Software Engineering

CSC440/640
Prof. Schweitzer
Week 6

General Use Case Feedback

- Don't Just Think About the "Happy Path"
  - What are realistic alternative flows?
- Are there any important business rules which can be documented? Required Fields? Unique Constraints, Validations?
- Be Consistent
  - Someone shouldn't be able to look at a document and tell how many different people worked on it. A team document should look like it was written by "the team".
- Individual Team Feedback will be Delivered After Class

Object-Oriented Analysis

- OOA is a semiformal analysis technique for the object-oriented paradigm
  - There are over 60 equivalent techniques
  - Today, the Unified Process is the only viable alternative
- During this workflow
  - The classes are extracted
- Remark
  - The Unified Process assumes knowledge of class extraction

The Analysis Workflow

- The analysis workflow has two aims
  - Obtain a deeper understanding of the requirements
  - Describe them in a way that will result in a maintainable design and implementation
- There are three types of classes:
  - Entity classes
  - Boundary classes
  - Control classes

Entity Class

- Models long-lived information
- Examples:
  - Account Class
  - Investment Class

Boundary Class

- Models the interaction between the product and the environment
- A boundary class is generally associated with input or output
- Examples:
  - Investments Report Class
  - Mortgages Report Class

Control class

- Models complex computations and algorithms
- Example:
  - Estimate Funds for Week Class

UML Notation for These Three Class Types

- Stereotypes (extensions of UML)

Extracting the Entity Classes

- Perform the following three steps incrementally and iteratively
  - Functional modeling
    - Present scenarios of all the use cases (a scenario is an instance of a use case)
  - Class modeling
    - Determine the entity classes and their attributes
    - Determine the interrelationships and interactions between the entity classes
    - Present this information in the form of a class diagram
  - Dynamic modeling
    - Determine the operations performed by or to each entity class
    - Present this information in the form of a statechart

Scenarios

- A use case provides a generic description of the overall functionality
- A scenario is an instance of a use case
- Sufficient scenarios need to be studied to get a comprehensive insight into the target product being modeled

Noun Extraction

- A two-stage process
- Stage 1. Concise problem definition
  - Describe the software product in single paragraph
  - Buttons in elevators and on the floors control the movement of n elevators in a building with m floors. Buttons illuminate when pressed to request the elevator to stop at a specific floor; the illumination is canceled when the request has been satisfied. When an elevator has no requests, it remains at its current floor with its doors closed

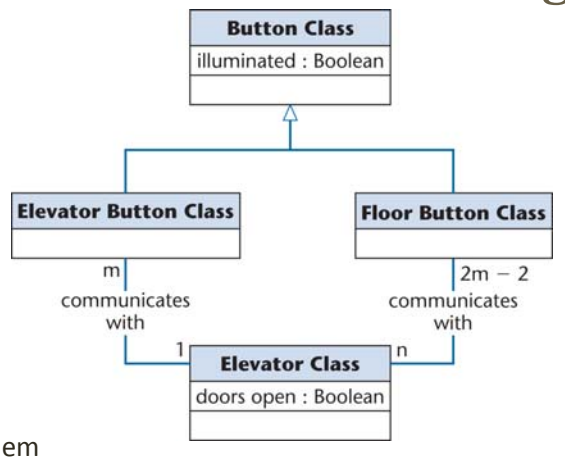
Noun Extraction

- Stage 2. Identify the nouns
  - Identify the nouns in the informal strategy
  - Buttons in elevators and on the floors control the movement of n elevators in a building with m floors. Buttons illuminate when pressed to request the elevator to stop at a specific floor; the illumination is canceled when the request has been satisfied. When an elevator has no requests, it remains at its current floor with its doors closed
- Use the nouns as candidate classes

Noun Extraction

- Nouns
  - button, elevator, floor, movement, building, illumination, request, door
  - floor, building, door are outside the problem boundary — exclude
  - movement, illumination, request are abstract nouns — exclude (they may become attributes)
- Candidate classes:
  - Elevator Class and Button Class
- Subclasses:
  - Elevator Button Class and Floor Button Class

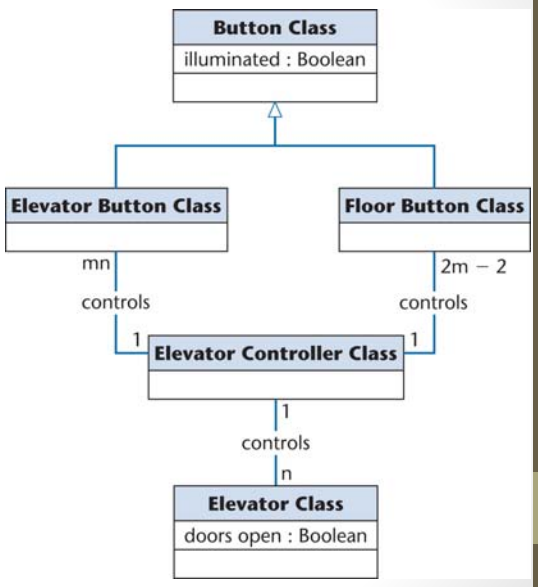
First Iteration of Class Diagram



- Problem
  - Buttons do not communicate directly with elevators
  - We need an additional class: **Elevator Controller Class**

Second Iteration of Class Diagram

- All relationships are now 1-to-n
  - This makes design and implementation easier



CRC Cards

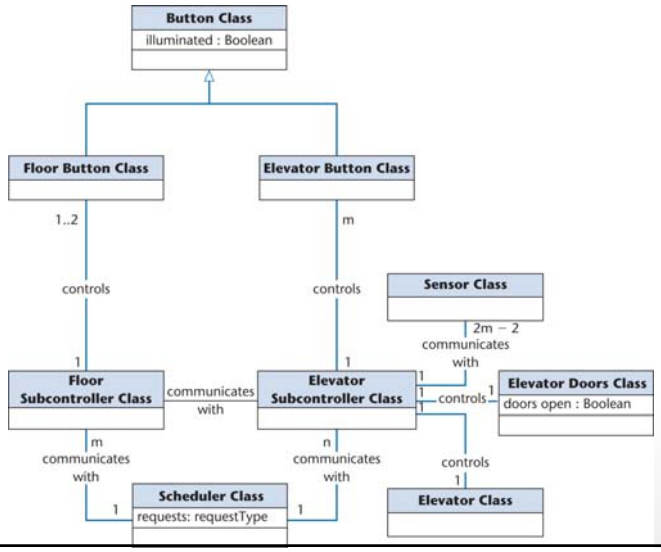
- Used since 1989 for OOA
- For each class, fill in a card showing
  - Name of Class
  - Functionality (Responsibility)
  - List of classes it invokes (Collaboration)
- Now CRC cards are automated (CASE tool component)
- Strength
  - When acted out by team members, CRC cards are a powerful tool for highlighting missing or incorrect items
- Weakness
  - If CRC cards are used to identify entity classes, domain expertise is needed

The Test Workflow: Object-Oriented Analysis

- CRC cards are an excellent testing technique

CLASS **Elevator Controller Class**
RESPONSIBILITY 1. Turn on elevator button 2. Turn off elevator button 3. Turn on floor button 4. Turn off floor button 5. Move elevator up one floor 6. Move elevator down one floor 7. Open elevator doors and start timer 8. Close elevator doors after timeout 9. Check requests 10. Update requests
COLLABORATION 1. **Elevator Button Class** 2. **Floor Button Class** 3. **Elevator Class**

Fourth Iteration of Class Diagram



Extracting the Boundary and Control Classes

- Each
  - Input screen,
  - Output screen, and
  - Report

  is modeled by its own boundary class
- Each nontrivial computation is modeled by a control class

Iteration and Incrementation

- The phrase “iterate and increment” also includes the possibility of having to decrement what has been developed to date
  - A mistake may have been made, and backtracking is needed
  - As a consequence of reorganizing the UML models, one or more artifacts may have become superfluous
  - Called *Refactoring*

Use-Case Realization

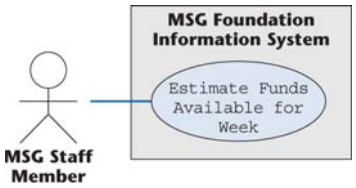
- The process of extending and refining use cases is called use-case realization
- The verb "realize" is used at least 3 different ways:
  - Understand ("Harvey slowly began to realize that he was in the wrong classroom");
  - Receive ("Ingrid will realize a profit of $45,000 on the stock transaction"); and
  - Accomplish ("Janet hopes to realize her dream of starting a computer company")
- In the phrase "realize a use case," the word "realize" is used in this last sense

Use-Case Realization

- The realization of a specific scenario of a use case is depicted using an interaction diagram
  - Either a sequence diagram or collaboration diagram
- Consider use case Estimate Funds Available for Week
- We have previously seen
  - The use case
  - The description of the use case

Estimate Funds Available for Week Use Case

- Use-case diagram



Estimate Funds Available for Week Use Case

- Description of use case

Brief Description

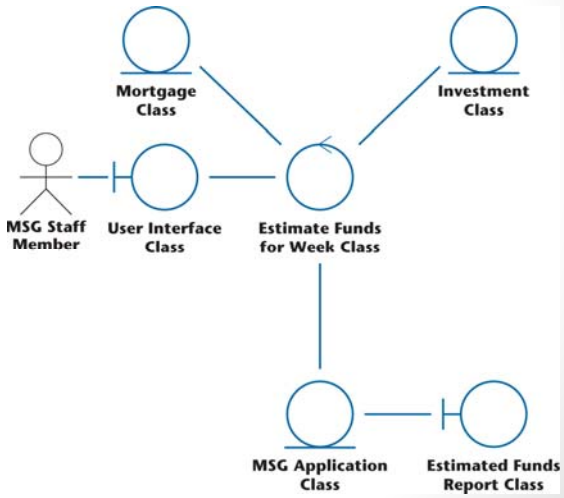
The `Estimate Funds Available for Week` use case enables an MSG Foundation staff member to estimate how much money the Foundation has available that week to fund mortgages.

Step-by-Step Description

1. For each investment, extract the estimated annual return on that investment. Summing the separate returns and dividing the result by 52 yields the estimated investment income for the week.
2. Determine the estimated MSG Foundation operating expenses for the week by extracting the estimated annual MSG Foundation operating expenses and dividing by 52.
3. For each mortgage:
   - 3.1 The amount to be paid this week is the total of the principal and interest payment and $\frac{1}{52}$nd of the sum of the annual real-estate tax and the annual homeowner's insurance premium.
   - 3.2 Compute 28 percent of the couple's current gross weekly income.
   - 3.3 If the result of Step 3.1 is greater than the result of Step 3.2, then the mortgage payment for this week is the result of Step 3.2, and the amount of the grant for this week is the difference between the result of Step 3.1 and the result of Step 3.2.
   - 3.4 Otherwise, the mortgage payment for this week is the result of Step 3.1, and there is no grant this week.
4. Summing the mortgage payments of Steps 3.3 and 3.4 yields the estimated total mortgage payments for the week.
5. Summing the grant payments of Step 3.3 yields the estimated total grant payments for the week.
6. Add the results of Steps 1 and 4 and subtract the results of Steps 2 and 5. This is the total amount available for mortgages for the current week.
7. Print the total amount available for new mortgages during the current week.

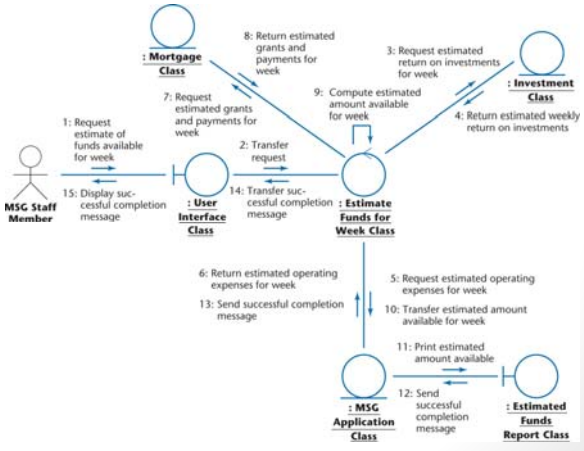
Estimate Funds Available for Week Use Case

- Class diagram (classes that enter into the use case)



Estimate Funds Available for Week Use Case

- Collaboration diagram (of the realization of the scenario of the use case)

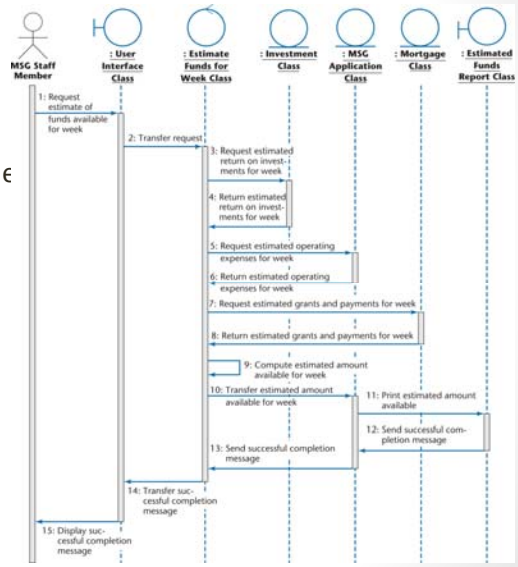


Estimate Funds Available for Week Use Case

- The collaboration diagram shows the objects as well as the messages, numbered in the order in which they are sent in the specific scenario

Estimate Funds Available for Week Use Case

- Sequence diagram equivalent to the collaboration diagram (of the realization of the scenario of the use case)



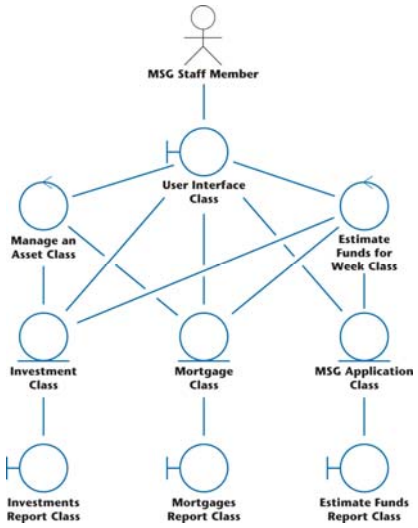
Interaction Diagrams

- The strength of a sequence diagram is that it shows the flow of messages and their order unambiguously
  - When transfer of information is the focus of attention, a sequence diagram is superior to a collaboration diagram
- A collaboration diagram is similar to a class diagram
  - When the developers are concentrating on the classes, a collaboration diagram is more useful than the equivalent sequence diagram

Incrementing the Class Diagram

- In the course of realizing the various use cases
  - Interrelationships between classes become apparent
- Accordingly, we now combine the realization class diagrams

Combining the Realization Class Diagrams



Break

Data and Actions

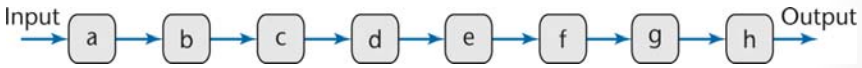
- Two aspects of a product
  - Actions that operate on data
  - Data on which actions operate
- The two basic ways of designing a product
  - Operation-oriented design
  - Data-oriented design
- Third way
  - Hybrid methods
  - For example, object-oriented design

Design and Abstraction

- Classical design activities
  - Architectural design
  - Detailed design
  - Design testing
- Architectural design
  - Input: Specifications
  - Output: Modular decomposition
- Detailed design
  - Each module is designed
    - Specific algorithms, data structures

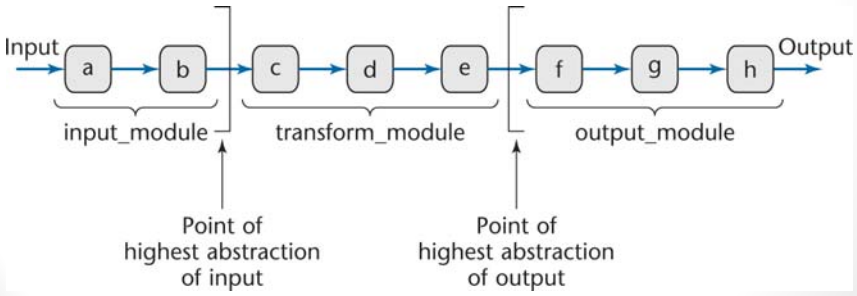
Operation-Oriented Design

- Data flow analysis
  - Use it with most specification methods (Structured Systems Analysis here)
- Key point: We have detailed action information from the DFD



Data Flow Analysis

- Every product transforms input into output
- Determine
  - "Point of highest abstraction of input"
  - "Point of highest abstract of output"

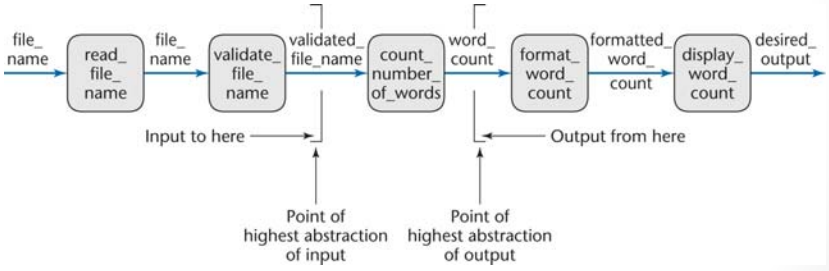


Data Flow Analysis

- Decompose the product into three modules
- Repeat stepwise until each module has high cohesion
  - Minor modifications may be needed to lower the coupling

Mini Case Study: Word Counting

- Example:
  - Design a product which takes as input a file name, and returns the number of words in that file (like UNIX wc)



Data-Oriented Design

- Basic principle
  - The structure of a product must conform to the structure of its data
- Three very similar methods
  - Michael Jackson [1975], Warnier [1976], Orr [1981]
- Data-oriented design
  - Has never been as popular as action-oriented design
  - With the rise of OOD, data-oriented design has largely fallen out of fashion

Object-Oriented Design (OOD)

- Aim
  - Design the product in terms of the classes extracted during OOA
- If we are using a language without inheritance (e.g., C, Ada 83)
  - Use abstract data type design
- If we are using a language without a type statement (e.g., FORTRAN, COBOL)
  - Use data encapsulation

Object-Oriented Design Steps

- OOD consists of two steps:
- Step 1. Complete the class diagram
  - Determine the formats of the attributes
  - Assign each method, either to a class or to a client that sends a message to an object of that class
- Step 2. Perform the detailed design

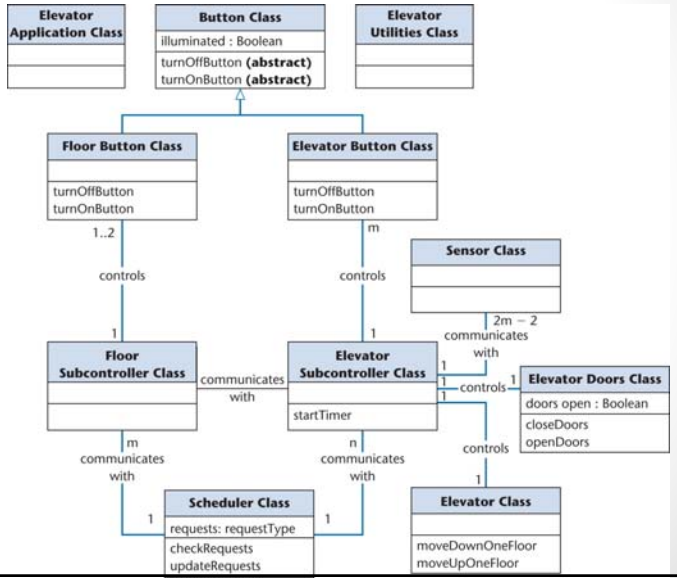
Object-Oriented Design Steps

- Step 1. Complete the class diagram
  - The formats of the attributes can be directly deduced from the analysis artifacts
- Example: Dates
  - U.S. format (mm/dd/yyyy)
  - European format (dd/mm/yyyy)
  - In both instances, 10 characters are needed
- The formats could be added during analysis
  - To minimize rework, never add an item to a UML diagram until strictly necessary

Object-Oriented Design Steps

- Step 1. Complete the class diagram
  - Assign each method, either to a class or to a client that sends a message to an object of that class
- Principle A: Information hiding
- Principle B: If an operation is invoked by many clients of an object, assign the method to the object, not the clients
- Principle C: Responsibility-driven design

Detailed Class Diagram: Elevator Problem



The Design Workflow

- Summary of the design workflow:
  - The analysis workflow artifacts are iterated and integrated until the programmers can utilize them
- Decisions to be made include:
  - Implementation language
  - Reuse
  - Portability

The Design Workflow

- The idea of decomposing a large workflow into independent smaller workflows (packages) is carried forward to the design workflow
- The objective is to break up the upcoming implementation workflow into manageable pieces
  - Subsystems
- It does not make sense to break up the MSG Foundation case study into subsystems — it is too small

The Design Workflow

- Why the product is broken into subsystems:
  - It is easier to implement a number of smaller subsystems than one large system
  - If the subsystems are independent, they can be implemented by programming teams working in parallel
    - The software product as a whole can then be delivered sooner

The Design Workflow

- The *architecture* of a software product includes
  - The various components
  - How they fit together
  - The allocation of components to subsystems
- The task of designing the architecture is specialized
  - It is performed by a software *architect*

The Design Workflow

- The architect needs to make *trade-offs*
  - Every software product must satisfy its functional requirements (the use cases)
  - It also must satisfy its nonfunctional requirements, including
    - Portability, reliability, robustness, maintainability, and security
  - It must do all these things within budget and time constraints
- The architect must assist the client by laying out the trade-offs

The Design Workflow

- It is usually impossible to satisfy all the requirements, functional and nonfunctional, within the cost and time constraints
  - Some sort of compromises have to be made
- The client has to
  - Relax some of the requirements;
  - Increase the budget; and/or
  - Move the delivery deadline

The Design Workflow

- The architecture of a software product is critical
  - The requirements workflow can be fixed during the analysis workflow
  - The analysis workflow can be fixed during the design workflow
  - The design workflow can be fixed during the implementation workflow
- But there is no way to recover from a suboptimal architecture
  - The architecture must immediately be redesigned

The Test Workflow: Design

- Design reviews must be performed
  - The design must correctly reflect the specifications
  - The design itself must be correct
- Transaction-driven inspections
  - Essential for transaction-oriented products
  - However, they are insufficient — specification-driven inspections are also needed

Challenges of the Design Phase

- The design team should not do too much
  - The detailed design should not become code
- The design team should not do too little
  - It is essential for the design team to produce a complete detailed design
- We need to "grow" great designers
- Potential great designers must be
  - Identified,
  - Provided with a formal education,
  - Apprenticed to great designers, and
  - Allowed to interact with other designers
- There must be a specific career path for these designers, with appropriate rewards

Should Architects Code?

Next Steps

- Next Week
  - Chapter 15 for Next Week
  - Open Discussion
- Project Next Steps
  - Class Diagram Due in Two Weeks (11/2)
  - 1 Sequence Diagram Due