**Modeling Requirements Use Cases**

**Use Case**
- A case or situation where a system is used to fulfill one or more of users’ requirements
- Capture requirements
  - “Shall” requirements and “Should” requirements
- Affects every other facet of system design
- Starting point for test case design
- Priority and risk analysis can be performed for use cases

**Actor**
- Things outside your system
  - Interact with the system
  - Doesn’t have to be actual people
  - Cannot control inside the system
  - Can be further refined

- A Contents Management System (CMS) Example

**Linking Actors and Use Cases**
- Identify use cases
  - System functionality used by actor(s)
  - Complexity can be different across use cases
  - Provide some measurable results to the users
- Communication lines
  - Show “involvement” relations
- System boundary
Use Case Description

- Diagrams are not enough
  - No specific rules for the description
- Example contents
  - Goal in context
  - Preconditions
  - Steps to execute
  - Successful end condition
  - Failed end condition
  - Primary (and secondary) actors
  - Trigger
  - Main flow
  - Extensions …
- Use case description completes the use case!

Use Case Relationships

- In essence, use case relations are about breaking down system functionality into chunks.
  - Inclusion
  - Specialization
  - Multiple flows
- Inclusion <<include>>
  - Represent shared behavior in a separate use case
  - Means "complete" reuse of shared behavior

Example: <<include>> relationship

The content management system shall allow an administrator to create a new personal Wiki, provided the personal details of the applying author are verified using the Author Credentials Database.

Example: Create a new Blog Account

<table>
<thead>
<tr>
<th>Use case name</th>
<th>Create a new Blog Account</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related Requirements</td>
<td>Requirement A.1</td>
</tr>
<tr>
<td>Goal in Context</td>
<td>A new existing author requests a new blog account from the Administrator,</td>
</tr>
<tr>
<td>Preconditions</td>
<td>The system is limited to recognized authors and so the author needs to have appropriate proof of identity.</td>
</tr>
<tr>
<td>Successful End Condition</td>
<td>A new blog account is created for the author.</td>
</tr>
<tr>
<td>Failed End Condition</td>
<td>The application for a new blog account is rejected.</td>
</tr>
<tr>
<td>Primary Actors</td>
<td>Administrator.</td>
</tr>
<tr>
<td>Secondary Actors</td>
<td>Author Credentials Database.</td>
</tr>
<tr>
<td>Trigger</td>
<td>The Administrator asks the CMS to create a new blog account.</td>
</tr>
<tr>
<td>Main Flow</td>
<td>Create a new Blog Account</td>
</tr>
<tr>
<td></td>
<td>Action</td>
</tr>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>The Administrator asks the system to create a new blog account.</td>
</tr>
<tr>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>The Administrator selects an account type.</td>
</tr>
<tr>
<td></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>The Administrator enters the author's details.</td>
</tr>
<tr>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>The author's details are verified using the Author Credentials Database.</td>
</tr>
<tr>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>The new blog account is created.</td>
</tr>
<tr>
<td></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>A summary of the new blog account's details are emailed to the author.</td>
</tr>
<tr>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Branding Action</td>
</tr>
<tr>
<td></td>
<td>The Author Credentials Database does not verify the author's details.</td>
</tr>
<tr>
<td></td>
<td>The author's new blog account application is rejected.</td>
</tr>
</tbody>
</table>

Benefit?
Use Case Relationships

- Specialization
  - Some use cases work in different modes or special cases
  - A use case with small changes for a collection of specific situations – i.e., use case generalization (or inheritance)

- Only need to specify extra or modified steps for specific cases
- Inherit both behaviors and use case relationships
  - Every step in the general use case has to occur in the specialized use cases.
  - Use case relationships in the general use case has to be inherited.

Example: inheritance relationship

Use Case Relationships

- Shared optional behavior <<extend>>
  - Similar to <<include>>, but the reuse is optional, dependent on runtime or system implementation decision

Use Case Overview Diagram

- Sketch a bird’s eye view system’s context or domain
Modeling System Workflows with Activity Diagrams

**Modeling System Workflow**

- Compared to use cases, activity diagram is to specify how your system will accomplish its goals.
- Visual representation of use case’s flow
- Activity diagram is object-oriented equivalent of flow chart and data-flow diagram
- Process view of system model

**Example**

- Consider a use case “create a new blog account” discussed previously
  - Activity diagram visualize steps for use cases better
- Modeling notation
  - Initial node
  - Final node
  - Action (≠ activity)
  - Edge
  - Decision

**Activity Frame, Decisions, and Merges**

- Activity frame is useful to show multiple activities in a diagram
  - can be omitted
- Decisions are used to model different sequence of actions depending on a condition
  - Branched flows join at a merge node

**Why need merge node?**
Activity Frame, Decisions, and Merges

- Beware of having multiple guards evaluated to true
  - Which one to follow?

Concurrent Tasks and Timed Event

- Represent parallel tasks by forks and joins
  - Fork
  - Join

- Represent timed events with hourglass symbol

Inter-Activity Calls

- Useful to avoid too big diagram and not to repeat same action sequence multiple times

Showing Objects

- Important data objects can be modeled with actions in an activity diagram
  - Multiple ways to show objects
    - Object node
    - Action inputs and outputs
    - Same object in different state can appear multiple times
    - Input to and output from an activity
Showing Objects

- Showing object nodes

- Action inputs and outputs using pins

Sending and Receiving Signals

- Represent interactions with external entities
  - Sent asynchronously and wait at receiving action (behaves like a synchronous call)

Ending Activities and Flows

- Interrupted flow
  - Interruption region

- Flow final node
  - Terminates its own path – not the whole activity
Swimlanes (or Partitions)

- Represent different actors (or groups, roles) in a system or in an organization in a diagram

Complex Activity Diagrams

- Connectors to untangle diagrams
  - Improve readability
- Expansion region for repeated activity for a data collection