OBJECT ORIENTATION CONCEPTS

CSCI 467/567 Introduction to Software Engineering, Fall 2017

1

Today

- Review Software Engineering Concepts
- Four Ways of Managing Complexity
- Review System Analysis and Design Principles
- Basic Object Orientation Concepts
- Identify objects and use cases from complete functional requirement statements.

2

Basic Concepts and Terms

- **Functional Requirements** = all the functions the system must do (or allow a user to do)
  - Example: A customer shall be able to add an item to a shopping basket.
- **Non-Functional Requirements** = other characteristics or attributes the system must have
  - Example: The system must be available 24 - 7.
- **Problem/Application Domain (Business Community)** = All aspects of the user's problem/situation
  - Physical environment
  - The users
  - Work processes
- **Solution Domain (Technical Community)** = All aspects of the application development including: objects, design, code, test and implementation.

3

Software Complexity

- Average new program: 200,000 lines
  - Problem domain (user community): what clients want is often COMPLEX: takes lots of code to implement system to meet their needs.
  - Software is extremely flexible
    - **VERY** easy to change code
    - **HARD** to change is **correctly**
  - A requirement can be viewed as "simple, yet complex."
    - Why?

4

Why are software systems so complex?

Traditional view

- The problem domain(business community) is difficult to understand.
- The development process is very difficult to manage.
- Software offers extreme flexibility.

Bruegge & Dutoit 5

Managing Complexity

Have to do *advance planning*:

1. Build models (abstraction)
2. Use decompose (divide and conquer)
3. Identify and use hierarchical relationships
4. Use Five Principles of Analysis and Design

6

Manage Complexity by Using Abstraction

- Model: high-level description without details
- Object model: What is the structure of the system? What are the objects and how are they related?
- Functional model: What are the functions of the system? How is data flowing through the system?
- Dynamic model: How does the system react to external events? How is the event flow in the system?

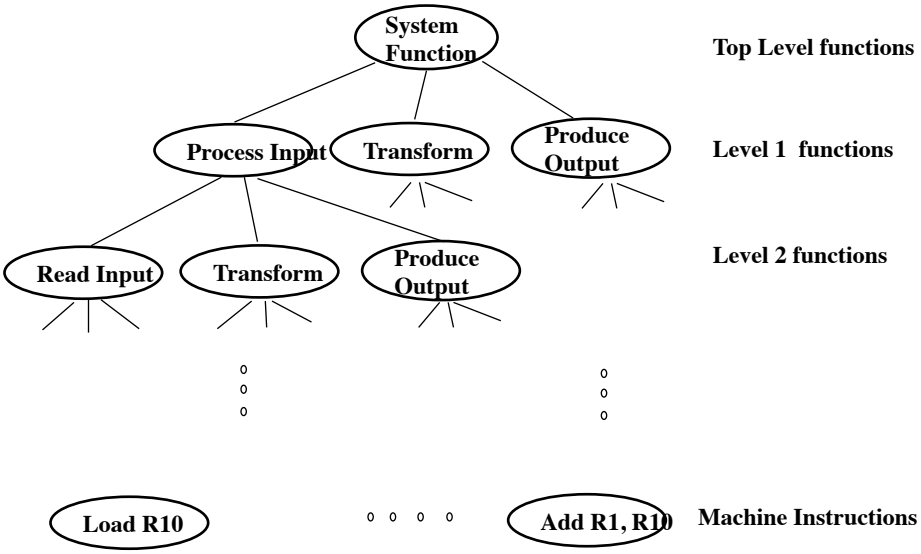
7

Manage Complexity by Using Decomposition

- Functional decomposition
- Object-Oriented decomposition

8

Functional Decomposition



Bruegge & Dutoit

9

Problems with Functional Decomposition

- Functionality is spread all over the system
- Maintainer must understand the whole system to make a single change to the system
- Consequence:
  - Codes are hard to understand
  - Code that is complex and impossible to maintain
  - User interface is often awkward and non-intuitive

Bruegge & Dutoit

10

Functional Decomposition

- Each function decomposed into steps; one module/ step
- Modules can be decomposed into smaller modules
- If badly done:
  - Maintainer must understand the whole system to make a single change to the system
  - Consequences:
    - Code is hard to understand
    - Code is complex and impossible to maintain

11

Object-Oriented Decomposition

- The system is decomposed into classes "objects"
- Each class is a "thing" in the application domain.
- Classes can be correctly decomposed into smaller classes.
- More about objects later.

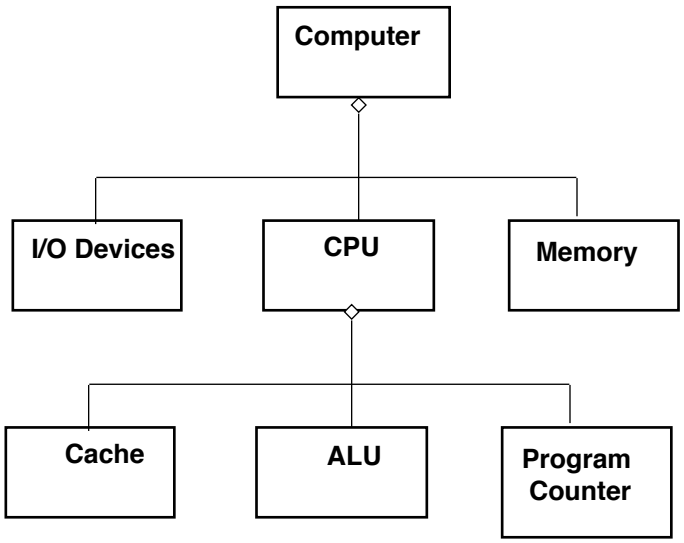
12

Manage Complexity by Using Hierarchies

- Part-of Hierarchy
- Kind-of Hierarchy

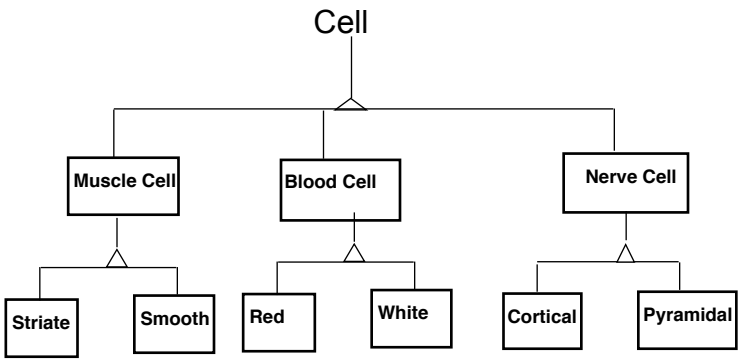
13

Part-of Hierarchy



14

Kind-of Hierarchy



15

Five Principles of Analysis of Design

1. Project must be well-defined in writing and **limited in scope.**
   - *Everything* in writing.
   - Written contracts.
     - Formal contract if working for customer.
     - Letter of understanding if in-house assignment.
       - If necessary, write your own (tactfully).

16

Principles of Analysis of Design

- In any case, state:
  - Scope
    - » Trying to avoid ***scope creep,*** the result of four very dangerous words: "While you're at it..."
  - Schedule.
  - Deliverables.
  - Necessary resources.
  - Acceptance criteria.
    - » Must be measurable.
    - » "Timeliness of reporting" versus "report produced by 9am."

17

Principles of Analysis of Design

2. **Partition** large complex problems into smaller, more easily handled ones.
   - Top down, forest first.
3. Highly maintainable **documents** as well as system.
   - Must keep pace with business environment.
   - Avoid redundancy.

18

Principles of Analysis of Design

4. Use **graphics** whenever possible.
   - Can communicate faster, without using as much jargon.
5. Build a **paper model** before real thing.
   - Can test on paper.
   - Can show to users and get their verification.
   - All called "walkthroughs."
     - Why bother?

19

Next

Object Orientation Concepts

20

What is an object?

Business Community

people, places, things

Technical Community

A data structure or function of a data structure. Has properties.

21

What is an Object

22

What is an Object?

A thing or something you want to do with that thing.

23

Instance

- You have a particular car that was manufactured in a factory.
- Your car has a vehicle **identification** number (VIN) that uniquely identifies it.
- Your particular car is an **instance** of a car.

24

Class and Object

- **Car** is the name of the **class** from which this instance was created.
- Each time a new car is manufactured, a new instance from the class of cars is created, and each instance of the car is referred to as an **object**.

25

Object

- An instance of a class of cars
- **Your car** might be the color red, have a black interior, be a convertible or hardtop, and so on.

26

What can you do with an object?

- What can you do with your car?
- You perform certain actions with your car:

  Drive it

  Fill it with gas

  Wash it

  Service it

27

Instance and Methods

- A unique occurrence of a class is an **instance**, and the actions that are performed on the instance are called **methods**.
- In some cases, a method can be applied to an instance of a class or to a class itself.

28

Instance and Methods

- For example, washing *your* car applies to an instance.
- All these methods can be considered **instance methods:**

  drive_it()
  fill_with_gas()
  wash_it()
  Service_it()

29

Instance and Methods

- Finding out how many types of cars a manufacturer makes would apply to the class, so it would be a **class method**.

30

Properties

Suppose there are **two cars** that came off the assembly line and are **almost identical**:

Same interior

Same paint color, etc

31

Object Properties

They might start out the same, but as each car is used by its respective owner, its unique **characteristics or properties** change.

One car may end up with a **scratch on it**, and the other might have **more miles on it**.

32

Object Properties

- Each instance has **initial characteristics** acquired from the factory, plus its **current characteristics**.

- **Object characteristics can change dynamically**. As you drive your car, the gas tank becomes depleted, the car gets dirtier, and the tires get a little more worn.

33

Object State

- Applying a method to an object can affect the **state** of that object.
- If your method is to "fill up my car with gas," after that method is performed, your car's gas tank will be full.
- The method then will have impacted the state of the car's gas tank.

34

Key Concepts

- **Objects** are unique representations from a **class**, and each object contains some information (**data**) that is typically **private to that object**.
- The **methods** provide the means of accessing and changing that data.

35

Identifying Objects using Noun-Verb Analysis

- Look/listen for nouns or noun phrases
- Nouns are words that represent **people, places, things, or ideas**.
- Nouns are words that identify:
  - That person is *John*
  - That place is a *warehouse*
  - That things is a *bike*
  - That idea is *responsibility*

36

Review: Nouns

- **Proper noun** = name of particular noun and are spelled with capital letter
  - New York City, Babe Ruth
- **Common nouns** = do not name particular person, places or things
  - city, athlete, nurse, building
- **Abstract noun** = identifies an idea, quality, or state of mind
  - liberty, intelligence, happiness

37

Nouns

- **Plural nouns** = more than one, noun ends in s, x, ch, z, sh, or ss
  - books, buses, foxes, lunches, waltzes, dishes, bosses
- **Possessive nouns** = nouns that show ownership:
  - Tom's book (possessive of singular noun)
  - Authors' books (possessive of plural noun)
  - Men's race, Children's hour

38

Nouns

- **Collective noun** = a group of persons, places or things
  - class, band, team, audience, United States
- **Predicate noun** = used as a subject complement and follows a linking verb
  - John F. Kennedy was the President back then.

39

Verbs

- A verb is a word that expresses action or a state of being
  - Action: enters, issues, approves
  - State of being: looks, is, were, seems
- Active verb = when the object is performing an action.
  - Ron enters the customer data.
- Transitive verb = action verb that "transmits" the action from the subject to the direct object.
  - The buyer **issued** three purchase orders.

40

Results from Noun-Verb Analysis

- Nouns may/will become objects
- Verb may/will become use cases
- Can you quickly identify objects and a use case from this complete functional requirement statement?

  "The iProc system must allow authorized buyers to create a new item in the item master. Each item contains a unique item number, description, minimum order quantity, purchase price, effective date, and preferred vendor."

41

Summary

We've learned…

- Software Engineering Concepts
- Four Ways of Managing Complexity
- Principles of System Analysis and Design Basic Object Orientation Concepts
- How to quickly identify objects and use cases from a complete functional requirement statement.

42