INF 202
Intro to Data and Databases

Spring 2023
Dr. M. Abdullah Canbaz

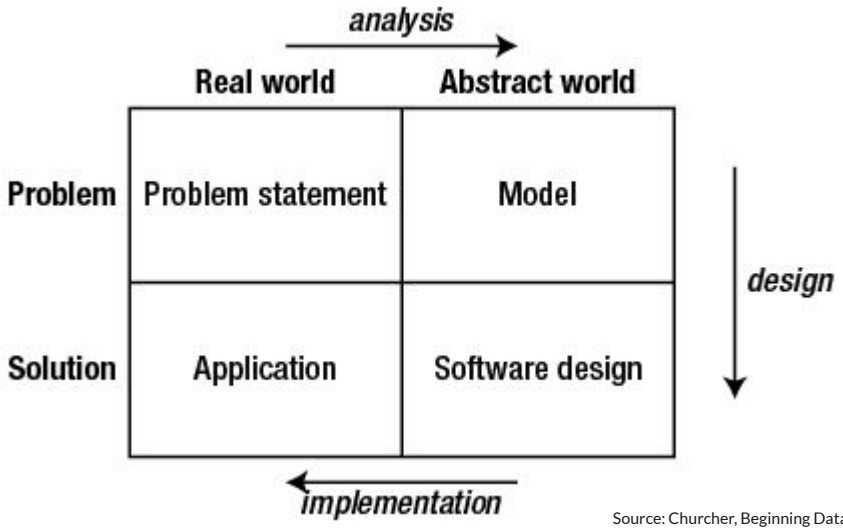
Class 2
Jan. 24, 2022

Recap - Data and Databases

- What is data?
  - Data is distinct pieces of information, specifically ones transmitted or stored by a computer, usually formatted in a specific way.
- What is a database?
  - A database is a collection of data organized in such a way that a computer program can quickly search for and retrieve desired pieces of data.

System Lifecycle

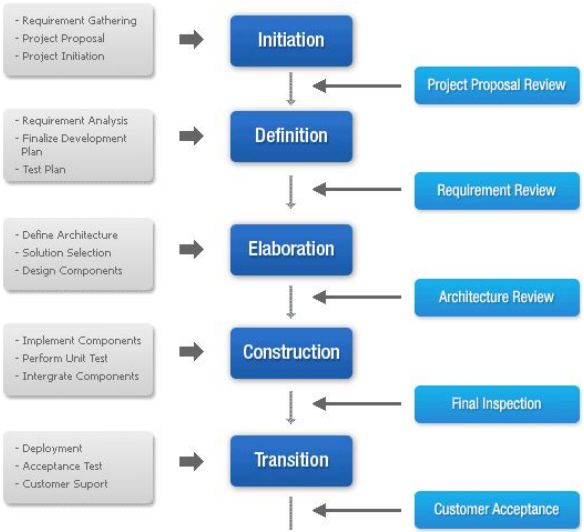
The system lifecycle is the various phases of development through which a computer system passes.

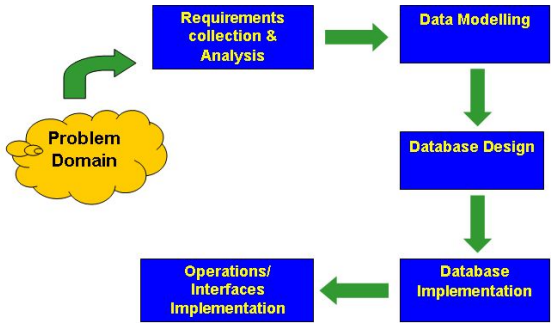


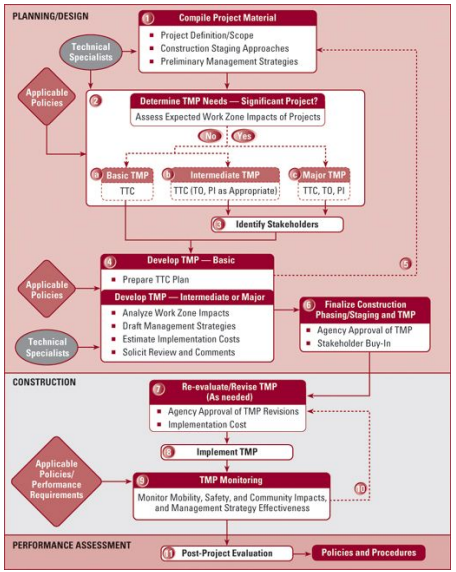
Source: Churcher, Beginning Database Design

System Lifecycle

Different organizations will have different specifics, but they should all share common elements.







Database System Lifecycle

- Initial Problem and Mission Statement
- Analysis of Use Cases
- Simple Data Model
- Iterate Until Requirements are Set
- Design
- Implementation
- Maintenance
- Repeat

Database System Lifecycle

- This course will not cover the Maintenance and Repeat steps, but in the real world, they will take up the majority of the effort over the lifespan of the project.
- Good design at the beginning of the project will make those steps much easier going forward.

Database System Lifecycle: Design Objectives

- Identify the tasks needed to satisfy the project objectives
- Identify the data needed to support the task long-term
  - What needs to be supported and maintained?
  - Do not focus too much on the task itself, rather focus on the underlying data items. The data should be independent of the task.
- Abstract from the real world
  - The analysis and design inherent in the database lifecycle are processes that abstract from the real world to get something that can be implemented in a “rules-driven” database.
- Avoid common data mistakes - this will make the maintenance effort easier:
  - Mishandling keywords and categories
  - Repeated information
  - Designing to a single report
- Accept that you will get things wrong and need to improve them - that’s what the iterative process is for

Database System Lifecycle: Design Principles

- Design is subjective, and different designers may produce different, equally valid interpretations.
- With the assistance and active involvement of the users, successive iterations of the design process should lead to adequate choices for the system required.

Database System Lifecycle

- Initial Problem and Mission Statement
- Analysis of Use Cases
- Simple Data Model
- Iterate Until Requirements are Set
- Design
- Implementation
- Maintenance
- Repeat

Initial Problem and Mission Statement

- The Initial Problem and Mission Statement stage includes setting the mission and goals of the database project.
- You can't design a good system unless you know the purpose of that system.
- When devising the system mission/goal statement -- in fact, all throughout the design and implementation process -- checking with the end user is paramount.
- You can design and implement an impressive system, but if it doesn't do what the end user wants, it's a waste of time and money.

Initial Problem and Mission Statement

- Good questions to ask when creating a mission/goals statement:
  - What is the purpose of your organization/enterprise?
  - Why do you feel that you need a database?
    - Do you really need it?
    - Can it be solved without requiring a database?
  - How do you know that a database will solve your problem?

Database System Lifecycle

- Initial Problem and Mission Statement
- Analysis of Use Cases
- Simple Data Model
- Iterate Until Requirements are Set
- Design
- Implementation
- Maintenance
- Repeat

Use Cases

- Once we define our initial problem description (statement), one way to represent them is with Use Cases.
- Use Cases are part of the Unified Modeling Language (UML)
  - A popular diagramming notation & techniques used by software architects to depict various aspects of the software building process
- Use Cases are descriptions of how different types of users interact with your proposed system.
- Use Cases are set of actions (tasks) and functions that a system needs to perform based on a user or system (think robots) interaction

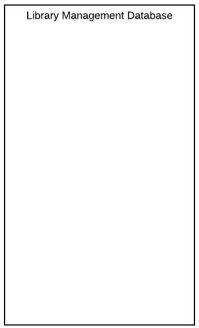
Use Case Diagrams

- A Use Case Diagram is a diagram that models the functionality of the proposed system using Actors (users) and identified use cases
- Use Case Diagrams are high-level diagrams
  - Looking at the diagram, one should be able to see what the proposed system can do

- Consists of the following elements
  - Proposed System
  - Actors
  - Use Cases
  - Relationships

Use Case Diagrams: System

- Denotes the proposed system you are building
  - Could be a website, a business process, database, a mobile app or anything
- Represented by a rectangle in a Use Case diagram
  - Put the name of the [proposed] system at the top
  - Helps identifies the scope of the system
  - All use cases identified should be within the rectangle
  - Anything outside the rectangle not part of the [proposed] system



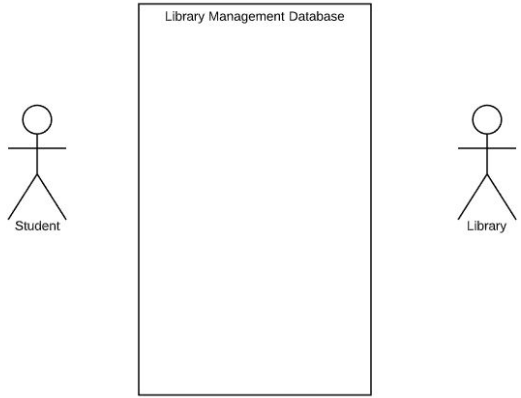
Use Cases: Actors

- Use cases define who uses the system (also known as “Actors”) and what they do with it (also known as “Activities” or “Tasks”)

Use Case Diagrams: Actors

- Depicted by a stick figure in UML Use Case diagram
- Denotes someone who is going to use your [proposed] system
  - Could be a person
  - Could be an organization
  - Could be an external system
  - Could be a device like an washing machine or toaster or refrigerator
- Actors are external objects to the [proposed] system
  - Need to placed outside the system i.e. rectangle container
  - Think in terms of abstract categories or roles
    - Do not name actors as “Joe” or “Peter”
    - Name them as “Student”, “Customer”, “Admin”

Use Cases: Types of Actors



- Primary Actors
  - Initiates the use of the [proposed] system
  - Should be to the left of the rectangle [proposed] system
- Secondary Actors
  - Reacts to the actions performed by the system
  - Should be to the right of the rectangle [proposed] system
- In a Library Management System:
  - Think of the “Student” actor as the one who initiates and uses the system like “Checking out a book”, “Viewing list of books”
  - Think of the “Library” actor as the one who would only react to provide a list of books back to the student who initiated the request.

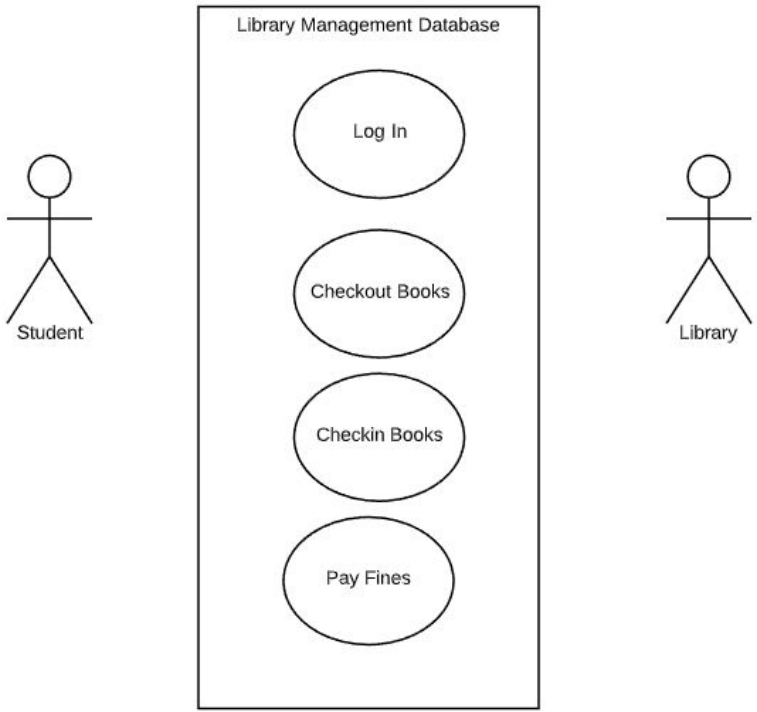
Use Cases: Stakeholders

- Next, we need to decide who are the most important users of the system.
- The following questions are key:
  - Who is the system for?
  - Who is involved with the system?
  - Who is important to the system?

Use Cases

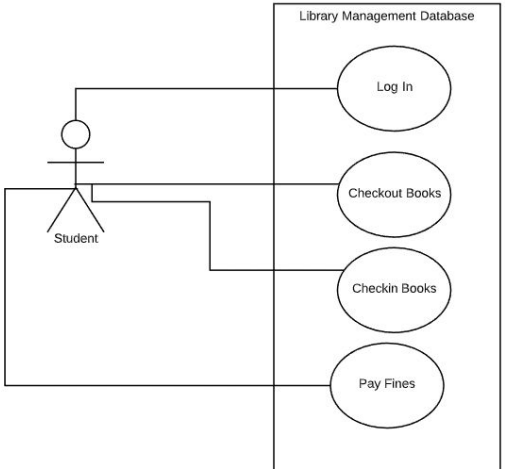
- You start to describe what your [proposed] system really does
- Denoted by an oval shape
- Represents an action that accomplished a specific task within the system
- Placed within the rectangle container to depicts tasks happening within the system
- In the proposed Library Management Database, a student would:
    - Log In
    - Checkout Books
    - Checkin Books
    - Pay Fines

Use Cases


Library Management Database
Log In
Checkout Books
Checkin Books
Pay Fines
Student
Library

Use Cases: Relationships

- Each Actor should interact with at least one of the identified use cases within our system
- Draw a connecting solid line between the Actor and identified use case
    - For e.g. Student [Actor] → Log In [use case]
- This type of relationship is known as “Association” and signifies a basic communication or interaction between the actor and the use case (action the user performs)



Use Cases: Relationships

- There are additional types of relationships in a Use Case diagram depicting the relationship between the Actor and Use cases
- Some of the other relationships include
  - Include
  - Extend
  - Generalization

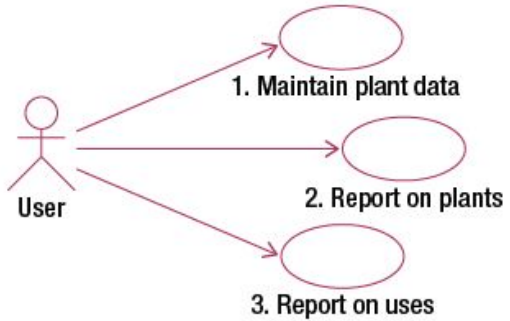
Use Cases: Input and Output Cases

- When designing your use cases, it can be useful to think of them in terms of input and output use cases.
- Input use cases involve users (or automated processes) entering information into the system. If the use case involves user input, there will need to be a user interface (UI) somewhere in the system.
- Output use cases involve reporting on the information in some way. This can involve informal reporting such as displaying information on the screen in a form or web page, or more formal reporting such as an Excel sheet or formatted printable report.

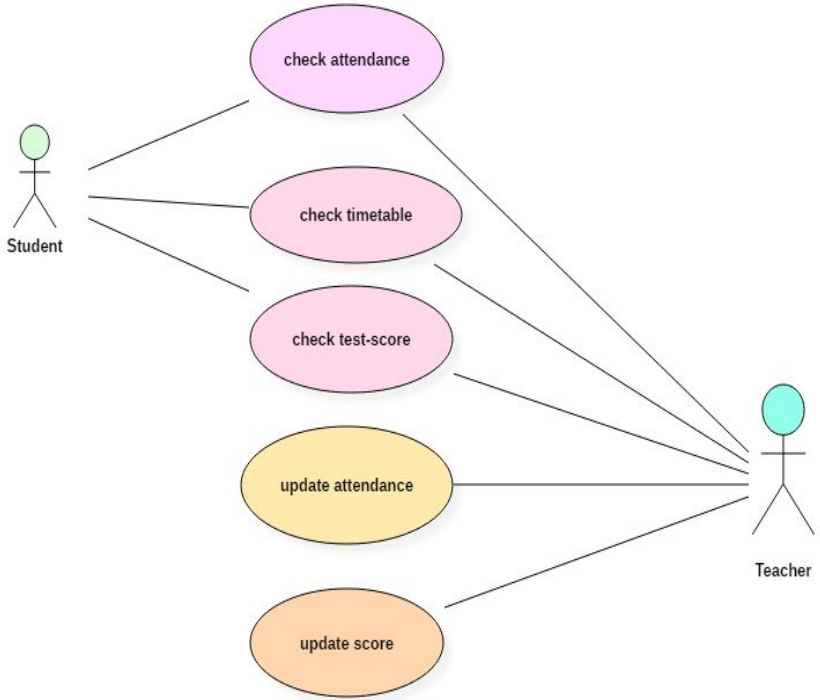
Use Cases: Database Systems

- For a database system or project, the most common use case tasks would be:
  - Entering or Adding data to a database
  - Updating or Maintaining data in a database
  - Deleting data from a database
  - Searching data from a database (also known as reading data from a database)
  - These 4 tasks are known as “CRUD” → Create, Read, Update, Delete
- We will learn database CRUD operations in detail later in the course. For the use case part, remember the users (Actors) of your proposed database system are interacting performing one of the above tasks.

Use Cases: Example 1

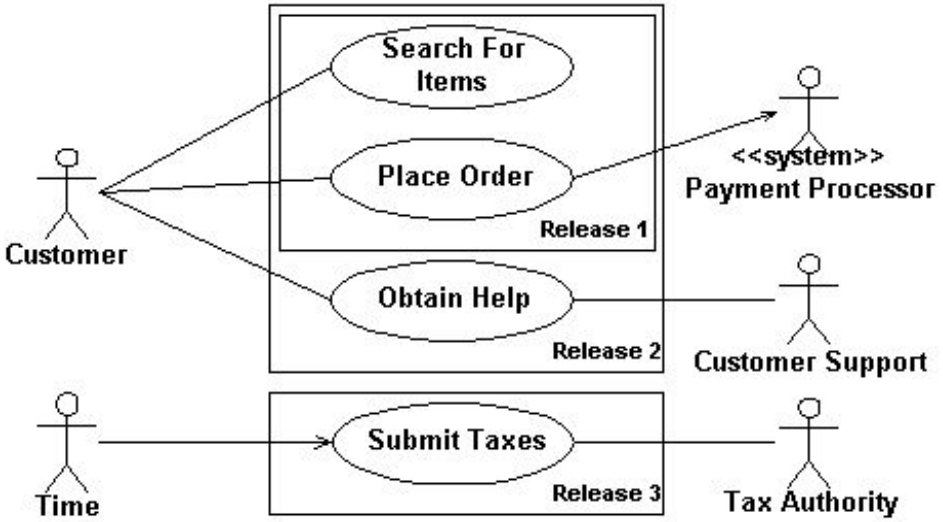


Use Cases: E


check attendance
check timetable
check test-score
update attendance
update score
Student
Teacher

Use Cases: Example 3

This example demonstrates how a use case diagram can show iterative releases of the system.



Consider the following problem statement:

A county health department begins receiving grants from various other departments and organizations. Each grant is to be used for certain activities by the department's employees, and ONLY those activities. Every month, the grant provider wants to know how much of the grant has been spent so far. The health department wants to track when the total grant amount has been used, so it can reassign the employees working on those activities to other tasks.

Consider the following problem statement:

Emerald Corporation decides to begin offering its employees automatic deductions of their home and auto insurance premiums from their weekly paychecks. Emerald chooses several insurance providers, and the employees apply for insurance through one of those providers. The policy details are sent to Emerald, who then calculates the weekly premium amount and deducts it from their paycheck. The employees need to see how much was deducted for each policy on their pay stubs.

Lucidchart Exercises

- Diagram the following in a use case diagram:
  - System: Travel Booking App
  - Actors: Vacationer, Airline
  - Use Cases: Browse Catalog, Order Package, Track Order, Update Catalog