Modeling Functionality with Use Cases (Part 1 – SysML Concepts)

Section Objectives

- In this Section, you will learn:
  - How to model Use Cases in SysML

Overview

- This section will discuss:
  - Use Case Concepts
    - What are Use Cases?
    - Why model Use Cases?
    - Use Case Diagram Components
    - Use Case Descriptions
    - Elaborating Use Cases
    - How to develop Use Cases
  - Use Case modeling for In-Class Project

What are Use Cases?

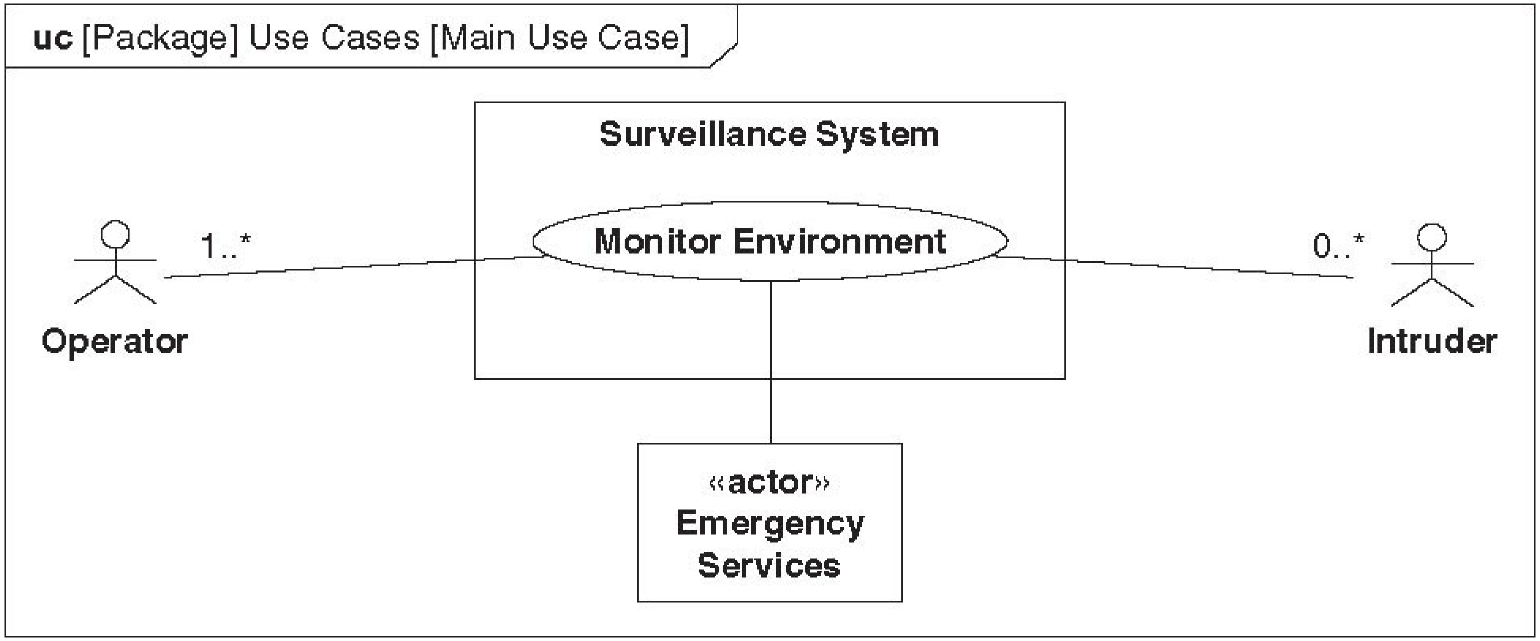
- A Use Case models who or what will use the system and what they will be able to do with it.
  - Describes the functionality that a system must provide to achieve user goals.
- The collection of Use Cases constitutes all of the defined ways that the system may be used.
- Identify Use Cases by finishing the statement, "I need to…"
- A Use Case diagram consists of a set of Actors and Use Cases and the relationships between them.
- A Use Case description is captured in text, and consists of Actors, pre-conditions, assumptions, post-conditions, and primary, alternative, and/or exception flows.

Why Model Use Cases?

- Use Cases can be used to model anything that performs a function, and thus is a useful tool for the system analysis phase of systems engineering
- Use Cases:
  - Communicate the system's functionality and behavior to the customer and end user
  - Focus on 'who' will use the system and 'what' they will be able to do with it, not on 'how' a system will do something
  - Identify interfaces between the systems and its users.
  - Model the system from a user's point of view
  - Help define functional requirements of the system

Use Case Diagram Components

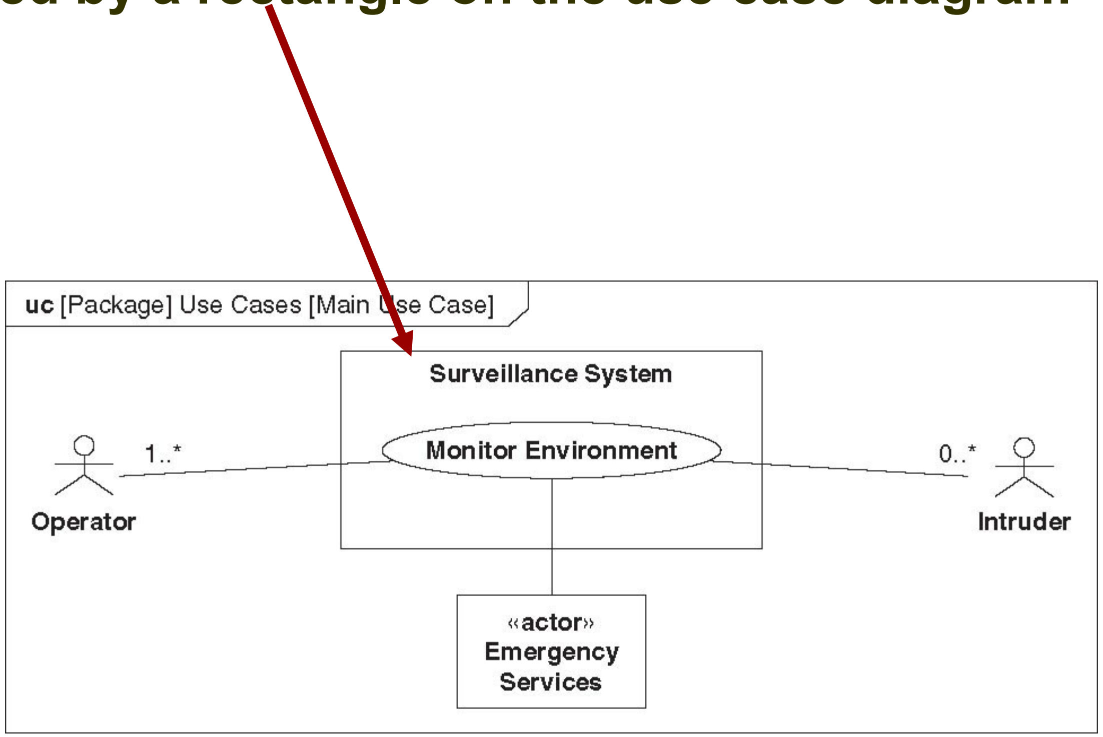
- Use Case diagrams are comprised of the following:
  - System
  - Actors
  - Use Cases
  - Relationships



© 2008 Elsevier, Inc.: A Practical Guide to SysML

System

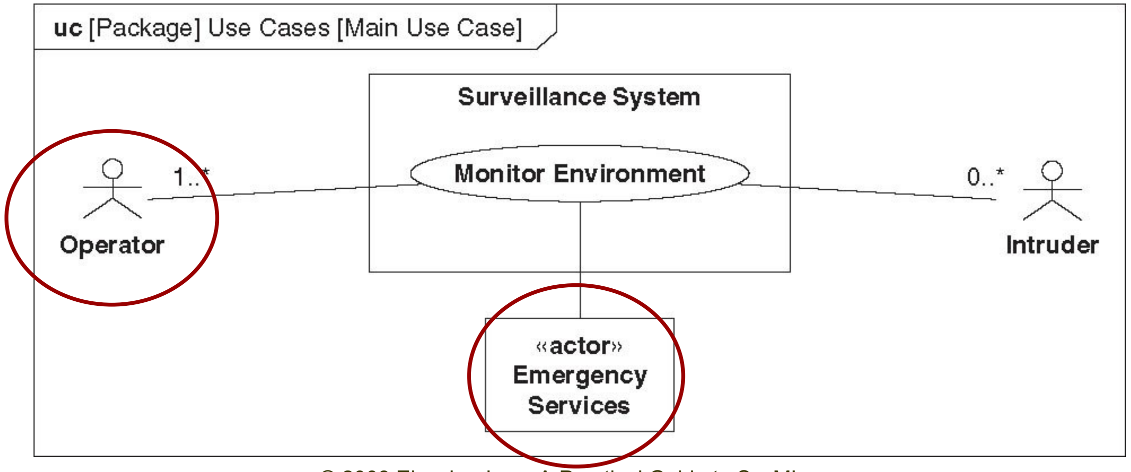
- Provides the functionality in support of the use cases
- Represents a system being developed
- Also called the ‘subject’ or ‘system under consideration’
- Represented by a rectangle on the use case diagram



© 2008 Elsevier, Inc.: A Practical Guide to SysML

Actors

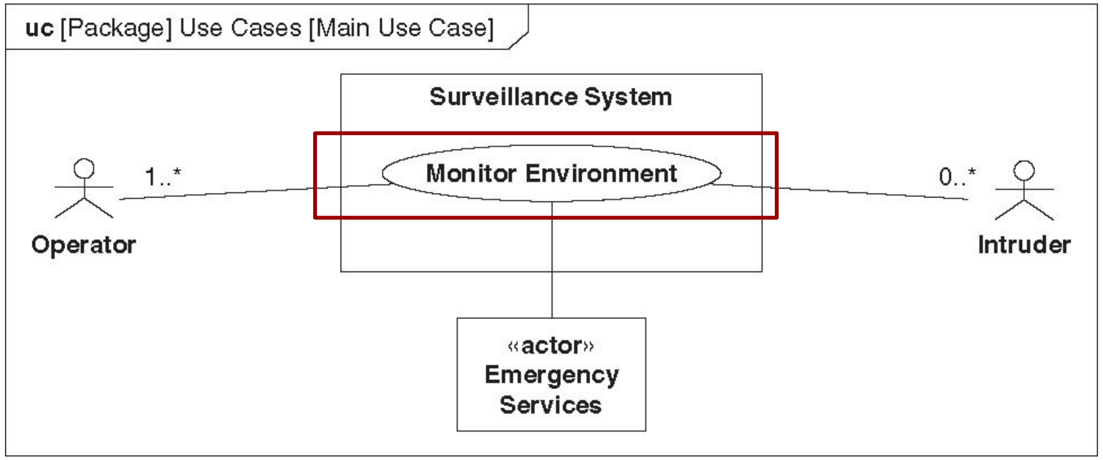
- Used to represent something that uses the system
  - Not 'part' of the system
    - Depicted outside of the system 'box'
  - Actors interface with the system
- Can be a person or another system
- Usually depicted by a stick figure and/or block with <<actor>> label
- Name the Actors based on the **role** they perform as a user of the system (e.g. Operator, Customer, etc)



© 2008 Elsevier, Inc.: A Practical Guide to SysML

Use Cases

- Represent the functions that a system will perform
- Depicted by an oval with the Use Case name inside
- Name should consist of a verb and a noun that describe the functionality of the system (e.g. Record Grades, Monitor Environment)

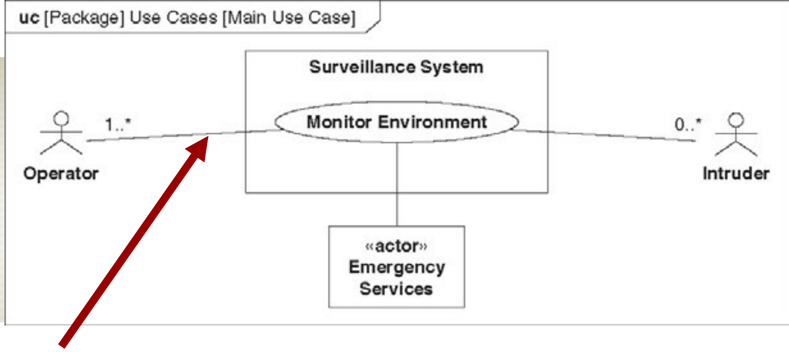


© 2008 Elsevier, Inc.: A Practical Guide to SysML

Relationships on a Use Case Diagram

- Relationships between Actors and Use Cases
- Relationships between Use Cases
  - Include
  - Extend
  - Generalize/Specialize

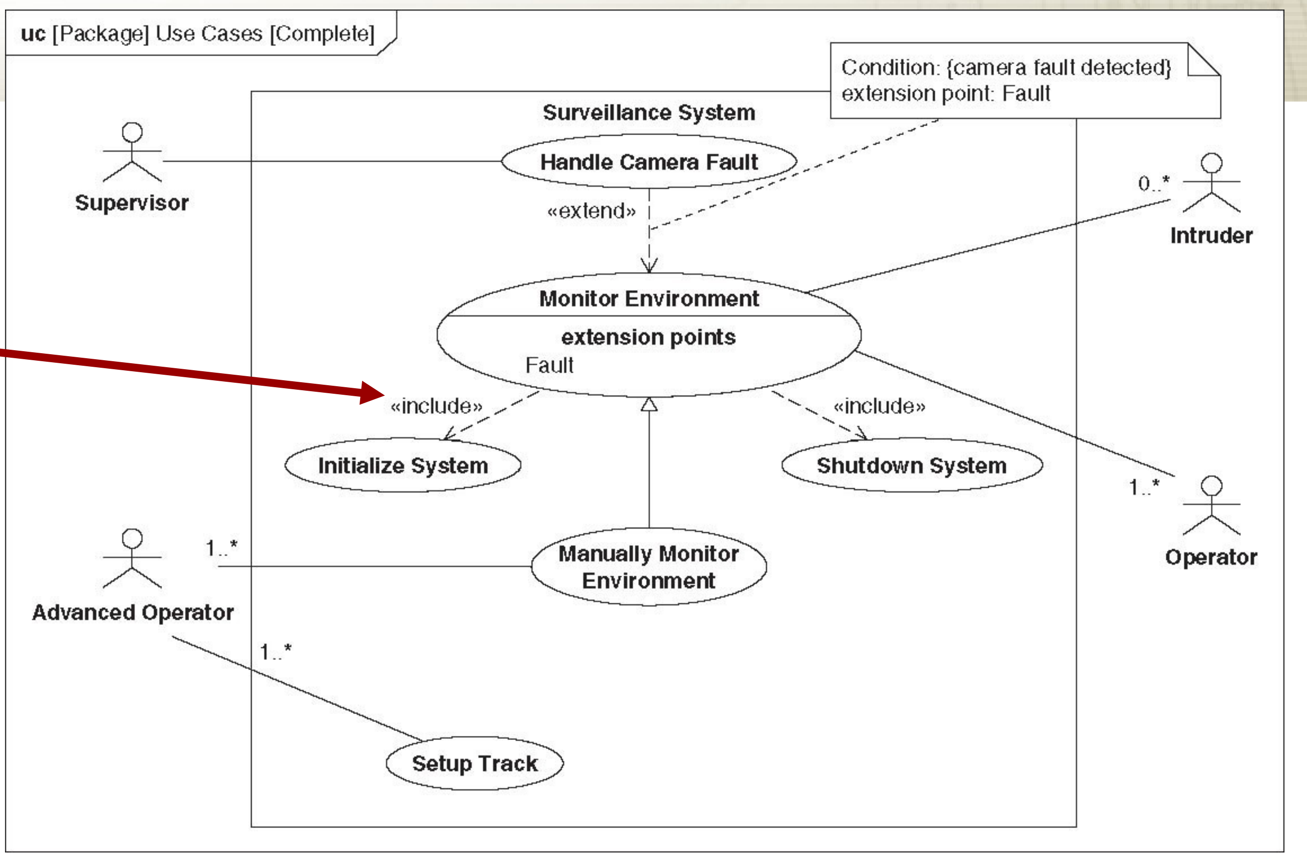
Relationships between Actors and Use Cases



- Depicts association between Actors and Use Cases
  - A Primary Actor initiates a Use Case
- Relationships are depicted by a solid line
  - Multiplicity depicts the number of Actors that relate to a single Use Case
    - 0..1 (zero or one) - default
    - 1..* (one to many)
    - 0..* (zero to many)
- An Actor can be associated with multiple Use Cases
- A Use Case can be associated with multiple Actors

Relationships Between Use Cases (Include)

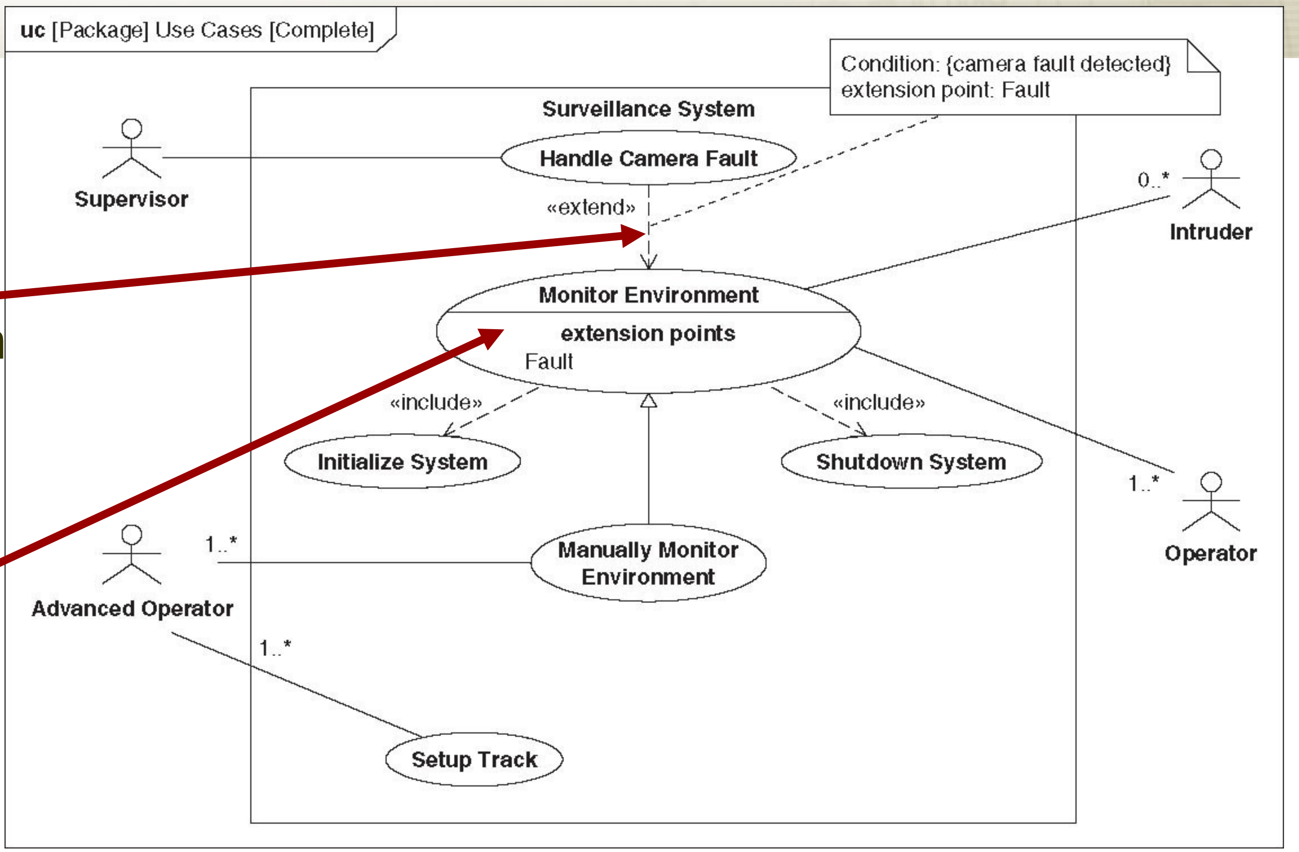
- Uses UML 'dependency' relationship
- Depicts shared (or re-used) functionality
- The included Use Case is always performed by the base Use Case



© 2008 Elsevier, Inc.: A Practical Guide to SysML

Relationships Between Use Cases (Extend)

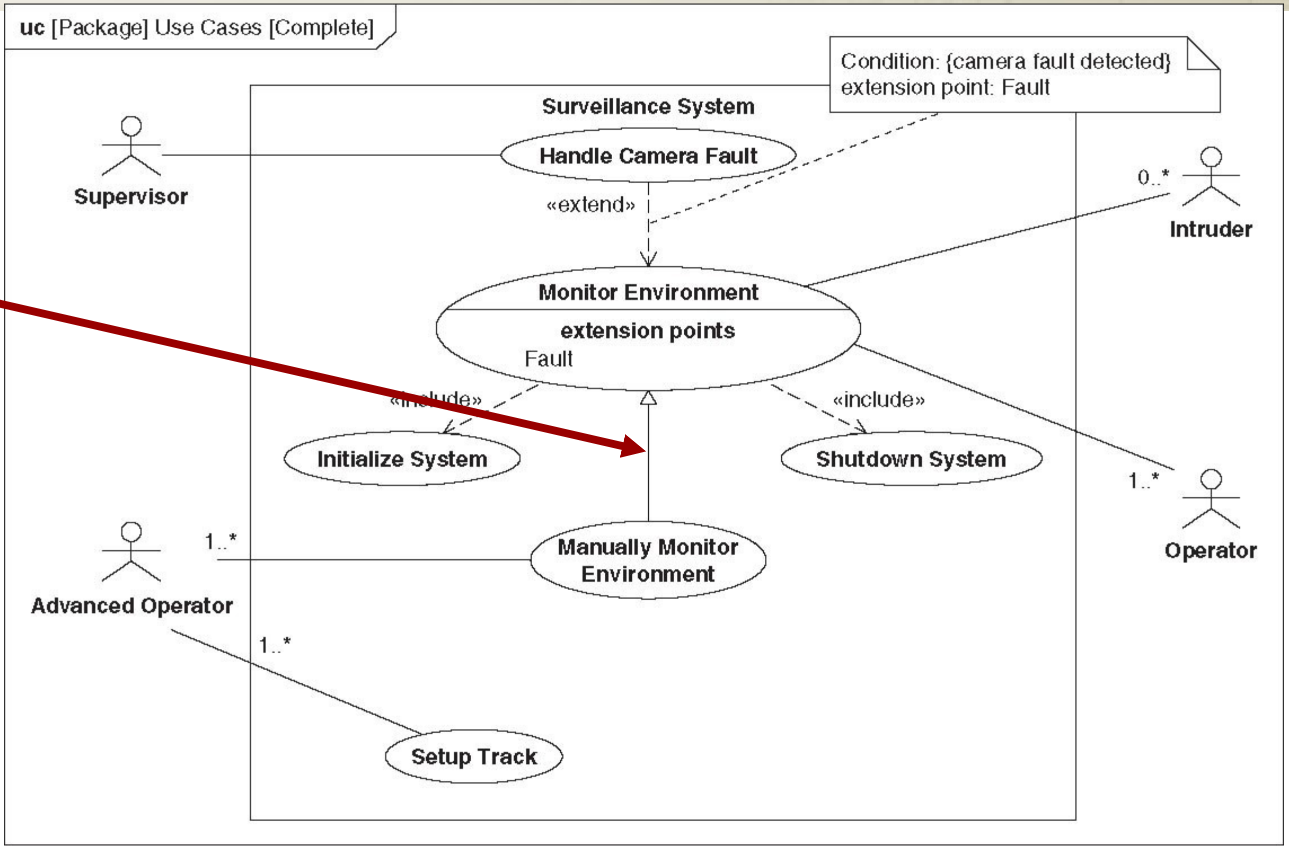
- Uses UML 'dependency' relationship
- Depicts optional functionality, which is only performed when a particular condition is met
- Extension points on the Use Case indicate the condition that allows the Use Case to 'call' the extending Use Case



© 2008 Elsevier, Inc.: A Practical Guide to SysML

Relationships Between Use Cases (Generalize/Specialize)

- Uses UML 'generalization' relationship
- Indicates that the 'child' Use Case inherits functionality from the 'parent' Use Case
- The parent Use Case is a more general Use Case than the child
- Read as: The child Use Case is a specialization of the parent Use Case



© 2008 Elsevier, Inc.: A Practical Guide to SysML

Use Case Descriptions

- Text that captures the details of a Use Case
  - Actor that initiates the Use Case
  - Assumptions for the Use Case
  - Pre-condition for the Use Case
  - Sequence of steps for each scenario*
    - Primary Flow
    - Alternative Flows
    - Exception Flows
  - Post-condition for each scenario
  - The Actor that benefits from the Use Case

- *Describe what the system should do, not how it does it

Introducing Scenarios

- Scenario – an instance of a Use Case
  - One path through the flow of events for the Use Case
- Used to describe how Use Cases are realized
- Each Use Case is a set of scenarios
  - Primary flow
  - Alternative flows
  - Exception flows
- Scenarios can be captured in Activity Diagrams and Sequence Diagrams (to be discussed in later sections)

Elaborating Use Cases

- Providing more detail of the Use Case and its scenarios, using
  - Activity diagrams
  - Sequence diagrams
  - State machine diagrams
- Activity diagrams – used when the use case scenario contains considerable control logic, inputs and outputs, and/or algorithms that transform data
- Sequence diagrams – used for use case scenarios that are largely message-based
- State machines – used when the interaction between Actors and the system is asynchronous and not easily depicted by an ordered sequence of events

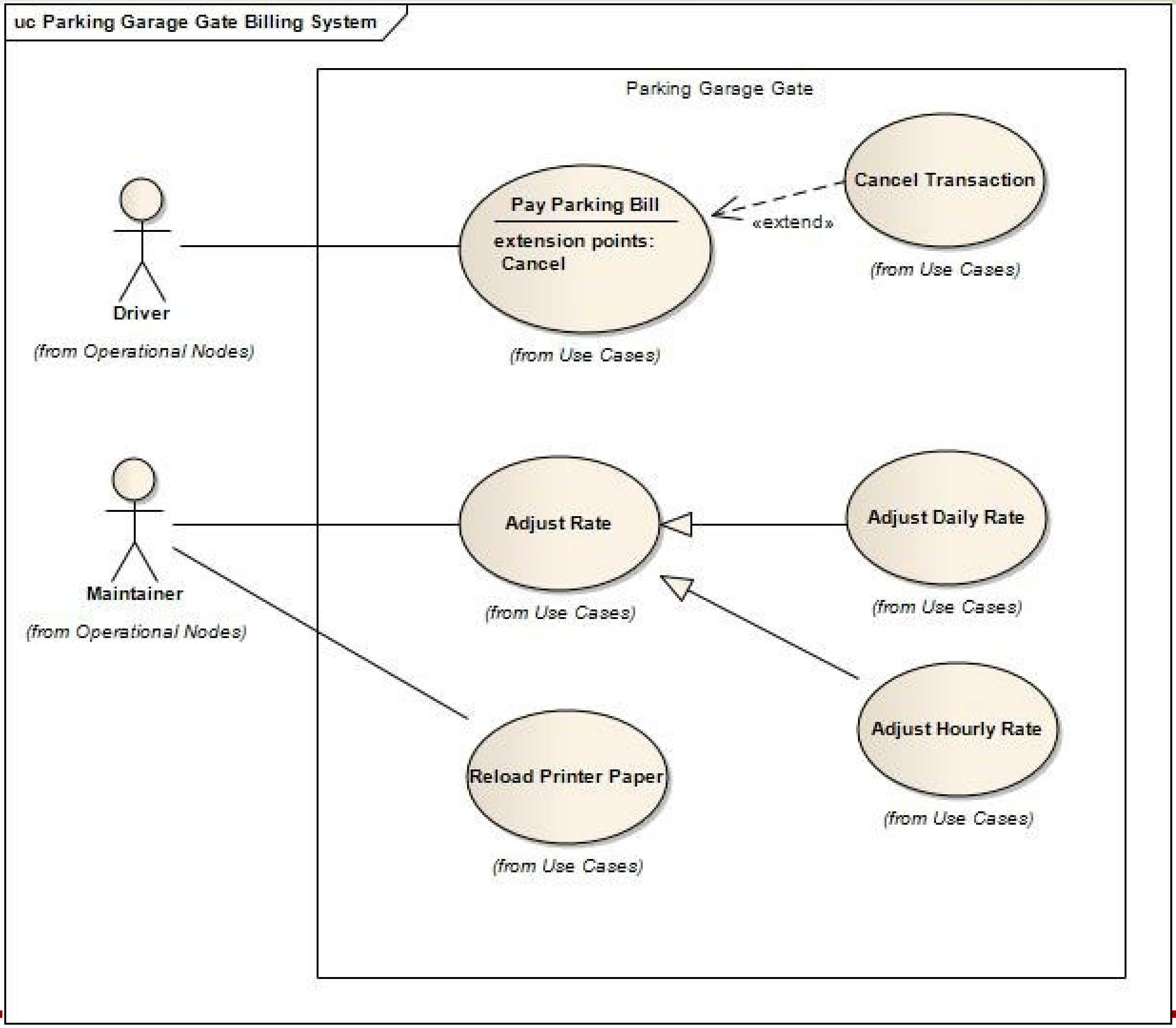
How to Develop Use Cases

- Identify the Actors and Use Cases
  - Elicitation/Interviews
  - Documentation:
    - CONOPS
    - System Specifications
  - Ask:
    - Who or what are the users of the system?
    - What will the system do for the users, to help them get what they want from the system?
- Detail each Use Case
  - Produce Use Case Description
  - Identify Primary Path
- Identify Common and Exception Functions (for include and extend relationships)
- Build the Use Case Model in a diagram
  - Showing Actors and Use Case relationships

Use Case Modeling for In-Class Project

- Use Case Model for Parking Garage Gate Project
- Define
  - Use Cases
  - Actors
  - Relationships between Actors and Use Cases
  - Relationships between Use Cases

Use Case Model for Parking Garage Gate



Summary

- Use Cases capture the functionality of a system must provide to achieve user goals
- Use Case diagrams are made up of:
  - System
  - Actors
  - Use Cases
  - Relationships
- Use Case can be elaborated through:
  - Activity diagrams
  - Sequence diagrams
  - State machine diagrams