# Requirement Process

## The Requirement Process

- Requirement Elicitation
  - Gather requirement from our users
- Requirement Analysis
  - Analyze the requirement and build a model for the potential system

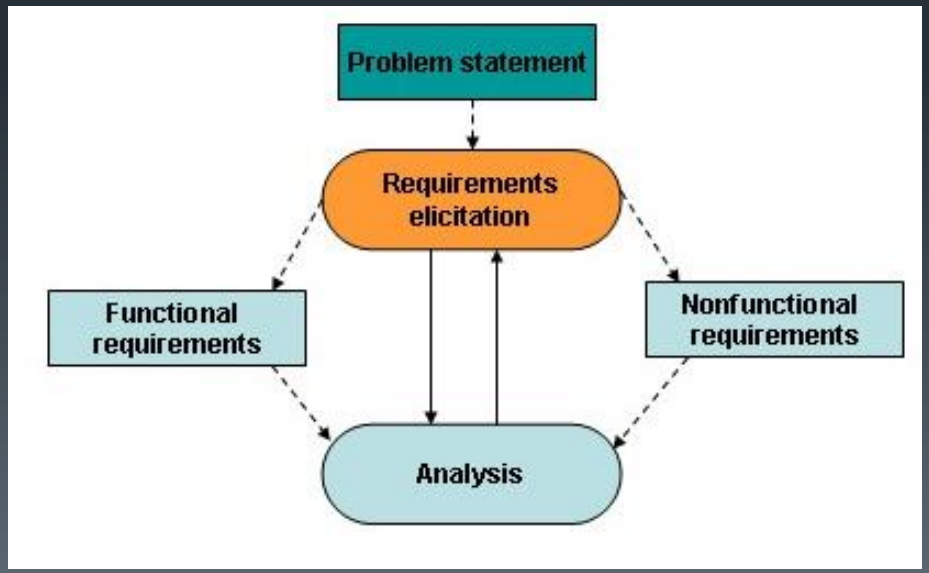
# Requirement elicitation

# What is Requirement elicitation

Requirements elicitation is about bringing out the requirements that originally reside in people's minds.

- A process among users, clients and developers for **defining the new system**
- "Elicit" means "**to bring out** or **to draw out**"
- Requirement elicitation tries to
  - bring out what the user already knows about the potential system (that's comparatively easy)
  - draw out what the user doesn't know or think of about the system

## What does requirement elicitation consume and produce?



# Requirement concepts

- **Completeness**: no scenarios are omitted
- **Consistent**: no contradictory requirements
- **Unambiguous**: the requirement specification cannot be interpreted in two mutually exclusive ways
- **Correctness**: accurately reflects the client's requirements
- **Realistic**: the system can be implemented within constraints
- **Verifiable**: repeatable tests can be designed to demonstrate that the system fulfills the requirements
- **Traceable**: if each requirement can be traced to the appropriate functions the system provides and vice versa

# Types of Requirement

- Functional Requirement
  - What the features/functions should support
  - Describe the interaction between the user and the system without concerning the implementation
- Non-functional Requirement
  - Refers to those requirements that are not directly related to the features of the system
  - Some examples are:
    - Usability
    - Reliability
    - Performance
    - Supportability

# A Collaborative Process

- Requirement elicitation is a collaborative process that requires several groups to involve
  - Users & domain experts are more specialized in their domain and have a better idea of what the system behaves, *but they are not the expert of software development*
  - Developers are technically competent to build software system, *but they are not specialize in the user environment*

# Common Problems in Requirement Process

- Developers and Users are from different worlds.
- They speak different languages and express requirement in different view points.
- Users do not understand
  - The technical jargon;
  - Why developers are always talking about jargon.
- Developers do not understand
  - The user environment and business processes;
  - Why users know so little about technical jargon.

**Use case model is designed to bridge the gap between developers and users**

# Requirement Elicitation Activities

- Requirement elicitation involves several activities:
- Define the initial problem statement
  - Identifying functional requirements
  - Identifying non-functional requirements
- Identifying actors
- Identifying scenarios
- Identifying use cases
- Refining use cases
- Identifying relationships among use cases

- Outcome of Requirement Elicitation = Use Case Model

# Use Case Modeling

- Models the ‘actors’ outside a system and their interactions with that system
- Every way that an ‘actor’ uses a system is called a Use Case

# Reasons for Use Cases

- No information system exists in isolation
- Most systems interact with humans or other automated systems (actors) that use the system for some purpose
- Actors expect the system to behave in a predictable way
- Use Cases specify the behavior of the system
- Helps visualize the system

# Elements of Use Case Models

- Problem statement
- Use case
- Actor
- Relationship
- Use Case Diagram
- Scenario
- System Boundary
- Use case description

# Problem Statement

- The problem statement, comes up by project manager & client, describes the problem that the system addresses
- A common problem statement describes:
  - The current situation
  - The functionalities supported
  - The target environment the system would run on
  - The deliverables expected by the client
  - The acceptance criteria

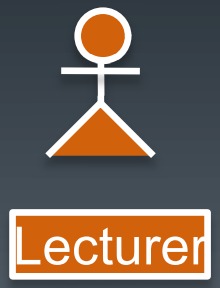
# Use Case

- A Use Case is an interaction between the system and a person or another system to achieve a result
- A required “bit” of functionality
- It yields an observable result of value to an actor (and hence a developer)
- Typically named with a verb than a noun
  - “Do something to something”

# Actor

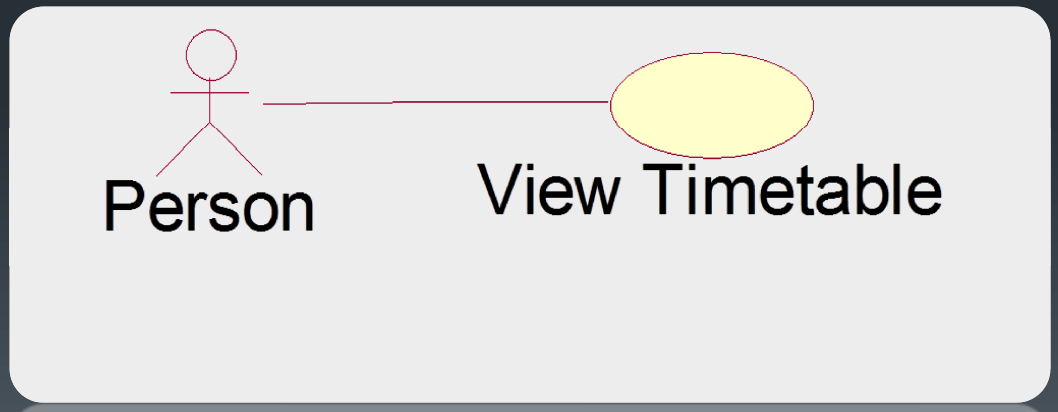
- A coherent set of roles that users of Use Cases play when interacting with Use Cases
- Roles not users or people
- User may have more than one role

- Actors are external entities of the system
- Questions that helps you identify the potential actor:
  - Who are going to rely on the system to perform their work?
  - Who is going to maintain and manage the system?
  - What external system will the system interact with?

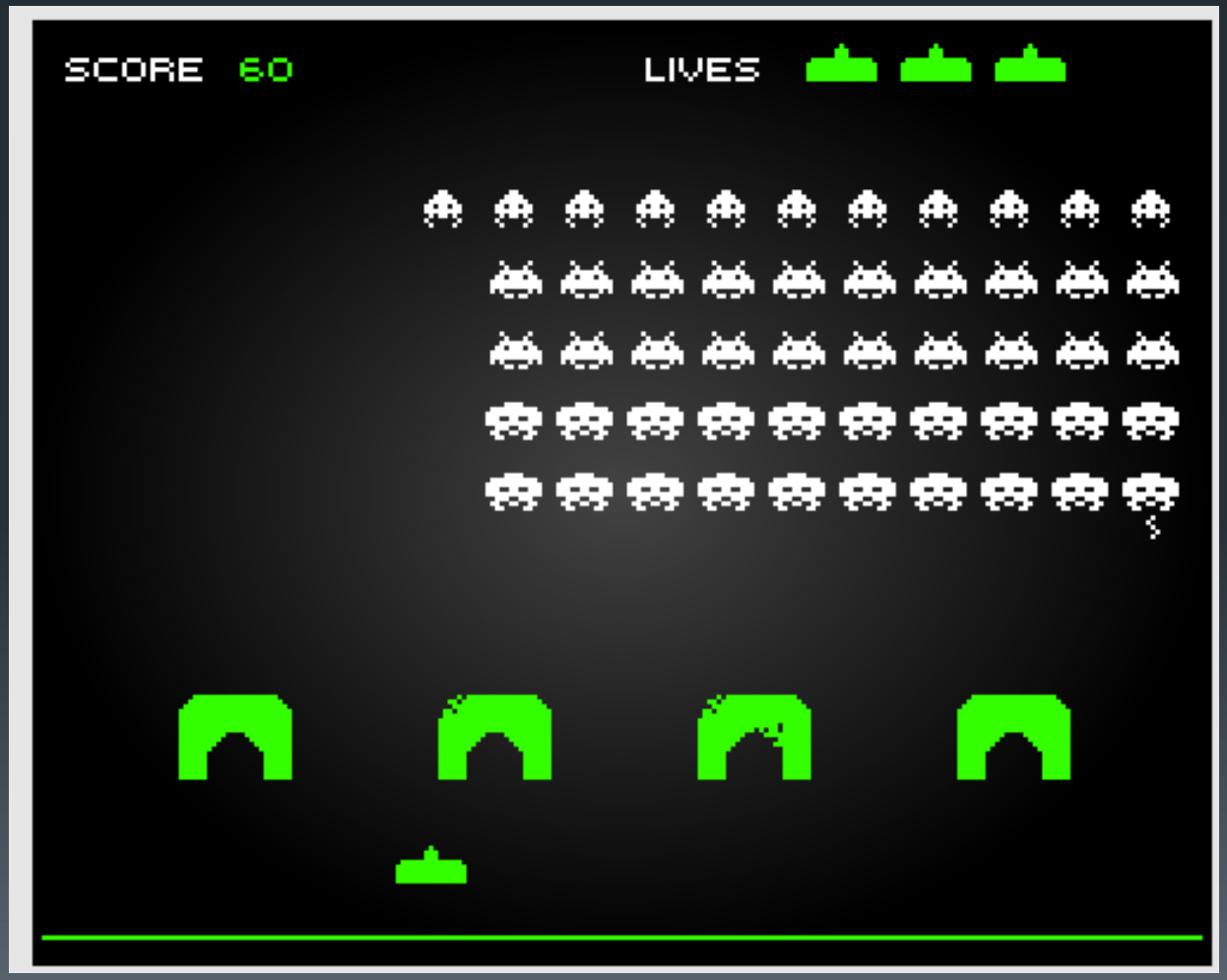



# Use Case Diagram

- A diagram that shows a set of Use Cases and Actors and their relationships
- Use Case diagrams address a user-centric view of a system
- Show a required “bit” of functionality



# Draw use case diagram –space invader



# Draw use case diagram

# Scenario

- A single path through a Use Case
- Use case is usually a collection of scenarios
- Included as part of use case description

# Identifying Scenarios

- Consider all the possibilities in the use case
- Identify the normal (primary) scenario associated with the use case
  - What typically happens
  - Ignore any complications
- Identify the exceptions to the normal action:-
  - Alternative scenarios
    - Things that could happen
  - Exceptional scenarios
    - Things that could be considered errors that need to be caught
- Each scenario should be presented as a series of simple numbered steps in text format

# Simple Scenario – Make Tea

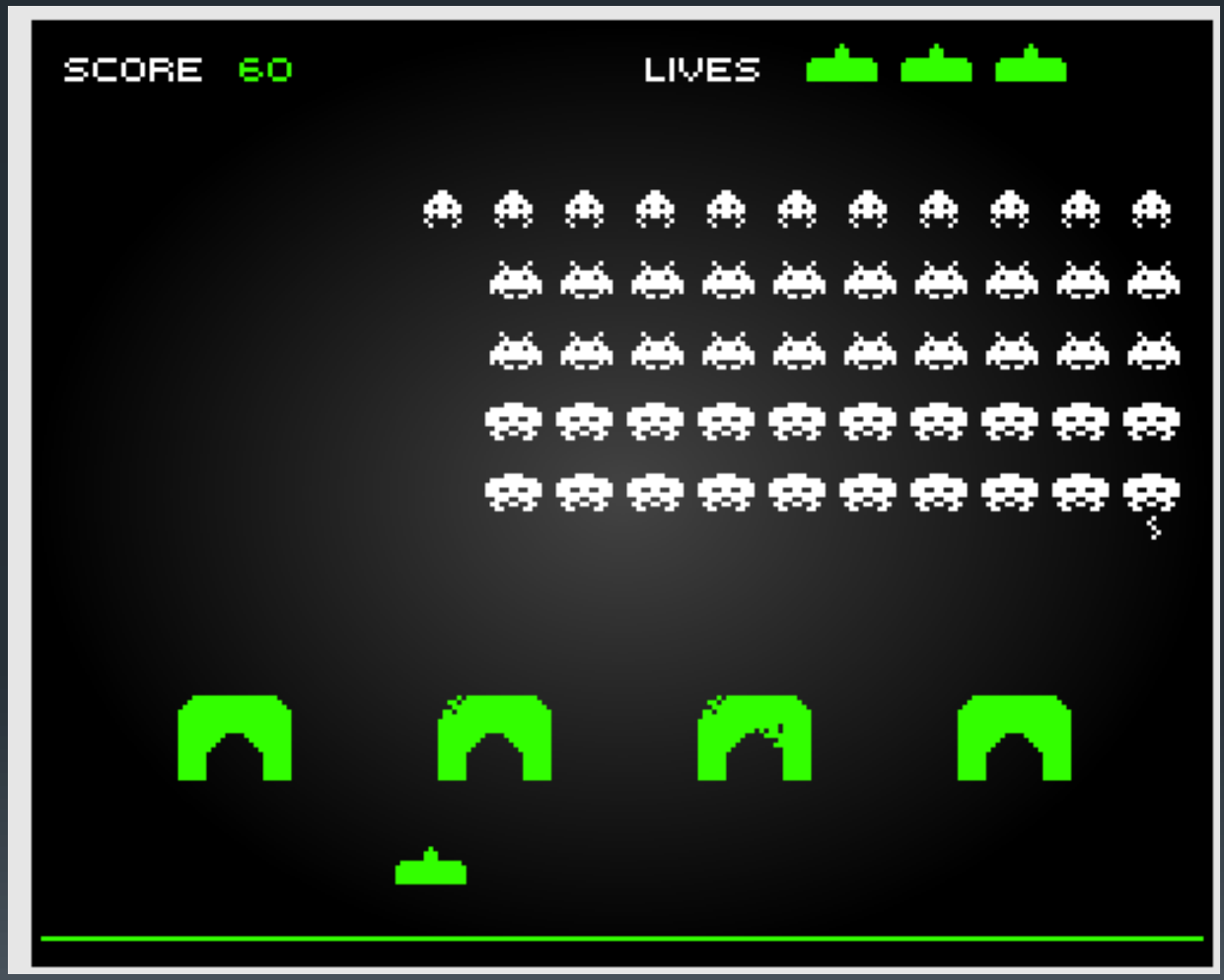
- Normal Scenario:
  1. Switch on kettle to boil water. [A1: Kettle empty]
  2. Place tea bag and milk in mug. [E1: No milk]
  3. When kettle has boiled pour boiling water into mug.
  4. Let tea brew and then remove tea bag and put tea bag in the bin.
- Alternative Scenarios:

  A1. Kettle empty
  1. Fill Kettle with water.
  2. Use case continues from step 1.
- Exceptional Scenarios:

  E1. No milk
  1. Sorry, have tea without milk.
  2. Use case terminates.

What other scenarios are there?

# Use Case Scenarios-“move left” identify scenario

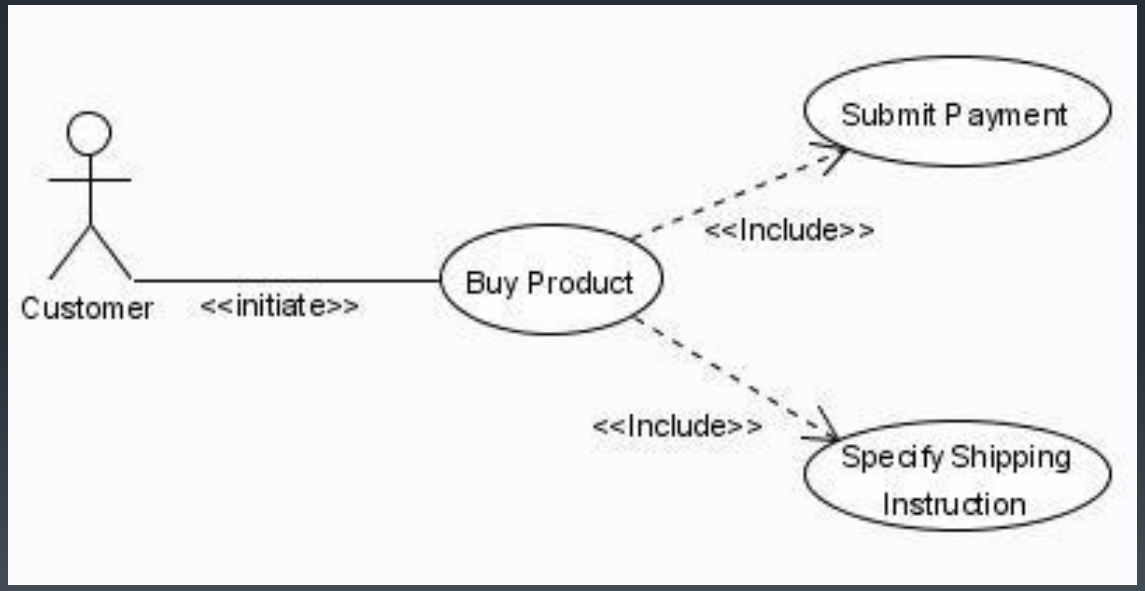


# Use Case Scenarios (for the Move Left use case)

| | |
|---|---|
| Normal Scenario | 1. Player indicates that the base moves to left<br>2. The system moves the base left |
| Alternative Scenario | 1. Player indicates that the base moves to left<br>2. A bullet arrives where the base has moved to<br>3. The base explodes |
| Exceptional Scenario | 1. Player indicates that the base moves to left<br>2. The base is at the extreme left of the screen<br>3. The base stays where it is |

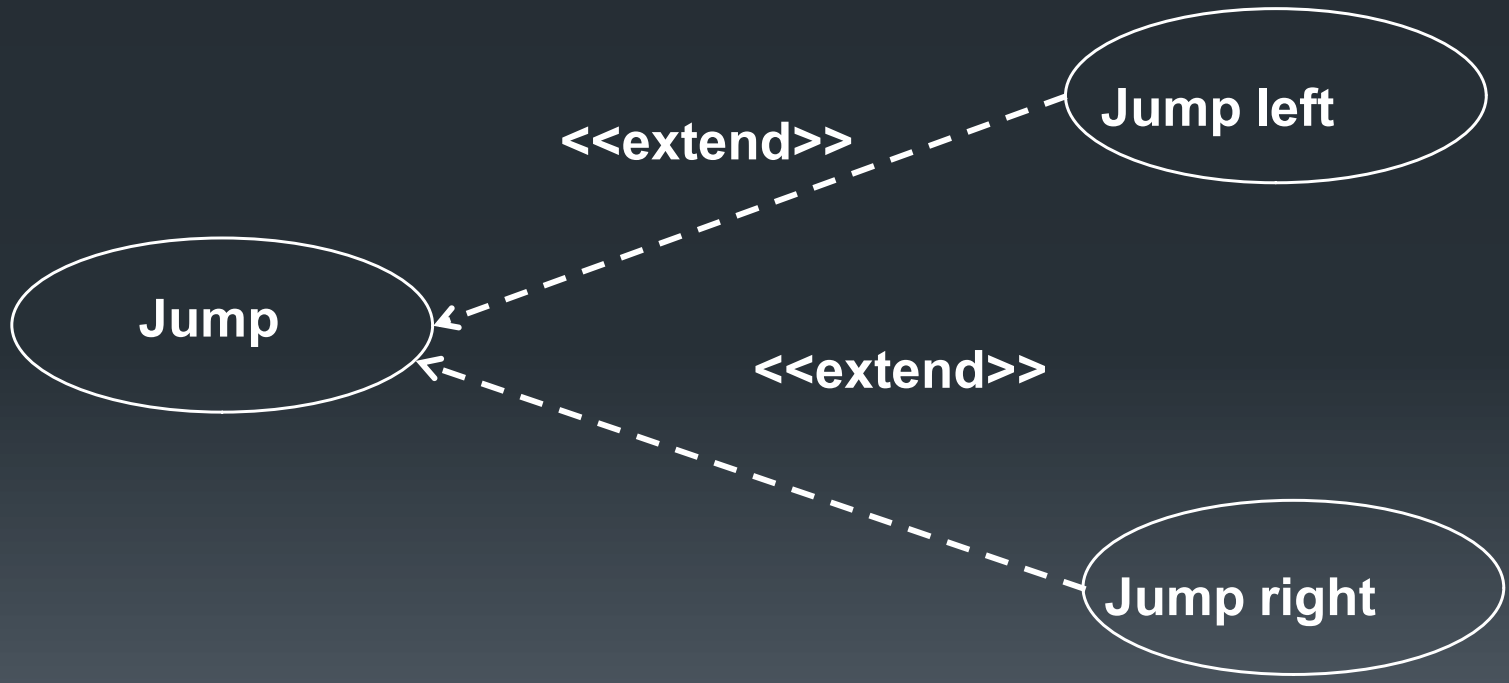
# Refining use case- include

- *include* relationship occurs when you have a chunk of behavior that is similar across more than one Use Case
  - use in two or more separate Use Cases to avoid repetition
  - a significant part of a use case
  - <<*include*>>

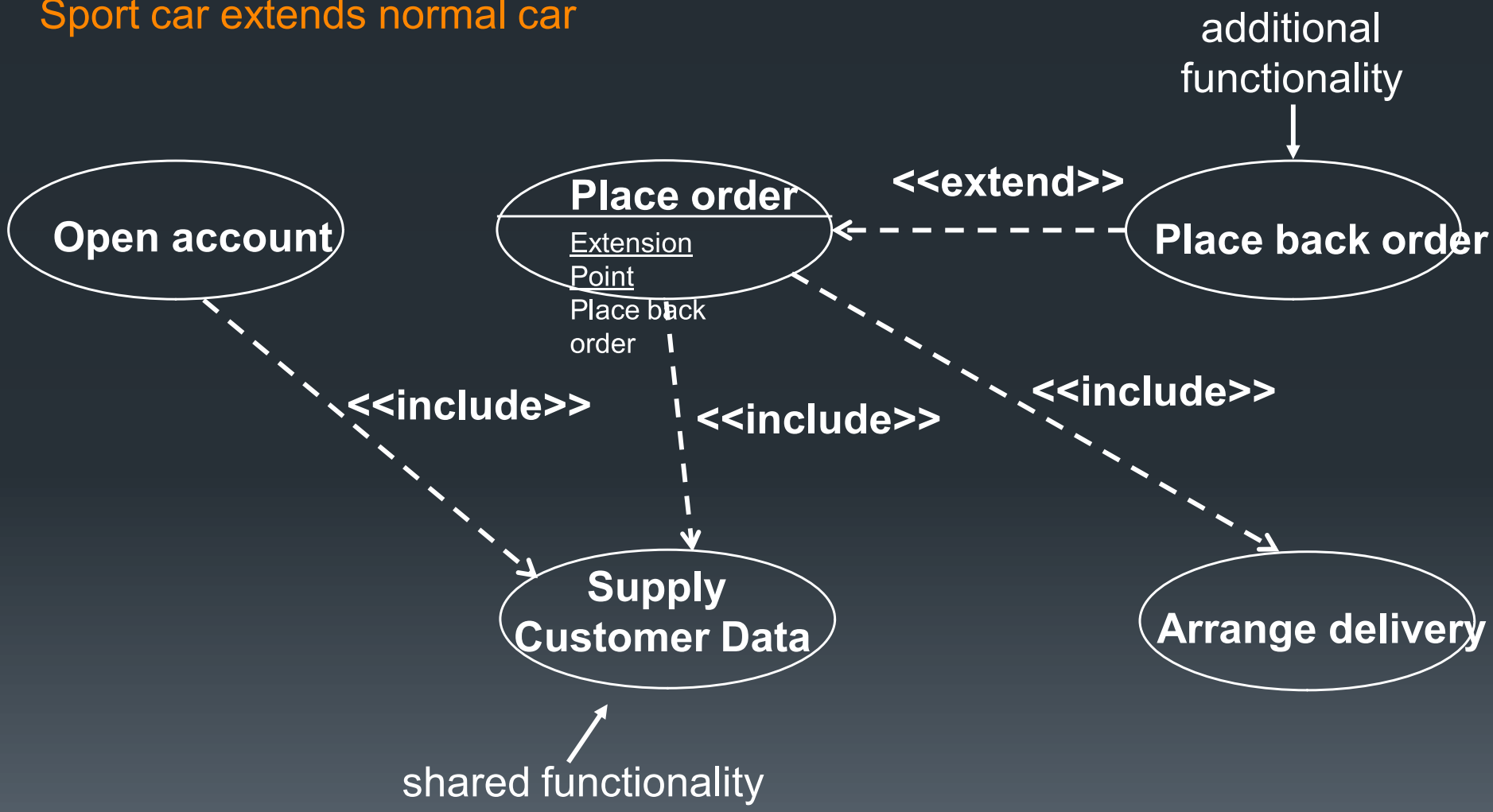


# Refining use case-extend

- *extend* relationship where you have one Use Case which adds functionality to another Use Case
  - any Use Case can have more than one extend
  - use when describing a variation on or in addition to normal behavior



Include->has a e.g. house has a window, car has a window
So house & car both include window
Extend -> is a e.g. Sports car is a normal car adding extra functionality
Sport car extends normal car



# The use of Use Cases

- Use cases are used to capture functional requirements
- Most use case modeling will happen during the early part of a project
- They drive the rest of development
  - You build what the client wants!
- Used as part of planning, testing and evaluation
- New Use Cases will continue to emerge as project iterates

# Possible problems with Use Case modeling

- Danger of mistaking requirements for design
  - You are analysing not designing - avoid technical detail
- Possibility of missing requirements if too much emphasis is placed on actors
- Incomplete picture
  - Non-functional requirements
  - Usability requirements