think these 'services' impact the design of classes?
  - How might a client access these services?
  - What is encapsulation? How does it relate to reusability? Self-governance? Design?

- **Discuss**: What is meant by 'responsibilities?'
  - Encapsulation of data and services?

Thinking in Terms of Objects and UML - 3

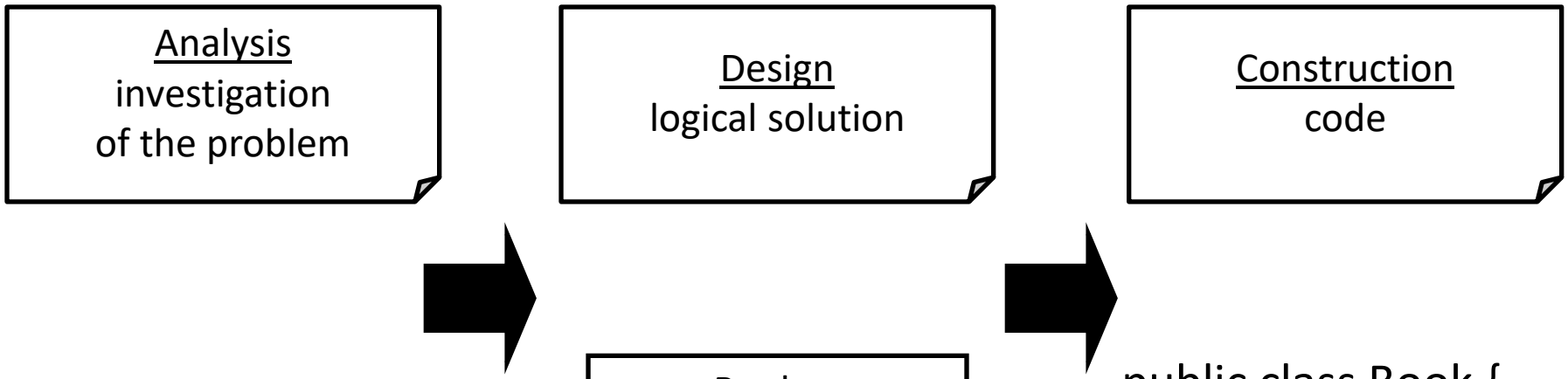
- **Object-Oriented Analysis (Overview)**
  - An **investigation** of the **problem** (rather than how a solution is defined)
  - During OO analysis, there is an emphasis on finding and describing the objects (or concepts) in the **problem domain**.
  - For example, concepts in a Library Information System include *Book*, and *Library*.
  - High level views found in the application domain.
  - Oftentimes called **domain objects; entities.**

Thinking in Terms of Objects and UML - 4

- **Object-Oriented Design**
  - Emphasizes a conceptual **solution** that fulfills the requirements.
  - Need to define sof**tware objects** and how they **collaborate** to meet the requirements.
  - For example, in the Library Information System, a *Book* software object may have a *title* attribute and a *getChapter* method.
    - What are the methods needed to process the attributes?
- Designs are **implemented** in a programming language.
  - In the example, we will have a *Book* class in Java.

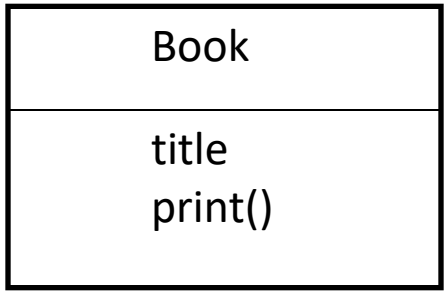
Thinking in Terms of Objects and UML – 5

From Design to Implementation



Book
(concept)

Domain concept



Representation in
analysis of concepts

```
public class Book {
        public void print();
        private String title;
}
```

Representation in an object-oriented programming language.

Can you see the services / responsibilities in the Book class?

Thinking in Objects and UML-6

- Then too, there are sets of **proven design solutions** to problems that are considered 'best practices.'
  - Certain 'groupings' of classes with specific responsibilities / interfaces.
  - These provide specific solutions to specific problems.
  - Called **Design Patterns**

- We will discuss (much later) these standard patterns and how to **apply** them to develop solutions to common design problems.

Thinking in Objects and UML-7

- Of course, design (solution to requirements) 'assume' a robust **requirements analysis** has taken place.

- **Use Cases** are often used to capture **stories** of requirements and are often views as 'constituting' the **functional** requirements, but NOT the software quality factors (non-functional requirements).

- Use Cases are **not specifically designed** to be object-oriented, but rather are meant to capture how an application will be used.

- Many methods for capturing requirements.
- We will concentrate on Use Cases (ahead).

Basic Terms: Iterative, Evolutionary, and Agile

1. Introduction

- ***Iterative*** - the entire project will be composed of min-projects and will iterate the same activities again and again (but on different part of the project AND with different emphases) until completion.
- ***Evolutionary (*or *incremental)*** - the software grows by increments (to be opposed to the traditional, and somewhat old-fashioned, Waterfall model of software development).
- ***Agile*** - we will use a light approach to software development rather than a very rigid one (which may be needed for a safety-critical system for example)
- This kind of approach seems better at treating software development as a **problem solving activity**; also the use of objects makes it amenable.

Our Approach:

- We need a Requirements Analysis approach with OOA/OOD need to be practiced in a framework of a development process.

- We will adopt an agile approach (light weight, flexible) in the context of the Unified Process, which can be used as a sample iterative development process.
  - Within this process, the principles can be discussed.

- Please note that there are several other contexts that may be used, such as Scrum, XP, Feature-Driven Development, Lean Development, Crystal Methods and others…and we will look at a few of these.

Why the Unified Process:

- The Unified Process is a popular iterative software development process.

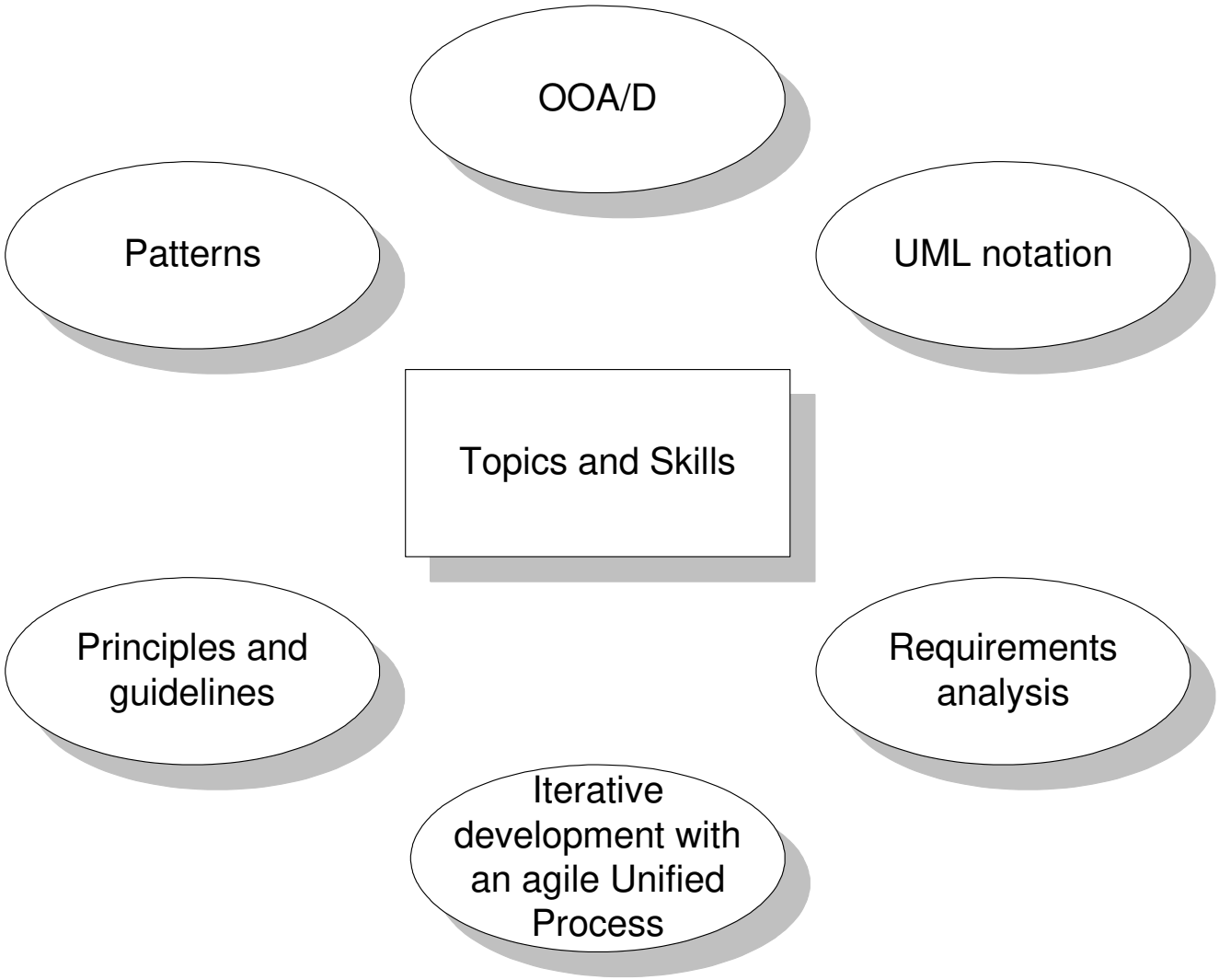
- Iterative and evolutionary development involves relatively early programming and testing of a partial system, in repeated cycles.

- It typically also means that development starts before the exact software requirements have been specified in detail;

- Feedback (based on measurement) is used to clarify, correct and improve the evolving specification:
- This is in complete contrast to what we usually mean by engineering!

2. What is the Unified Process?

- The UP is very flexible and open and can include other practices from other methods such as Extreme Programming (XP) or Scrum for example.
  - e.g. XP's test-driven development, **refactoring** can fit within a UP project; So can Scrum's **daily meeting**.
  - Being **pragmatic** in adapting a particular process to your needs is an important skill : all projects are different.


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

The Rush to Code

- Critical ability to develop is to think in terms of objects and to artfully **assign responsibilities to software objects**.
- Talk at great length in COP 3538 about encapsulation and assigning methods to objects where the data is defined…
- One cannot design a **solution** if the **requirements** are not understood.
- One cannot **implement** the design if the **design** is faulty.
- If I could only stop my students….☺

The Rush to Code

- Analysis: - investigate the **problem** and the **requirements.**
  - What is needed? Required functions? Investigate domain objects.
  - Problem Domain
  - The **Whats** of a system.
  - **Do the right thing (analysis)**

- Design:
  - Conceptual solution that meets requirements.
  - Not an implementation
  - E.g. Describe a database schema and software objects.
  - Avoid the CRUD activities and commonly understood functionality.
  - The Solution Domain
  - The '**Hows**' of the system
  - **Do the thing right (design)**

What is Object-Oriented Analysis and Design

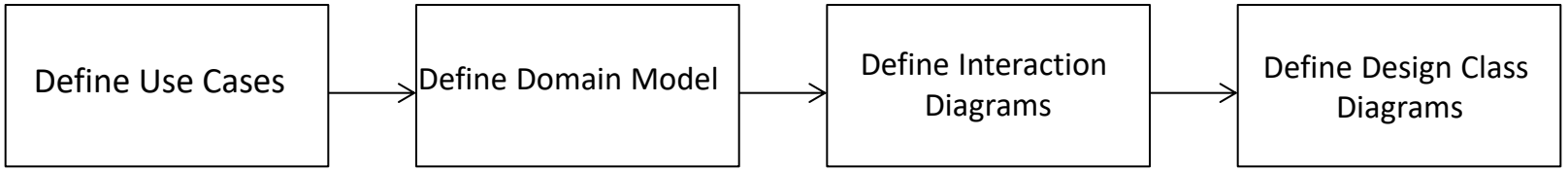
- OOA: we find and describe **business objects** or concepts in the **problem domain**

- OOD: we define how these **software objects <u>collaborate</u>** to meet the requirements.
  - Attributes and methods.

- OOP: Implementation: we implement the design objects in, say, Java, C++, C#, etc.

Homework Assignment

- Using the model below, develop a two-three page discussion



outlining the four activities listed and present the major features of each.

A short definition and example of a domain model, interaction diagram, and class diagram is sufficient, but be prepared to discuss each of these.

Also, have a general idea about use cases – what they are designed to do and what they are not designed to do.