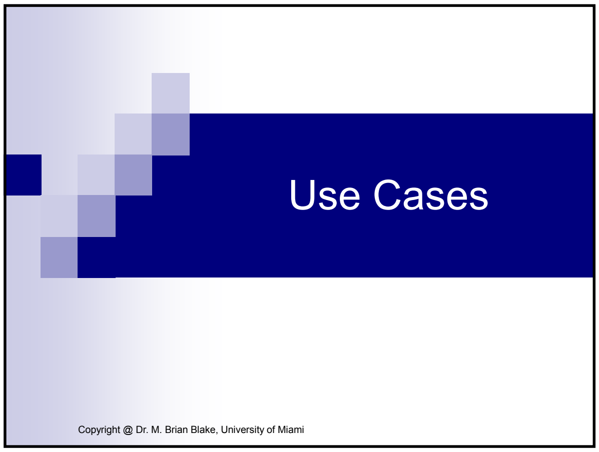
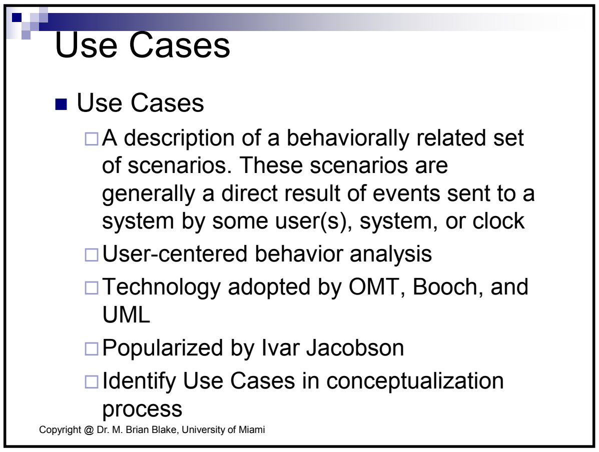
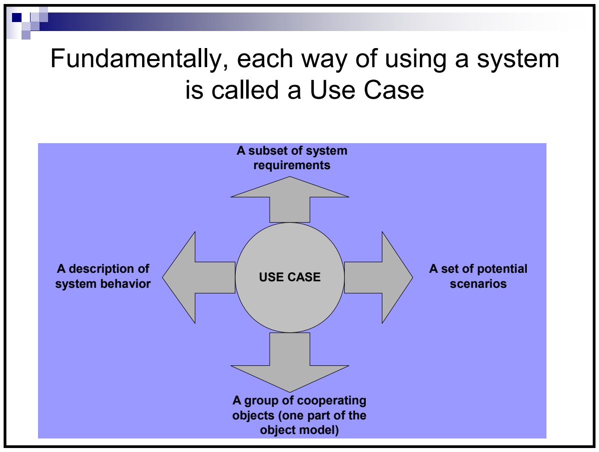


Use Cases



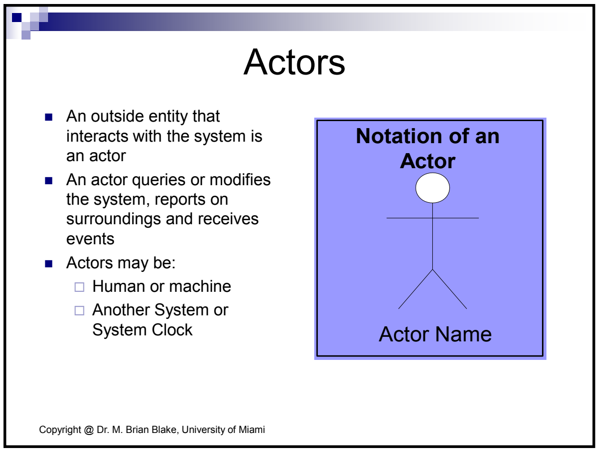
Use Cases

- Use Cases
  - □ A description of a behaviorally related set of scenarios. These scenarios are generally a direct result of events sent to a system by some user(s), system, or clock
  - □ User-centered behavior analysis
  - □ Technology adopted by OMT, Booch, and UML
  - □ Popularized by Ivar Jacobson
  - □ Identify Use Cases in conceptualization process



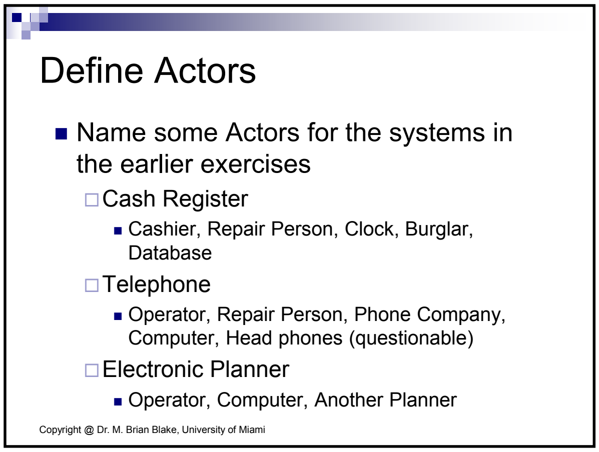
Fundamentally, each way of using a system is called a Use Case

A subset of system requirements
A description of system behavior
USE CASE
A set of potential scenarios
A group of cooperating objects (one part of the object model)



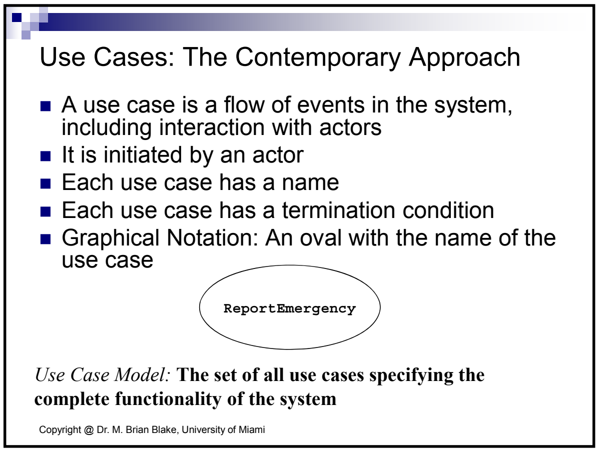
Actors

- An outside entity that interacts with the system is an actor
- An actor queries or modifies the system, reports on surroundings and receives events
- Actors may be:
  - □ Human or machine
  - □ Another System or System Clock



Define Actors

- Name some Actors for the systems in the earlier exercises
  - □ Cash Register
    - Cashier, Repair Person, Clock, Burglar, Database
  - □ Telephone
    - Operator, Repair Person, Phone Company, Computer, Head phones (questionable)
  - □ Electronic Planner
    - Operator, Computer, Another Planner

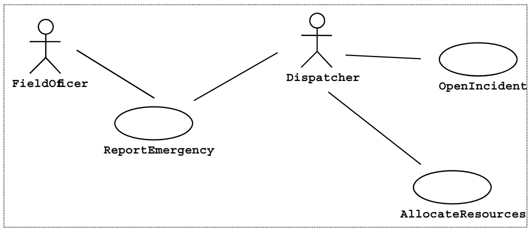


Use Cases: The Contemporary Approach

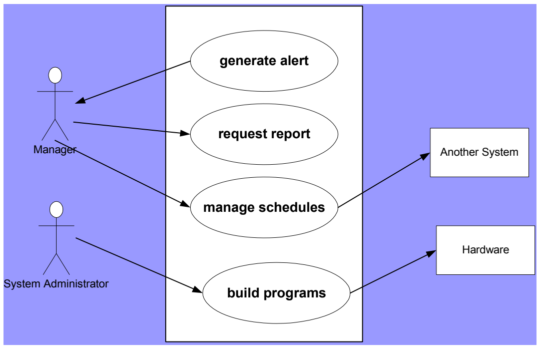
- A use case is a flow of events in the system, including interaction with actors
- It is initiated by an actor
- Each use case has a name
- Each use case has a termination condition
- Graphical Notation: An oval with the name of the use case

Use Case Model: **The set of all use cases specifying the complete functionality of the system**

Example: Use Case Model for Incident Management



Use Case (Simple Example)



Heuristics: How do I find use cases?

- Select a narrow vertical slice of the system (i.e. one scenario)
  - Discuss it in detail with the user to understand the user's preferred style of interaction
- Select a horizontal slice (i.e. many scenarios) to define the scope of the system.
  - Discuss the scope with the user
- Use illustrative prototypes (mock-ups) as visual support
- Find out what the user does
  - Task observation (Good)
  - Questionnaires (Bad)

Use Case Process

- Identify actors and their purpose
- Develop a Candidate Use Case List
- Graphically depict all Use Cases in the system
- Review with the user
- Describe each Use Case, concentrating on control
- Review the Use Case descriptions

Use Case Example: ReportEmergency

- Use case name: ReportEmergency
- Participating Actors:
  - Field Officer (Bob and Alice in the Scenario)
  - Dispatcher (John in the Scenario)
- Exceptions:
  - The FieldOfficer is notified immediately if the connection between her terminal and the central is lost.
  - The Dispatcher is notified immediately if the connection between any logged in FieldOfficer and the central is lost.
- Flow of Events: **on next slide.**
- Special Requirements:
  - The FieldOfficer's report is acknowledged within 30 seconds. The selected response arrives no later than 30 seconds after it is sent by the Dispatcher.

Use Case Example: ReportEmergency Flow of Events

- The **FieldOfficer** activates the "Report Emergency" function of her terminal. FRIEND responds by presenting a form to the officer.
- The FieldOfficer fills the form, by selecting the emergency level, type, location, and brief description of the situation. The FieldOfficer also describes possible responses to the emergency situation. Once the form is completed, the FieldOfficer submits the form, at which point, the **Dispatcher** is notified.
- The Dispatcher reviews the submitted information and creates an Incident in the database by invoking the OpenIncident use case. The Dispatcher selects a response and acknowledges the emergency report.
- The FieldOfficer receives the acknowledgment and the selected response.

Order of steps when formulating use cases

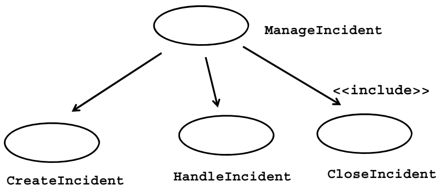
- First step: name the use case
  - Use case name: ReportEmergency
- Second step: Find the actors
  - Generalize the concrete names ("Bob") to participating actors ("Field officer")
  - Participating Actors:
    - Field Officer (Bob and Alice in the Scenario)
    - Dispatcher (John in the Scenario)
- Third step: Then concentrate on the flow of events
  - Use informal natural language

Use Case Associations

- A use case model consists of use cases and use case associations
  - A use case association is a relationship between use cases
- Important types of use case associations: Include, Extends, Generalization
- Include
  - A use case uses another use case ("functional decomposition")
- Extends
  - A use case extends another use case
- Generalization
  - An abstract use case has different specializations

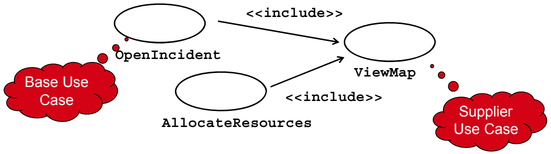
<<Include>>: Functional Decomposition

- Problem:
  - A function in the original problem statement is too complex to be solvable immediately
- Solution:
  - Describe the function as the aggregation of a set of simpler functions. The associated use case is decomposed into smaller use cases



<<Include>>: Reuse of Existing Functionality

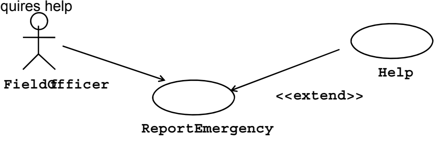
- Problem:
  - There are already existing functions. How can we *reuse* them?
- Solution:
  - The *include association* from a use case A to a use case B indicates that an instance of the use case A performs all the behavior described in the use case B ("A delegates to B")
- Example:
  - The use case "ViewMap" describes behavior that can be used by the use case "OpenIncident" ("ViewMap" is factored out)



Note: The base case cannot exist alone. It is always called with the supplier use case

<Extend>> Association for Use Cases

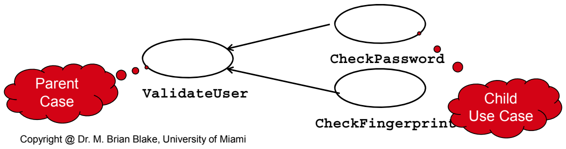
- Problem:
  - The functionality in the original problem statement needs to be extended.
- Solution:
  - An *extend association* from a use case A to a use case B indicates that use case B is an extension of use case A.
- Example:
  - The use case "ReportEmergency" is complete by itself , but can be extended by the use case "Help" for a specific scenario in which the user requires help



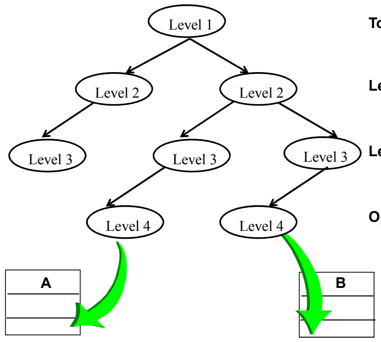
Note: The base use case can be executed without the use case extension in extend associations.

Generalization association in use cases

- Problem:
  - You have common behavior among use cases and want to factor this out.
- Solution:
  - The generalization association among use cases factors out common behavior. The child use cases inherit the behavior and meaning of the parent use case and add or override some behavior.
- Example:
  - Consider the use case "ValidateUser", responsible for verifying the identity of the user. The customer might require two realizations: "CheckPassword" and "CheckFingerprint"



From Use Cases to Objects



Top Level Use Case

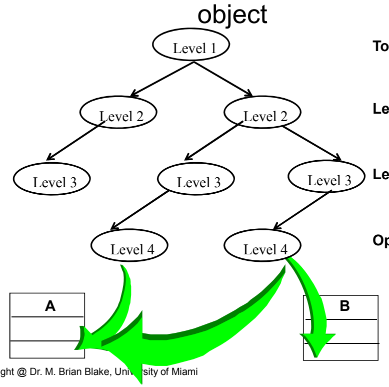
Level 2 Use Cases

Level 3 Use Cases

Operations

A and B are called Participating Objects

Use Cases can be used by more than one object



Top Level Use Case

Level 2 Use Cases

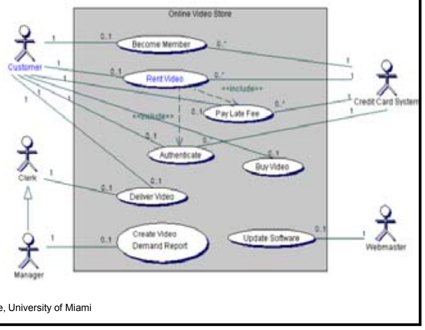
Level 3 Use Cases

Operations

Participating Objects

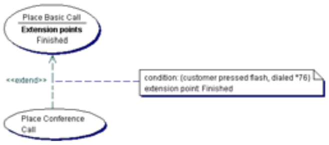
UML 2.0: Multiplicity

- Multiplicity
  - □ How many sessions or how many types of interactions
  - □ e.g. Only rent one video at a time, Only interact with one type of credit card system, can pay an infinite amount of late fees
  - □ Samples:
  - □ 1 , 0..1, 1..n, 1..#, 0..#, etc.



UML 2.0: Conditional *extends*

- Condition on <<extends>>
  - □ Explains the circumstance when an extends relationship is relevant
  - □ To show this condition, we add a *note* to the extend relationship indicating the condition under which we extend the *basic call* use case. We also call out the extension point used by the extension.



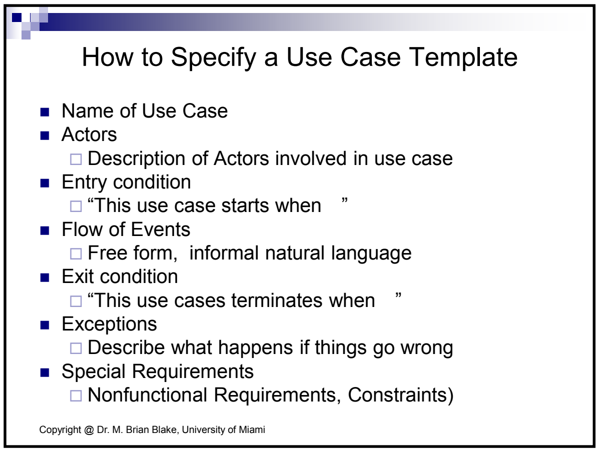
condition: {customer pressed flash, dialed *76}
extension point: Finished

OMT Behavior Analysis

- During conceptualization
  - □ Identify actors and roles
  - □ Diagram the Use Cases
- During analysis
  - □ Describe Use Case behavior in detail
- During design
  - □ Design object interactions to satisfy Use Cases
- During test
  - □ Structure tests to verify Use Cases

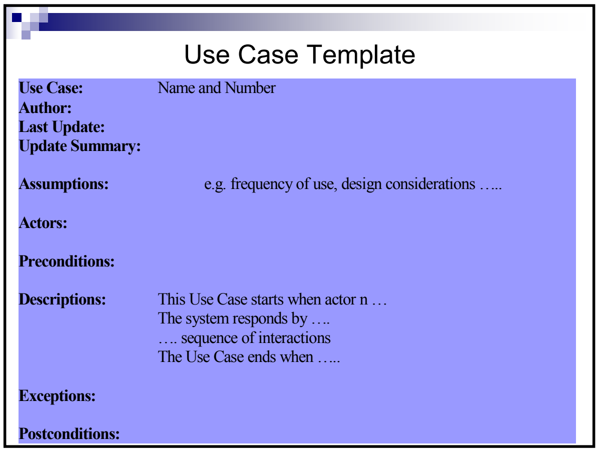
Exercise: Use Case Real World Problems

- Develop Use Case Diagrams for your Frisbee projects



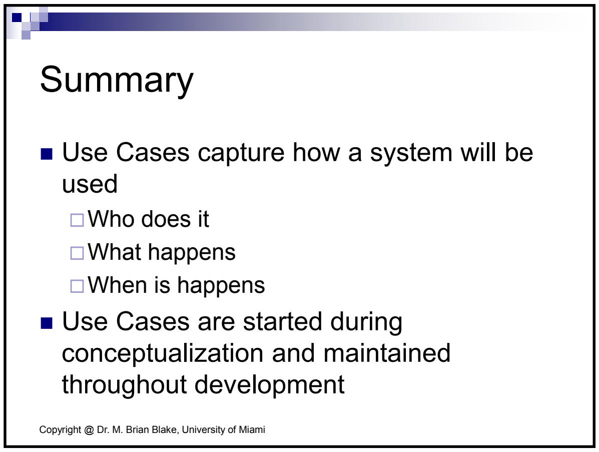
How to Specify a Use Case Template

- Name of Use Case
- Actors
  - □ Description of Actors involved in use case
- Entry condition
  - □ “This use case starts when ”
- Flow of Events
  - □ Free form, informal natural language
- Exit condition
  - □ “This use cases terminates when ”
- Exceptions
  - □ Describe what happens if things go wrong
- Special Requirements
  - □ Nonfunctional Requirements, Constraints)



Use Case Template

Use Case:	Name and Number
Author:	
Last Update:	
Update Summary:	
Assumptions:	e.g. frequency of use, design considerations …..
Actors:	
Preconditions:	
Descriptions:	This Use Case starts when actor n … The system responds by …. …. sequence of interactions The Use Case ends when …..
Exceptions:	
Postconditions:	



Summary

- Use Cases capture how a system will be used
  - □ Who does it
  - □ What happens
  - □ When is happens
- Use Cases are started during conceptualization and maintained throughout development