

Engenharia de Software (14341, 16230, 15386)

Software Design Principles, UML, Patterns and Quality Attributes #1

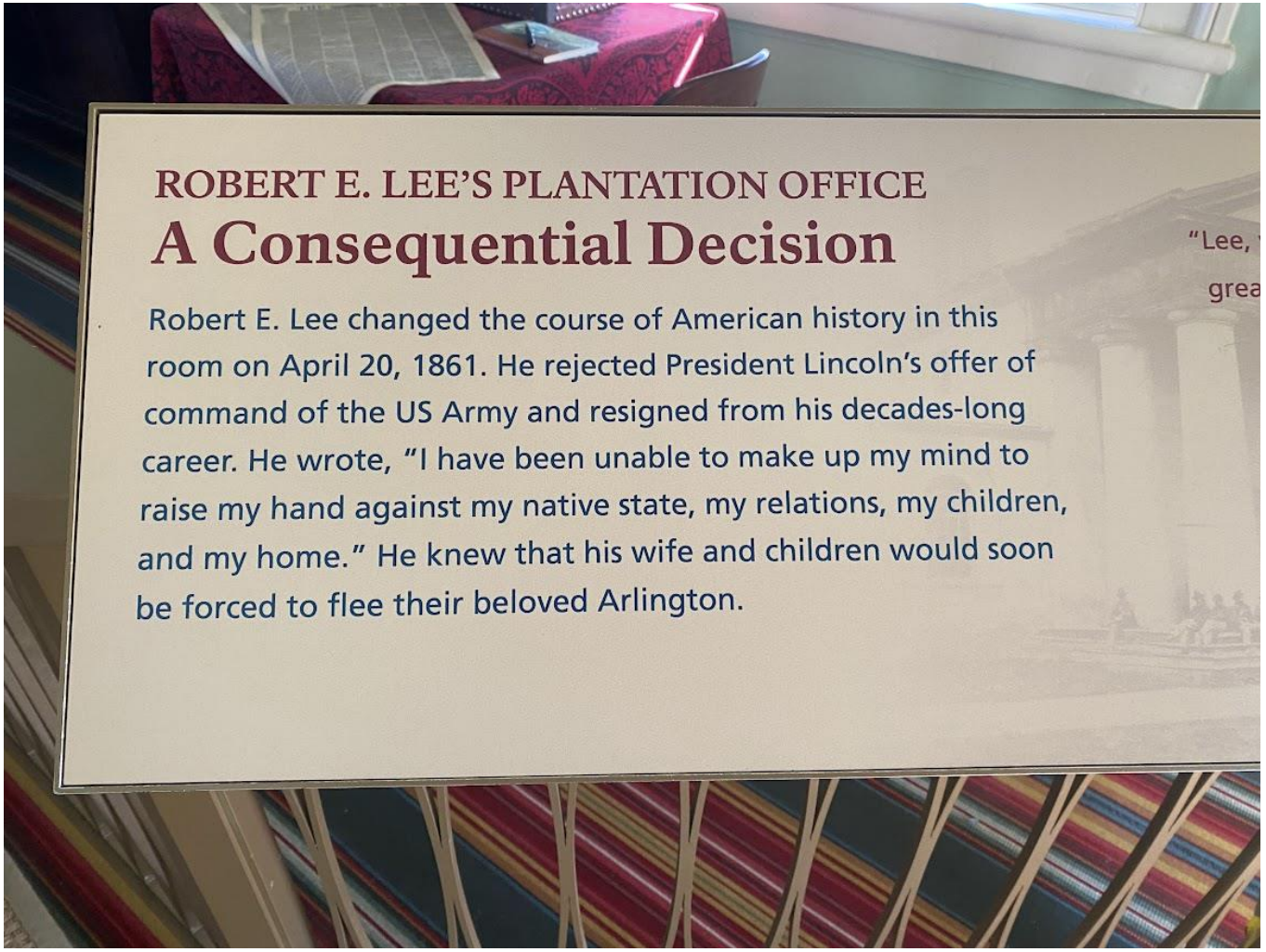
(adapted from *Software Engineering: International Version (10th Edition)*, Ian Sommerville, Pearson, 2015)

Topics covered




PROFESSIONAL SKILL
SPOTLIGHT
DEVELOP • APPLY • GROW

Ethics


ROBERT E. LEE'S PLANTATION OFFICE
A Consequential Decision
Robert E. Lee changed the course of American history in this room on April 20, 1861. He rejected President Lincoln's offer of command of the US Army and resigned from his decades-long career. He wrote, "I have been unable to make up my mind to raise my hand against my native state, my relations, my children, and my home." He knew that his wife and children would soon be forced to flee their beloved Arlington.

System modeling

- System modeling is the process of developing abstract representations of a system.
- System modeling has now come to mean representing a system using some kind of graphical notation, which is now almost always based on notations in the **Unified Modeling Language (UML)**.
- System modeling helps the analyst to understand the functionality of the system and models are used to communicate with customers.

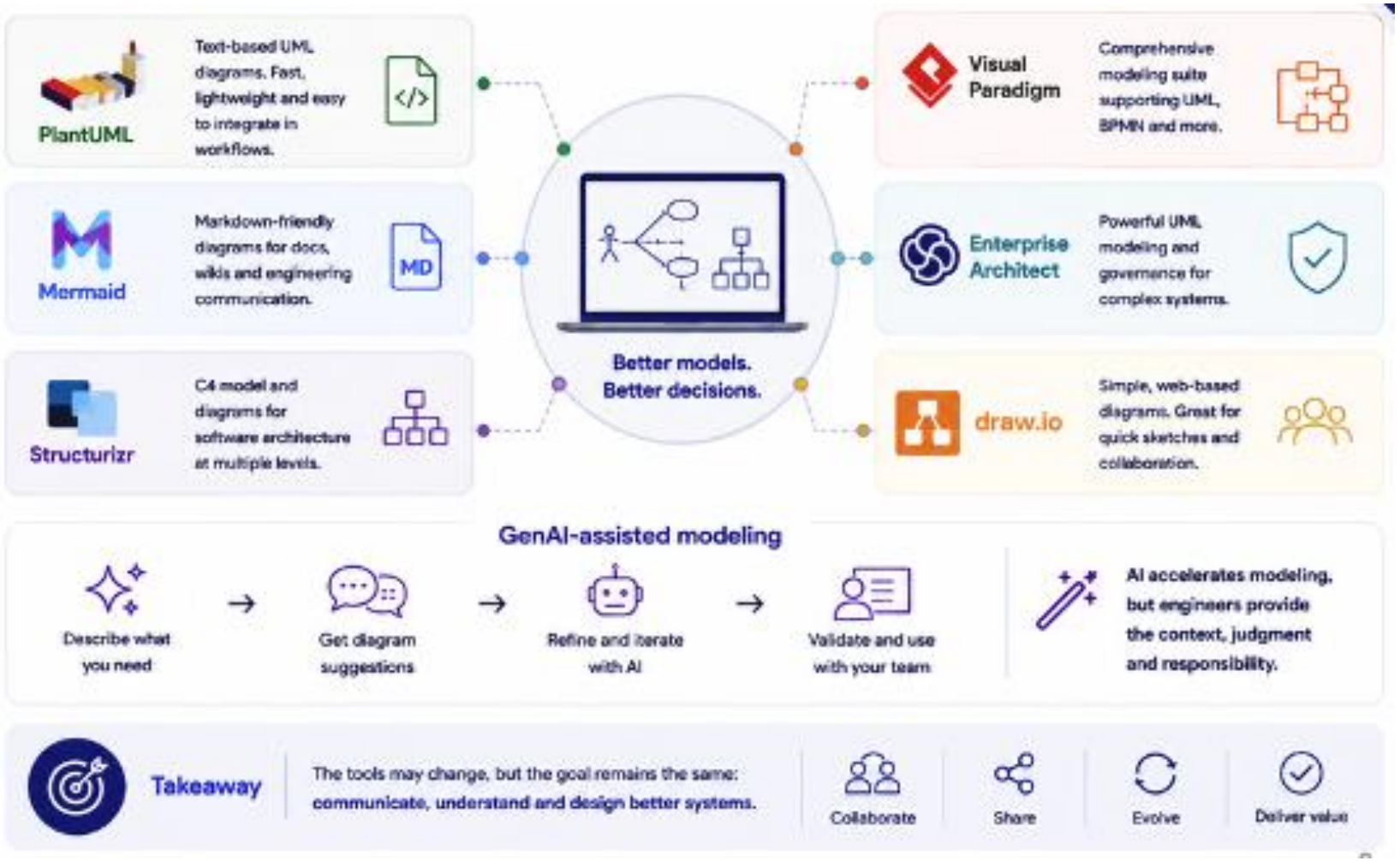
Existing and planned system models

- Models of the existing system are used during requirements engineering. They help clarify what the existing system does and can be used as a basis for discussing its strengths and weaknesses. These then lead to requirements for the new system.
- Models of the new system are used during requirements engineering to help explain the proposed requirements to other system stakeholders. Engineers use these models to discuss design proposals and to document the system for implementation.
- In a model-driven engineering process, it is possible to generate a complete or partial system implementation from the system model.

Benefits of using graphical notation in system modeling

- ✧ Clear Visualization: Provides an intuitive, standardized way to represent both the structure and behaviour of a system.
- ✧ Reduced Ambiguity: Minimizes misunderstandings by replacing vague text descriptions with precise diagrams.
- ✧ Early Validation & Verification: Allows stakeholders to review, confirm, and refine requirements before costly development stages.
- ✧ Improved Communication: Creates a common language for developers, analysts, testers, and business stakeholders.

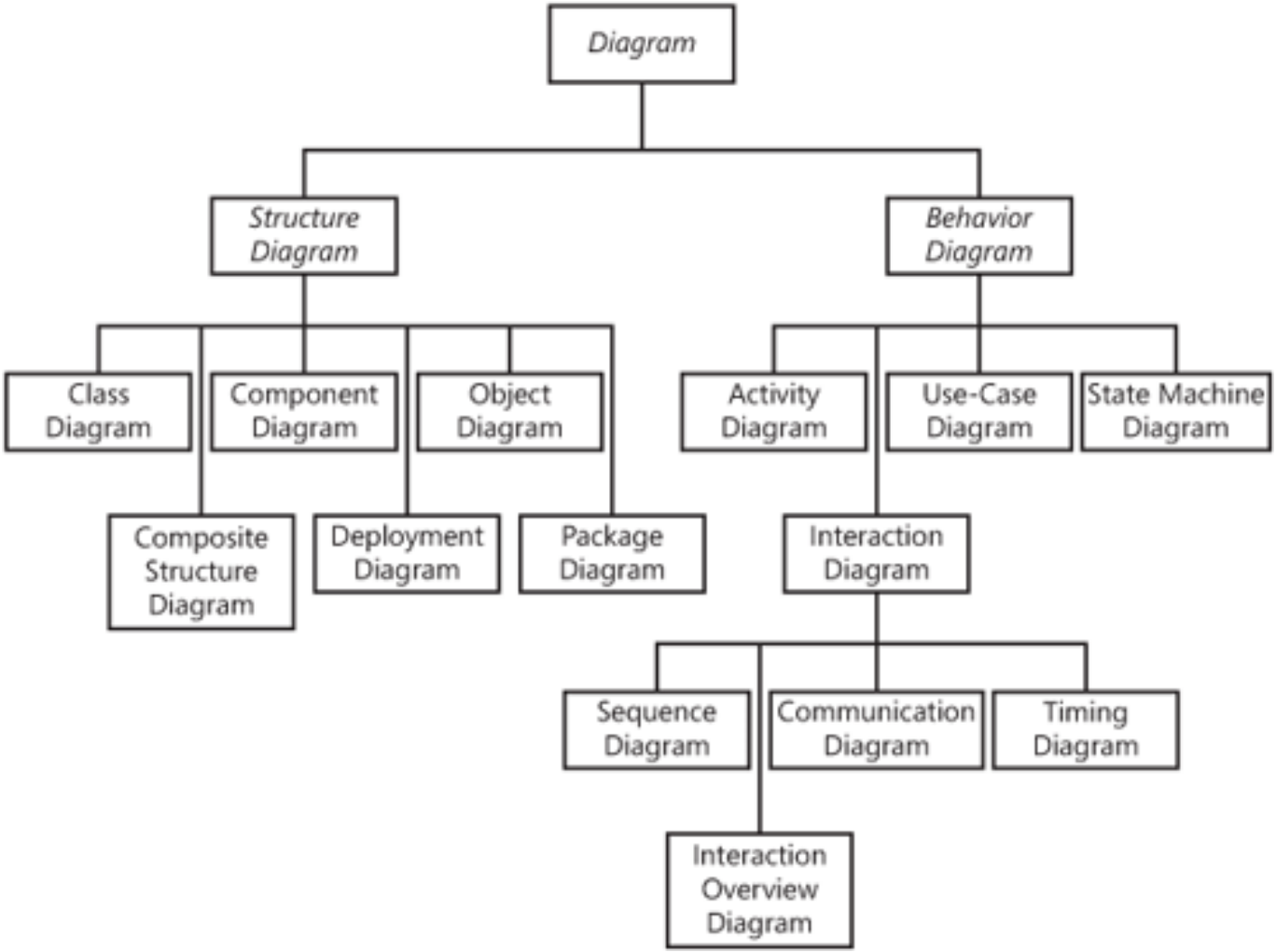
Modern system modeling ecosystem



System perspectives

- An external perspective, where you model the context or environment of the system.
- An interaction perspective, where you model the interactions between a system and its environment, or between the components of a system.
- A structural perspective, where you model the organization of a system or the structure of the data that is processed by the system.
- A behavioral perspective, where you model the dynamic behavior of the system and how it responds to events.

Hierarchy of UML diagrams



UML diagram types

- Activity diagrams, which show the activities involved in a process or in data processing (i.e. show workflows in processes).
- Use case diagrams, which show the interactions between actors and the system.
- Sequence diagrams, which show interactions between actors and the system and between system components over time.
- Class diagrams, which show the object classes in the system and the associations between these classes.
- State diagrams, which show how the system reacts to internal and external events.

Use of graphical models

- As a means of facilitating discussion about an existing or proposed system
  - Incomplete and incorrect models are OK as their role is to support discussion.
- As a way of documenting an existing system
  - Models should be an accurate representation of the system but need not be complete.
- As a detailed system description that can be used to generate a system implementation
  - Models have to be both correct and complete.

Interaction models

Interaction models

- Modeling user interaction is important as it helps to identify user requirements.
- Modeling system-to-system interaction highlights the communication problems that may arise.
- Modeling component interaction helps us understand if a proposed system structure is likely to deliver the required system performance and dependability.
- Use case diagrams and sequence diagrams may be used for interaction modeling.

Use case modeling

- Use cases were developed originally to support requirements elicitation and now incorporated into the UML.
- Each use case represents a discrete task that involves external interaction with a system.
- Actors in a use case may be people or other systems.
- Represented diagrammatically to provide an overview of the use case and in a more detailed textual form.

Use case diagrams

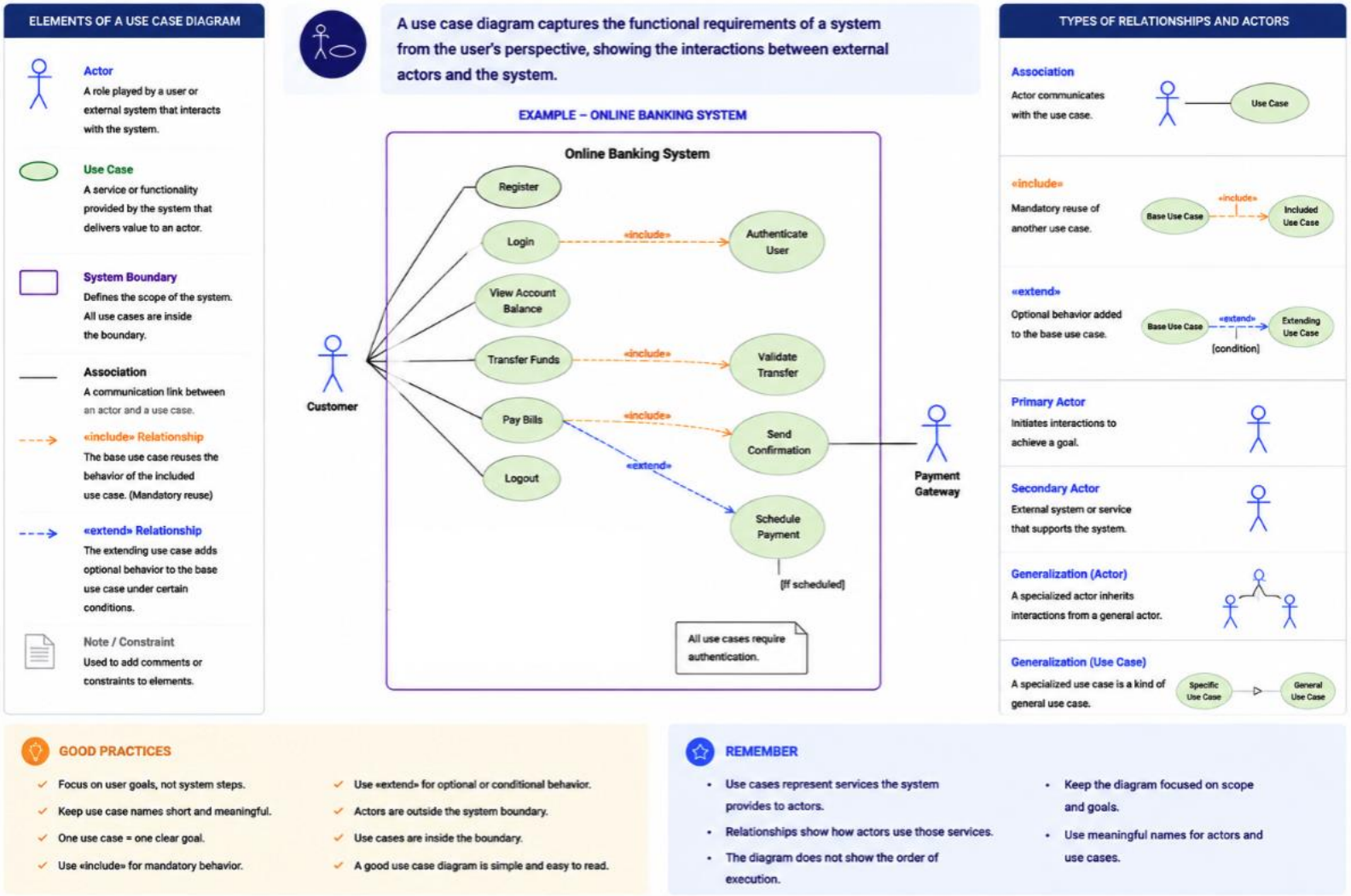
✧ Purpose

- **Capture System Functionality**: Represent what the system should do from the *user's perspective*.
- **Identify Actors & Interactions**: Show all external entities (actors) and how they interact with the system.

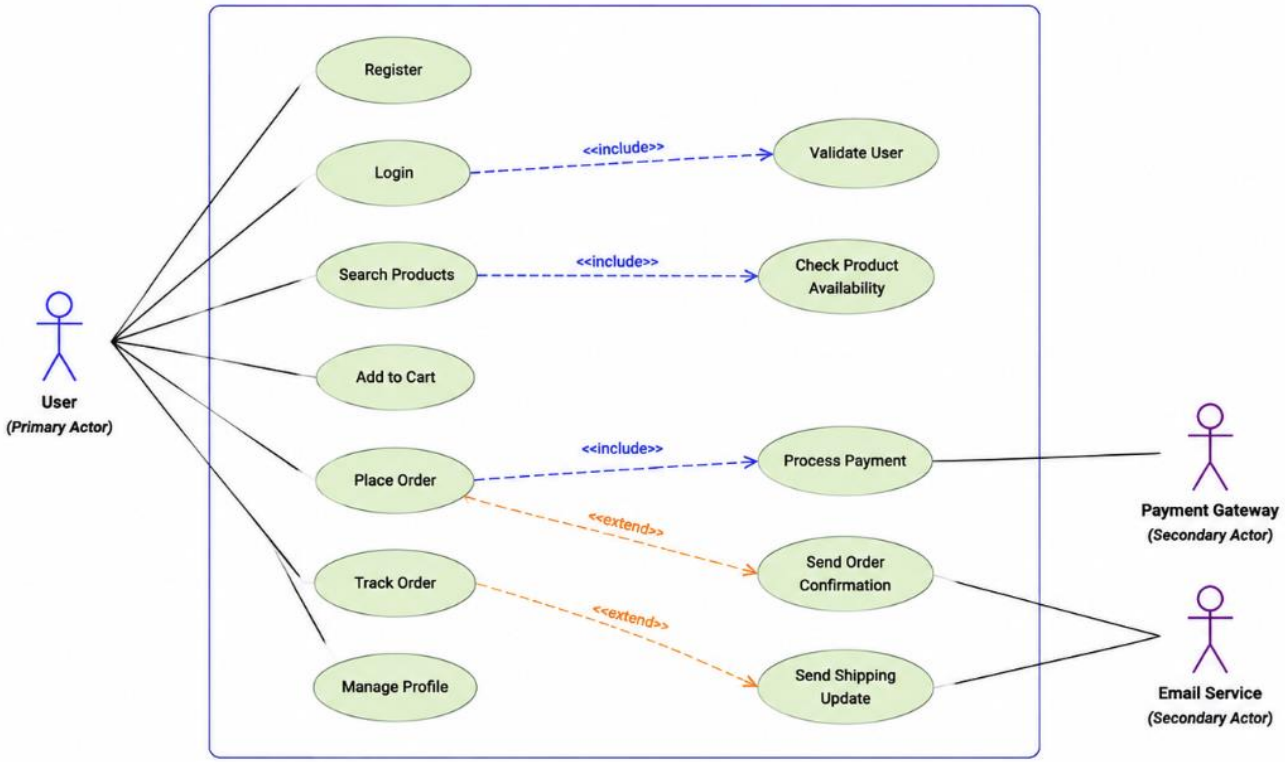
✧ Best Practices

- **Keep It High-Level:** Focus on main functionalities, not low-level details.
- **Avoid Technical Overload:** Limit specialized notation or internal system design details.
- **Ensure Clarity:** Make it easy for non-technical stakeholders to understand.

Use case diagrams



Use case diagram for e-commerce system



Legend

Primary Actor
Initiates interactions to achieve goals.

Secondary Actor
External system or service that supports the system.

Use Case
Functionality provided by the system.

Association
Communication between an actor and a use case.

<<include>>
Mandatory reuse. The base use case always includes the behavior.

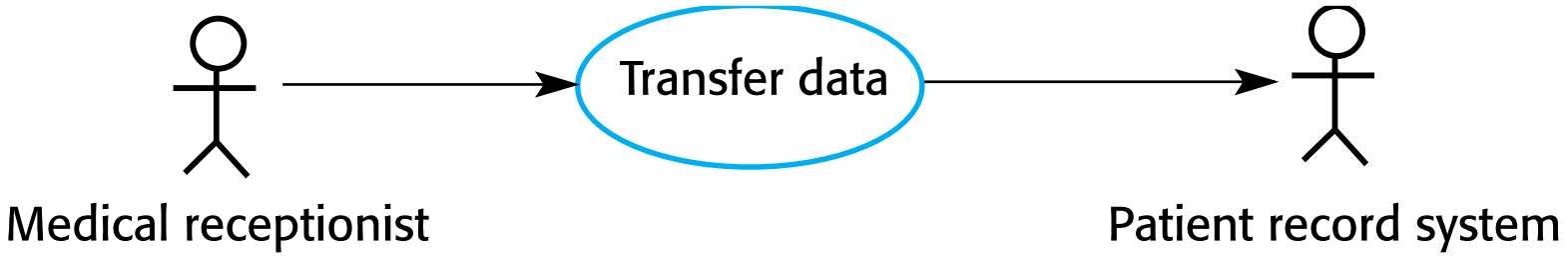
<<extend>>
Optional or conditional behavior added to the base use case.

Notes

- Use cases are inside the system boundary.
- Actors are outside the system boundary.
- <<include>> represents mandatory behavior.
- <<extend>> represents optional behavior.

Transfer-data use case

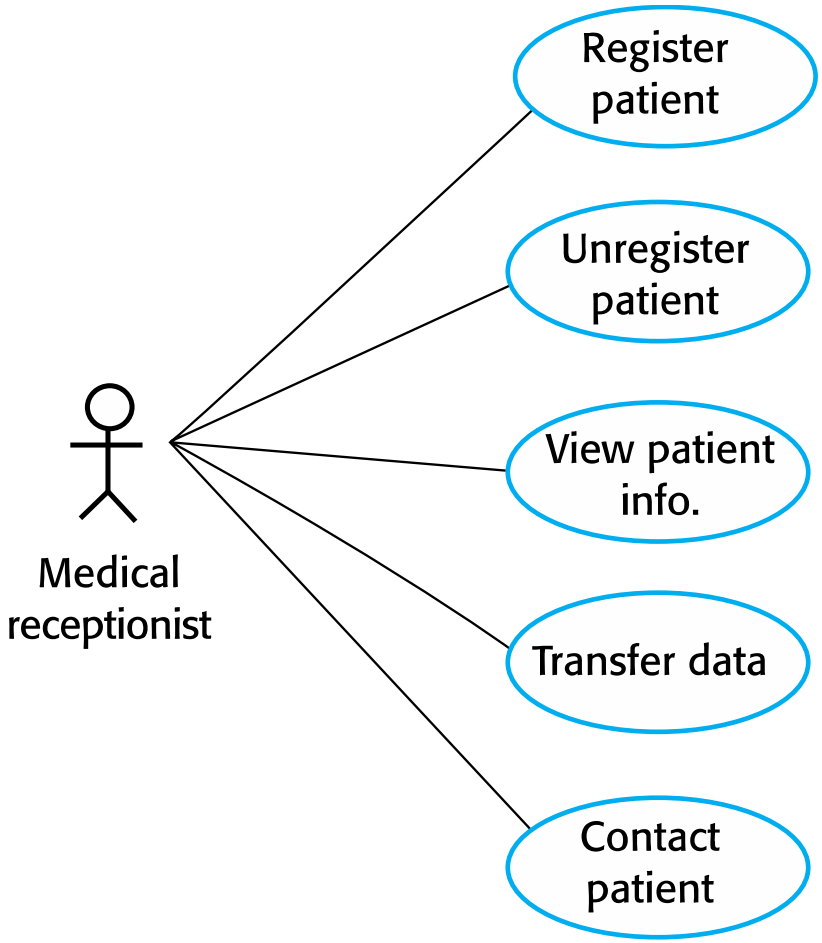
✧ A use case in the Mentcare system



Tabular description of the 'Transfer data' use case

MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the Mentcare system to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient's diagnosis and treatment.
Data	Patient's personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

Use cases in the Mentcare system involving the role 'Medical Receptionist'



Sequence diagrams

- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.
- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.
- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.

Sequence diagrams

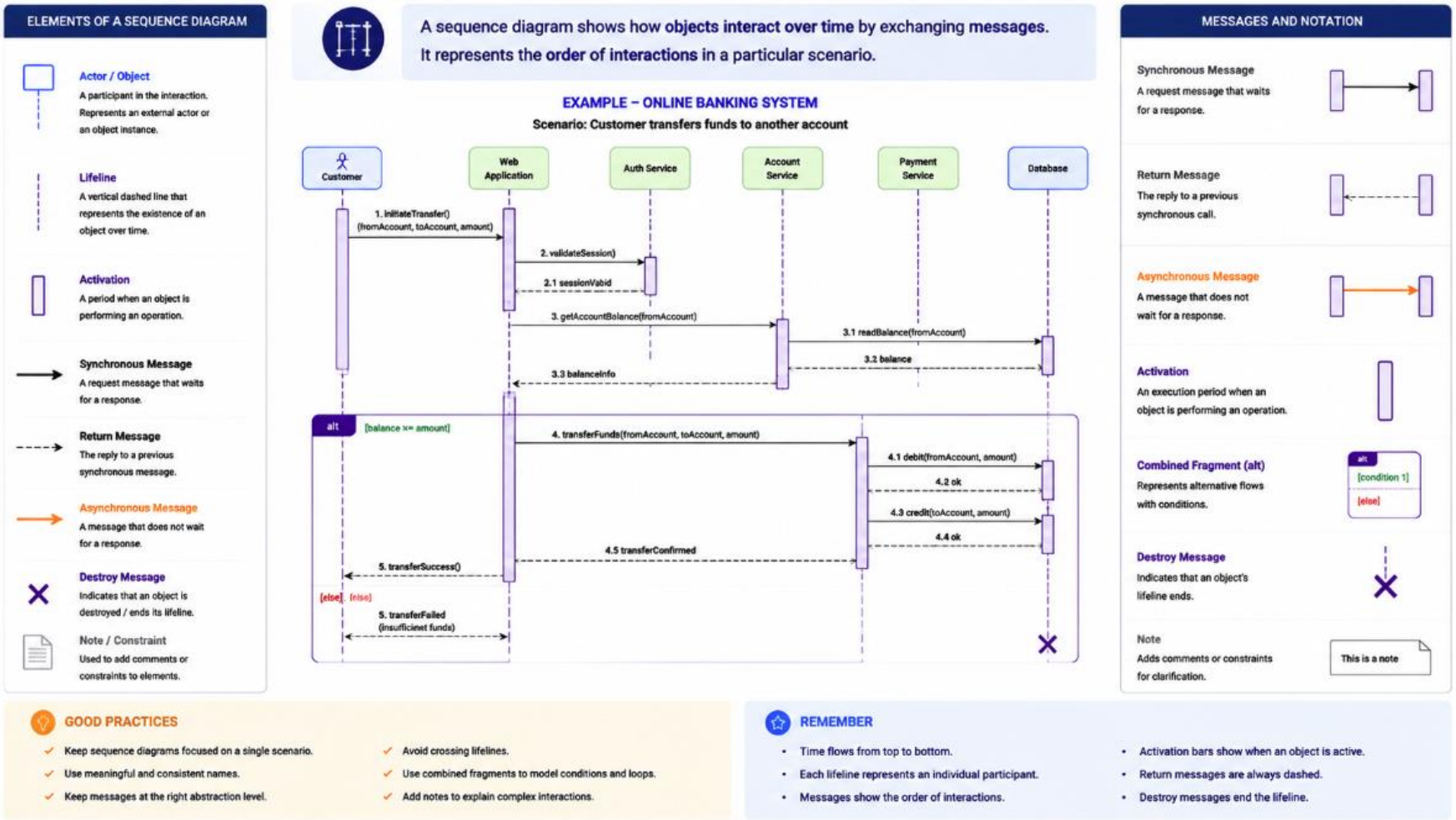
✧ Purpose

- **Model Interactions Over Time:** Show how objects or components interact in a specific time order.
- **Clarify Workflows & Responsibilities:** Identify which part of the system is responsible for each step in the process.

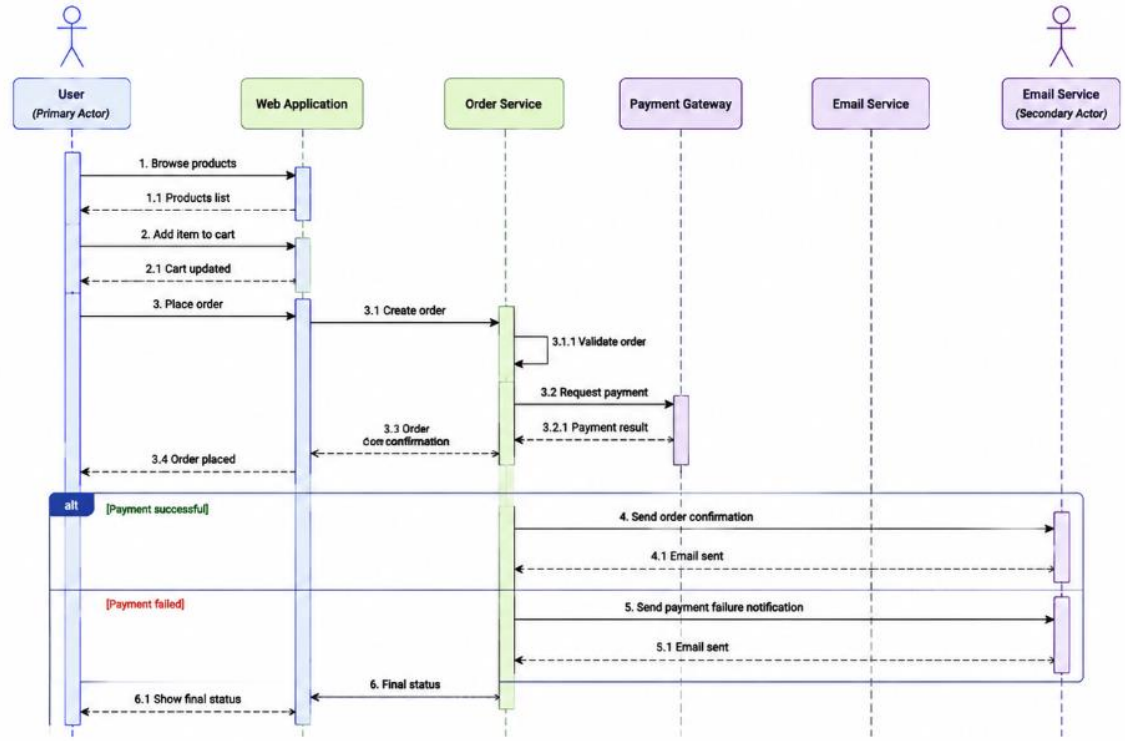
✧ Best Practices

- **Focus on Key Interactions:** Avoid clutter—show only the essential messages and participants.
- **Use Clear Message Names:** Ensure each message/action is understandable at a glance.
- **Maintain Logical Order:** Keep the sequence intuitive and consistent with the system's behavior.
- **Highlight Initiator and Outcome:** Make clear what triggers the interaction and the expected results.

Sequence diagrams



Sequence diagram for e-commerce system



LEGEND

Primary Actor
Initiates the interaction with the system.

Secondary Actor
External system or service that interacts with the system.

Combined Fragment (alt)
Represents alternative flows with conditions.

Lifeline
Represents an element (actor or object) that exists over time.

Activation
Represents the period when an element is executing an operation.

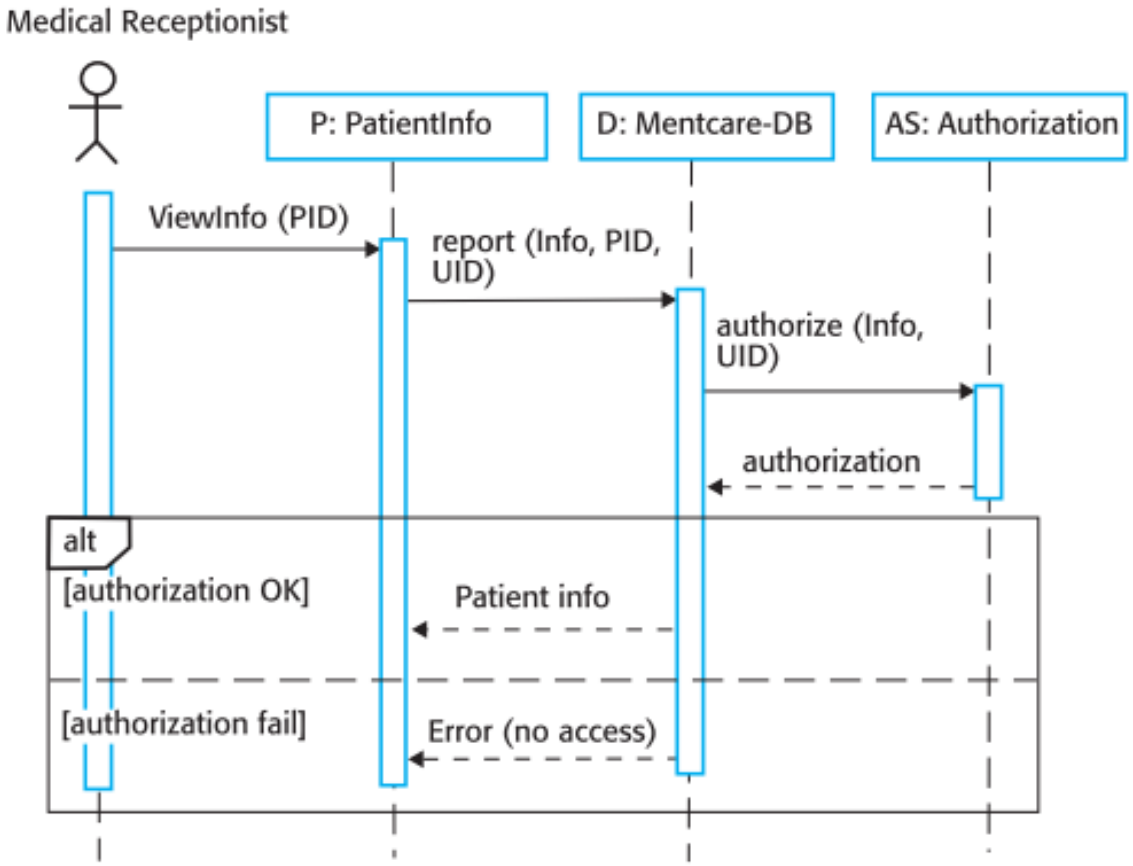
Synchronous Message
A request that waits for a response.

Return Message
The reply to a previous message.

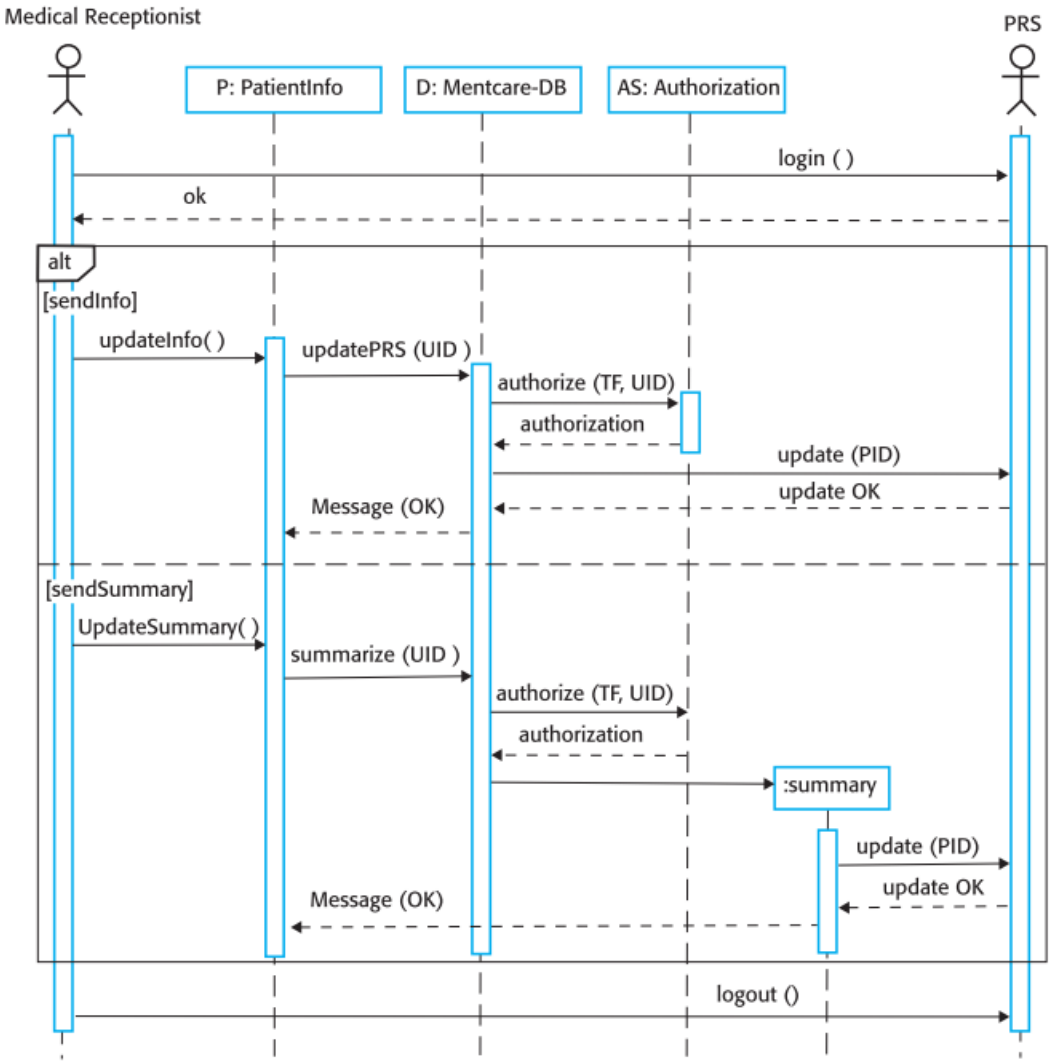
NOTES

- Time flows from top to bottom.
- Dashed vertical lines are lifelines.
- Messages show the order of interactions.
- Activation bars show when an element is active.
- The secondary actors (Payment Gateway and Email Service) are external systems that the application interacts with.

Sequence diagram for View patient information



Sequence diagram for Transfer Data



Context models

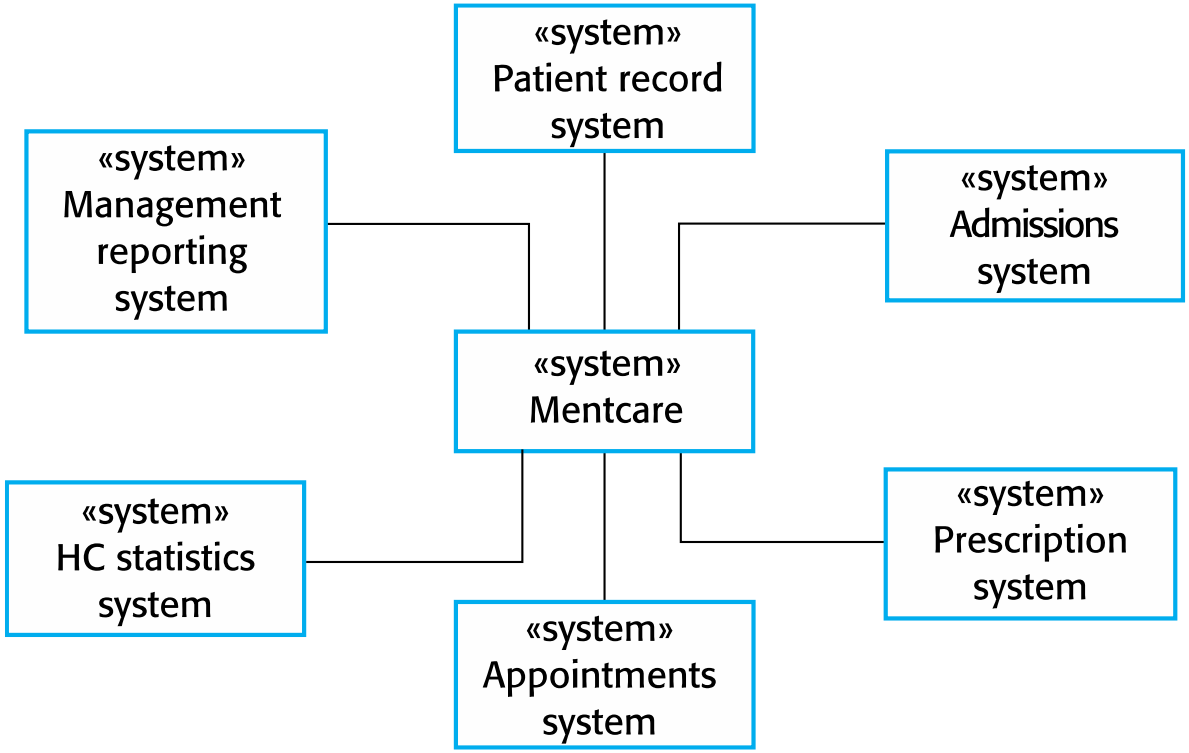
Context models

- Context models are used to illustrate the operational context of a system - they show what lies outside the system boundaries.
- Social and organisational concerns may affect the decision on where to position system boundaries.
- Architectural models show the system and its relationship with other systems.

System boundaries

- System boundaries are established to define what is inside and what is outside the system.
  - They show other systems that are used or depend on the system being developed.
- The position of the system boundary has a profound effect on the system requirements.
- Defining a system boundary is a political judgment
  - There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.

The context of the Mentcare system



Process perspective

- Context models simply show the other systems in the environment, not how the system being developed is used in that environment.
- Process models reveal how the system being developed is used in broader business processes.
- UML activity diagrams may be used to define business process models.

Activity diagrams

✧ Purpose

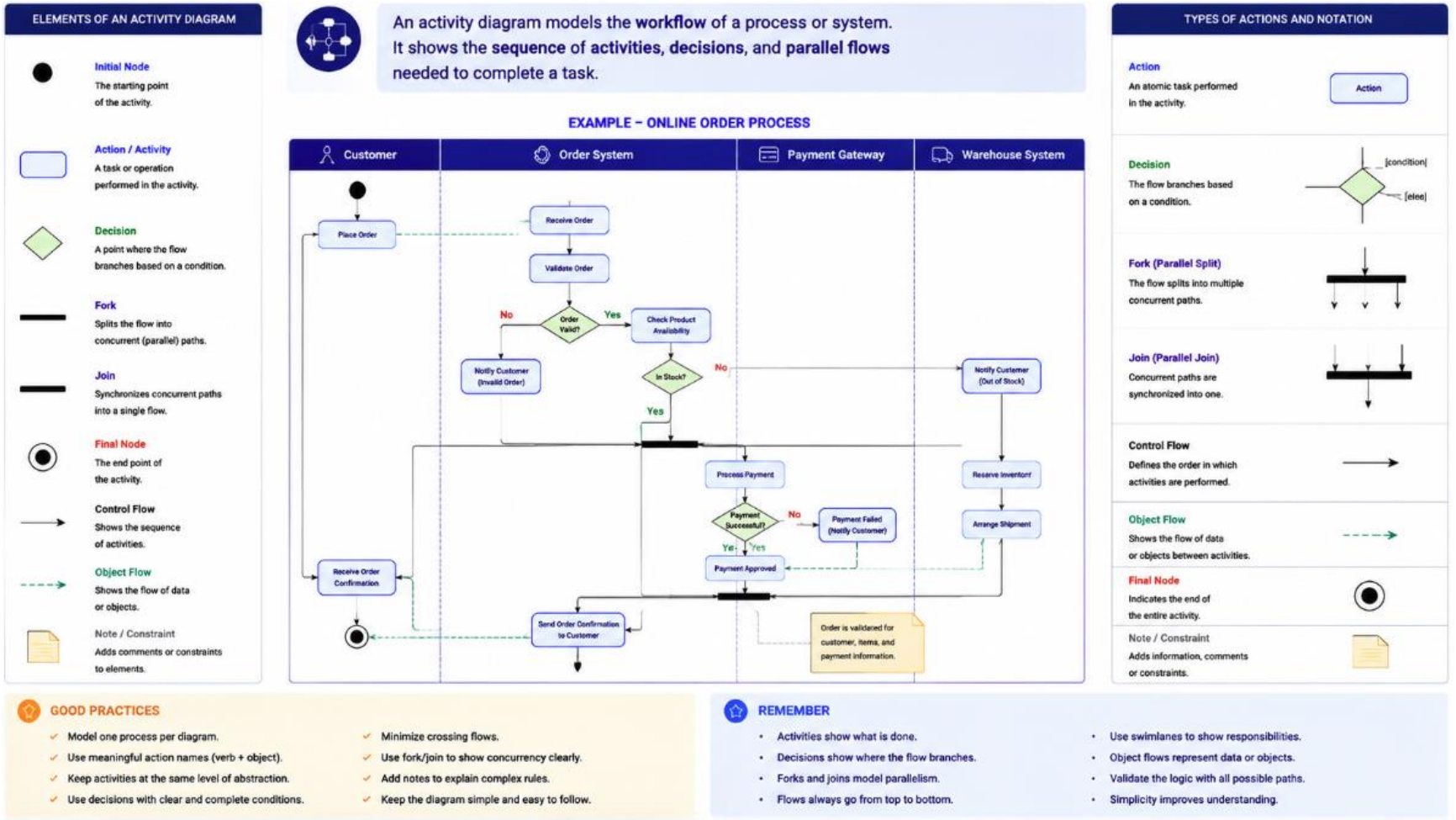
- **Model workflows of activities and actions**
  - Represents the step-by-step sequence of operations in a process.
  - Shows how activities transition from one to another, including the start and end points.
- **Represent parallel and conditional flows clearly**
  - Illustrates where processes can branch (decision points) or run in parallel (fork/join nodes).
  - Helps identify opportunities for concurrency and dependencies between tasks.

Activity diagrams

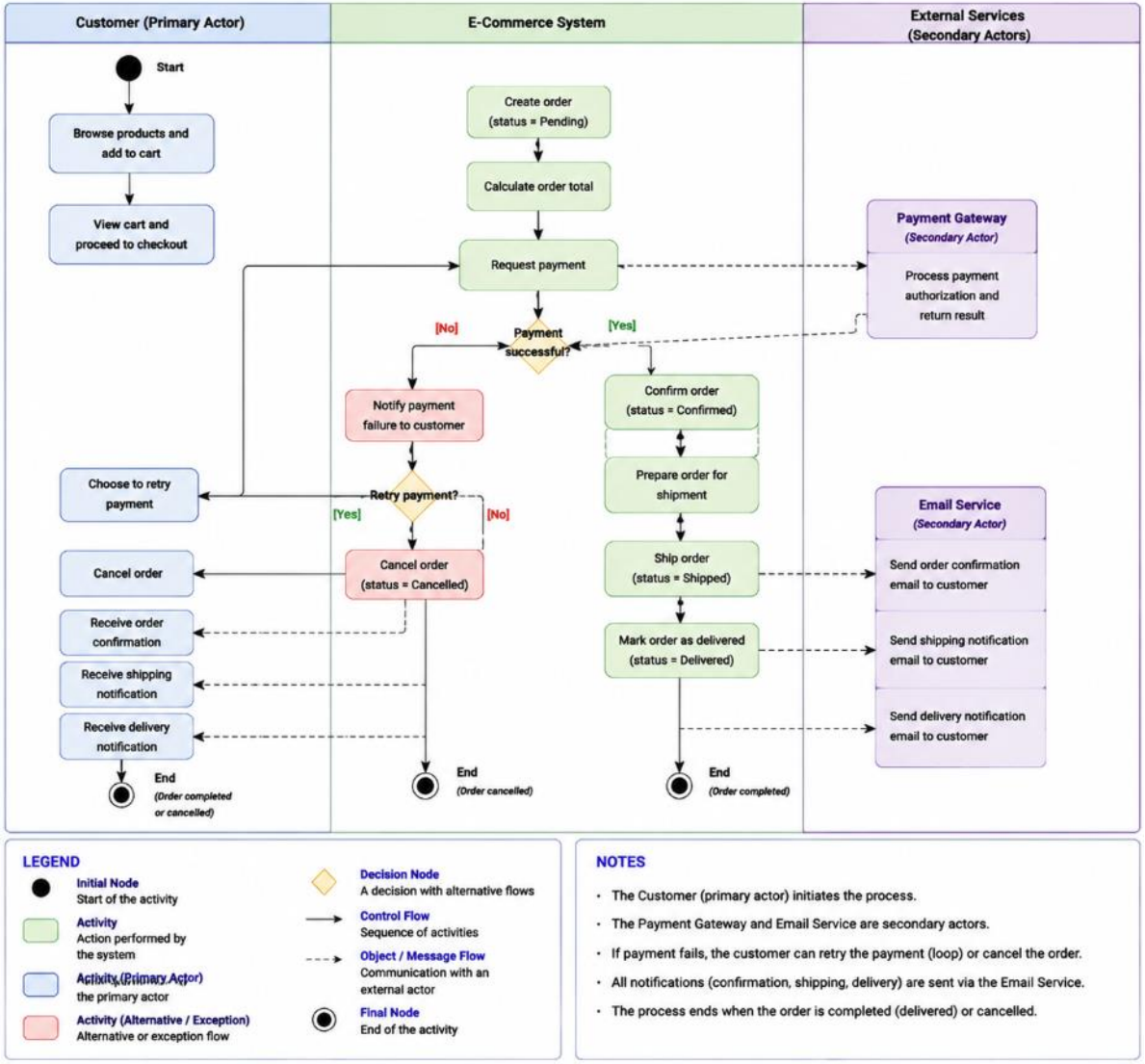
✧ Best Practices

- **Use swimlanes to show responsibilities**
  - Clearly assign each activity to a role, system, or department.
  - Swimlanes make it easy to identify ownership and collaboration points.
- **Keep decision points simple and easy to interpret**
  - Use clear yes/no or condition labels on branches.
  - Avoid overly complex conditions in one decision node; break them into multiple steps if necessary.

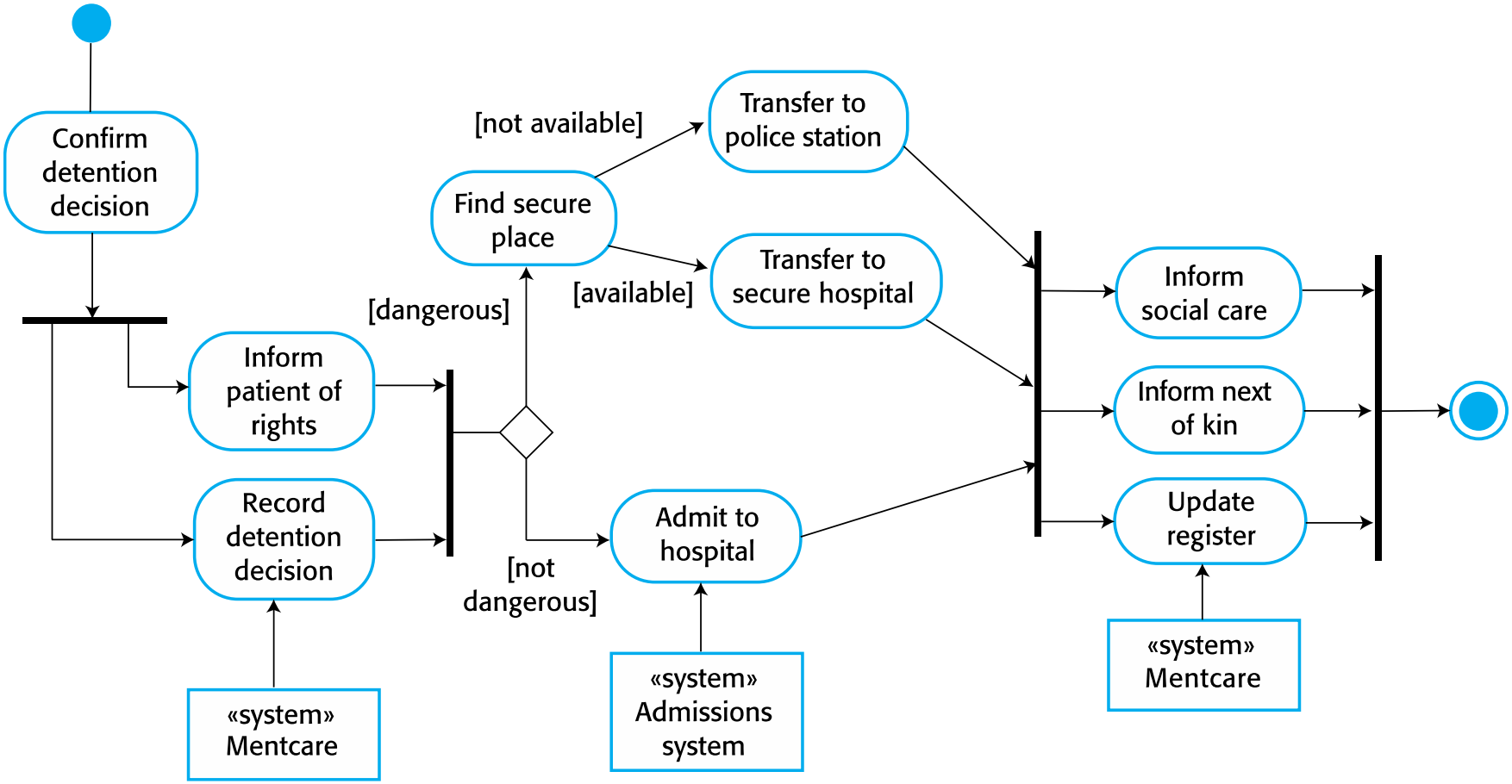
Activity diagrams



Activity diagram for e-commerce system



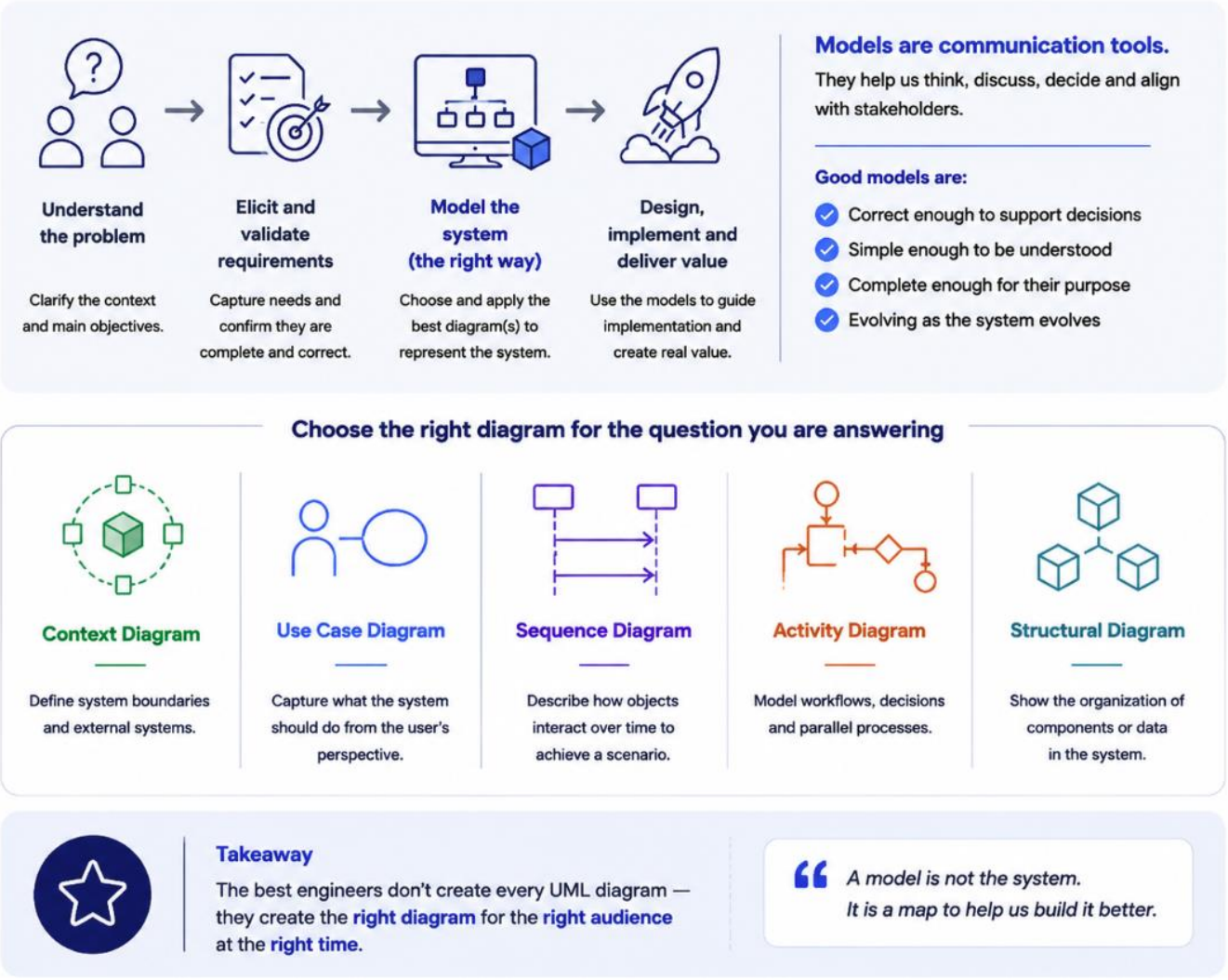
Process model of involuntary detention




IN-CLASS EXERCISE
PRACTICE • APPLY • UNDERSTAND


Work individually.
Hand in your work on a sheet of paper with your student number and name.
The exercise will be given by the instructor during the class.

Engineering insight



Key points

1



Model from multiple perspectives

Use external (context), interaction, structural and behavioral perspectives to capture the system completely and accurately.

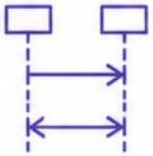
2



Use cases capture what matters

Use case diagrams represent system functionality from the actor's point of view and help elicitate and validate requirements.

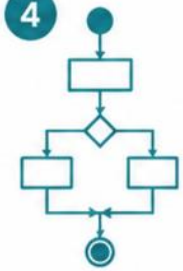
3



Sequences show how things happen

Sequence diagrams describe object interactions over time to realize scenarios and achieve system goals.

4



Activities model workflows and logic

Activity diagrams represent processes, decisions, parallel flows and business or system workflows.

5



Diagrams form a coherent view

Different UML diagrams answer different questions. Together they provide a complete understanding of the system.

6



Good models evolve with the system

Models are living artifacts. Keep them simple, accurate and up to date as the system evolves.



BRIDGE TO THE NEXT LECTURE

Now that we understand how to model systems using UML diagrams, next we will move to Software Design and learn how to transform models into high-quality solutions.



Next Lecture (T08):

Software Design
Design concepts and patterns.

QUESTIONS?

Let's clarify, explore and learn together!

Concepts

Anything you didn't understand?

Applications

How do we apply these ideas?



Decisions

Want to discuss any trade-offs?

Next Steps

What would you like to learn next?

No question is too small!

Your questions help everyone learn and grow.

Ask freely, challenge ideas, and let's build understanding **together!**