Discussion 2: List the two most important diagrams in one of the three modeling spaces

Discussion Topic:

List the two most important diagrams in one of the three modeling spaces. Provide a sample diagram created in Lucidchart. In response to your peers, debate with examples.

My Post:

Hello Class,

In Software Engendering (SE), Unified Modeling Language (UML) diagrams are essential tools for communicating ideas and understanding systems. UML diagrams are integral to SE because they present a suite of modeling artifacts that are a globally accepted standard for SE (Unhelkar, 2018 a). In other words, they provide a standard visual language for representing the structure, behavior, and interactions within software systems (Tim, 2024). Furthermore, these diagrams are organized within distinct modeling spaces (MOPS, MOSS, and MOAS) that guide their application to specific aspects of the software development process (Unhelkar, 2018b). They help software developer teams to plan, design, and communicate with stakeholders. UML version 2.5 defines 14 types of diagrams, use case, activity, package, class, profile, sequence, communication, interaction overview, object, state machine, composite structure, component, deployment, and timing diagrams. The table below provides a brief description of each diagram.

Table 1
UML Diagrams

Diagram Name	Description	Category: - Structural - Behavioral	Property: - Static - Dynamic
1- Use Case Diagram	Shows system functionality from a user's perspective; identifies actors and their interactions with the system.	Behavioral	Static
2- Activity Diagram	Models the flow or processes within a system, including workflows and dependencies.	Behavioral	Static
3- Class Diagram	Depicts classes, their attributes, operations, and relationships, both business and technical.	Structural	Static
4- Sequence Diagram	Models interactions between objects based on their timelines, emphasizing the order of messages.	Behavioral	Dynamic
5- Interaction Overview Diagram	High-level overview of interactions in a system; references sequence or other interaction diagrams.	Behavioral	Static
6- Communication Diagram	Similar to sequence diagrams, but emphasis on relationships between objects through messages.	Behavioral	Dynamic
7- Object Diagram	Shows objects (instances of classes) and their links at a specific point in time; depicts multiplicities.	Structural	Dynamic
8- State Machine Diagram	Models the runtime lifecycle of an object; shows the different states of an object and how they change.	Behavioral	Dynamic
9- Composite Structure Diagram	Represents the internal structure of an object/component with its runtime scenario, interfaces, and realizations.	Structural	Dynamic
10- Component Diagram	Shows the structure of the system in terms of components and their dependencies, also displays the physical layout.	Structural	Static
11- Deployment Diagram	Depicts the hardware architecture of the system; shows the distribution of software components across nodes.	Structural	Static
12- Package Diagram	Represents subsystems or areas of the system's organization. Also models dependencies between packages.	Structural	Static
13- Timing Diagram	Models the concept of time and the way in which an object's state changes over time.	Behavioral	Dynamic
14- Profile Diagram	Enables extensions to UML with project-specific elements such as stereotypes, tags, and constraints.	Structural	Static

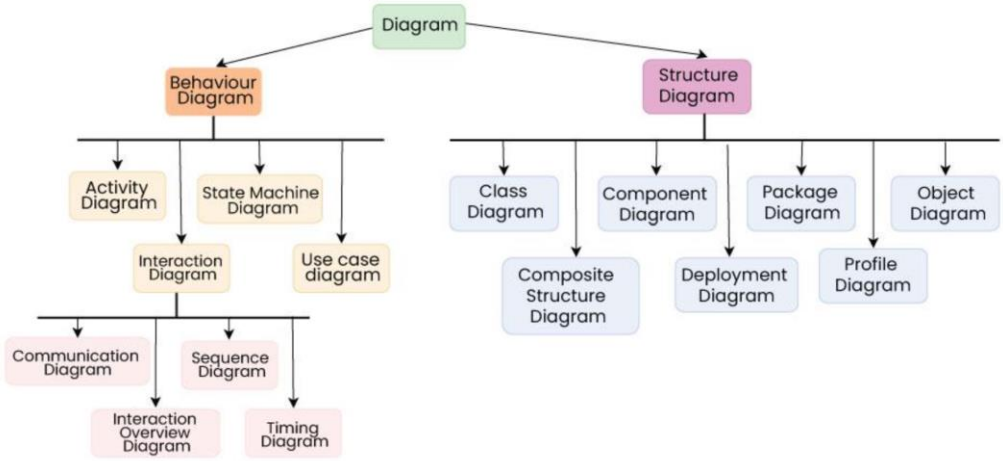
Note: Data from "Chapter 2 - Review of 14 Unified Modeling Language diagrams" (Unhelkar, 2018 a).

As shown in Table 1, there are two main categories of UML diagrams, structural and behavioral. A structural diagram illustrates the way a system is organized/structured whereas a behavioral diagram illustrates the flow of activity, actions, or interactions (behaviors) within the system (Unhelkar, 2018 a). A diagram can represent either a static or dynamic view of a system. Static diagrams illustrate the structure of a system at a specific point in time, whereas dynamic diagrams capture the system's changes over a period of time or during execution, emphasizing its time-dependent aspects.

The diagram below categorizes the diagrams into structural and behavioral groups and adds the interaction subgroup to the behavioral category. An interaction diagram shows interactions between the components of a system, and it can also depict how a system as a whole interacts with external entities (Dwivedi, 2019).

Figure 1

UML Diagrams Diagram



Note: From "Unified Modeling Language (UML) Diagrams*"* by GeeksforGeeks (2024).

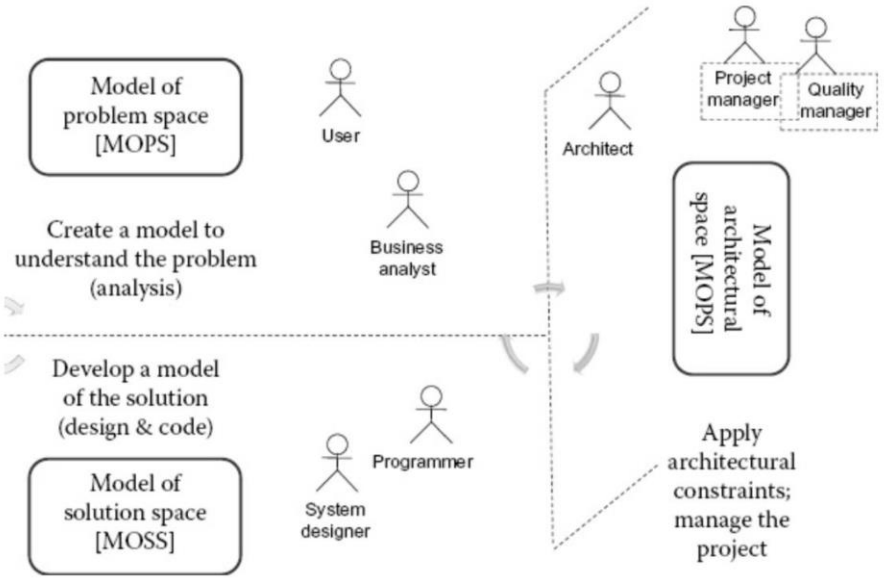
Each UML diagram plays a role in modeling different areas of a software system. These areas can be divided into three categories called modeling spaces with each diagram responsible for modeling within those spaces (Unhelkar, 2018 b). The three modeling spaces are:

1. **Model of Process Space (MOPS) or Problem Space:** It models "*what"* the business problem or user is. MOPS' goal is to understand and model the business requirements.
2. **Model of Solution Space (MOSS):** It models "*how"* the solution of the problem will be implemented. MOSS' goal is to represent the system's structure, behavior, and interactions using diagrams like class diagrams, sequence diagrams, and object diagrams.
3. **Model of Architectural Space (MOAS):** It models the "*big picture"* and the overall technical environment. MOAS' goals are to define architectural constraints, manage the project, and ensure quality.

These spaces are crucial for organizing and structuring the software development process, without them the use of UML can degenerate into incorrect or excessive modeling (Unhelkar, 2018 b).

The figure below illustrates how the three models relate to each other, to the different actors, and to the software development process.

Figure 2
Modeling Spaces



Note: From "Software Projects and Modeling Spaces: Package Diagrams. *Software Engineering with UML,"* Figure 3.3, by Unhelkar (2018 b).

Each UML diagram has varying levels of importance within the different modeling spaces. The table below maps each diagram to each modeling space, assigning up to 5 '*' to show the diagram's level of importance within a mapped space, with 5 '*' being the highest level of importance (Utmost Importance).

Table 2

*Importance of each UML Diagram to Respective Modeling Space (with 5 * for Utmost Importance to That Particular Space)*

UML Diagrams	MOPS (Business Analyst)	MOSS (Designer)	MOAS (Architect)
Use case	*****	**	*
Activity	*****	**	*
Class	***	*****	**
Sequence	****	*****	*
Interaction overview	****	**	**
Communication	*	***	*
Object	*	****	***
State machine	***	****	**
Composite structure	*	*****	****
Component	*	***	*****
Deployment	**	**	*****
Package	***	**	****
Timing	*	***	***
Profile	*	**	***

Note: From "Software Projects and Modeling Spaces: Package Diagrams. Software Engineering with UML," Table 3.2, by Unhelkar (2018 b).

For example, in the Model of Solution Space (MOSS), the three most important diagrams are the class, sequence, and composite structure diagrams. Where each diagram plays a different role in modeling the solution:

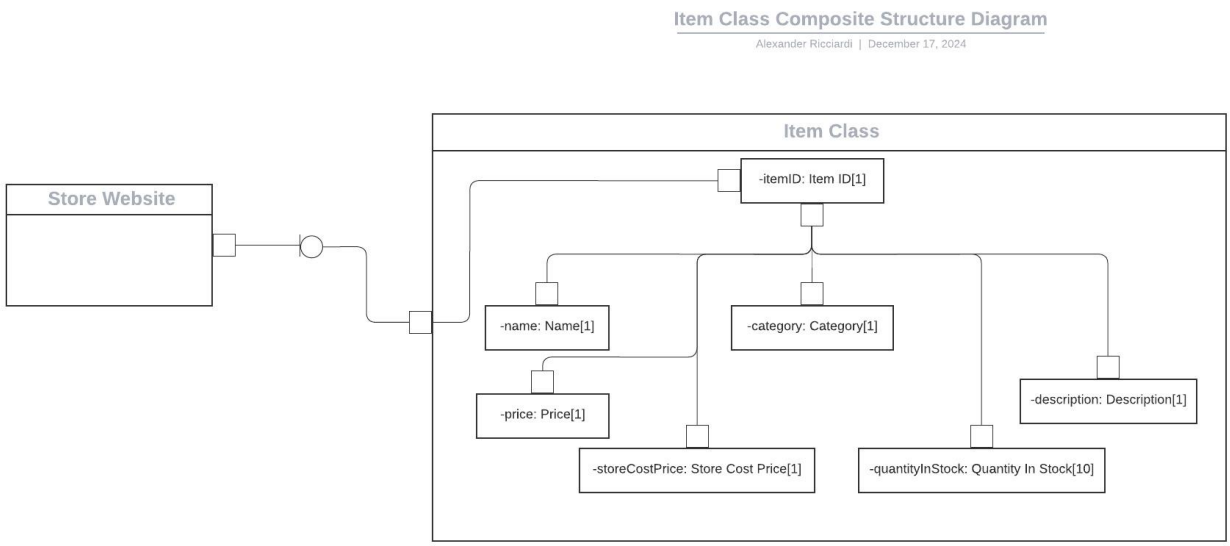
- Class diagrams illustrate detailed designs and programming constructs. They can also model relational database tables. Class diagrams define the system's structure, showing the classes, their attributes, methods, and the relationships between them.
- Sequence diagrams illustrate detailed models of interactions within the system. They depict the dynamic exchange of messages between objects over time.
- Composite structure diagrams illustrate the internal structure of a classifier (like a class or component), that is the functionality of a group of objects and components, including their interfaces and realizations. It is only a UML diagram used to model the physical components of a system or business.

(Unhelkar, 2018 b)

For example, let's explore the composite structure diagrams of a simple item class. See the figure below.

Figure 3

Item Class Composite Structure Diagram

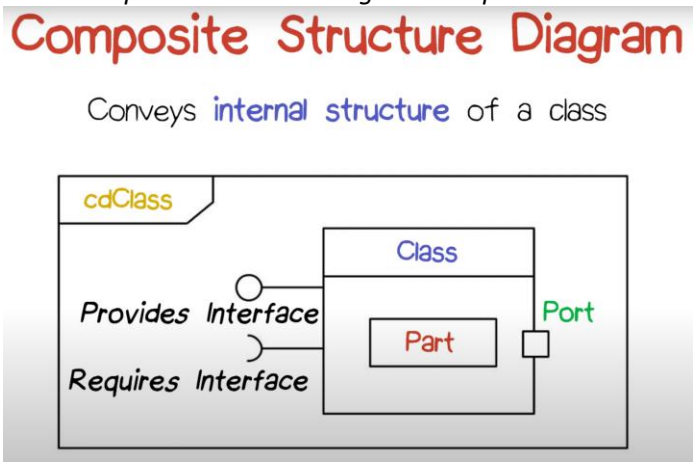


Note that the item class provides an interface for the website component of the project, allowing the website to access and display item information such as name, price, and availability. The figure below provides a basic illustration of components that can be found in a

In other words, the provided interface indicates what the class offers to the external environment, while the required interface specifies what the class needs from other components to function correctly. This interaction between classes is essential for creating a functional software architecture composite structure diagram.

Figure 4

Basic Composite Structure Diagram Components



Note: From “Composite Structure Diagram” by Udacity (2015)

To summarize, UML diagrams are essential for communicating ideas and understanding systems. The 14 UML diagram types are categorized as either structural or behavioral and can represent either static or

dynamic view of a system. Additionally, these diagrams within modeling spaces—MOPS, MOSS, and MOAS— have varying levels of importance. Thus, selecting the most relevant diagrams for each stage of the software development lifecycle is essential. As shown in the example of the Item Class Composite Structure Diagram is an application of UML within the appropriate modeling space (MOSS in this case). Ultimately, the strategic use of UML diagrams throughout the software development lifecycle is essential to avoid incorrect or excessive modeling, as they guide, and empower software engineers to create representations of problems, solutions, structures, interactions, and relationships within complex systems.

-Alex

References:

Dwivedi, N. (2019, September 9). Type of UML models. *Software Design: Modeling with UML* [Video]. LinkedIn Learning. https://www.linkedin.com/learning/software-design-modeling-with-uml/types-of-uml-models?u=2245842

GeeksforGeeks (2024, October 23). *Unified Modeling Language (UML) diagrams*. GeeksforGeeks. https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/

Tim. (2024, November 5). *Top 7 most common UML diagram types for software architecture*. Icepanel. https://icepanel.io/blog/2024-11-05-top-7-most-common-UML-diagram-types

Udacity (2015, February 23). *Composite structure diagram* [Video]. YouTube. https://www.youtube.com/watch?v=pJyuKhD86Ro

Unhelkar, B. (2018 a). Chapter 2 - Review of 14 Unified Modeling Language diagrams. *Software engineering with UML*. CRC Press. ISBN 9781138297432

Unhelkar, B. (2018 b). Chapter 3 - Software projects and modeling spaces: Package diagrams. *Software engineering with UML*. CRC Press. ISBN 9781138297432