From Requirements to Design

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

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”…

[Diagram of a use-case diagram showing the flow of interactions between the Homeowner, Camera, and Interface. The diagram includes stages for entering passwords, selecting surveillance or cameras, and viewing video output.]
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

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

<table>
<thead>
<tr>
<th>Class: FloorPlan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Responsibility:</td>
</tr>
<tr>
<td>Defines floor plan name/type</td>
</tr>
<tr>
<td>Manages floor plan positioning</td>
</tr>
<tr>
<td>Scales floor plan for display</td>
</tr>
<tr>
<td>Scales floor plan for display</td>
</tr>
<tr>
<td>Incorporates walls, doors and windows</td>
</tr>
<tr>
<td>Shows position of video cameras</td>
</tr>
</tbody>
</table>

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)