System Design & Analysis

LECTURE 06

History of Object Model

- The early computer languages that were used for mathematical and scientific operations
  - Freed the programmer from complexities of assembly language
- Growth in size & complexity to evolution in languages to accommodate new features
  - E.g. automation for business application
- Slowly, language evolved even more
  - Software moved closer to problem domain and further away from the machine

Object Model

- The newer languages that were created; used classes and objects as building blocks instead of such programs and algorithms
- These languages build upon object model.
- The object model is based upon three important concepts
  - object oriented analysis
  - object oriented Design
  - object oriented Programming

Object Oriented Analysis

- Object Oriented analysis is a method of analysis that examines the requirements from the perspective of the classes and objects found in the vocabulary of the problem domain.

  Grady Booch

- The is the first step for developing an object-oriented system
- Investigation of problem and requirements rather solution
- Focus at the behavior of the system independent of its domain
- Examines the real-world environment in which the system will operate
  - Includes people and things that will interact to produce some results
  - Analyzed in a basic abstract forms, in multiple iterations
  - These abstractions later become classes in the problem domain

Example - Library

- Member requests a book from the library
- Librarian will search for the book
- If found, the book is issued
  - The dataset is updates with the issued book
- If unavailable, the request is placed in a queue
- If not found, an order may be placed for the book

Analysis

- Highest level analysis or abstraction
- This will continue until the objects are uniquely identified
- The objects are refined as abstractions with the key characteristics
- Emphasis on investigation rather than the solution

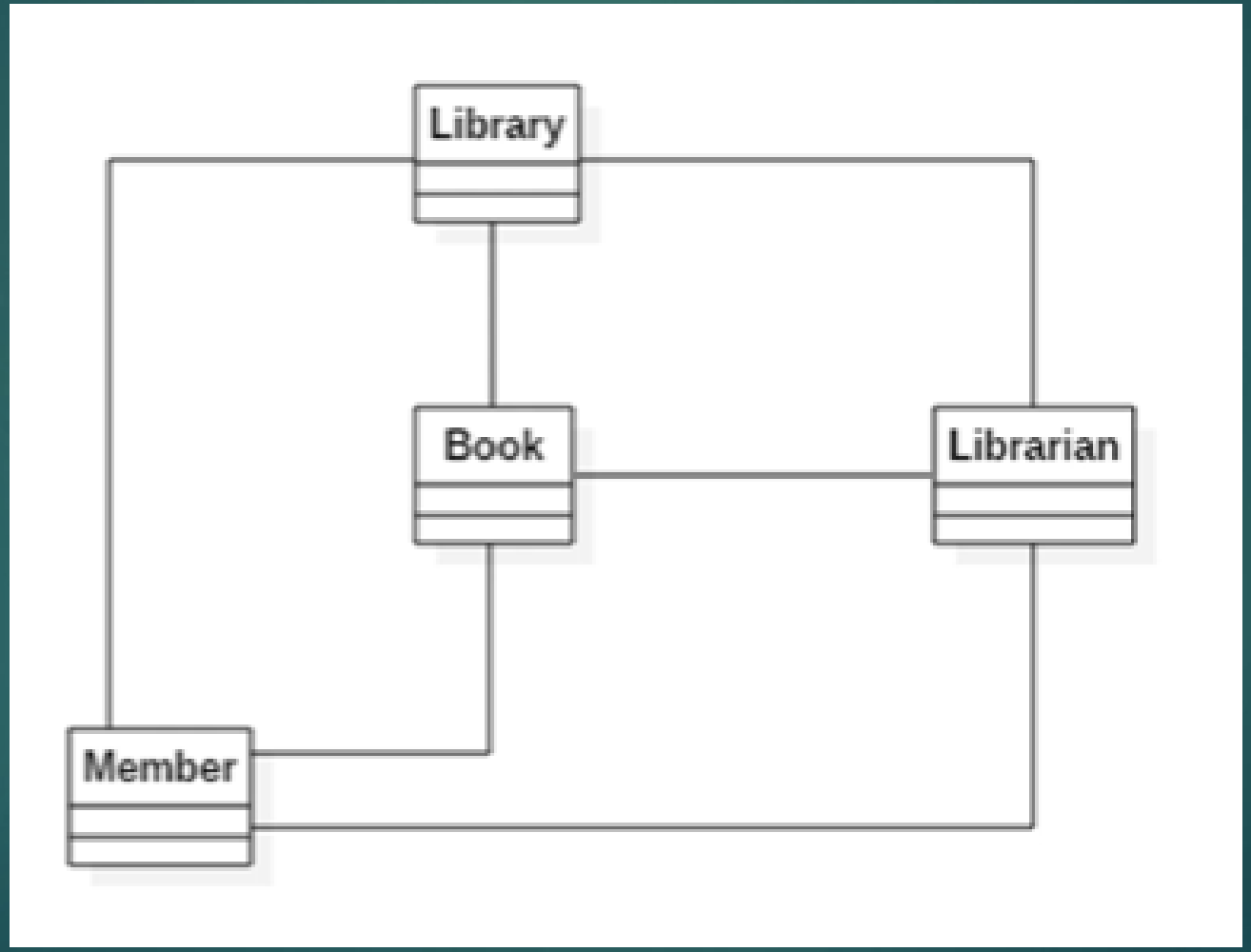
Object Oriented Design

- object oriented design is a method of design encompassing the process of object oriented decomposition and a notation for depicting both logical and physical as well as starting and dynamic models of a system under design object oriented design

  Grady Booch

- Focus on OO decomposition
- Used different notation to express different models of the system
- High level concepts of analysis are mapped onto classes
- Hierarchical association among the classes are designed
- Results in detail description that specifies how the system is to be built using different technologies

Example - Library



Object Oriented Programming

- It is a method of implementation in which programs are organized as cooperate to collection of objects, each of which represents an instance of some class, and whose classes are all members of a hierarchy of classes united via inheritance relationships
- Uses objects as a fundamental blocks
- Classes & Objects are related through relationships
- Requires programming language that supports
  - Object as abstraction
  - Classes
  - Generalization of classes
- languages like Java or C++ C sharp can be used to write objects or into programs.

Summary

- The object model builds upon three important concepts
- object oriented analysis, design and programming.
- Analysis part focuses on the problem
- Design is about OO decomposition and expression of the design using some notations
- Programming is the implementation using OO languaes

Object Oriented Analysis
Use Case

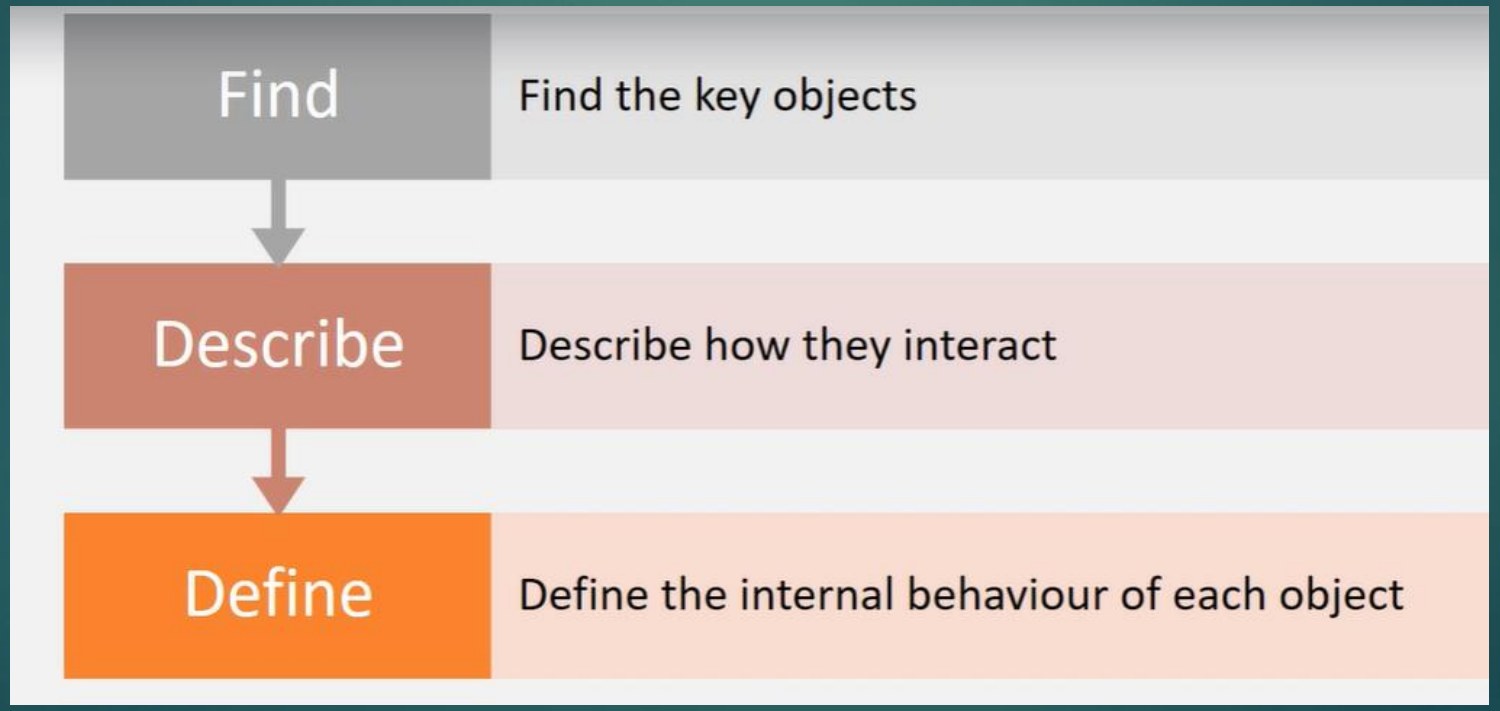
Use case

- OO analysis focuses on the problem and requirements
- Focus in what the system supposed to do
- Implementation details are mostly ignores
- An analysis model is created
- Done through user stories or use cases
- Involves key abstraction fro the problem domain
- Constraints of technologies are ignored

OO Analysis

- In OO analysis we performed OO decomposition

Steps



Example: Car game(Reckless Driver)

- Object: keep driving as long as you can & earn money by smashing objects. Spend money on powerups
- Smash into traffics, side objects (hydrant, phone booth, etc.)
- Each hit will damage the player car and reduce its health
- Repair through health pickups
- Amount if damage caused is accumulated by the player
- Smash anything, except trees & police cars (causes instant player death)
- Powerups are available
- heavy armour: protects players car from damage
- Bomb: destroys objects in the path of the player car

Key Objects

- Player car
- Traffic cars
- Fire hydrant
- Phone booth
- Newspaper box
- Bus stand bench
- Hotdog cart
- Tree
- etc

Example Gameplay

- The player car smashed into traffic car
- Show sparks at point of contact
- Play crash audio
- Apply damage to player car & reduce health
- Computer damage caused to traffic car
- Accumulate damage as cash

Use Case

- Defines a model of a system behavior
- Uses natural language
- Designers and programmers use a common reference
- Decomposes layer system specification into small manageable parts
- Smaller parts are easily described, hence easily implemented
- Models functional behaviors of the system

Use case Description

Description	Details
Goal	Use case's place within the system and why it is important
Preconditions	What is required before use case execution
Successful End Condition	System condition after successful execution
Failed End Condition	System condition after failed execution
Primary Actors	Actors that may trigger the use case
Secondary Actors	Actors that participate but are not main players in a use case's execution
Trigger	Event that causes the use case to execute
Main Flow	Important steps in a use case's normal execution
Extensions	Alternative steps in use case execution

Gameplay

Description	Details
Goal	Remain alive as long as you can
Preconditions	Player has enough health
Successful End Condition	Player earns cash
Failed End Condition	Player dies
Primary Actors	Player
Secondary Actors	System
Trigger	Player starts the game

Main Flow

Description	Details
Main Flow	1. Player starts the game
	2. Player smashes into the traffic car
	3. Play crash audio
	4. Show sparks
	5. Apply damage to player car & reduce health
	6. Compute damage caused to traffic car
	7. Accumulate damage as cash

Alternate Flow

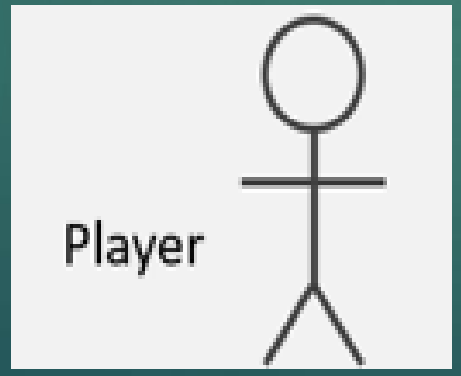
Description	Details
Alternate Flow	2a. Player smashes into police car
	5a. Player health reaches 0

Use Case Diagram

- Describes discrete unit of behavior that has a clearly defined scope
- Illustrates what should be done to achieve the goal of the use case
- This ultimately maps into program code
- Shown as an oval shape with the description of the behavior

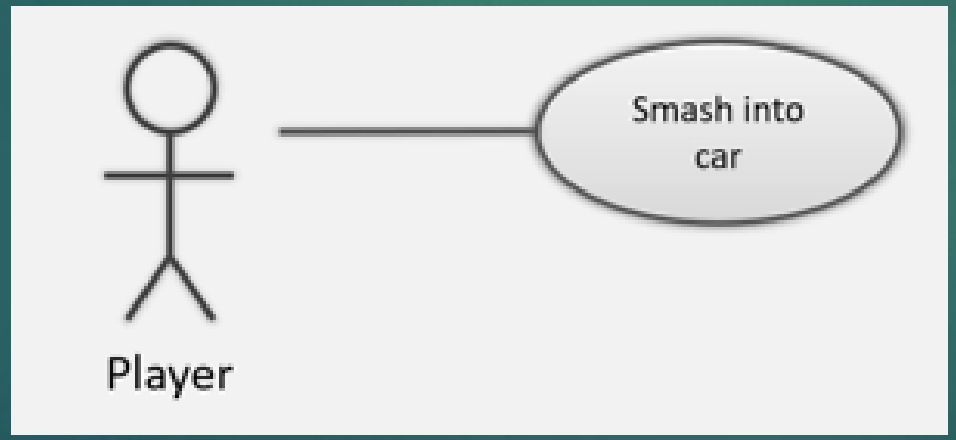
Actor

- External entity that interacts with the system
- Could be a person, system or some external entity
- Maybe defined outside the system
- Acts as a black box and cannot be changed
- Appears as a stick figure



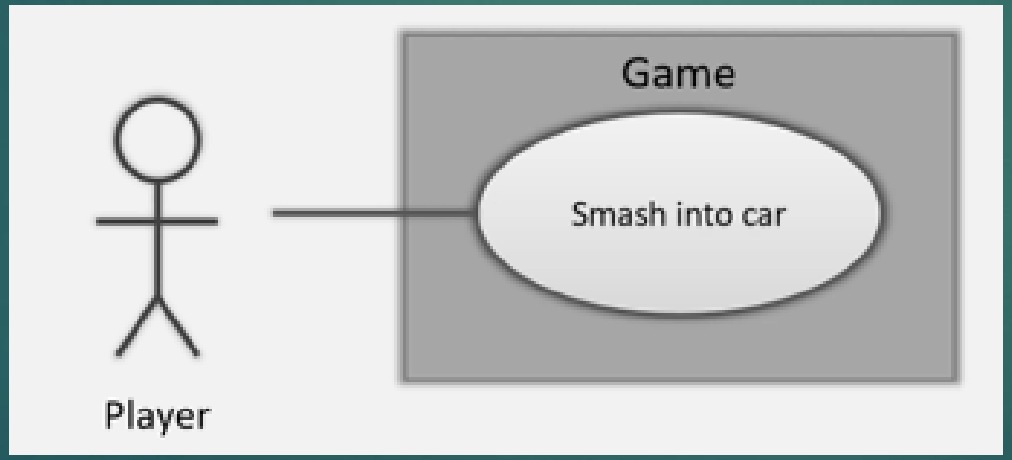
Communication line

- Connects an actor and a use case
- Indicated participation of an actor
- Appears as a line between an actor & the use case

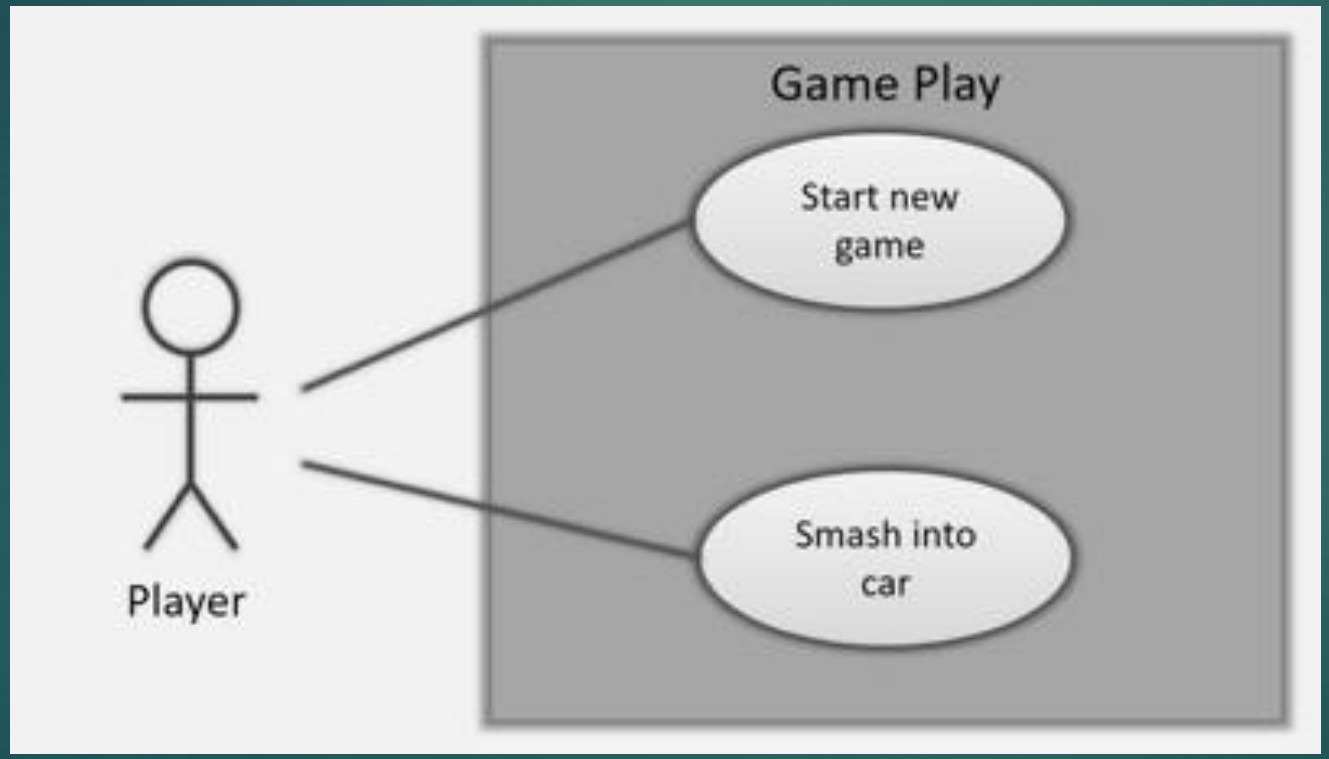


System Boundaries

- Indicates separation between actors and use cases (internal to the system)
- Marked by a bounded box around the use cases



Example



Modelling Tools

- Enterprise Architect
- Microsoft Visio
- Star UML
- Papyrus
- Visual Paradigm