Principles

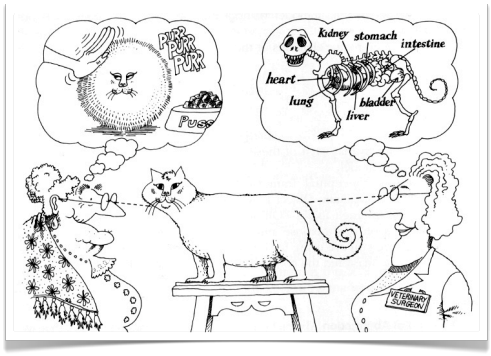
of object-oriented design

- Abstraction – Hide details
- Encapsulation – Keep changes local
- Modularity – Control information flow
  High cohesion • weak coupling • talk only to friends
- Hierarchy – Order abstractions
  Classes open for extensions, closed for changes • Subclasses that do not require more or deliver less • depend only on abstractions

Goal: *Maintainability* and *Reusability*

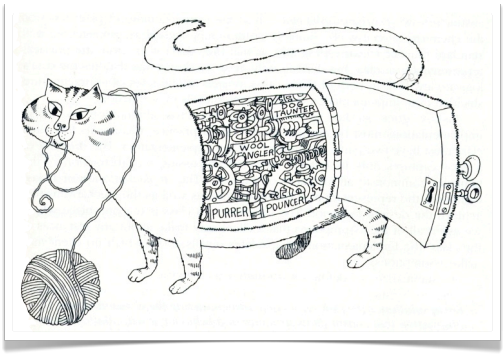
1

Abstraction



2

Encapsulation



3

Modularity

4

Principles of Modularity

- High cohesion – Modules should contain functions that logically belong together
- Weak coupling –Changes to modules should not affect other modules
- Law of Demeter – talk only to friends

Call your Friends

A method M of an object O should only call methods of

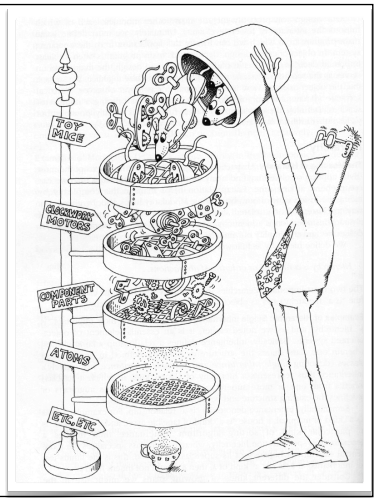
1. O itself
2. M's parameters
3. any objects created in M
4. O's direct component objects

"single dot rule"

http://en.wikipedia.org/wiki/Law_of_Demeter

Hierarchy

"Hierarchy is a ranking or ordering of abstractions."



Hierarchy principles

- Open/Close principle – Classes should be open for extensions
- Liskov principle – Subclasses should not require more, and not deliver less
- Dependency principle – Classes should only depend on abstractions

From Requirements to Design

- Describe requirements as *use cases*
- Refine use cases to *alternate scenarios*
- Identify *classes* and *operations*

See Pressman, chapter 8 for the remainder of this lecture

Initial Use Case

Use case: *display camera views*
Actor: *homeowner*

If I'm at a remote location, I can use any PC with appropriate browser software to log on to the SafeHome Web site. I enter my user ID and two levels of passwords and, once I'm validated, I have access to all the functionality. To access a specific camera view, I select "surveillance" and then "select a camera". Alternatively, I can look at thumbnail snapshots from all cameras by selecting "all cameras". Once I choose a camera, I select "view"…

SAFEHOME

Use-Case Template for Surveillance

Use-case: Access camera surveillance—display camera views (ACS-DCV).
Primary actor: Homeowner.
Goal in context: To view output of camera placed throughout the house from any remote location via the Internet.
Preconditions: System must be fully configured; appropriate user ID and passwords must be obtained.
Trigger: The homeowner decides to take a look inside the house while away.

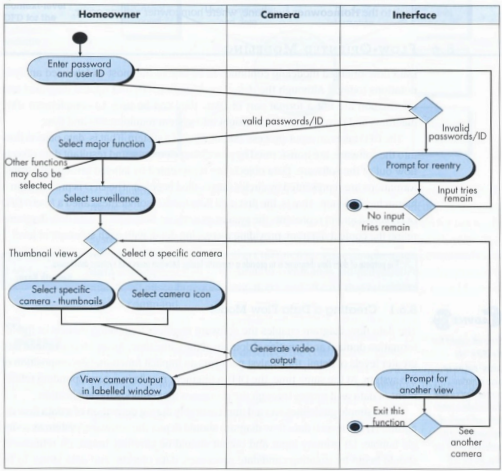
Scenario:

1. The homeowner logs onto the *SafeHome Products* Web site.
2. The homeowner enters his or her user ID.
3. The homeowner enters two passwords (each at least eight characters in length).
4. The system displays all major function buttons.
5. The homeowner selects "surveillance" from the major function buttons.
6. The homeowner selects "pick a camera."
7. The system displays the floor plan of the house.
8. The homeowner selects a camera icon from the floor plan.
9. The homeowner selects the "view" button.
10. The system displays a viewing window that is identified by the camera ID.
11. The system displays video output within the viewing window at one frame per second.

Exceptions:

1. ID or passwords are incorrect or not recognized—see use-case: "validate ID and passwords."
2. Surveillance function not configured for this system—system displays appropriate error message; see use-case: "configure surveillance function."
3. Homeowner selects "view thumbnail snapshots for all cameras"—see use-case: "view thumbnail snapshots for all cameras."
4. A floor plan is not available or has not been configured—display appropriate error message and see use-case: "configure floor plan."
5. An alarm condition is encountered—see use-case: "alarm condition encountered."

Priority: Moderate priority, to be implemented after basic functions.
When available: Third increment.
Frequency of use: Infrequent.



Swimlane diagram for Access camera surveillance–display camera views functions

Requirements for

Potential Classes

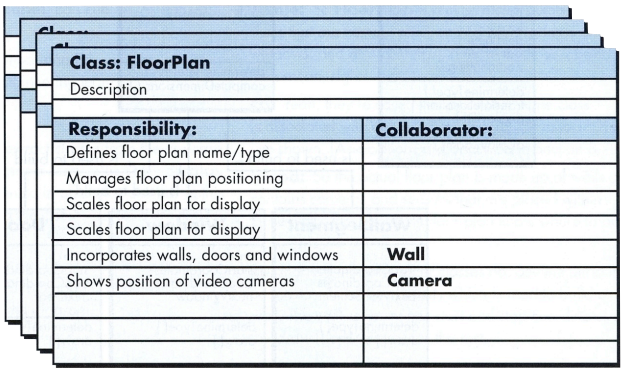
1. Retained Information
   The information is necessary for the system to function
2. Needed Services
   The potential class must have a set of potential operations
3. Multiple Attributes
   We are focusing on potential classes with more than one attribute
4. Common Attributes and Operations
   The attributes and operations apply to all instances of the class
5. Essential Requirements
   External entities – producers and consumers of information – almost always become classes

These are requirements a potential class has to fulfill to be retained

Classes and Methods

- *Class-Responsibility-Collaborator (CRC) modeling* is a simple means for identifying and organizing classes
- Makes use of virtual or actual *index cards*

A CRC index card



Class: FloorPlan	
Description	
Responsibility:	**Collaborator:**
Defines floor plan name/type	
Manages floor plan positioning	
Scales floor plan for display	
Scales floor plan for display	
Incorporates walls, doors and windows	**Wall**
Shows position of video cameras	**Camera**

CRC Responsibilities

- System intelligence should be *distributed across classes* (→ modularity)
- State responsibilities *as general as possible* (→ abstraction)
- Information and related behavior goes into the same class (→ encapsulation)
- Information about one thing should be *localized in a single class* (→ modularity)
- Responsibilities should be *shared among related classes* (→ hierarchy)