System Analysis & Design

Use case

Use case

- visual representation of the interactions between users (actors) and a system to achieve goals
- part of UML
- used in requirements gathering and system design phases of software development

Components

Actors
- user / entity that interacts with the system

Use cases
- specific function goal that an actor wants to achieve
- sequences of actions

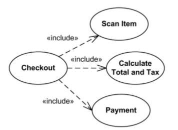
System Boundary
rectangle that defines the scope of the system all use cases that the system performs

Relationships
how actors interact with the use case

<INCLUDE>
<EXTEND>
<GENERALIZATION>

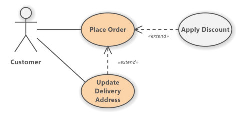
INCLUDE RELATIONSHIP

- signify that a use case contains the behavior of another use case
- common behavior



EXTENDED RELATIONSHIP

- use case extends behavior of another use case
- optional / conditional behavior

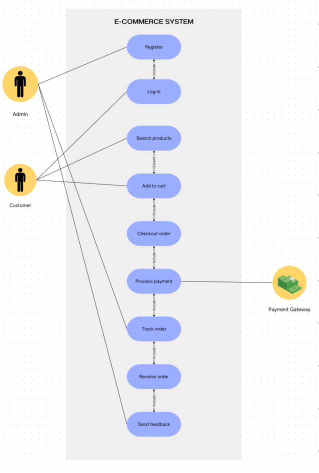


Purpose and Benefits

- Requirements Gathering
- Design Foundation
- Communication
- Project Scope

Steps

1. Identify Actors : who will interact with the system
2. Identify use cases : goals of the actors and needs to achieve goals
3. Define Relationship : associations between actors and use cases
4. Draw the system Boundary : encapsulate within a rectangle
5. Review and Refine : w/ stakeholders and feedback

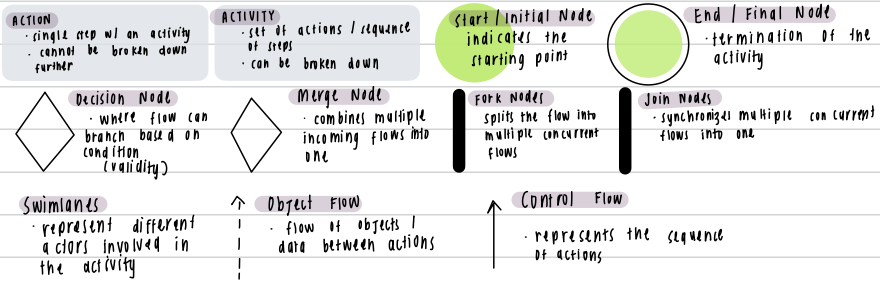


Activity Diagram

Activity Diagram

- type of UML diagram that represents the dynamic aspects of a system
- illustrates the flow of control / object flow emphasizing the sequence and conditions
- shows how activities are coordinated to accomplish a goal

Components



ACTION
- single step w/ an activity
- cannot be broken down further

ACTIVITY
- set of actions / sequence of steps
- can be broken down

Start / Initial Node
indicates the starting point

End / Final Node
- termination of the activity

Decision Node
- where flow can branch based on condition (validity)

Merge Node
- combines multiple incoming flows into one

Fork Nodes
splits the flow into multiple concurrent flows

Join Nodes
- synchronizes multiple concurrent flows into one

Swimlanes
- represent different actors involved in the activity

Object Flow
- flow of objects / data between actions

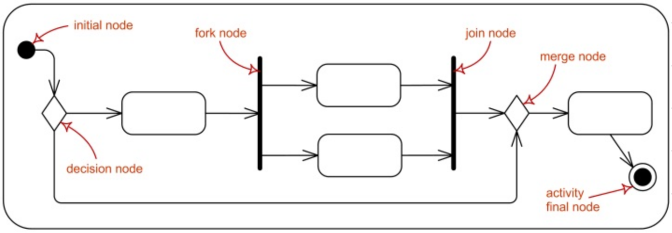
Control Flow
- represents the sequence of actions

Purpose & Benefit

1. Process visualization
2. Requirement clarification
3. System Behavior Modeling
4. Workflow anasysis

Steps

1. Identify Activity : determine the actions and activities in the process
2. Determine Sequences: establish the order of activities and decision
3. Define start and end points
4. Add decision points : represents conditional branches
5. Include Forks and joints : represent parallel processes
6. Use swim lanes
7. Draw Control and object flow : show sequence and data movement



System Design Strategies

System Design

- essential developing effective information systems
- creating systems that are both functional and user-friendly
- inc. logical and physical design, interface design

1. Logical and Physical Design

Logical Design abstract representation of data flows, inputs and outputs of system

independent of physical considerations

focuses on what the system will do

a. Data Flow Diagram (DFDs)

- represents the flow of information within the system (how data moves)

b. Entity-Relationship Diagrams (ERDs)

- depicts the relationships between data entities in the system

c. Modeling Processes

- logical design involves detailed system processeses (flowcharts / pseudocodes)

Physical Design logical design $\xrightarrow{\text{translate}}$ physical components <
hardware
software

a. Hardware specifications

- determine necessary hardware components

b. Software specifications

- software platform, OS, and applications

c. Database Design

- map logical data model to specific database management systems (DBMS)

d. Interface Design

- UI layout and specifications <
user requirements
system functionality

2. Interface Design

- defines how users interact with the system
- creates intuitive and user-friendly interfaces → effective user interfaces

a. UI Design

b. UX design

c. Prototyping

- mock-ups for test design
- helps gather user feedback before full-scale development

d. Accessibility

3. Database Design

- creating a detailed data model (structure of the data)

a. Conceptual Design

- high-level data model ; ER diagram usually

b. Logical Design

- specifies data types, relationships and constraints

c. Physical Design

- logical model ⟶ DBMS
- defining tables, indexes, views and other database objects

d. Normalization

- Organization of database < reduce redundancy / improve data integrity

e. Optimization

- enhancing the performance of the database < indexing / query Optimization

Data Flow Diagram

- graphical representation of the flow of data through an information system
- illustrates how data is processed by a system (inputs and outputs)
- visualization of data processing steps

Components

External Entities	Processes	Data Stores	Data Flows
sources of data outside the system -¦- rectangles	transform input data into output data -¦- circles/ovals	repositories -¦- open-ended rectangles / parallel lines	show movements -¦- arrows

Types

1. Context Diagram · provides an overview of the system and its interaction with external entities
2. Level 0 DFD · "fundamental system model" · breaks down the main process into subprocesses
3. Level 1 DFD · detailed breakdown of a process within level 0 DFD

Benefits of DFDs

Clarity, Communication, Analysis, Documentation

Entity-Relationship Diagram

- illustrates relationships between entities
- define the data structure

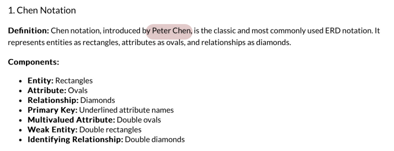
Components

Entities	Attributes	Relationships	Primary Key	Foreign Key	Cardinality
real-world object -¦- rectangle	properties / characteristics -¦- ovals	· interaction of entities -¦- diamonds	uniquely identifies an instance -¦- underlined	links the PK to other entities -¦- Dashedline	no. of instances -¦- 1, N, 1…N

Relationships

One-to-One	One-to-Many	Many-to-Many
1:1	1	M (*)

ERD Notations

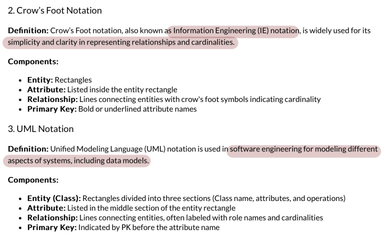


1. Chen Notation

Definition: Chen notation, introduced by Peter Chen, is the classic and most commonly used ERD notation. It represents entities as rectangles, attributes as ovals, and relationships as diamonds.

Components:

- **Entity:** Rectangles
- **Attribute:** Ovals
- **Relationship:** Diamonds
- **Primary Key:** Underlined attribute names
- **Multivalued Attribute:** Double ovals
- **Weak Entity:** Double rectangles
- **Identifying Relationship:** Double diamonds



2. Crow's Foot Notation

Definition: Crow's Foot notation, also known as Information Engineering (IE) notation, is widely used for its simplicity and clarity in representing relationships and cardinalities.

Components:

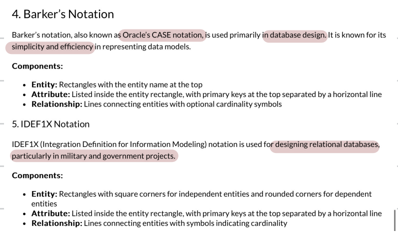
- **Entity:** Rectangles
- **Attribute:** Listed inside the entity rectangle
- **Relationship:** Lines connecting entities with crow's foot symbols indicating cardinality
- **Primary Key:** Bold or underlined attribute names

3. UML Notation

Definition: Unified Modeling Language (UML) notation is used in software engineering for modeling different aspects of systems, including data models.

Components:

- **Entity (Class):** Rectangles divided into three sections (Class name, attributes, and operations)
- **Attribute:** Listed in the middle section of the entity rectangle
- **Relationship:** Lines connecting entities, often labeled with role names and cardinalities
- **Primary Key:** Indicated by PK before the attribute name



4. Barker's Notation

Barker's notation, also known as Oracle's CASE notation, is used primarily in database design. It is known for its simplicity and efficiency in representing data models.

Components:

- **Entity:** Rectangles with the entity name at the top
- **Attribute:** Listed inside the entity rectangle, with primary keys at the top separated by a horizontal line
- **Relationship:** Lines connecting entities with optional cardinality symbols

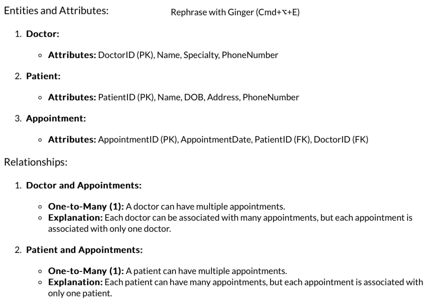
5. IDEF1X Notation

IDEF1X (Integration Definition for Information Modeling) notation is used for designing relational databases, particularly in military and government projects.

Components:

- **Entity:** Rectangles with square corners for independent entities and rounded corners for dependent entities
- **Attribute:** Listed inside the entity rectangle, with primary keys at the top separated by a horizontal line
- **Relationship:** Lines connecting entities with symbols indicating cardinality

Challenge 3: ERD Diagram



Entities and Attributes:

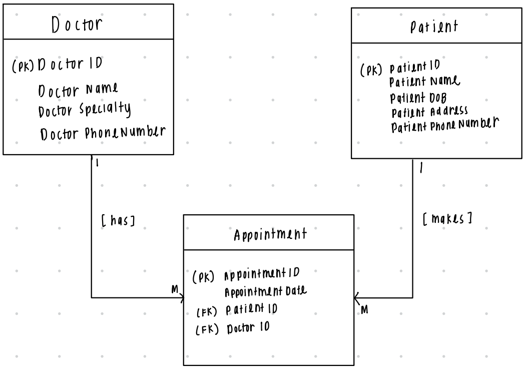
Rephrase with Ginger (Cmd+⌥+E)

1. **Doctor:**
   - **Attributes:** DoctorID (PK), Name, Specialty, PhoneNumber
2. **Patient:**
   - **Attributes:** PatientID (PK), Name, DOB, Address, PhoneNumber
3. **Appointment:**
   - **Attributes:** AppointmentID (PK), AppointmentDate, PatientID (FK), DoctorID (FK)

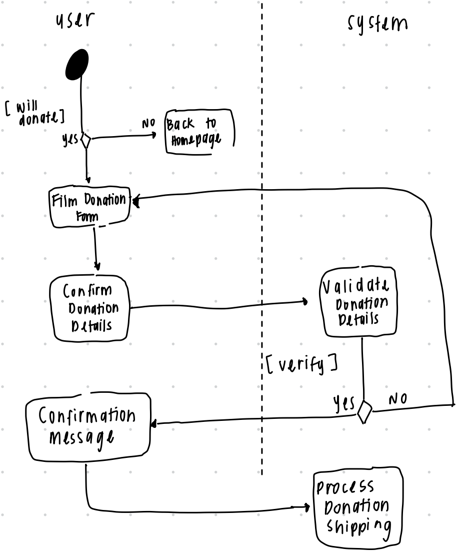
Relationships:

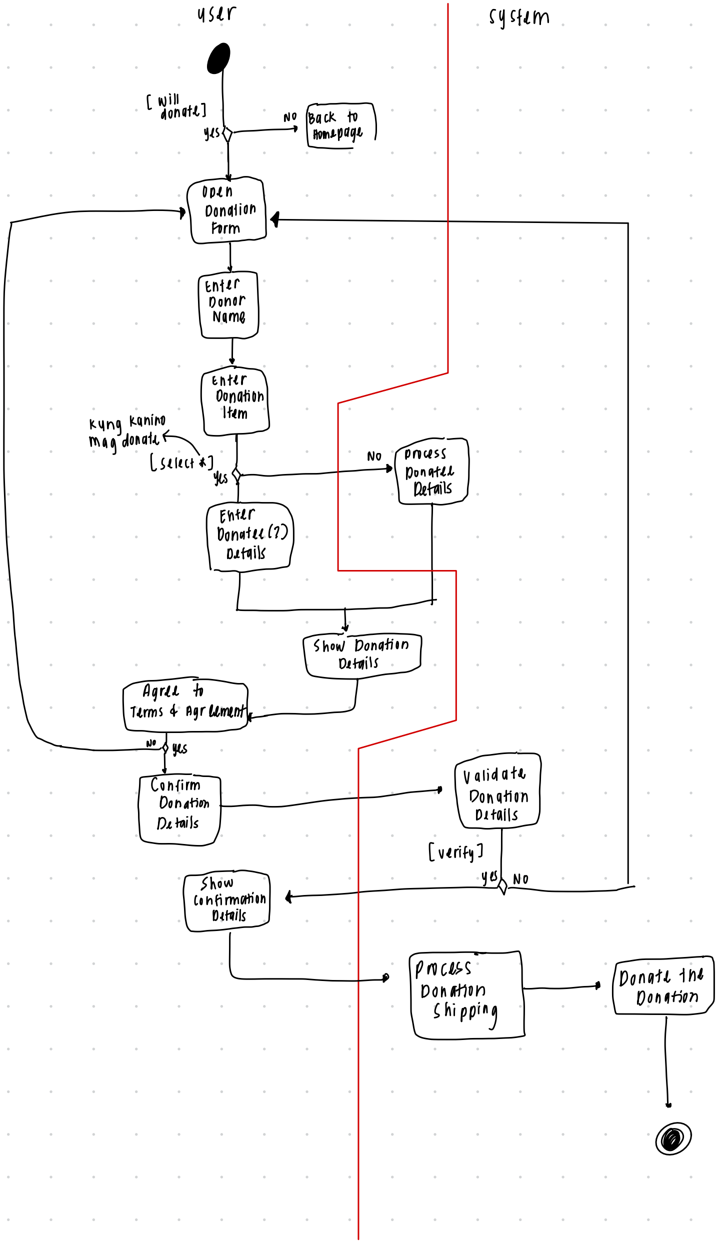
1. **Doctor and Appointments:**
   - **One-to-Many (1):** A doctor can have multiple appointments.
   - **Explanation:** Each doctor can be associated with many appointments, but each appointment is associated with only one doctor.
2. **Patient and Appointments:**
   - **One-to-Many (1):** A patient can have multiple appointments.
   - **Explanation:** Each patient can have many appointments, but each appointment is associated with only one patient.

HOSPITAL MANAGEMENT SYSTEM



DONATION ACTIVITY DIAGRAM




user
system
[will donate]
yes
No
Back to Homepage
Open Donation Form
Enter Donor Name
Enter Donation Item
kung kanino mag donate
[select *]
yes
No
Process Donatee Details
Enter Donatee (?) Details
Show Donation Details
Agree to Terms & Agreement
No
yes
Confirm Donation Details
Validate Donation Details
[verify]
yes
No
Show confirmation Details
Process Donation Shipping
Donate the Donation