.NET Mentoring Program Intermediate+ Practice_Q2_2024

TICKETING DOMAIN – INTRODUCTION

Legal Notice: This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

What diagram types do you have on your current project?

On the current project we have 2 types of diagrams:

- Use Case Diagram: This diagram depicts the interaction between actors (users or external systems) and the system itself. It focuses on the functionalities offered by the system and how users interact with them to achieve specific goals.
- Sequence Diagram: It illustrates the message flow between different participants in a system over time, focusing on the sequence of interactions for a specific use case.

What are the pros and cons of UML diagrams?

Pros of UML Diagrams

- Standardization and Communication: UML provides a standardized notation for modeling software systems. This common language allows developers, designers, and other stakeholders to understand the system's structure, behavior, and functionality even if they haven't worked on the project directly. This promotes clear communication and collaboration.
- Visualization and Documentation: UML diagrams are visual representations of a system, making it easier to understand complex concepts compared to text-based descriptions. They serve as effective documentation tools, capturing the system's design and functionalities for future reference and maintenance.
- Analysis and Design: By creating UML diagrams, developers can think through the system's requirements, identify potential issues, and plan the overall architecture before coding begins. This promotes better analysis, design decisions, and helps avoid coding mistakes later in the development process.
- Maintainability and Scalability: Well-documented UML diagrams can significantly improve the maintainability of a software system. When new developers join the project, they can quickly understand the system's design by referring to the diagrams. Additionally, UML diagrams can be adapted to accommodate future changes and system growth.

Cons of UML Diagrams

- Time Consumption: Creating and maintaining UML diagrams can be time-consuming, especially for complex systems. The effort required to keep the diagrams up-to-date with the evolving codebase needs to be balanced with the benefits.
- Complexity for Beginners: While UML provides a standardized notation, it can be overwhelming for beginners to learn all the symbols and semantics. This initial learning curve can be a barrier to adoption, especially for smaller teams.
- Over-Engineering: In some cases, UML diagrams can become overly complex and detailed, losing their effectiveness as communication tools. It's important to focus on creating clear and concise diagrams that capture the essential aspects of the system.

What are the pros and cons of UML diagrams?

- Structural Diagrams: These diagrams focus on the static structure of a system, emphasizing its components, their organization, and the relationships between them. They provide a blueprint of the system's building blocks.
- Behavioral Diagrams: These diagrams focus on the dynamic behavior of a system, illustrating how the components interact with each other and how the system responds to events. They depict the flow of information and actions within the system.

Legal Notice: This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

Feature	Structural Diagrams	Behavioral Diagrams
Focus	Static structure of the system	Dynamic behavior of the system
Key aspects	Components, relationships, organization	Interactions, messages, workflows
Examples	Class Diagram, Object Diagram, Component Diagram	Use Case Diagram, Sequence Diagram, Activity Diagram

Understanding the Differences:

- Structural Diagrams: These diagrams are like blueprints, showing the building blocks (classes, objects, components) that make up the system and how they are connected. They don't show how these elements interact or how the system behaves over time.
- Behavioral Diagrams: These diagrams focus on how the system comes to life. They show the communication between components, the sequence of messages exchanged, and the flow of activities that occur when the system responds to events or user interactions.

Name the relationship types in a use case diagram

There are five main relationship types used in a use case diagram:

- Association: This is the most basic relationship and indicates a simple interaction between an actor and a use case. An actor can participate in one or more use cases, and a use case can involve one or more actors. The association is typically represented by a solid line connecting the actor and the use case.
- Generalization: This relationship shows an inheritance hierarchy between actors or use cases. A specific actor (child) inherits the behavior of a more general actor (parent). Similarly, a specific use case (child) inherits the functionality of a more general use case (parent). The generalization relationship is depicted by a hollow arrow pointing from the specific element to the more general element.
- Include: This relationship signifies that one use case includes the functionality of another use case. The included use case provides a set of common steps that can be reused by other use cases. The include relationship is shown by a dotted arrow labeled with the stereotype "<<include>>" pointing from the base use case to the included use case.
- Extend: This relationship indicates that one use case extends the functionality of another use case under specific conditions. The extending use case adds optional steps or variations to the behavior of the base use case. The extend relationship is represented by a dashed arrow labeled with the stereotype "<<extend>>" pointing from the extending use case to the base use case.
- Dependency: This relationship is less common in use case diagrams but can be used to show a causal relationship between two use cases. One use case might depend on the successful completion of another use case before it can proceed. The dependency relationship is depicted by a dashed arrow pointing from the dependent use case to the use case it relies on.

What is the sequence diagram?

A sequence diagram, also known as a system sequence diagram (SSD), is a type of UML (Unified Modeling Language)

Legal Notice: This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.

interaction diagram that illustrates the sequence of messages exchanged between objects in a system over time. It focuses on the interactions between objects involved in a specific scenario or use case.
The key aspects of a sequence diagram:

- Participants: The diagram depicts participants in the interaction, which can be objects, actors (users or external systems), or even other services within the system itself. They are represented by vertical lifelines throughout the diagram.
- Messages: Messages exchanged between participants are shown as horizontal arrows between their lifelines. The sequence of these arrows represents the chronological order of message flow.
- Time Progression: As you move down the diagram, time progresses. This allows you to visualize how the interaction unfolds step-by-step.
- Object Activation: A participant lifeline is typically drawn as a solid line, and it becomes thicker during the period when the object is actively involved in processing a message. This highlights the active state of the object.

Benefits of Sequence Diagrams:

- Clear Communication: Sequence diagrams provide a clear visual representation of how objects interact, making it easier to understand complex system behavior. They promote better communication between developers, designers, and other stakeholders.
- Detailed Message Flow: They allow you to see the specific messages exchanged between objects, including parameters and return values. This level of detail helps identify potential issues and ensure proper coordination between objects.
- Scenario Visualization: Sequence diagrams are ideal for visualizing specific scenarios or use cases within a system. By focusing on a particular interaction, you can analyze the flow of information and identify potential bottlenecks or areas for improvement.

Legal Notice: This document contains privileged and/or confidential information and may not be disclosed, distributed or reproduced without the prior written permission of EPAM®.