Building the Analysis Model 3

Suradet Jitprapaikulsarn

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Object-Oriented Analysis (OOA)

- Scenario-based model ⇨ Use-case, Activity diagram, Swim lane diagram
- Class-based model ⇨ Class diagram, Analysis package, CRC model, Collaboration diagram
- Flow-oriented model ⇨ Data Flow diagram, Control Flow diagram Process narrative
- Behavior model ⇨ State diagram, Sequence diagram

Object-Orient Concepts

- Must be understood to apply class-based elements of the analysis model
- Key concepts:
  - Classes and objects
  - Attributes and operations
  - Encapsulation and instantiation
  - Inheritance

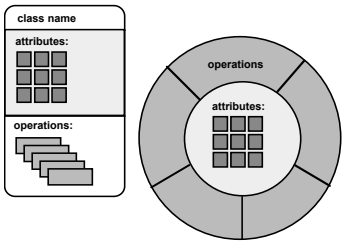
Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Classes

- object-oriented thinking begins with the definition of a **class**, often defined as:
  - template
  - generalized description
  - "blueprint" ... describing a collection of similar items
- a **metaclass** (also called a **superclass**) establishes a hierarchy of classes
- once a class of items is defined, a specific instance of the class can be identified

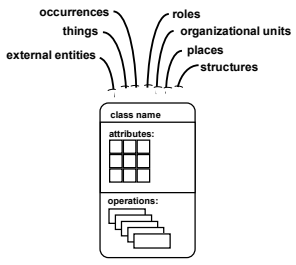
Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Building a Class



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

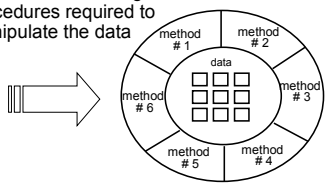
What is a Class?



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Encapsulation/Hiding

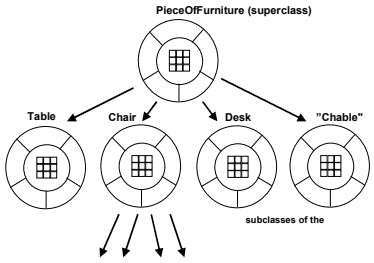
The object encapsulates both data and the logical procedures required to manipulate the data



Achieves “information hiding”

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

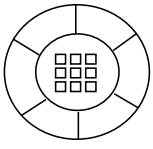
Class Hierarchy



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Method (a.k.a. Operation, Service)

An executable procedure that is encapsulated in a class and is designed to operate on one or more data attributes that are defined as part of the class.
A method is invoked via message passing.



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Scenario-Based Modeling

"[Use-cases] are simply an aid to defining what exists outside the system (actors) and what should be performed by the system (use-cases)." Ivar Jacobson

1. What should we write about?
2. How much should we write about it?
3. How detailed should we make our description?
4. How should we organize the description?

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Use-Cases

- a scenario that describes a "thread of usage" for a system
- ***actors*** represent roles people or devices play as the system functions
- ***users*** can play a number of different roles for a given scenario

Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Use-Cases

- A collection of user scenarios that describe the thread of usage of a system
- Each scenario is described from the point-of-view of an "actor"—a person or device that interacts with the software in some way
- Each scenario answers the following questions:
  - Who is the primary actor, the secondary actor (s)?
  - What are the actor's goals?
  - What preconditions should exist before the story begins?
  - What main tasks or functions are performed by the actor?
  - What extensions might be considered as the story is described?
  - What variations in the actor's interaction are possible?
  - What system information will the actor acquire, produce, or change?
  - Will the actor have to inform the system about changes in the external environment?
  - What information does the actor desire from the system?
  - Does the actor wish to be informed about unexpected changes?

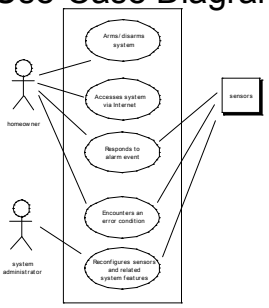
Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Use-Case Template

- **Name**
- **Identifier [Optional]**
- **Description**
- **Actors [Optional]**
- **Status [Optional]**
- **Frequency**
- **Pre-conditions**
- **Post-conditions**
- **Extended use case [Optional]**
- **Included use cases [Optional]**
- **Assumptions [Optional]**
- **Basic course of action**
- **Alternate course of action**
- **Change history [Optional]**
- **Issues [Optional]**
- **Decisions**

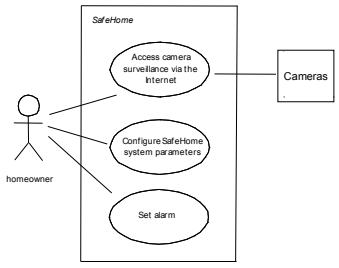
Derived from Scott W. Ambler, *The Object Primer*, 2nd Edition, Cambridge University Press, 2001

Use-Case Diagram



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

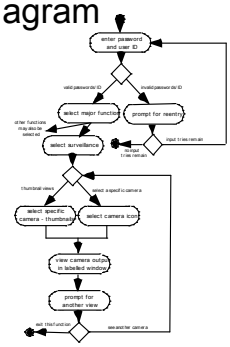
Use-Case Diagram



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Activity Diagram

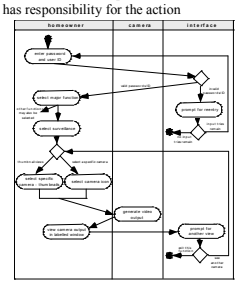
Supplements the use-case by providing a diagrammatic representation of procedure flow



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005

Swimlane Diagram

Allows the modeler to represent the flow of activities described by the use-case and at the same time indicate which actor (if there are multiple actors involved in a specific use-case) or analysis class has responsibility for the action described by an activity rectangle



Derived from Roger S. Pressman, *Software Engineering: A Practitioner's Approach*, 6th Edition, McGraw-Hill, 2005