

## Discussion-5 differences of class diagrams used in analysis with those use in design

### **Discussion Topic:**

Discuss two key differences between class diagrams in analysis and those in design. Use applicable examples (diagrams or pseudocode) in your post.

### **My Post:**

Hello Class,

In Software Engineering (SE), Unified Modeling Language (UML) class diagrams are one of six UML types of structural diagrams (IBM, 2021). Class diagrams are used to model object processes and the static structures of a system. In other words, they are the blueprints of systems and subsystems. However, these diagrams can be used in different stages of system design. This post gives an overview of the class diagram and explores the difference between class diagrams used in the problem space (MOPS), which can be used to analyze and capture a system's functional requirements and components, and those used in the solution space (MOSS), which focuses on designing a system by modeling what the system needs to do.

### **Class Diagrams Overview**

Class diagrams are useful in many stages of the SE process, especially during the development life cycle. In the analysis phase (MOPS), they help analyze and capture problem requirements and components. In the design phase (MOSS), they help design solutions. IBM in its “Rational Software Modeler” documentation describes class diagrams as an essential tool for software development that can help:

*you to understand the requirements of your problem domain and to identify its components. In an object-oriented software project, the class diagrams that you create during the early stages of the project contain classes that often translate into actual software classes and objects when you write code. Later, you can refine your earlier analysis and conceptual models into class diagrams that show the specific parts of your system, user interfaces, logical implementations, and so on. Your class diagrams then become a snapshot that describes exactly how your system works, the relationships between system components at many levels, and how you plan to implement those components.*

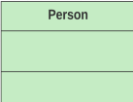
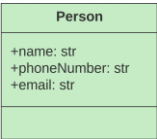
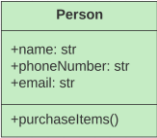


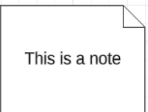
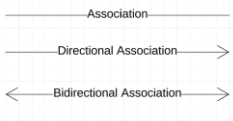
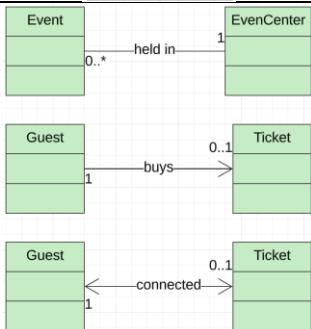
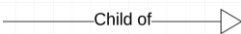
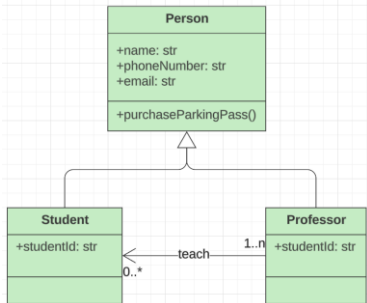
(IBM, 2021, p1)


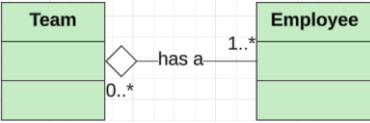

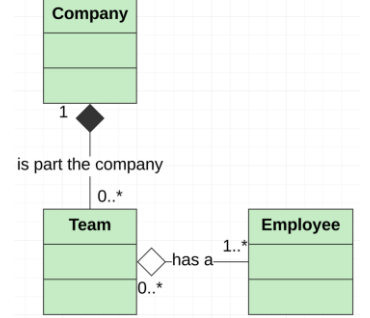
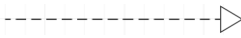
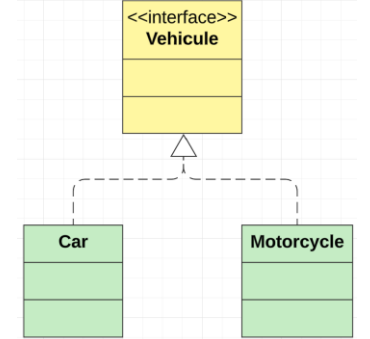




In other words, class diagrams can be used to illustrate, define, and document the structural features of a system. UML class diagrams are the most widely used notation standard in SE for object-oriented modeling.

### **UML Class Diagram Notation Overview**

In object-oriented based software development UML Diagram notation is used to illustrate advanced class diagrams and class-to-class relationships. The table below illustrates the different components found in a UML class diagram.

**Table 1**  
*UML Class Diagram Notation*

Component	Definition	Example:
<b>Class</b> 	Represents an object or objects that share a common structure and behavior.	Person, Doctor, Department.
<b>Attributes</b> 	Represent properties of a class that describe a range of data values that instances of the class may hold.	+name: str +phoneNumber: str +email: str
<b>Operations</b> 	Represent Functions or methods that can be applied to or by objects of a class.	+purchaseItems()
<b>Multiplicity</b> 	Indicates the number of instances of one class linked to one instance of another class.	1..*, 0..*
<b>Notes</b> 	Represents annotations, it is used to add comments or explanations to a diagram.	
<b>Association</b> 	Represent a relationship between two classes where objects of one class are connected to or use the services of objects of another class.	
<b>Inheritance</b> 	Represents a relationship where one class (subclass) inherits the attributes, operations, and relationships of another class (superclass). Also known as generalization.	

<b>Aggregation</b> 	Represent a specialized form of association representing a "whole-part" or "has-a" relationship. Also referred to as "by reference" implying that the relationship is only a reference (pointer) to an object	
<b>Composition</b> 	Represent a stronger form of aggregation where the "part" cannot exist independently of the "whole."	
<b>Realization</b> 	Represents a relationship between an interface and the class that realizes or implements it.	
<b>Interface</b> 	Represents a collection of operation signatures without implementation details.	 <p>The "Patient" class is a substantial class containing numerous operations. However, not all of these operations are needed by the "PatientForm" class, it uses only one of the "Patient" interfaces, the "Patient Registration Interface" subclass.</p>
<b>Dependency</b> 	Represents a relationship indicating that one class (the client) depends on another class (the supplier). Changes in the supplier may affect the client.	

*Note:* Data from several sources (Unhelkar, 2018; Dwivedi; 2019; Colorado State University Global, 2025). Shapes made with Lucid App.

As shown in Table 1 UML class programs provide an extensive set of notations for modeling the static structure of a system, including classes, attributes, operations, and various types of relationships between them. These diagrams can be used to analyze and design systems.

### Differences Between Analysis and Design Class Diagrams

In the context of SE, it is important to understand the difference between design classes and analysis classes. In problem space, class diagrams are used to analyze and capture a system's functional

requirements and components. They act like detectives investigating and understanding the problem at hand (Navlaniwesr, 2024). In the solution space, they focus on what a system needs to do, without diving into how it will be done. In other words, analysis class diagrams focus on understanding the problem domain, while design class diagrams detail the solution.

### Analysis Class Diagrams

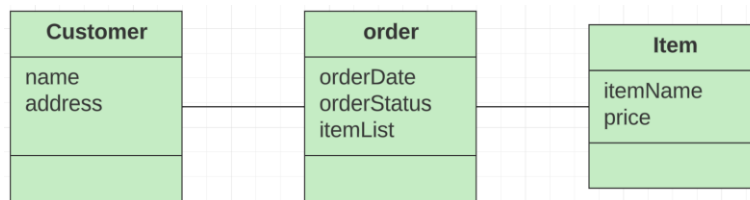
Analysis class diagrams are abstract. They are usually based on case diagrams, and they are often the initial step in understanding the problem domain and the detailed requirements of a system. They are characterized by the following:

- They focus on understanding the problem domain and capturing the requirements of a system (Navlaniwesr, 2024).
- They are illustrations of a system's components, attributes, and relationships.
- They portray a high-level view of a system without delving into implementation specifics.
- They capture user needs, business processes (use cases), and external system interactions.

Below is an example of analysis class diagrams illustrating a very simple product ordering system.

**Figure 1**

*Simple Product Ordering Analysis Class Diagram*



*Note:* the diagram shows basic entities/classes (Customer, Order, Product) with simple attributes. Relationships are basic associations.

### Design Class Diagrams

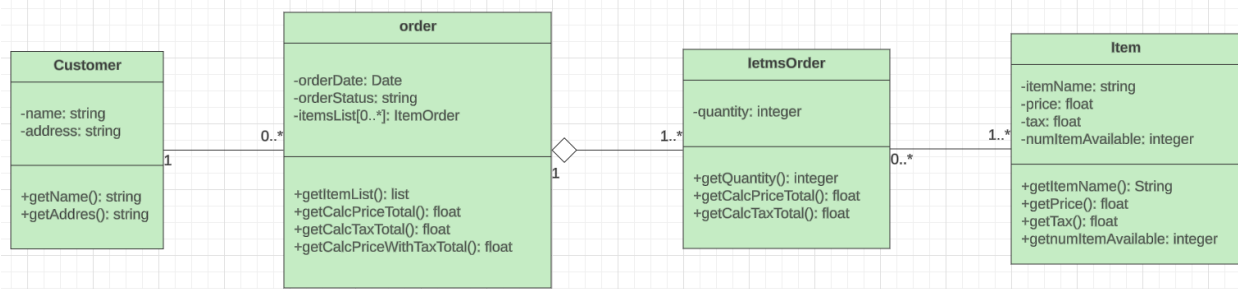
Design class diagrams are blueprints illustrating what a system needs to do. They contain more details than the class diagram developed during analysis in the problem space (Unhelkar, 2018). They model in detail relationships, objects' attributes, and functionality, as well as visualize multiplicities. They are characterized by the following:

- They model a system structure and operation signatures (Navlaniwesr, 2024). A signature is the interface of an operation, that is the information needed to call a specific operation such as parameters and what the operation returns.
- They illustrate in detail a class's methods, attributes, and interactions.
- They incorporate implementation details, such as data structures and algorithms.
- They translate functional requirements into structures that can be developed.

Below is an example of design class diagrams illustrating the simple product ordering system used earlier to illustrate analysis class diagrams.

**Figure 2**

*Simple Product Ordering Design Class Diagram*



*Note:* the diagram added details to the design class diagram, an extra class “ItemOrder” with an aggregation relationship to the “order” class

As shown by Figures 1 and 2, analysis and design class diagrams differ in detail. Most importantly, they differ fundamentally in their purpose, with analysis class diagrams focusing on “*what*” a system does, while design class diagrams focus on designing a solution or system by defining what the system needs to do. In other words, analysis class diagrams are used to analyze a system, while design class diagrams are used to design a system.

To summarize, class diagrams are used to model systems and subsystems. They are the blueprints of systems and subsystems. They are visual blueprints representing system structure stages which are used in various stages of the development lifecycle, from the initial analysis of the problem domain (MOPS) to the detailed design of the solution (MOSS). Analysis and design class diagrams differ not only in detail but more importantly in their purpose, with analysis class diagrams modeling “*what*” a system does, and design class diagrams modeling a solution or system by defining what the system needs to do or “*how*” it needs to do it. In object-oriented software development, UML class diagrams are crucial for analyzing and designing complex systems, making them an indispensable tool in the software engineer's toolkit.

-Alex

#### References:

Colorado State University Global. (2024). *Module 5 - Software design concepts: Class designs* [Interactive lecture]. CSU Global Computer Science Department Canvas.

[https://csuglobal.instructure.com/courses/104036/pages/module-5-overview?module\\_item\\_id=5372244](https://csuglobal.instructure.com/courses/104036/pages/module-5-overview?module_item_id=5372244)

Dwivedi, N. (2019, September 23). Class diagrams: Relationships. *Software design: Modeling with UML*. LinkedIn Learning. <https://www.linkedin.com/learning/software-design-modeling-with-uml/class-diagrams-relationships?resume=false&u=2245842>

IBM (2021, March 5). Class Diagrams. *Rational software modeler*. IBM Documentation. <https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams>

Navlaniwesr (2024, April 18). *What is the difference between design classes and analysis classes?* GeeksforGeeks. <https://www.geeksforgeeks.org/what-is-the-difference-between-design-classes-and-analysis-classes/>

Unhelkar, B. (2018). Chapter 11 — Class Model-3: Advanced class designs. *Software engineering with UML*. CRC Press. ISBN 9781138297432