USE CASE MODELING

Hoang Huu Hanh, Hue University
hanh-at-hueuni.edu.vn

Agenda

- What is a Use Case?
- Benefits of the Use Cases
- Use Cases vs. Requirements document
- Developing the Use Case model
  - System
  - Actor
  - Use Case
  - Use Case Relationships
- Example: TVRS Use Cases

Hue University

What is a Use Case?

- Created by Ivar Jacobson (1994)
- "A use case is a sequence of transactions in a system whose task is to yield a measurable value to an individual actor of the system"
- Describes WHAT the system (as a "Black Box") does from a user's perspective
- A set of scenarios tied together by a common user goal
- The Use Case Model is NOT an inherently object oriented modeling technique

Hue University

Benefits of Use Cases

- Captures functional requirements from user's perspective
- Gives a clear and consistent description of what the system should do
- A basis for performing system tests
- Provides the ability to trace functional requirements into actual classes and operations in the system

Benefits of Use Cases (cont.)

- Serves as a unit of estimation
- The smallest unit of delivery
  - Each increment that is planned and delivered is described in terms of the Use Cases that will be delivered in that increment

Use Cases vs. Requirements

- A requirements document states what the system shall do. Use Cases describe actions that the users take and the responses that the system generates
- Use Cases are sometimes used as a means to elicit requirements
- Requirements may be effectively documented as Use Cases
  - Better traceability
  - Easier user validation of functional requirements
  - Helps structuring the users manuals
  - A tool for finding classes

Use Cases vs. Requirements

- Use-Cases are not well suited for capturing non-functional requirements.
- We will view use-cases as one piece of our software requirements specification

UML's Use Case Diagrams

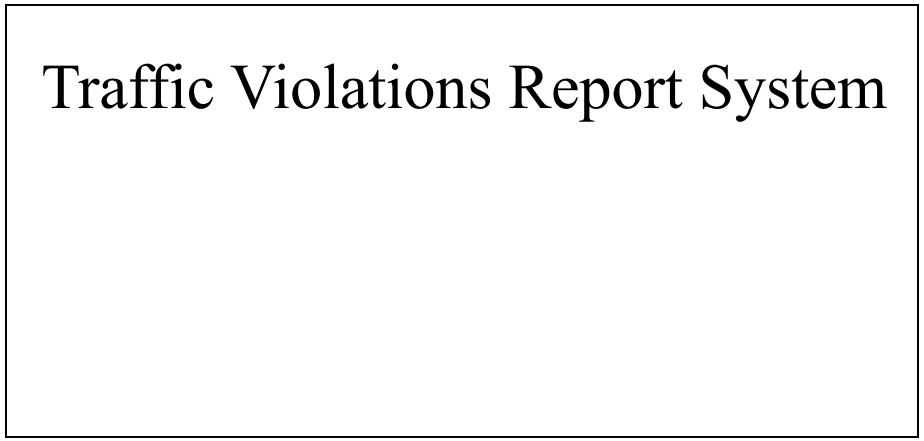
- A Use Case model is described in UML as one or more Use Case Diagrams (UCDs)
- A UCD has 4 major elements:
  - The system described
  - The actors that the system interacts with
  - The use-cases, or services, that the system knows how to perform
  - The relationships between the above elements

System

- As part of use-case modeling, the boundaries of the system developed must be defined
- A system does not necessarily have to be a software system
- Defining the boundaries of the system is not trivial
  - Which tasks are automated and which are manual?
  - Which tasks are performed by other systems?
    - The entire solution that we supply should be included in the system boundaries
    - Incremental releases

System (cont.)

- A system in a UCD is represented as a box
- The name of the system appears above or inside the box



Actor

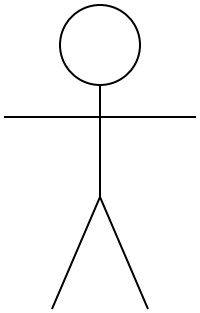
- Someone or something that interacts with the system (exchanges information with the system)
- An actor represents a role played with respect to the system, not an individual user of the system
- Example:
  - Clerk – Enters data
  - Supervisor – Allowed to modify/erase data
  - Manager – Allowed to view statistics.
- A single user may play more than one role

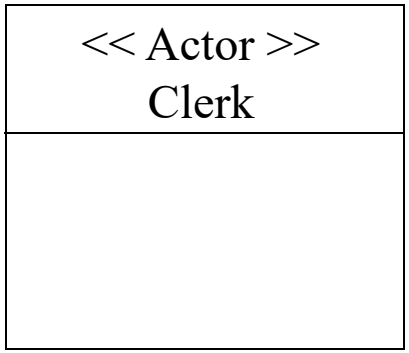
Actor (cont.)

- Actors have goals:
  - Add a Traffic Violation
  - Lookup a Traffic Violation
- Actors don't need to be human
  - May be an external system that interfaces with the developed system
- An actor has a name that reflects its role
- In this course a use case is always initiated by an actor

Hue University

Actor Icons

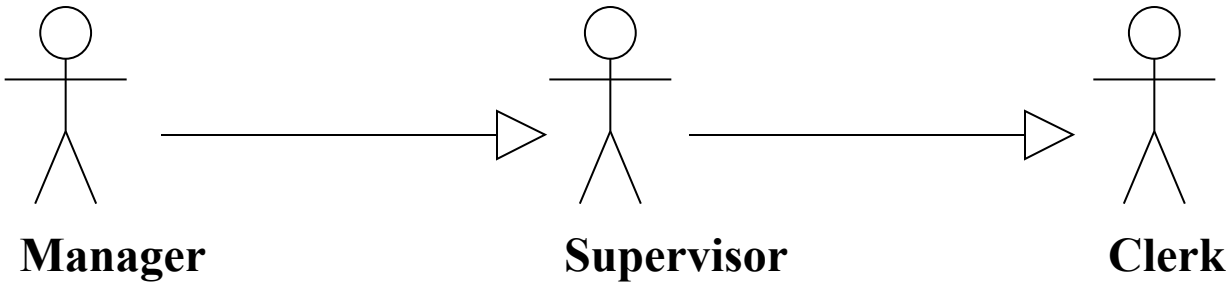




Hue University

Relationships between Actors

- When several actors as part of their roles, also play a more generalized role, it is described as generalization
- The behavior of the general role is described in an actor super-class
- The specialized actors inherit the behavior of the super-class and extend it in some way
- Relationships between actors are not always necessary



Identifying Actors

- Who will use the main functionality of the system?
- Who will need to maintain, administrate and keep the system working?
- With which other software/hardware systems does the system need to interact?
  - Other computer systems
  - Other applications on the same computer (I.e. X client/server)
  - Pitfall: don't confuse interaction with input

Identifying Actors

- Who
  - uses the system?
  - installs the system?
  - starts up the system?
  - maintains the system?
  - shuts down the system?
  - gets information from the system?
  - provides info to the system?
- Does anything happen at a preset time?

What about time?

- Treat time as an actor.
- Time actor can initiate a use case.

Hue University

Use Case

- Represent a complete functionality as perceived by an actor
  - A use case satisfies an actor's goal
- Always initiated by an actor
- A use case provides a value to an actor
- A use case is complete
  - Don't divide a use case into smaller use cases that implement each other (functional decomposition)

Use Case (cont.)

- The scenarios of a use case are normally described textually
  - A simple and consistent specification about how the actors and the system interact
  - Use case description template
- Describe at the level of user intentions and system responses
  - Free of technology and mechanism details, especially those related to user interface

UC Description Template

- Name
  - Name of use case, usually close to the user's goal
- Actors
- Goal description
- Reference to requirements (Optional)
  - Backward trace-ability
- Pre-conditions
  - The necessary conditions that have to be met before the use case can be performed
  - Could be other Use Cases as well (not equivalent to <<include>> at the beginning of the description)

UC Description Template (cont.)

- Description
  - A description of the basic or normal course that should be taken by the system if the system should perform as intended (everything goes right!)
- Post-conditions
  - The state of the system after the use case is performed
  - The value delivered to the actor
  - Distinguishes between variations and exceptions
- Variations
  - Expected condition causing the branch
  - Description of the alternative course or name of the extending Use Case
- Exceptions
  - Unexpected condition causing the branch (conflicts with post-condition)
  - Description of the alternative course

Use-Cases Tips

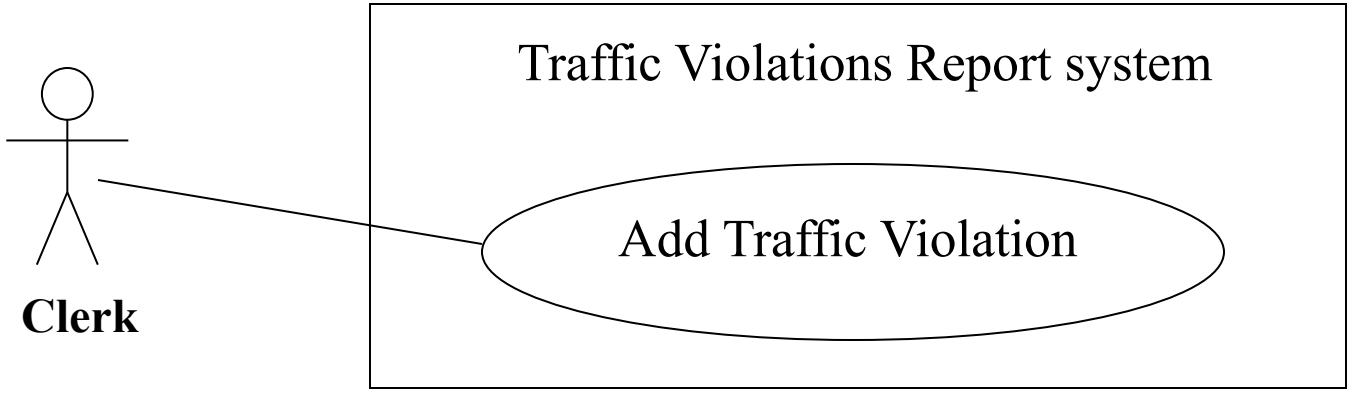
- Large Percentage begin and end with an actor.
- Scenarios (description) are written from the actor's point of view (therefore steps should be viewable to the actor)
- Use declarative statements.

Finding Use Cases

- For each of the actors previously defined:
  - Which services does the actor require from the system?
  - Does the system store info?
    - Read, create, destroy, modify, store information
  - Does the actor have to be notified about events in the system, or does the actor need to notify the system about something?
  - Could the actor's daily work be simplified?
    - Don't concentrate only on the current system

Use Case (cont.)

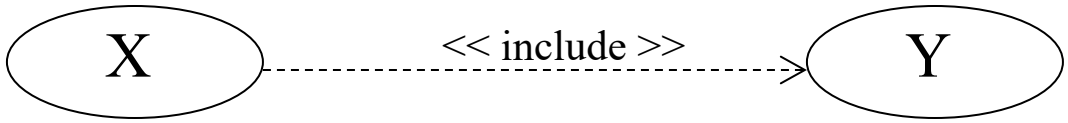
- Use Case Icon
  - An ellipsis containing the name of the Use Case
  - Placed inside the boundaries of the modeled system
  - Connected to at least one actor with a communication association



Except for specialized / extending use cases.

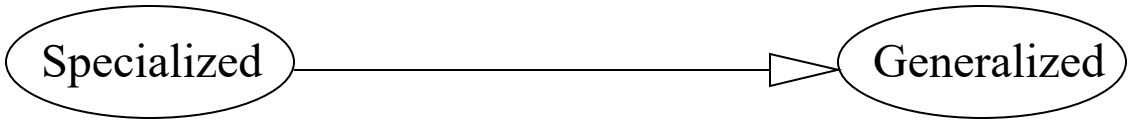
Use Case Relationships

- Include relationship ("uses" in older versions)
  - When a number of Use Cases have common behavior, this behavior can be modeled in a single use case that is used by the other use cases
  - X << includes >> Y indicates that the process of doing X always involves doing Y at least once
    - Beware of functional decompositions
  - The included Use Case must be complete
  - X must satisfy the pre-conditions of Y before including it



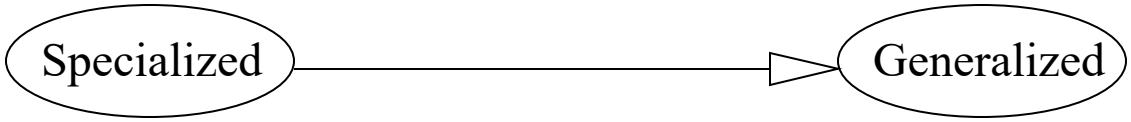
Use Case Relationships (cont.)

- Generalization relationship
  - Used when a number of Use Cases all have some subtasks in common, but each one has something different about it that makes it impossible to lump them all in a single use case
  - The generalized and specialized use cases must share the same goal
  - A specialized Use Case may capture an alternative scenario of the generalized Use Case
  - The generalized Use Case must be complete



Use Case Relationships (cont.)

- Generalization relationship (cont.)
  - The Specialized use case may interact with new actors.
  - The Specialized use case may add pre-conditions and post-conditions (AND semantics).



Recommended Workflow

1. Identify actors (and their relationships if necessary)
2. For each actor identified and until no new UC is discovered do
   a. Find all the goals of the actor
   b. Decide on the main course of success for each goal
   c. Create a Use Case for each of the goals
      - New actors/goals may be discovered
   d. Validate/correct existing Use Cases
3. Draw the Use Case diagram
   - Simplify model by repeating the process incase the produced diagram is too complex

Hue University

TVRS Use Cases

Example

Hue University

Example – TVRS Use Cases

- **Name**: Remove Traffic Violation
- **Actors**: Supervisor, OffendersDB.
- **Goal**: Remove an existing Traffic Violation
- References to requirements: 1.2.3, 1.3.2.4, …
- **Pre-conditions**:
  - Normal Course of "Lookup Traffic Violation" UC is completed, and the details of an existing Traffic Violation are displayed
- **Description**:
  1. Supervisor calls for deletion of the chosen Traffic Violation
  2. TVRS prompts Supervisor for confirmation

External System

Example – TVRS Use Cases

3. Supervisor confirms
4. TVRS requests OffendersDB to delete the Traffic Violation from the offender's record
5. OffendersDB approves that the Traffic Violation has been deleted
6. TVRS allows Supervisor to look up a new Traffic Violation as described in the "Lookup Traffic Violation" UC

- **Post-conditions**:
  - Removed Traffic Violation is no longer stored in the TVRS.
  - Traffic Violation is removed from the offender's record in the OffendersDB
  - "Lookup Traffic Violation" form is displayed

Example – TVRS Use Cases

- **Exceptions**:
  - 3a: Supervisor cancels:
    - 3a1: TVRS Continues to item 6 without removing the Traffic Violation
  - 5a: Traffic Violation is not removed from the OffendersDB
    - 5a1: TVRS displays an error message describing the failure
      5a2: TVRS continues to item 6 without clearing chosen Traffic Violation details, and without deleting the Traffic Violation

Example – TVRS Use Cases

- **Name**: Add Traffic Violation
- **Actors**: Clerk, PolicemenDB, OffendersDB
- **Goal**: Add a new Traffic Violation to the TVRS
- References to requirements: …
- **Pre-conditions**:
  - The Traffic Violation Management window is displayed
- **Description**:
  1. Clerk calls for addition of a new Traffic Violation
  2. TVRS displays an empty Traffic Violation Details form
  3. Clerk enters violation details and calls for saving the new Traffic Violation

Hue University

Example – TVRS Use Cases

4. TVRS prompts Clerk for confirmation.
5. Clerk confirms
6. TVRS asks the PolicemenDB whether or not the policeman is known
7. PolicemenDB replies that the policeman is known
8. TVRS asks the OffendersDB whether or not the offender is known
9. [Extension point] OffendersDB replies that the offender is known

…

Example – TVRS Use Cases

- **Post-conditions**:
  - New Traffic Violation is stored in the TVRS
  - TVRS displays an empty Traffic Violation Details form
- **Variations**:
  - 5a: Clerk cancels
    - 5a1: TVRS continues to item 2 without clearing the traffic violation details entered by clerk
  - 9a: OffendersDB replies that the offender is not known.
    - Described in Use Case "New Offender"
  - 7a: Policeman is not stored in the policemenDB
    - 7a1: TVRS displays an error message
    - 7a2: TVRS continues to item 2 without clearing Traffic Violation details entered by clerk
  - ...

Hue University

Example – TVRS Use Cases

- **Exceptions**:
  - 3a: Clerk cancels addition of the new Traffic Violation
    - 3a1: TVRS displays the "Traffic Violation Management" form
  - …

Example – TVRS Use Cases

- **Name**: New Offender [extends "Add Traffic Violaton"]
- **Actors**:
- **Goal**:
- **References to requirements**: …
- **Pre-conditions**:
  - Offender is not stored in the OffendersDB

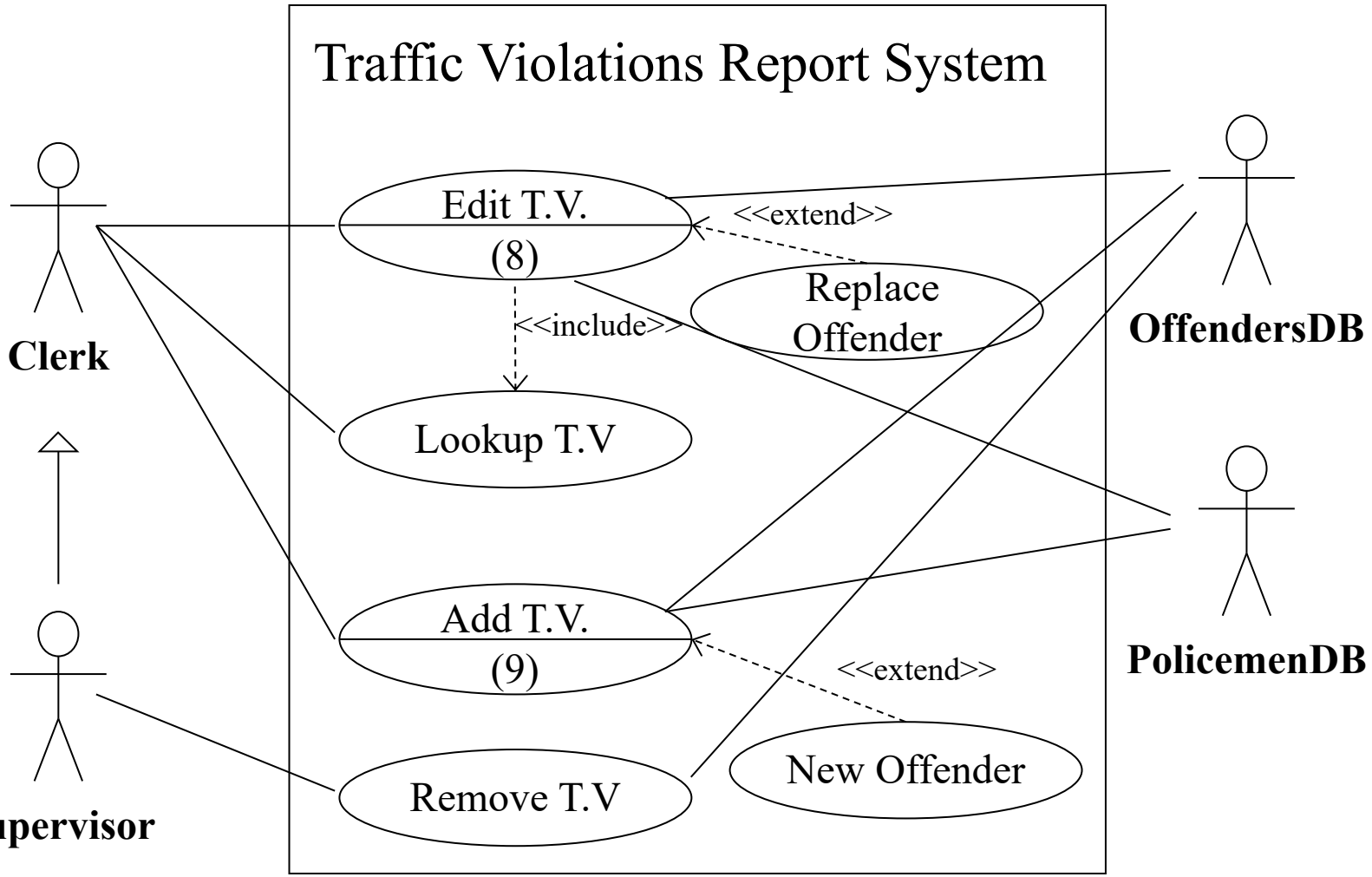
UC attributes are "inherited"

Hue University

Example – TVRS Use Cases

- **Description**:
  - 9a: OffendersDB replies that the offender is not known. [Add Traffic Violation]
  - 9b: TVRS displays an empty "Offender Details form"
  - 9c: Clerk enters offender details and calls for saving the new details
  - 9d: TVRS prompts Clerk for confirmation
  - 9e: Clerk confirms
  - 9f: TVRS requests OffendersDB to store the new offender
  - 9g: OffendersDB replies that offender was stored successfully
- **Post-conditions**:
  - New Offender is stored in the offenders DB
- ...

Example – TVRS Use Cases



Levels of Use-Cases

- Sea-Level
  - Discrete interactions between primary actor and system
- Kite-Level
  - Show how sea-level use-cases fit into wider business interactions

Use-Cases

- Provide the boundaries and scope of a project

Prioritize Use cases

- Based on risk
- Time
- Money

Thank You!

… feel tired? **YES!!!!**

Hue University