Unified Modeling Language

(A Brief Overview)
Thank you Dr. Owen!
Types of Diagrams

Objectives: visualize, specify, construct, and document a system

- **Structural**: focus on static aspects of system

- **Behavioral**: focus on dynamic aspects of system (changing parts)
Structural Diagrams

- **Class**: set of classes and their relationships
  - Interface: is a collection of operations that specify a service of a class

- **Object**: set of objects and their relationships

- **Component**: set of components and their relationships
  - component: physical realization of a logical grouping of elements (e.g., classes, interfaces)

- **Deployment**: set of nodes and their relationships
  - exists at run time; represents computational resource
  - node typically encloses one or more components
Behavioral Diagrams

- **Use case**: organize behaviors of system
  - user goals (high-level services of system)
  - perspective from external entities (actors)

- **Interaction Diagrams**
  - **Sequence**: focus on time ordering of messages
  - **Collaboration**: focus on structural organization of objects that send/receive messages

- **Statechart**: changing state of system driven by events

- **Activity**: focus on flow of control from one activity to another
Development Process

- High-Level capture of requirements
  - Use Case Diagram

- Identify major objects and relationships
  - Class diagram (object diagram)

- Create scenarios of usage
  - Interaction Diagrams
    - Sequence Diagram
    - Collaboration Diagram

- Generalize scenarios to describe behavior
  - State Diagram
  - Activity Diagram

- Refine to add implementation details
  - Implementation Diagrams
    - Component Diagram
    - Deployment Diagram
Use Cases and Scenarios
What is a Use-Case

- A use-case captures some user visible function
- This may be a large or small function
  - Depends on the level of detail in your modeling effort
- A use-case achieves a discrete goal for the user
- Examples
  - Format a document
  - Request an elevator
- How are the use cases found (captured or elicited)?
User Goals versus User Interactions

- Consider the following when formatting a document
  - Define a style
  - Change a style
  - Copy a style from one document to the next
    - versus
  - Format a document
  - Ensure consistent formatting of two documents

- The latter is a user goal
  - Something the user wants to achieve

- The former are user interactions
  - Things the user does to the system to achieve the goal
Goals and Interactions

● There is a place for both goals and interactions

● Understand *what* the system shall do
  ▪ Capture the user goals

● Understand *how* the user will achieve the goals
  ▪ Capture user interactions
  ▪ Sequences of user interactions

● Thus, start with the user goals and then refine the user goals into several (many) user interactions
Use-Case Diagrams (POST)

POST: Point of Sale Terminal

System Boundary

Use Case

POST

Buy Item

Log In

Refund a Purchased Item

Cashier

Customer

Adapted from Larman "Applying UML and Patterns"
Another Example

Financial Trading System

- Set Limits
- Update Accounts
- Analyze Risk
  - Value includes
- Price Deal
  - «includes»
- Capture Deal
  - «extends»
- Limit Exceeded

- Accounting System
- Trading Manager
- Trader
- Salesperson

Adapted from Fowler "UML Distilled"
Includes and Extends

- **Includes**
  - You have a piece of behavior that is similar across many use cases
  - Break this out as a separate use-case and let the other ones “includes” it
  - Examples include
    - Valuation
    - Validate user interaction
    - Sanity check on sensor inputs
    - Check for proper authorization

- **Extends**
  - A use-case is similar to another one but does a little bit more
  - Put the normal behavior in one use-case and the exceptional behavior somewhere else
    - Capture the normal behavior
    - Try to figure out what can go wrong in each step
    - Capture the exceptional cases in separate use-cases
  - Makes it a lot easier to understand
Includes

- You have a piece of behavior that is similar across many use cases

- Break this out as a separate use-case and let the other ones “include” it

- Examples include
  - Valuation
  - Validate user interaction
  - Sanity check on sensor inputs
  - Check for proper authorization
Extends

- A use-case is similar to another one but does a little bit more

- Put the normal behavior in one use-case and the exceptional behavior somewhere else
  - Capture the normal behavior
  - Try to figure out what can go wrong in each step
  - Capture the exceptional cases in separate use-cases

- Makes it a lot easier to understand
Setting the System Boundary

- The system boundary will affect your actors and use-cases
A Different Boundary

- Let us view the whole store as our system
Partial POST

POST

Buy Item

Log In

Refund a Purchased Item

Start Up

Manage Users

And a Lot More

Cashier

Customer

System Administrator

Manager

© 2007-08 Betty H.C. Cheng. This presentation is available free for non-commercial use with attribution under a creative commons license.
POST Use-Case

Use case: Buy Item

Actors: Customer (initiator), Cashier

Type: Primary

Description: The Customer arrives at the checkout with items to purchase.

The Cashier records the purchase items and collects a payment. On completion the Customer leaves with the items.
Use case: Buy Item

Actors: Customer (initiator), Cashier

Type: Primary and essential

Description: The Customer arrives at the checkout with items to purchase. The Cashier records the purchase items and collects a payment. On completion the Customer leaves with the items.

Cross Ref.: Requirements XX, YY, and ZZ

Use-Cases: Cashier must have completed the Log In use-case
Typical Course of Events

Actor Action
1. This use-case begins when a user arrives at the checkout
2. The cashier records purchase items
3. The cashier collects payment
4. The user leaves with items
Home Heating System

Image: http://www.heatingoil.com/

Image: http://www.homeheatingdr.com/

CSE 435: Software Engineering

© 2007-08 Betty H.C. Cheng. This presentation is available free for non-commercial use with attribution under a creative commons license.
Home Heating System

- Fuel Valve
- Water Pump
- Controller
- Hot Water
- Burner
- Water Valve
- Home
- Temp Sensor
- Control Panel
- Fuel
Home Heating Requirements

The purpose of the software for the Home Heating System is to control the heating system that heats the rooms of a house. The software shall maintain the temperature of each room within a specified range by controlling the heat flow to individual rooms.

- The software shall control the heat in each room.
- The room shall be heated when the temperature is 2F below desired temp.
- The room shall no longer be heated when the temperature is 2F above desired temp.
- The flow of heat to each room shall be individually controlled by opening and closing its water valve.
- The valve shall be open when the room needs heat and closed otherwise.
- The user shall set the desired temperature on the thermostat.
- The operator shall be able to turn the heating system on and off.
- The furnace must not run when the system is off.
Home Heating Requirements

The purpose of the software for the Home Heating System is to control the heating system that heats the rooms of a house. The software shall maintain the temperature of each room within a specified range by controlling the heat flow to individual rooms.

- When the furnace is not running and a room needs heat, the software shall turn the furnace on.
- To turn the furnace on the software shall follow these steps:
  - open the fuel valve
  - turn the burner on
- The software shall turn the furnace off when heat is no longer needed in any room.
- To turn the furnace off the software shall follow these steps:
  - close fuel valve
  - turn burner off
Home Heating Use-Case Diagram

- Home Owner
  - Home Heating
    - Power Up
    - Power Down
    - Change Temp.
Home Heating Use-Cases

Use case: Power Up

Actors: Home Owner (initiator)

Type: Primary and essential

Description: The Home Owner turns the power on. Each room is temperature checked. If a room is below the desired temperature the valve for the room is opened, the water pump started. If the water temp falls below threshold, the fuel valve is opened, and the burner ignited.

If the temperature in all rooms is above the desired temperature, no actions are taken.

Cross Ref.: Requirements XX, YY, and ZZ

Use-Cases: None
Modified Home Heating

Home Heating

- Power Up
- Power Down
- Change Temp.

Adjust Temp

Temp. High

Temp. Low

«includes»

Home Owner

© 2007-08 Betty H.C. Cheng. This presentation is available free for non-commercial use with attribution under a creative commons license.
Modified: Home Heating Use-Cases

Use case: Power Up
Actors: Home Owner (initiator)
Type: Primary and essential
Description: The Home Owner turns the power on. Perform Adjust Temp. If the temperature in all rooms is above the desired temperature, no actions are taken.

Cross Ref.: Requirements XX, YY, and ZZ
Use-Cases: Perform Adjust Temp
Modified:
Home Heating Use-Cases

Use case: Adjust Temp
Actors: System (initiator)
Type: Secondary and essential
Description: Check the temperature in each room. For each room below target, open room valve, start pump if not started. If water temp falls below threshold, open fuel value and ignite burner.
Cross Ref.: Requirements XX, YY, and ZZ
Use-Cases: Temp-Low, Temp-high
HACS

- Homework assignment and collection are an integral part of any educational system. Today, this task is performed manually. What we want the homework assignment distribution and collection system (HACS for short) to do is to automate this process.

- HACS will be used by the instructor to distribute the homework assignments, review the students’ solutions, distribute suggested solution, and distribute student grades on each assignment.

- HACS shall also help the students by automatically distributing the assignments to the students, provide a facility where the students can submit their solutions, remind the students when an assignment is almost due, remind the students when an assignment is overdue.
HACS Use-Case Diagram

System Admin

Instructor

HACS

Configure HACS
Distribute Assignments
Post Solutions
Distribute Grade

Remind Student
Get Assignment
Submit Assignment
Get Solution
Get Grade

Student
Use case: Distribute Assignments

Actors: Instructor (initiator)

Type: Primary and essential

Description: The Instructor completes an assignment and submits it to the system. The instructor will also submit the due date and the class the assignment is assigned for.

Cross Ref.: Requirements XX, YY, and ZZ

Use-Cases: Configure HACS must be done before any user (Instructor or Student) can use HACS
Alternate HACS

HACS

Configure HACS

Distribute Assignments

Post Solutions

Distribute Grade

Remind Student

Submit Assignment

System Admin

Instructor

Student
Alternate HACS Use-Cases

Use case: Distribute Assignments
Actors: Instructor (initiator), Student
Type: Primary and essential
Description: The Instructor completes an assignment and submits it to the system. The instructor will also submit the delivery date, due date, and the class the assignment is assigned for. The system will at the due date mail the assignment to the student.

Cross Ref.: Requirements XX, YY, and ZZ
Use-Cases: Configure HACS must be done before any user (Instructor or Student) can use HACS
When to use Use-Cases

- always!!!

- Requirements is the toughest part of software development
  - Use-Cases is a powerful tool to understand
    - Who your users are (including interacting systems)
    - What functions the system shall provide
    - How these functions work at a high level

- Spend adequate time on requirements and in the elaboration phase
How it Fits Together

Preliminary Investigation Report

Requirements Specification

Use-Cases
  a. All High Level
  b. Some Expanded

Use-Case Diagram

Draft Conceptual Model

Glossary (data dictionary)

Prototypes

Budget, Schedule

Adapted from Larman “Applying UML and Patterns”