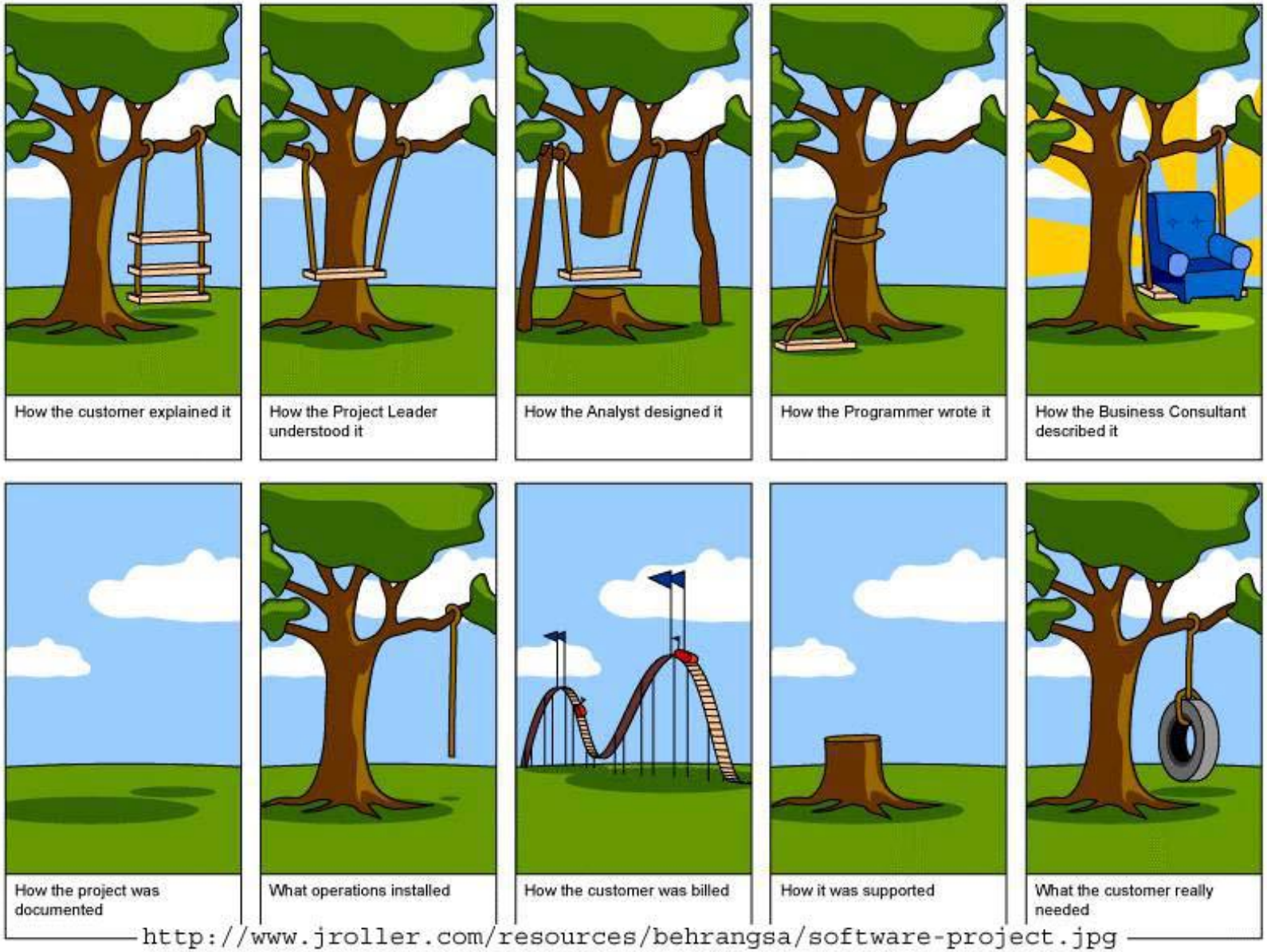
CT1404 Lecture 2

Requirement Engineering

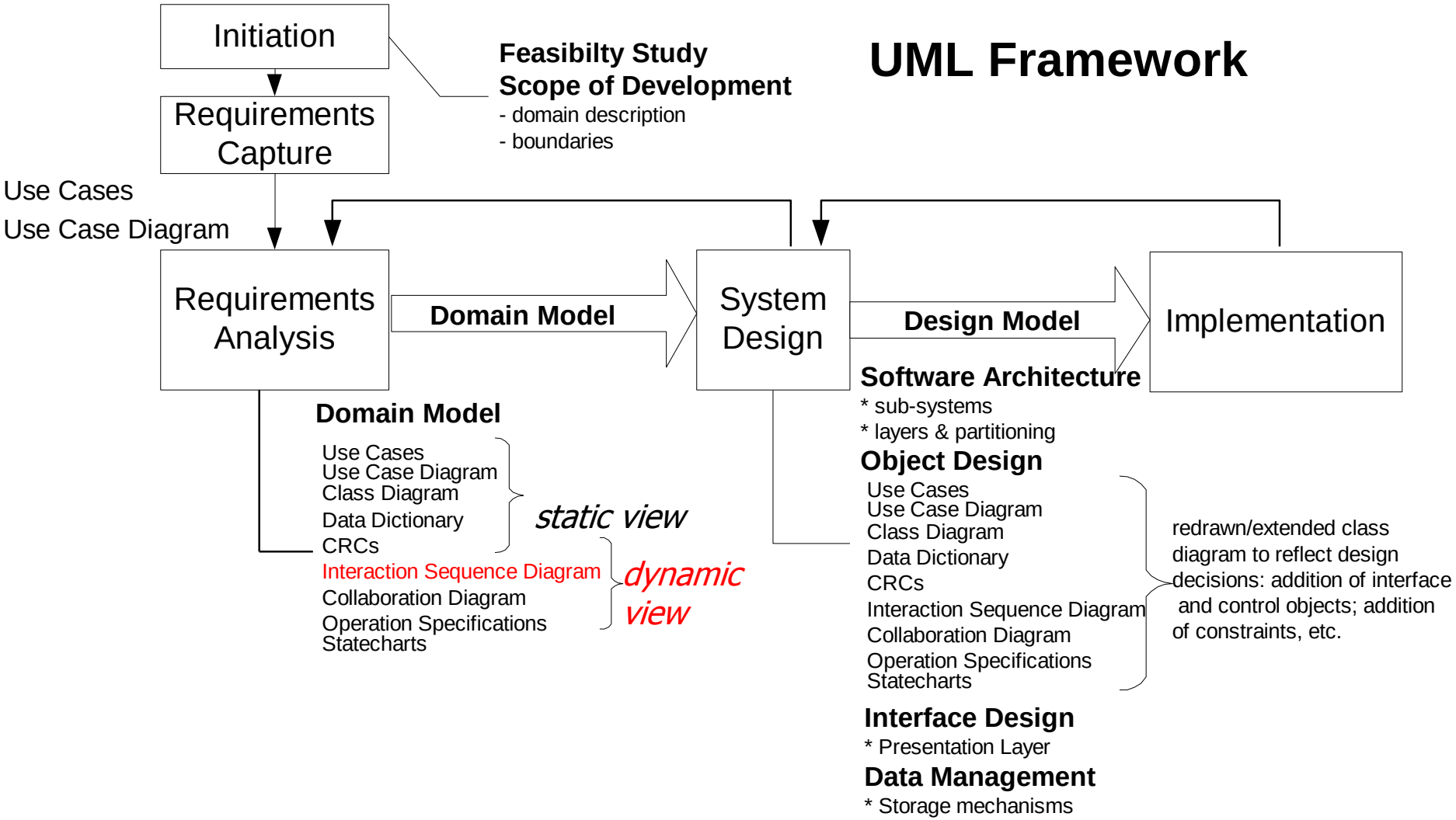
Today's Lecture

- Concept of system requirements
- Techniques to solicit requirements
- Elaboration of use case diagrams
- Organization and documentation of system requirements in the form of Use Cases



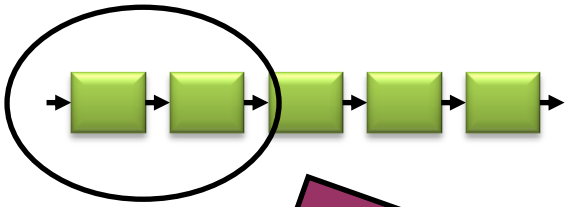
http://www.jroller.com/resources/behrangsa/software-project.jpg

UML Framework



CONCEPT OF SYSTEM REQUIREMENTS

Context of requirements analysis

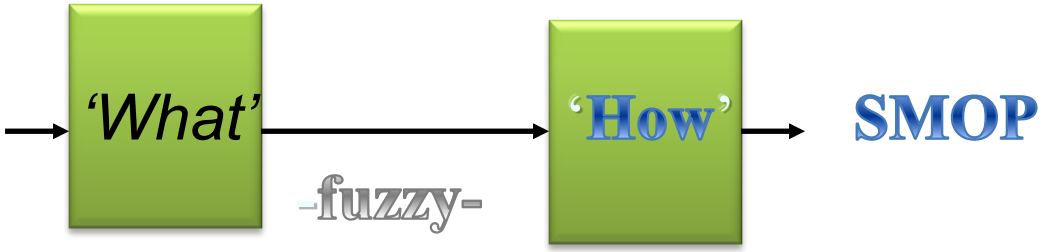


Build the right product

Build the product right

Requirements
(Analysis)

Design



SMOP

System requirements should set out **what** the system should do rather than **how** this is done

Type of Sys. Requirements

Functional requirement

- Describe a system **service** or **function**

- **Examples:**
  - What Systems do
  - Inputs, Outputs, Process

Non-functional requirements

- Describes a **constrain** the system placed or on the development process

- Non-Functional requirements may be further divided into *quality attributes* and *constraints*

Non-functional requirements

quality attributes

- Usability
- Reliability
- Supportability
- performance

constraints

- Hardware
- interface
- language
- Security

Techniques to solicit requirements

- The task of the requirements determination phase is to determine, analyze and negotiate requirements with the **stakeholders**

Stakeholders

- People or organisations that are affected by the proposed system or have a potential influence on the requirements
- Includes:
  - Client
  - User
  - Development
  - Others involved in context of use
- **Important to involve key stakeholders in requirements capture process**

Approaches to requirements capture

- Lots of methods & techniques
- **Main approaches**
  - Data- processing oriented
  - Task and activity oriented
  - Viewpoint oriented
  - Contextual method

- **Requirements capture techniques:**
  - Observation
  - Interview
  - Questionnaires
  - Scenario Walkthroughs
  - Focus groups
  - Prototypes.

Approaches to requirements capture

- **Result:**
  - Large legalistic documents
  - Easy to misinterpret
  - Changes hard to manage
  - Easy to miss / omit requirements

Approaches to requirements capture

- **Modern approach**
  - Model using UML
    - Use cases are used to capture **functional** requirements
      - Use Case Diagram
      - Set of Use Case descriptions
    - Perhaps augmented by an Activity Diagram.

Use Case Modeling with UML

- In combination with plain text
  - Large volume of requirements difficult to consume as a whole
  - Modeling provides constructs to help organize ideas
  - Visualization helps clarify complex ideas
  - Standards bring focus to key abstractions

Advantages of Use Cases

- Avoid **analysis paralysis**
  - Use cases help break up the problem so it can be solved incrementally.
  - Do just enough analysis to get started but we don't have to worry that it's hard to come back later and add more.
- Avoid **gold plating**
  - Using use cases to derive functional requirements avoids stating a functional requirement that is not directly tied to a user task needed to accomplish a business goal.

The Use Case Model

- Writing Requirements in Context”
- Identify user goals, corresponding system functions /business processes
- Brief, casual, and “fully-dressed” format

Actors

System

Goals

Stories- example of use case description

- Using a system to meet goals
- E.g. brief format
  - **Process Sale**: A customer arrives at a checkout with items to purchase. The cashier uses the POS system to record each purchased item. The system presents a running total and line-item details. The customer enters payment information, which the system validates and records. The system updates inventory. The customer receives a Receipt from the system and then leaves with the items.

Definitions

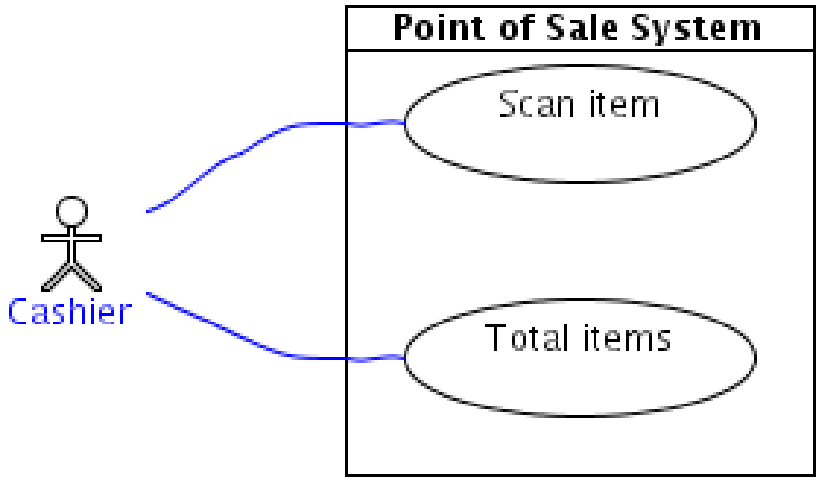
- **Actor**: something with a behavior; person, system, organization
- **Scenario**: specific sequence of actions and interactions between actor(s) and system (= one story; success or failure)
- **Use Case**: collection of related success and failure scenarios describing the actors attempts to support a specific goal

Use Case Modelling

- Use Case diagram
  - Diagram illustrating
    - Actors
    - Use cases

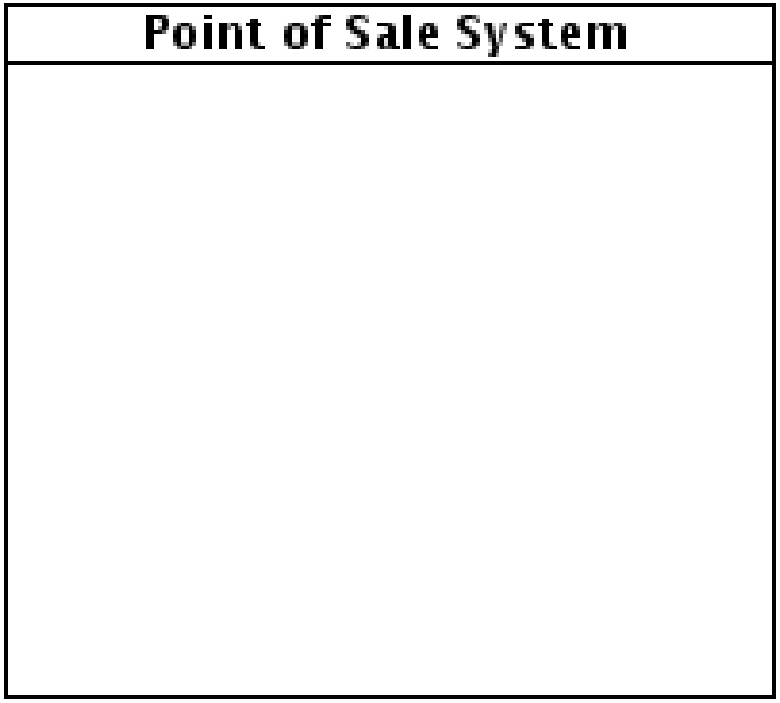
    In the system
- Use Case Description
  - Specification of what happens in each use case
    - Textural description
    - Diagrams

Use Case Diagrams



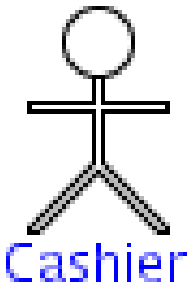
- A cashier uses the POS system to scan an item.
- A cashier uses the POS system to total items.

System Boundary

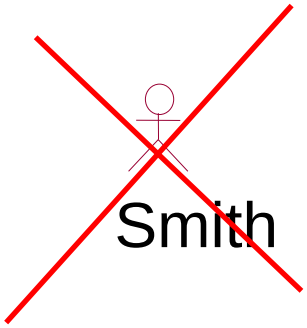


- Marks off the system as separate from its environment
- Actors are outside
- When no system boundary is shown, the system is assumed

Actor



- Someone or something outside the system that interacts with it
- Actors represent “roles” not individuals



Use Case

- A use case achieves a goal of value to an actor
- Defines a task which must be supported by the system
- What system is for not how it does it
- Starts with an active verb from the point of view of the system

Communications

- Called relations
- Lines between Actor and Use Case summarize interactions graphically
- Actors and use cases exchange information to achieve the goal but the details of interaction are not shown

Considerations for Use Case Diagrams

- Do not represent the flow of information or sequence of events.
- Do not represent communication between actors. Keep the focus on putting the system in context, not the actors. Actors may collaborate through a use case.
- Actors are not always roles played by a person. Actors may represent the role played by anything that acts on the system such as another system.

Use Case Diagram Summary

- Show the system in its environment
- Show what a system is used for
- From a behavioral perspective:
  - For capturing functional requirements
  - For enabling incremental specification
  - To understand who the system is for
- Details of interactions are not shown

How to find use cases (Guideline)

- Choose the system boundary
  - Software, hardware, person, organization
- Identify the primary actors
  - Goals fulfilled by using the system
- Identify goals for each actor
- Define use cases that satisfy goals
  - User-goal level use cases will be one-to-one with user goals
  - Exception: CRUD (Create, Retrieve, Update, Delete)
    - E.g., “edit user”, “delete user”, …
    - Collapse into a single use case
      (e.g., *Manage Users*)

Choose the system boundary

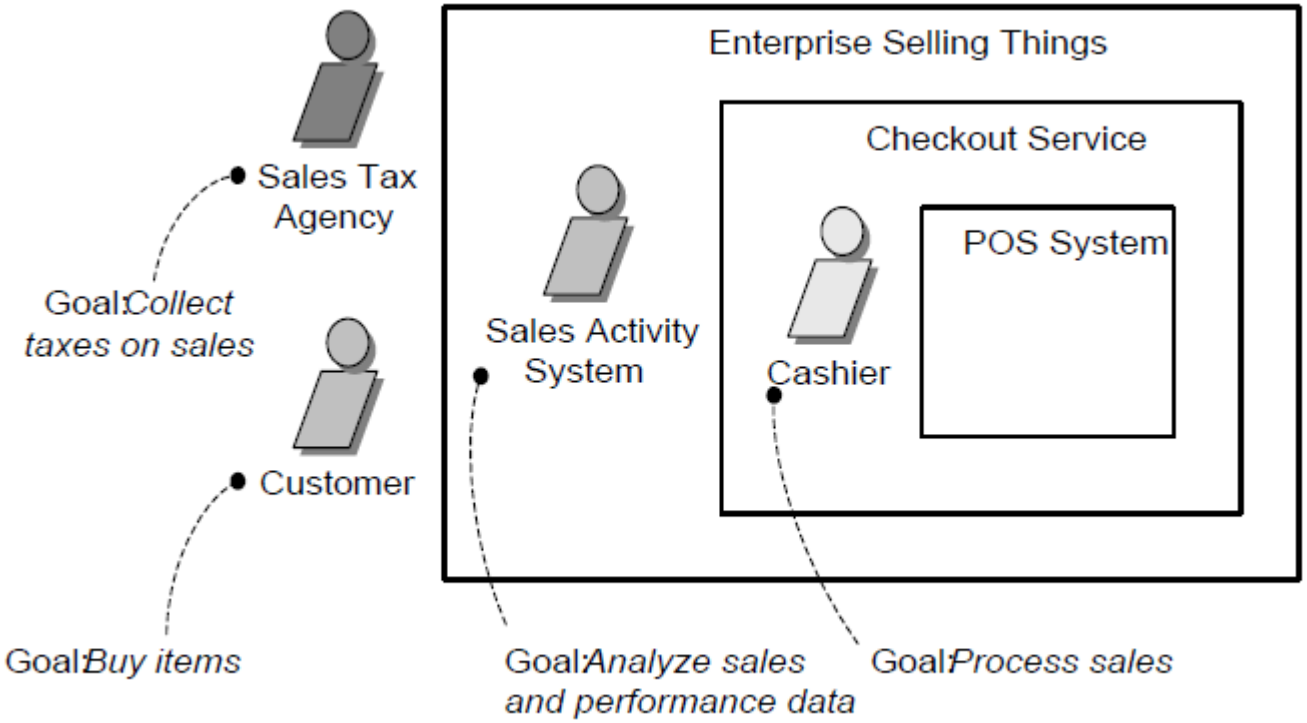
- In the case study, the POS system itself is the system under design;
- Everything outside of it is outside the system boundary
- Identify what is outside would help to identify the boundary

Find primary actor and goals

- Questions to help find actors and goals
  - Is “time” an actor because the system does something in response to a time event?
  - In addition to *human* primary actors, are there any external software or robotic systems that call upon services of the system?

Actor	Goal
Cashier	Process sales Process rentals Handle returns …
Manager	Start up Shut down
System Admin	Add user Modify user …
Sales activity System	Analyze sales and performance data

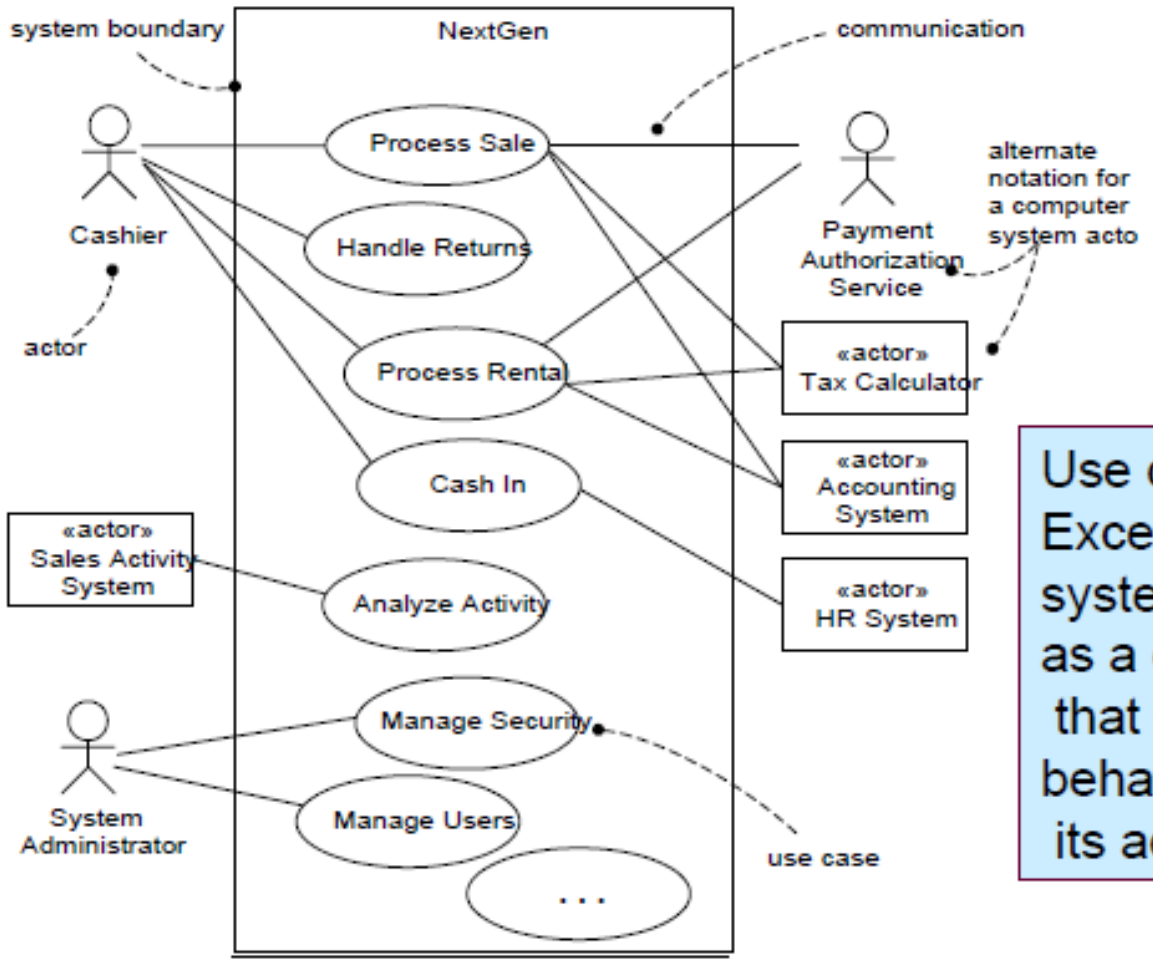
Actors, Goals & Boundaries



Valid or useful use cases

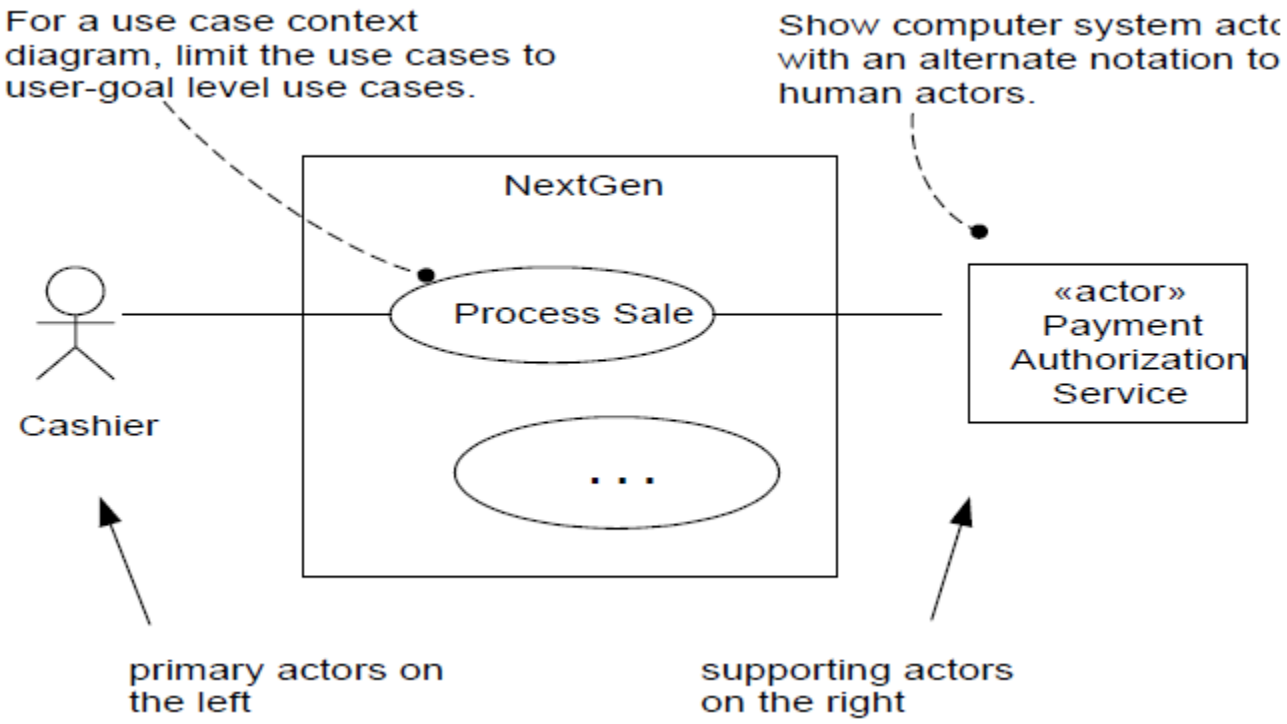
- Which of these is a valid use case?
  - Negotiate a Supplier Contract
  - Handle Returns
  - Log In
  - Move Piece on Game Board
- All are valid, but at different level; better question is “what is a useful level to express use cases”

Applying UML: Use Case Diagram



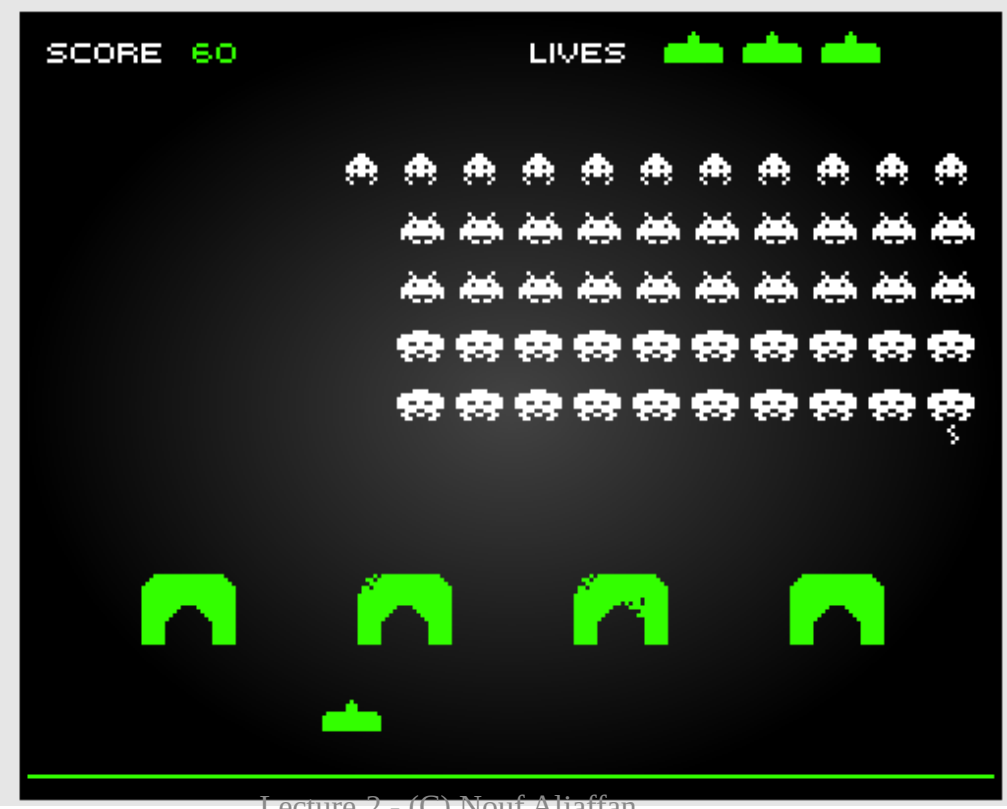
Use case diagram is an Excellent picture of the system context. It serves as a communication tool that summarizes the behavior of a system and its actors

Diagram notation

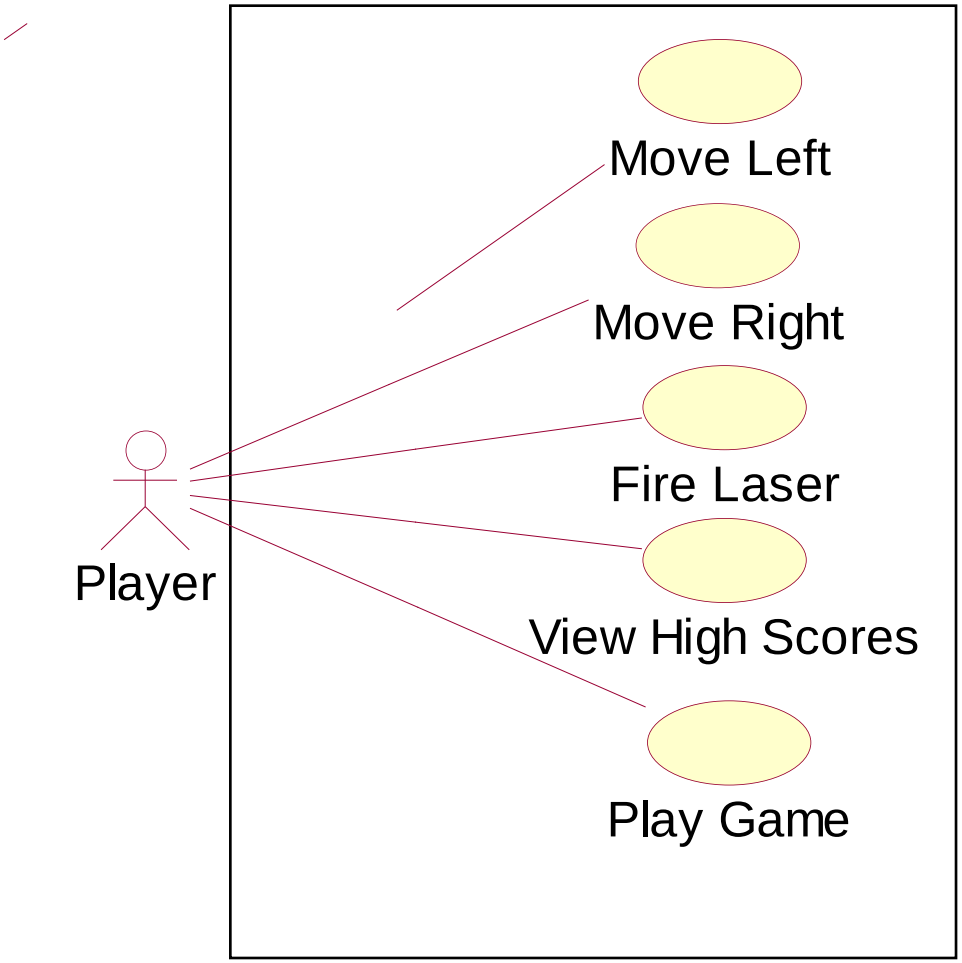


Exercise 1 – Use Case Diagram

- In groups of 3-4 spend 5 minutes attempting to draw a use case diagram for the space invader game below:



Exercise 1 Solution

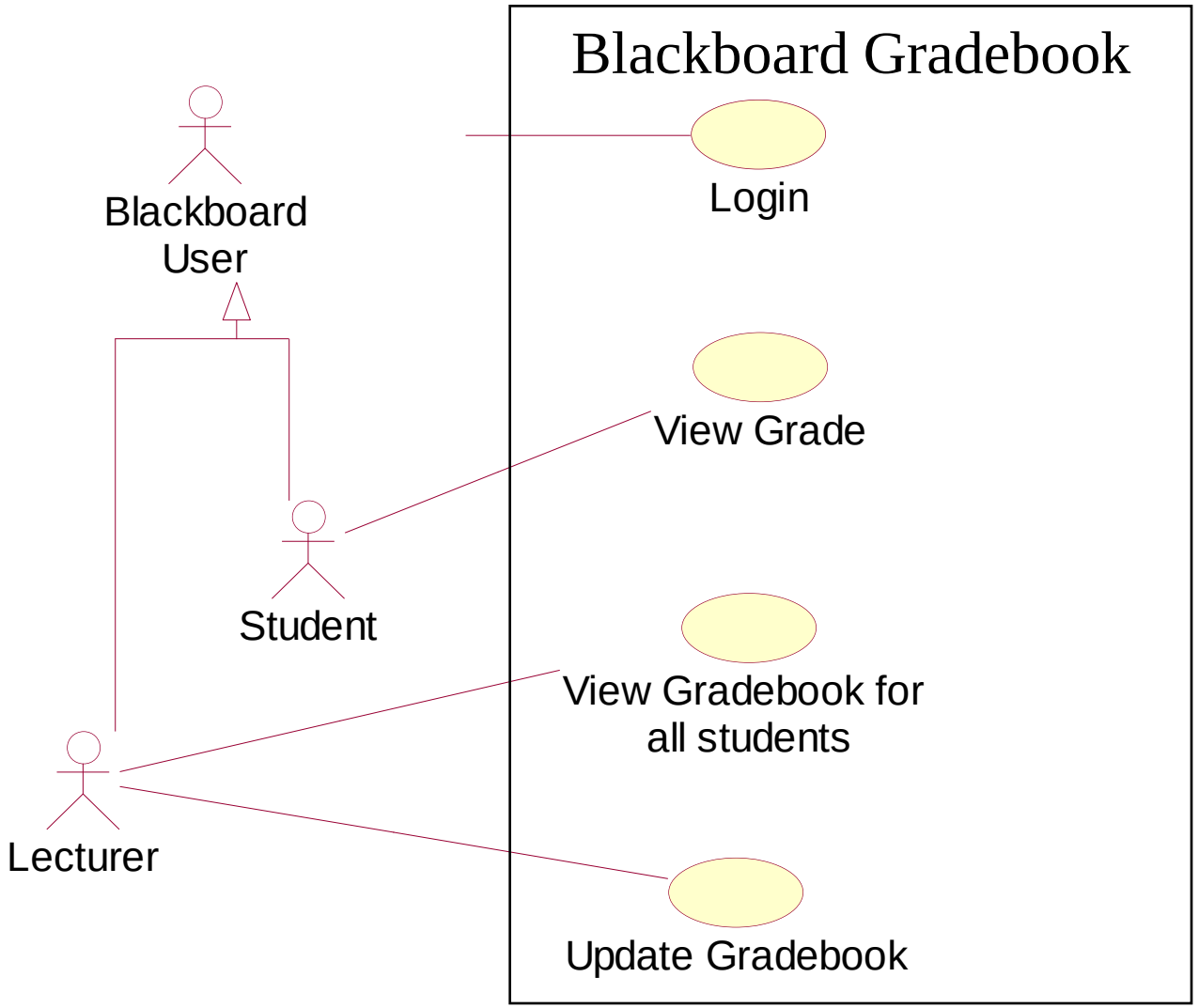


Relationships in use cases

- Between actor and use case
  - Actor uses
- Generalisation of actors
  - Types of users
- Use case stereotypes
  - <<extend>>
    - Optional
  - <<include>>
    - Mandatory
- Stereotype is a UML extension mechanism to indicate a type of behaviour

Generalisation of Actors

Question:
Is Login part of this system?



Use case variants : include and extend

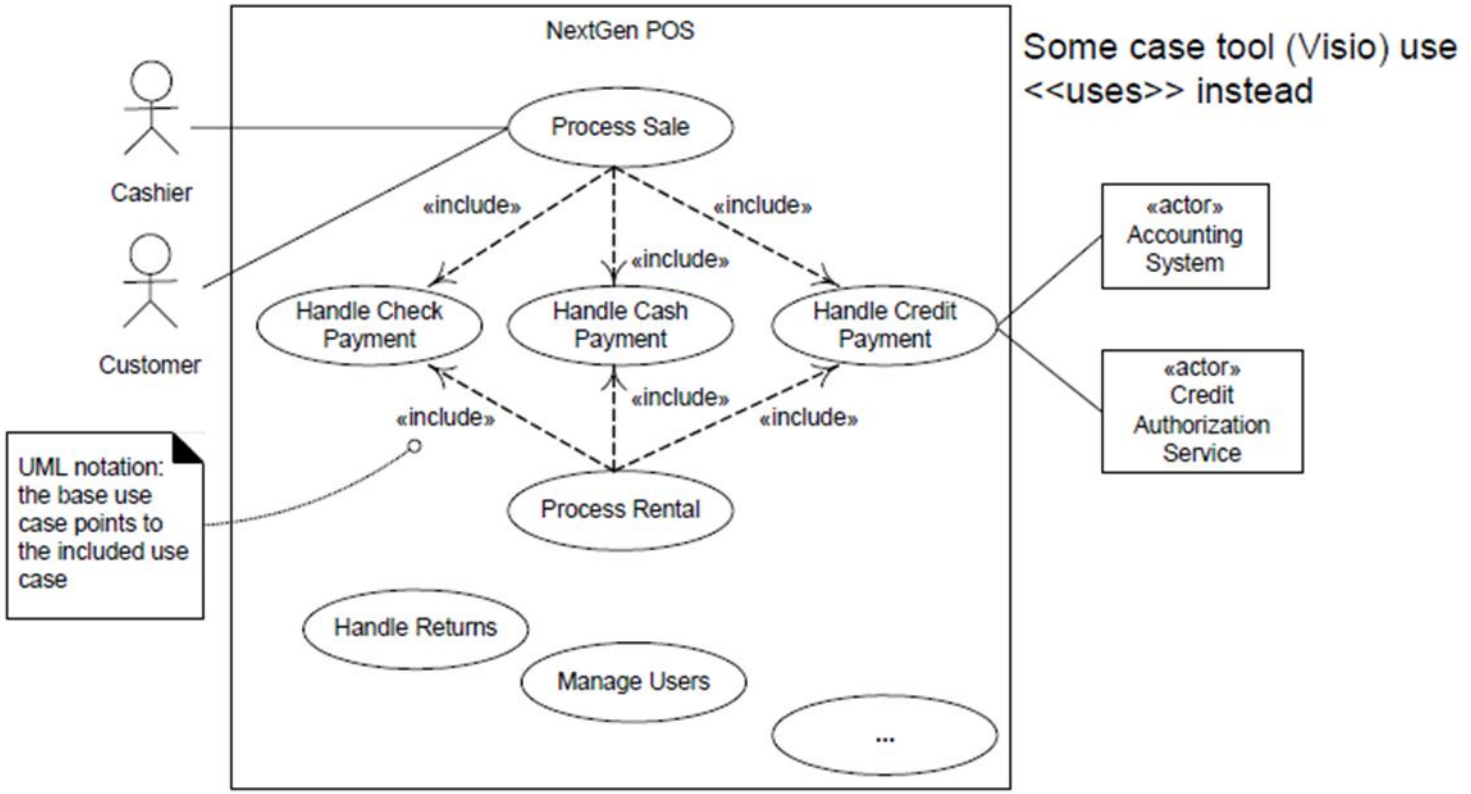
include

- Partial behaviour that is common across several use cases
  - E.g., PayByCredit occurs in several use cases:
  - Process Sale, Process Rental, and so on
- Solution: represent as a separate use case
  - "Refactoring and linking text to avoid duplication" (Larman)
- "Use << *include* >>

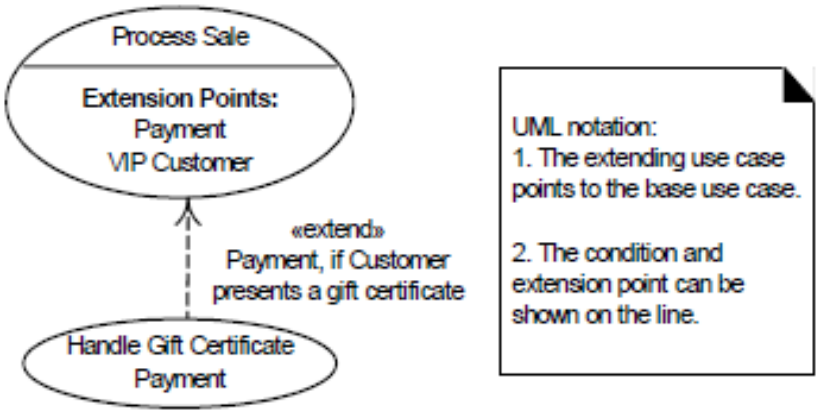
extend

- *How to add an extension to a use case that is frozen or too complicated?*
- *Solution: use extend to relate to an addition use case*
  - any Use Case can have more than one extend
  - use when describing a variation on or in addition to normal behavior
  - OPTIONAL BEHAVIOUR
    - Otherwise part of use case

<<include>>

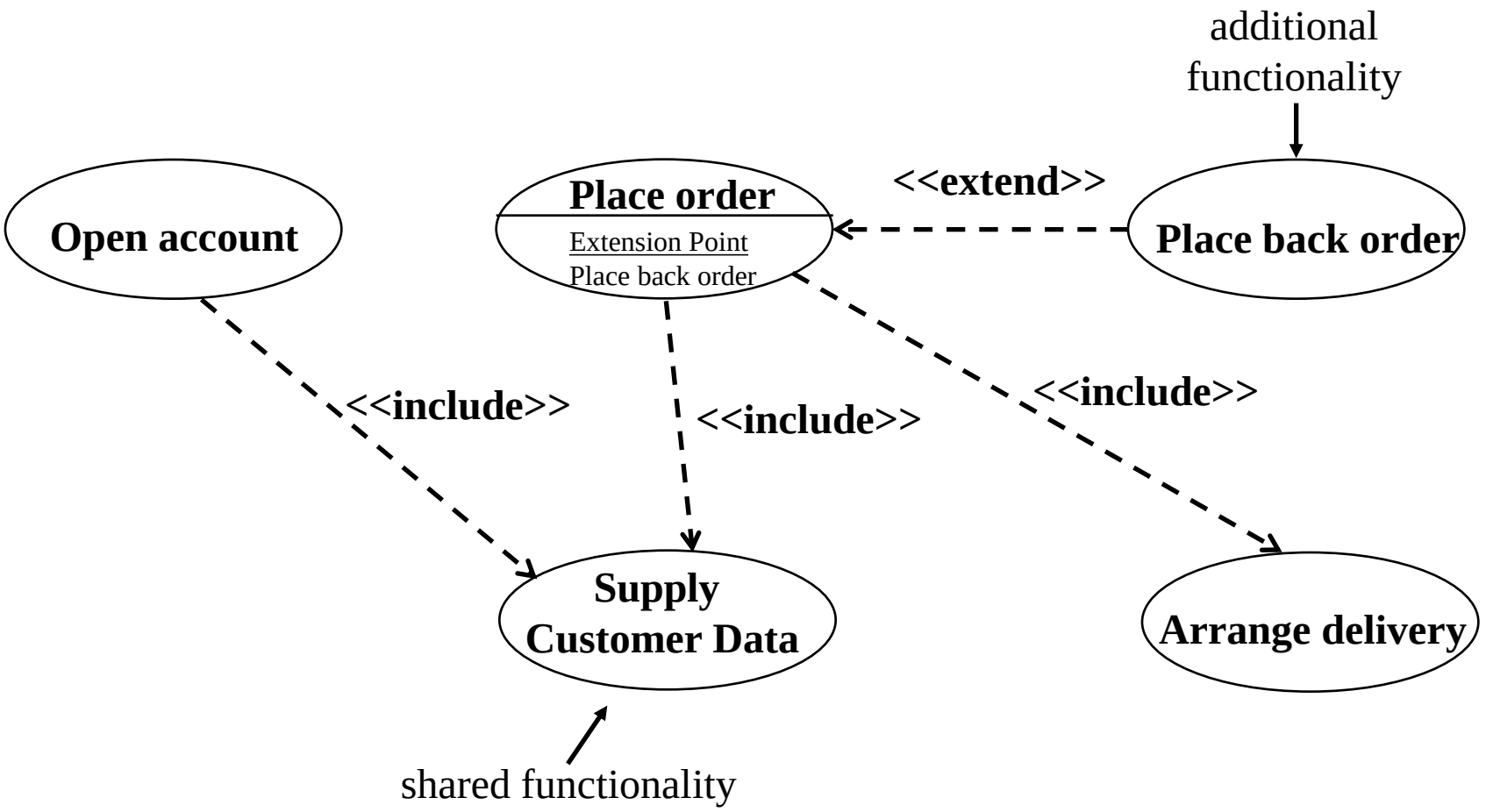

NextGen POS
Some case tool (Visio) use <<uses>> instead
Cashier
Customer
Process Sale
«include»
«include»
«include»
Handle Check Payment
Handle Cash Payment
Handle Credit Payment
«include»
«include»
«include»
Process Rental
Handle Returns
Manage Users
...
«actor» Accounting System
«actor» Credit Authorization Service
UML notation: the base use case points to the included use case

<<Extend>>



- Handle Gift Certificate could simply have been recorded by adding it as an extension in the Extensions section of Process Sale

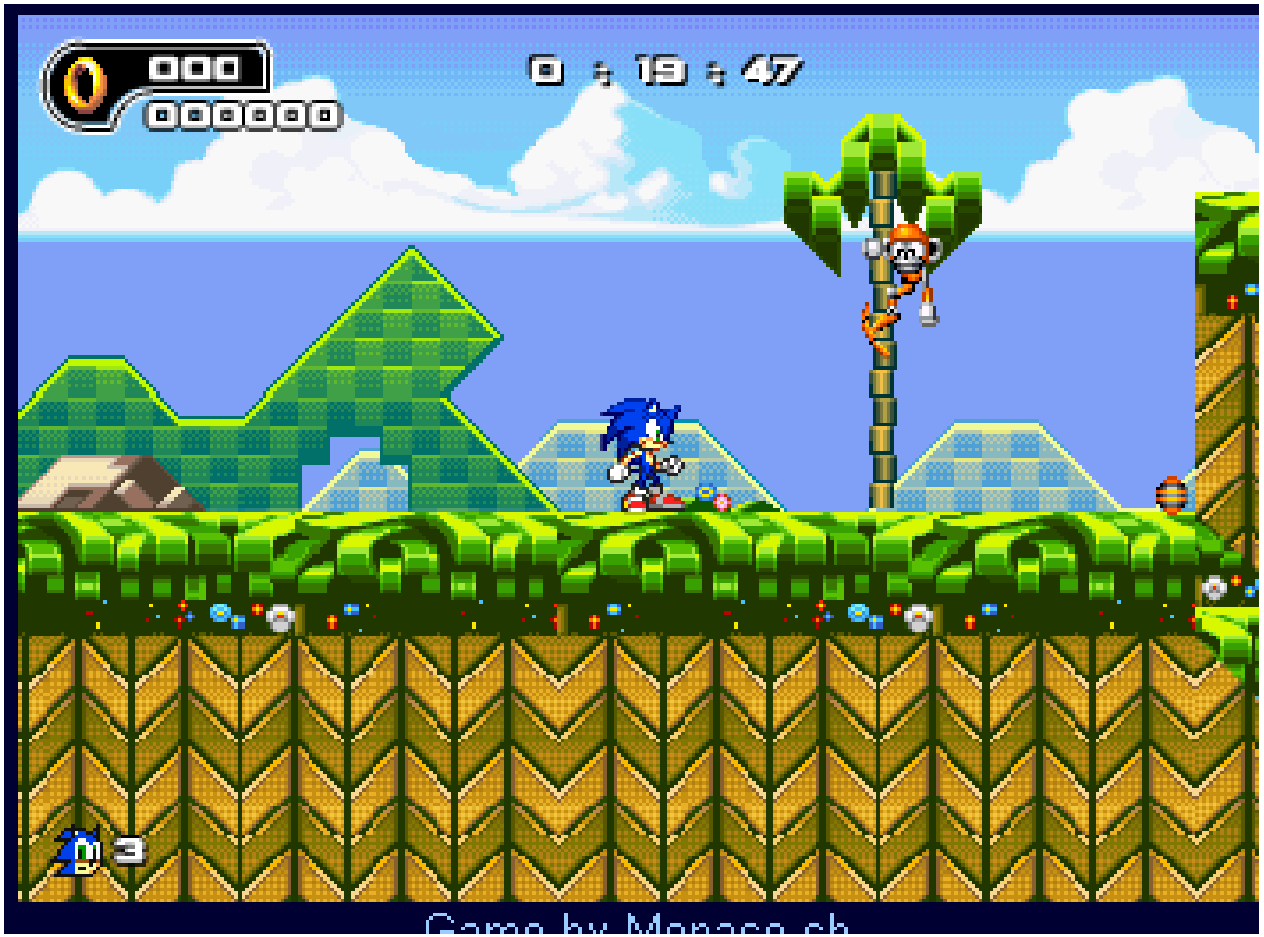
Example of Use Case Variants



Exercise 2

Consider Sonic the hedgehog.

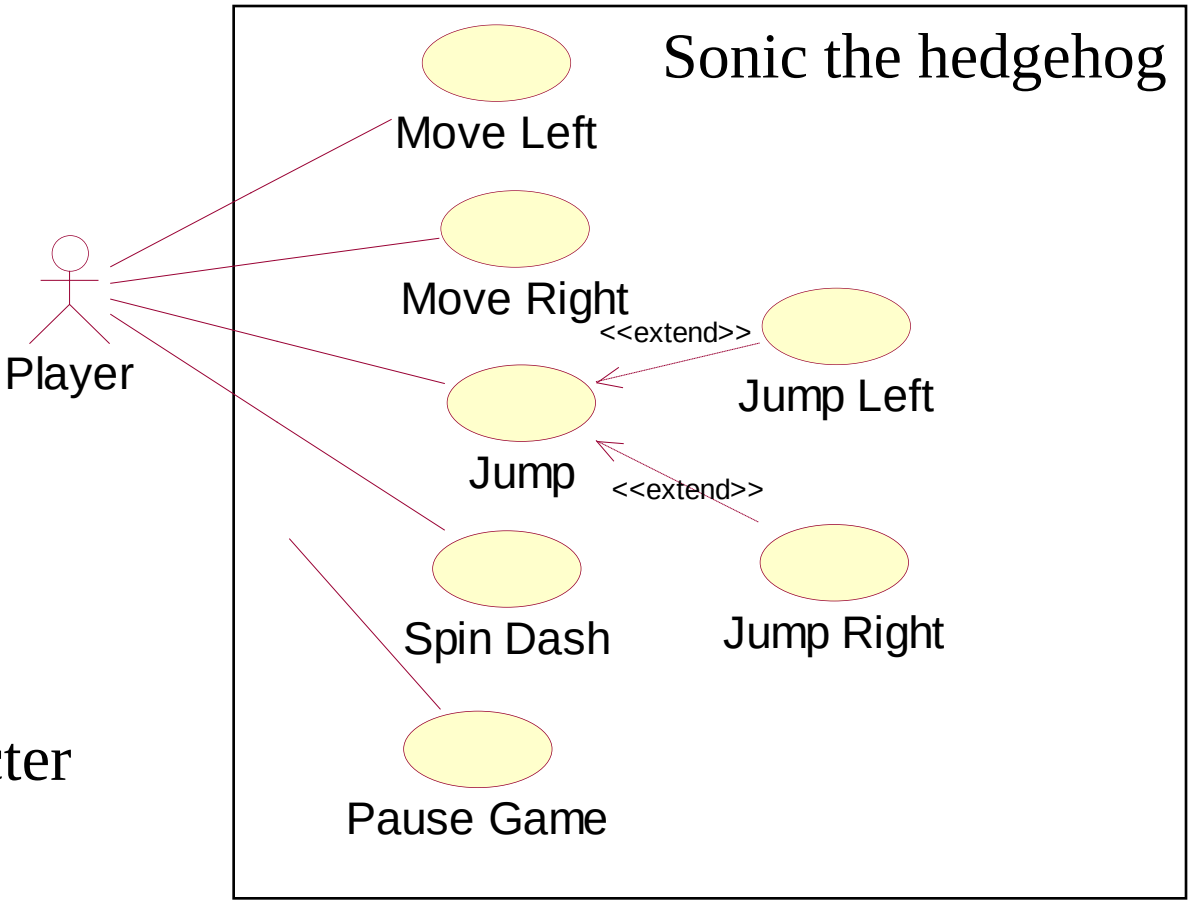
1. What can sonic do?
2. What are the use cases?
3. Are there any relationships
4. Draw the use case diagram



1. Move left
2. Move right
3. Jump
4. Jump left
5. Jump right
6. Spin Dash
7. Pause

http://www.ebaumsworld.com/games/play/1108/

Exercise 2 – Possible Solution



What about:
New game
Choose character
etc

Summary

- What is a use case
- How to draw a use case diagram
- Use case stereotypes
- Relationships between Actors
- Next Lecture
  - Use case descriptions