

UML Diagram Types

Dynamic Models

- activity diagrams
- statechart diagrams
- interaction diagrams
 - sequence diagrams
 - collaboration diagrams
- *use case diagrams*

Structural Models

- class diagrams
- object diagrams
- packages

Architectural Models

- component diagrams
- deployment diagrams


Use Case

def'n: a description of a set of sequences of actions, including variants, that a system performs to yield an observable result of value to an actor

- every interesting system interacts with human or automated actors
- specifies the behavior of a system or part of a system
- captures intended behavior, without specifying how behavior is implemented
- aids common understanding of end users and domain experts
- denote essential system or subsystem behavior
- basis for test cases as they evolve during development


Use Case Definition

- *set of sequences:* each sequence represents the interaction of the things outside the system and within the system
- system level functions that help visualize, construct, and document the intended behavior of your system during requirements capture and analysis
- represents functional requirements of system as a whole




Use Case Definition

- *interaction with actors*: coherent set of roles external to the system
- can be human, or automated



Use Case Definition

- *variants*: differences between closely related used cases
- specialized versions of other use cases
- common parts of other use cases
- extend the behavior of other use cases



Use Case Definition

- *tangible work*: some measurable accomplishment that, from the perspective of a given actor, is of value
- system level functions that help visualize, construct, and document the intended behavior of your system during requirements capture and analysis
- represents functional requirements of system as a whole

Use Case

Convention

- ellipse
- simple name (name) or path name (package::name)
- may have attributes, operations, state behavior

Actor

def'n: a coherent set of roles that users play when interacting with a use case

- role that a human, hardware device, or another system plays with system
- live outside of the system

Convention

- stick figure
- actors can be specializations of each other
- connected to use case by association, indicating communication between use case and actor

Flow of Events

- describe flow of events clearly enough for outsider to understand
- include how/when a use case starts/ends
- when the use case interacts with actors
- what objects are exchanged
- basic flow of alternative flows of behavior

Convention

- informal structured text
- formal structured text (pre and post conditions)
- pseudocode

Scenarios

def'n: specific sequence of actions that illustrates behavior

- scenarios: use cases as instances: classes
- first describe with text (flow of events)
- next, describe with interaction diagrams
 - main flow
 - exceptional flow

Organizing Use Cases

- *generalization*
 - similar to generalization between classes
 - e.g., child inherits behavior and meaning of parent use case
 - child may override or add to behavior of parent
 - child may be substituted for parent

Organizing Use Cases con't

- *include*
 - base use case explicitly incorporates the behavior of another use case at a location specified in the base
 - included base class never stands alone, but is instantiated as part of some larger base
 - avoids redundant description of same flow of events
- *Convention*
 - dependency from base use case to another use case
 - stereotype <<include>> or <<uses>> on dependency

Organizing Use Cases con't

■ *extend*

- base use case implicitly incorporates the behavior of another use case at a location specified indirectly by the extending use case
- base use case may stand alone, but under certain conditions behavior may be extended by behavior of another use case
- models part of a use case that the user may see as optional behavior
 - separates optional from mandatory behavior
 - model a conditionalsubflow
 - model several subflows that may be inserted at a certain point

■ Convention

- dependency from extended use case to base use case
- stereotype <<extend>> or <<extends>> on dependency

What it Means

■ Generalizing a Set of Behavior

- use generalization

■ Extracting Common Behavior

- use include

■ Distinguishing Variants

- use extend

Modeling Techniques

■ model behavior of an element


- entire system
- subsystem
- class

■ focus on what, not how

■ forum for domain experts and developers to meet on common ground


■ provide method of decomposition of complex problem

■ basis for testing each element during development



To Model

- Identify actors that interact with the element
- Organize actors into general and specialized roles
- Consider common interactions with use cases
- Consider exceptional interactions with use cases
- Organize behaviors using include and extend relationships



Hints and Tips

- Name a reasonable partition of the system
- Factor common behavior
- Factor variant behavior
- Describes flow of events clearly enough for outsider to understand
- Use scenarios that specify normal and variant behavior
