

# UML

cmisc435 - 1

## Objectives

- Documenting user requirements using the UML notation
- Description of the various components of UML
- The use of Use Cases

cmisc435 - 2

## Unified Modeling Language

- The UML is an international industry standard graphical notation for describing software analysis and designs.
- UML is a modeling language or graphical/diagrammatic notation for object-oriented programming - a way to express the "blueprints" of your system. Within UML, there are several types of diagrams:
  - Use case
  - Class
  - Sequence
  - State
  - Activity diagram

cmisc435 - 3

## Use cases

- A **scenario** is a sequence of actions that illustrates behavior. A scenario may be used to illustrate an interaction or the execution of a use case instance.
- **Use cases** are used to describe scenarios.
  - A use case is a specification of sequences of actions, including variant sequences and error sequences, that a system, subsystem, or class can perform by interacting with outside actors
- A set of scenarios makes up a use case

cmisc435 - 4

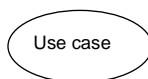
## Example scenario

---

- From game of Monopoly
- Goal: Get out of jail
  - A player is in Jail. The player clicks the "Get out of Jail" button. \$50 is decremented from their money. The player can then roll the dice and continue with the game.
  - A player is in Jail. The player clicks the "Get out of Jail" button. The player has less than \$50. The player becomes bankrupt and all the tradable cells he or she owns becomes available in the game. The player is out of the game.

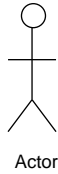
## UML symbols - use case

---



Describe as present-tense verb in active voice: e.g., *Get out of Jail*, *Solve problem*, *Print solution*

## Actors



- An actor is an entity that interacts with the system or needs to exchange information with the system.
  - ❑ supplies input information to the system
  - ❑ receives information from the system
  - ❑ both supplies input information to and receives information from the system

cmisc435 - 7

## Identifying actors

- Who uses the system?
- Who installs the system?
- Who starts up the system?
- Who maintains the system?
- Who shuts down the system?
- What other systems use this system?
- Who gets information from this system?
- Who provides information to the system?
- Does anything happen automatically at a present time?

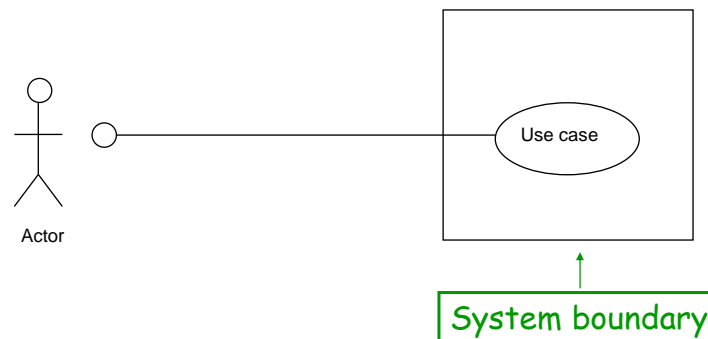
cmisc435 - 8

## Identifying the use cases

- What functions will the actor want from the system?
- Does the system store information? What actors will create, read, update or delete this information?
- Does the system need to notify an actor about changes in the internal state?
- Are there any external events the system must know about? What actor informs the system of those events?

cmisc435 - 9

## Use case diagram



- Only actions within the system boundary need to be modeled and built as part of the software

cmisc435 - 10

## Use case flow of events

### X Flow-of-Events for the <name> Use Case

**X.1 Preconditions.** What needs to happen (in another use case) before this use case can start? What state must the system be in before the use case?

**X.2 Main Flow.** The main flow is a series of declarative steps.

**X.3 Sub-flows.** Sub-flows break down the main flow and other sub-flows to improve document readability.

**X.4 Alternative Flows.** The alternative flows define exceptional behavior that can interrupt the normal flow. Often alternative flows indicate what is to be done under error conditions. To determine alternative flows, ask yourself, "What could possibly go wrong?" for each of the actions in the main flow and the sub-flows.

**Note:** X is a unique identifier for each use case.

## UC8 Flow of Events for the *Buy House* Use Case

### 8.1 Preconditions:

1. It is the player's turn.
2. The player has not rolled the dice.
3. The player has monopoly on one or more color groups.

### 8.2 Main Flow:

When a player has all the tradable cells in a color group, this player is said to have monopoly on the color group. A player may build house(s) in the property cells in the color groups the player has monopoly on by pressing the Buy House button before he or she rolls the dice [S1] [E1 - E2]. The price of the house is determined by the cell. After buying the house(s), the status of the player is updated and displayed on the game board [UC13].

### 8.3 Subflows:

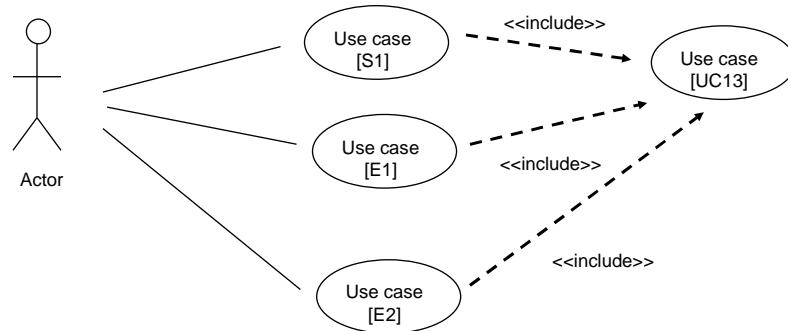
[S1] When the Buy House button is clicked, the Buy House dialog shows up. The player selects the monopoly color group and the number of houses from that dialog. After clicking on OK in the dialog box, the player pays the fee, and the houses are created. All the property cells in the selected color group have the same number of houses.

### 8.4 Alternative Flows:

- [E1] Nothing happens if the player does not have enough money.  
[E2] The player can build at most five houses in a cell.

**A use case is one path through this flow.**

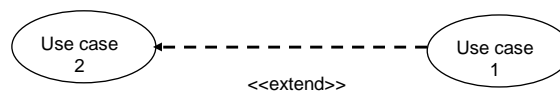
## Buy House use case [Inclusion of other use cases]



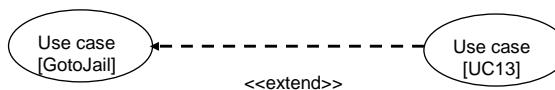
- [S1] - Own monopoly and buy house
- [E1] - Not enough money to buy houses
- [E2] - Have 5 houses on properties

cm435 - 13

## Extend relationship



- Use case 2 extends the actions of use case 1.



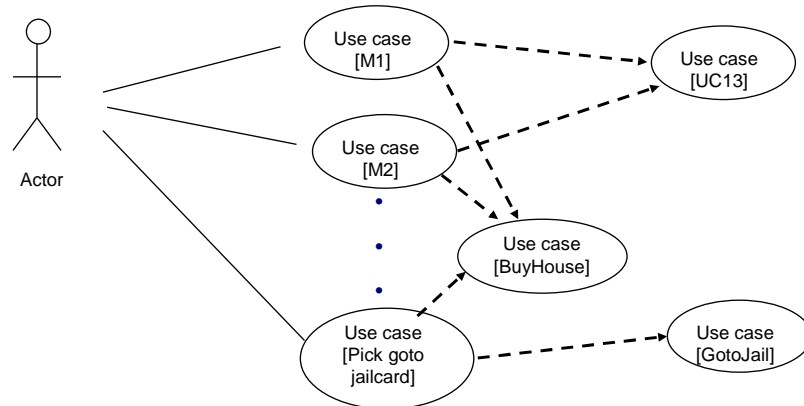
cm435 - 14

## Include versus extend

- Use Case X includes Use Case Y:
  - X has a subtask Y. In the course of doing X or a subtask of X, Y will **always** be completed (e.g., [UC13] displays game board)
- Use Case X extends Use Case Y:
  - Y performs a sub-task and X is a similar but more specialized way of accomplishing that subtask. X **only happens in an exception situation**. Y can complete without X ever happening. (e.g., [Goto Jail] can extend [UC13] if player is directed to go to jail.)

cmse435 - 15

## Actor makes move

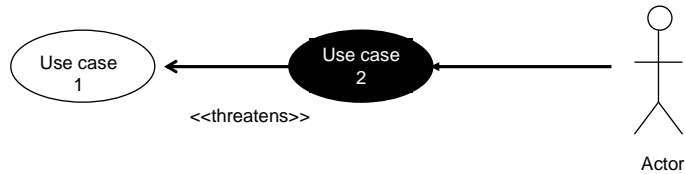


- [BuyHouse] case is first task and [UC13] is last task per move

cmse435 - 16



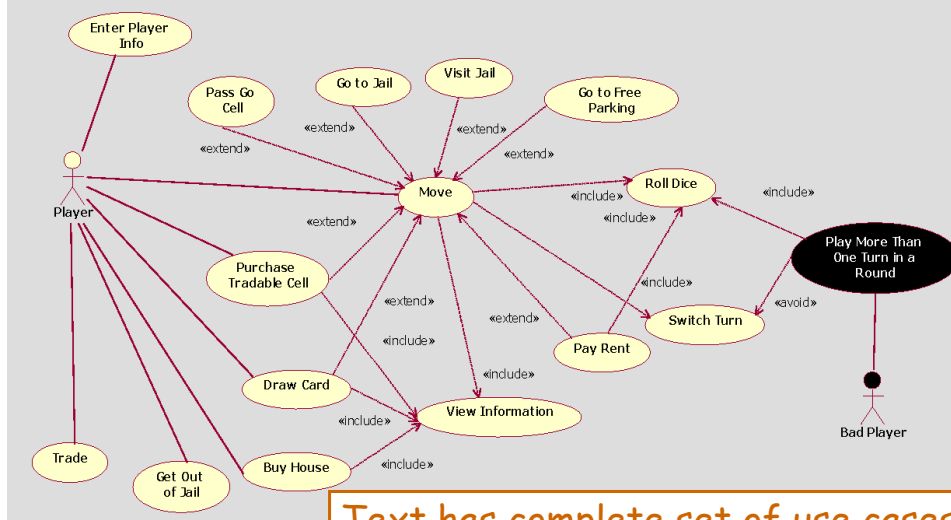
## Security: Misuse case



- Use case 2 is a misuse case by actor
  - E.g., Actor is "Hacker", use case 2 is "obtain password, and use case 1 is "Login"
- Need for mitigation of threats

cmssc435 - 17

## Example use cases for Monopoly



Text has complete set of use cases

cmssc435 - 18

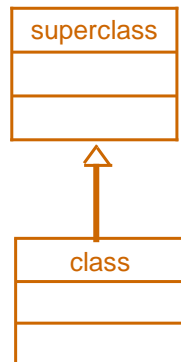
# Class diagrams

- Class diagrams are used in both the **analysis** and the **design** phases.
- The class diagram describes the types of objects in a system and the various kinds of static relationships that exist among them:
  - ❑ - private
  - ❑ # protected
  - ❑ + public
- You should have seen these in courses like CMSC330 in understanding C++ classes. They can represent the signatures of C++ classes.

ClassName
-Attribute1
#Attribute2
+Attribute3
-Operation1
#Operation2
+Operation3

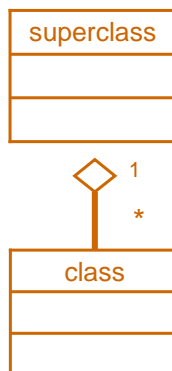
# Static relationships

- Inheritance (generalization):



class "is-a" superclass

- Aggregation (association):



superclass "has-a" class

- Composition (association):

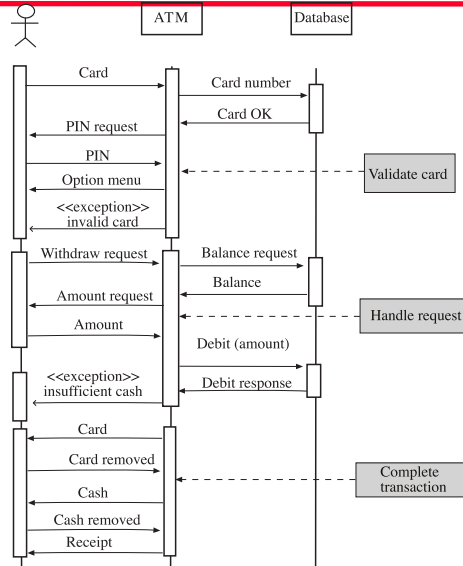


Class "exists within" superclass



## Sequence diagram of ATM withdrawal

- Seen earlier
- Structure
  - ❑ the vertical dimension represents time;
  - ❑ The horizontal dimension represents different objects.
  - ❑ Arrows represent flows from use cases



cmssc435 - 23

## State diagrams

- Based on finite state machines (Did you learn this in CMSC330 or elsewhere?)
  - ❑ A state is represented by a rounded rectangle.
  - ❑ A start state is represented by a solid circle.
  - ❑ A final state is represented by a solid circle with another open circle around it.
  - ❑ A transition is a change of an object from one state (the source state) to another (the target state) triggered by events, conditions, or time. Transitions are represented by an arrow connecting two states.

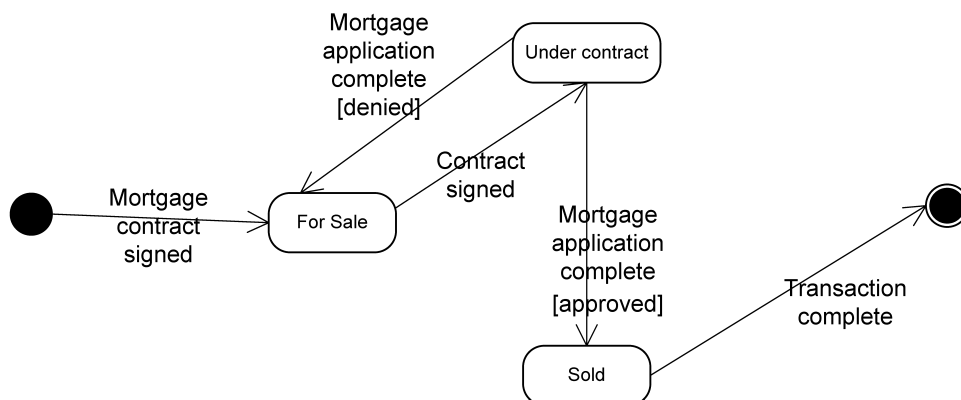
cmssc435 - 24

## Transitions

- Each transition has a label that comes in three parts. All the parts are optional: **trigger-signature** [guard]/activity
- The **trigger-signature** is usually a single event that triggers a potential change of state. A missing trigger-signature indicates that you take the transition.
- The **guard**, if present, is a Boolean condition that must be true for the transition to be taken. Only one transition can be taken out of a given state. If more than one guard condition is true, the choice of transition to fire is nondeterministic if no priority rule is given. A missing guard indicates that you always take the transition once the trigger-signature has fired.
- The **activity** is some behavior that's "executed" during the transition. A missing activity means that you don't do anything during the transition.

cmisc435 - 25

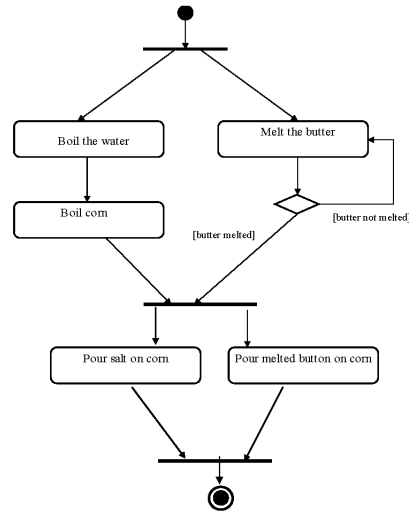
## Example state diagram



cmisc435 - 26

## Activity diagrams [Prepare corn on the cob]

- Notation a mix of state diagrams with addition of synchronization bars (from operating system Petri nets)
- Operation cannot flow pass a bar until all input arcs have been satisfied. Then all output arcs are activated.
- Operations between synchronization bars are asynchronous.



cmisc435 - 27

## UML summary

- 5 basic models
  - Use cases
  - Class diagrams
  - Sequence diagrams
  - State charts
  - Activity diagrams
- This is only a brief introduction
  - No time to study this in depth

cmisc435 - 28

## Key Points

---

- Identify all the actors of the system. Think about all the functionality that the actors want from the system. Consider what the various functions the actors are asking have in common. Abstract these as «include» use cases.
- A picture is worth a thousand words. Use case diagrams help to visualize what the system has to do. But, more importantly, the use case flow-of-events gets much more specific about what the customer wants - the variations and the exceptions.
- UML diagrams, sequence, charts, activity diagrams all represent pictorial views of the operations in a system.