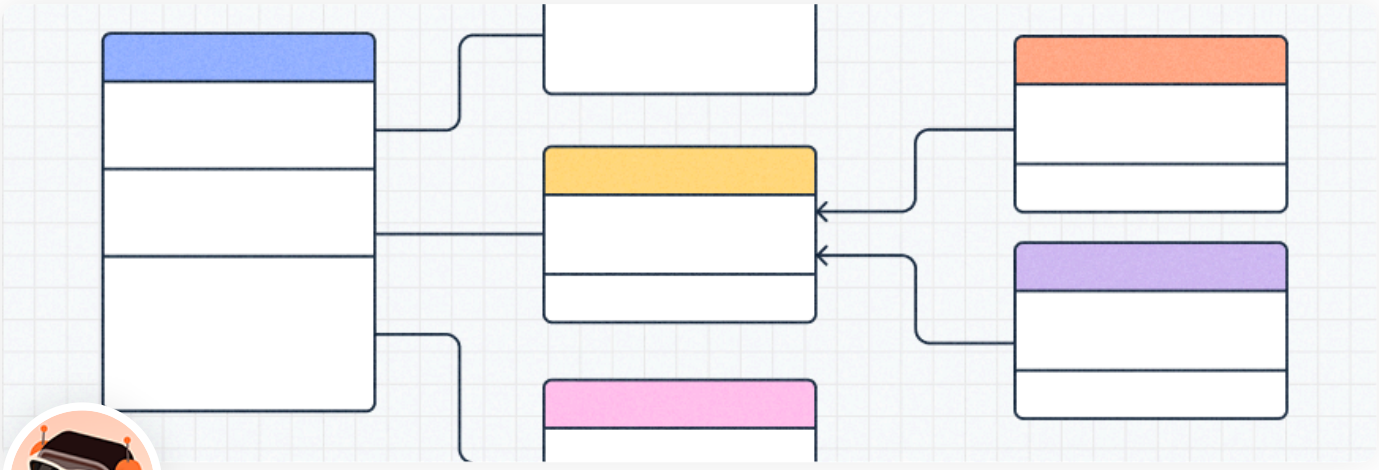
How to make a UML class diagram (and others) with examples



By Steve Tsentserensky

Updated March 12, 2025 • 11 min read

DIAGRAMMING

IN THIS ARTICLE (7)

Understanding systems is what UML diagrams are all about. Whether it's a UML class diagram, a sequence diagram or any of the other 10+ out there, there's a method to putting one together to create clarity. In this article, we'll walk you through how to make a UML diagram and showcase examples of each type.

What is a UML diagram?

Learning how to make one correctly is certainly important but let's pinpoint what they are first.

For starters, UML means Unified Modeling Language and it's a standardized visual language used most often in, but not exclusively, software development, IT and business systems.

The idea is to have a uniform way to represent the classes, objects, relationships and interactions within simple or complex systems to make it easier for developers and stakeholders to understand and communicate about the system.

Look at it as a way to bridge the gap between technical and non-technical folks.

Here's how to make a UML diagram

Ok, onto the main event: how to make a UML class diagram.

And because there are so many types of UML diagrams, we're going to focus on just one and make this specifically a UML class diagram tutorial because it's among the most commonly used.

A quick definition: UML class diagrams are used to show and define a system's static classes, attributes, methods and relationships between classes.

Lastly, before we start to draw a class diagram, it's important to know that UML diagrams have their own UML notation and symbols that must be followed.So make sure the flowchart software you're using can handle these.

1. Define the purpose and scope

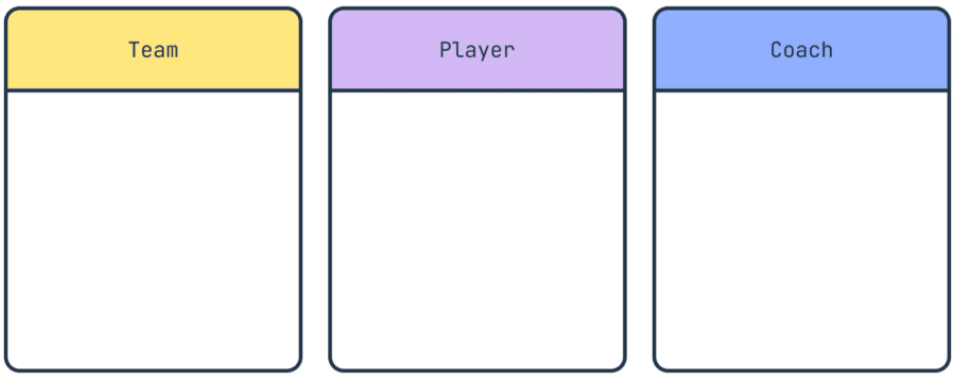
The first step of how to make a class diagram is simple, identify and describe the system you're modeling. This can be anything but to keep things straightforward and make the relationships that we document easier to understand, we'll use the NFL as an example.

2. Identify and label classes

Classes are the main objects or major components of a system; the highest level.

Using our simple NFL team example, we can break things down into 3 classes; Team, Player and Coach.

Open your diagram creator and select the 3-tiered class box label the top box with the name of each class.



3. Add attributes

Below the class name, in the middle box, is where you'd put attributes, which are the structural features and descriptions of a class.

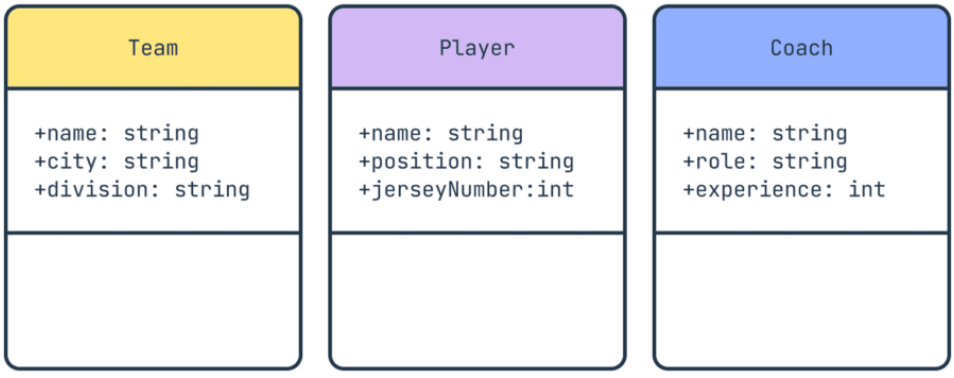
For the Team, we'd have attributes like name, city and division.

For Player, it would be name, position, jerseyNumber.

And for Coach, we can say name, role, experience.

After each attribute goes a colon followed by the type of data. For example, "String" would be used for most text and "Integer" for numbers.

The +,-, # and ~ are the visibility of each attribute, we'll explain in the next section.



4. Add methods

Also referred to as functions or operations are the behavioral features of a class, or what that class does.

In other words, they define what actions or operations can be performed on objects of that class.

For our NFL example, it might look like this:

Team: playGame(), hireCoach()

Player: playGame(), train()

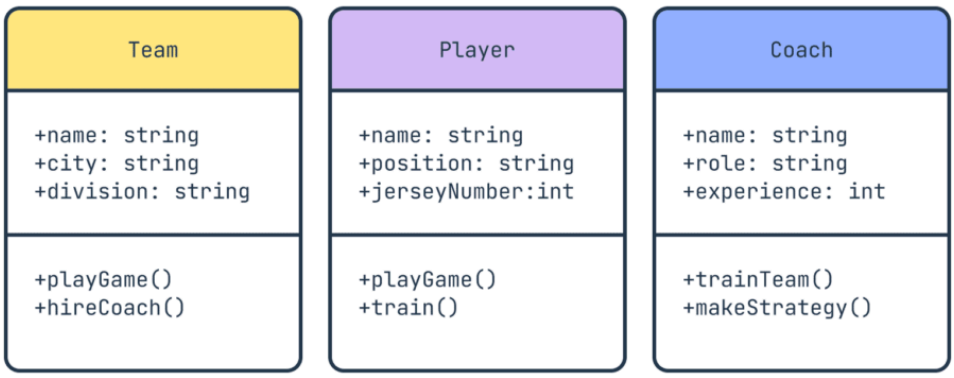
Coach: trainTeam(), makeStrategy()

The () represents a function that would be programmed at a later time.

Note that both attributes and methods appear in list form, with a lowercase letter and no space between words.

As for those symbols: +,-, # or ~, they're the visibility of that attribute or method and its access privileges to other classes.

– = Private, meaning no accesses from other classes
+ = Public, accessible to other classes
# = Protected, accessible by the same class or subclasses
~ = Package, can be used by any class in the same package



5. Show relationships

This is the fun part, class diagram relationships and showing how each class may interact with another.

Relationships can be defined in quite a few ways in class diagrams. To show the most important ones, we'll expand our NFL example a touch throughout this section.

Inheritance

Also known as generalization, this is a relationship where a subclass inherits attributes, methods and other relationships from a superclass.

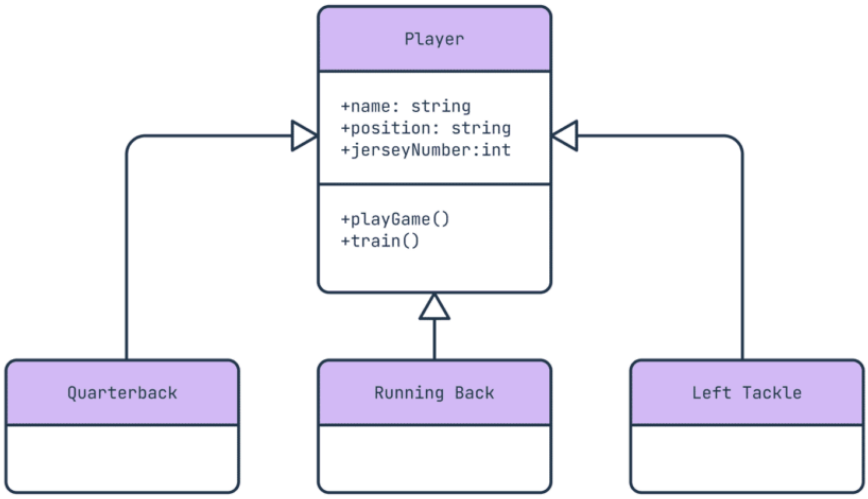
In the NFL example, if we add the classes of Quarterback, Running Back and Left Tackle, they would inherit the attributes and methods of the player class. Player would be the superclass or parent in the relationship and the position would be the child, or subclass.

If we change an attribute in the player class, all those connected to it will inherit the change.

In the class diagram, inheritance is shown with a solid line and hollow arrow.

A note about classes here, if we do add a bunch of positions, the player class turns into what's known as an abstract class. That's one that serves as a blueprint for others below it and provides common structure and behavior.

The way to indicate this on our NFL diagram is either <<Player>> ;or with italics, *Player*.



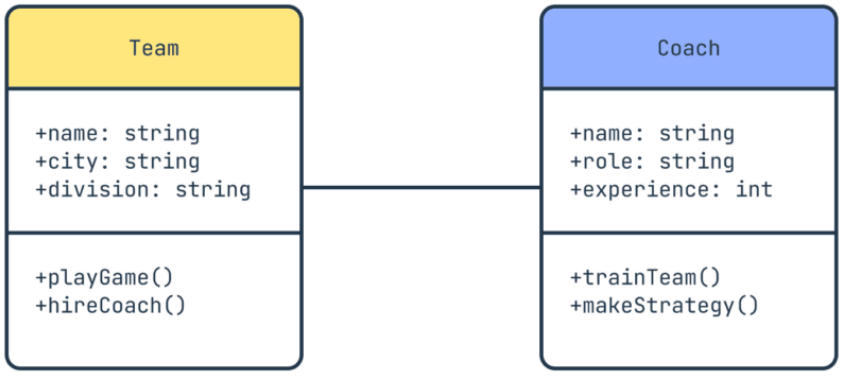
Association

Sometimes called simple association, this is a relationship that indicates one class is related to another but not necessarily dependent.

For example, in the NFL, coaches are associated with teams, teams can have multiple coaches and other teams have coaches.

We display this with a solid line connecting the associated classes.

The relationship can also be directional, if that’s the case, you can use a filled-in arrowhead with the head ending at the smaller class. So team would point and connect to player.



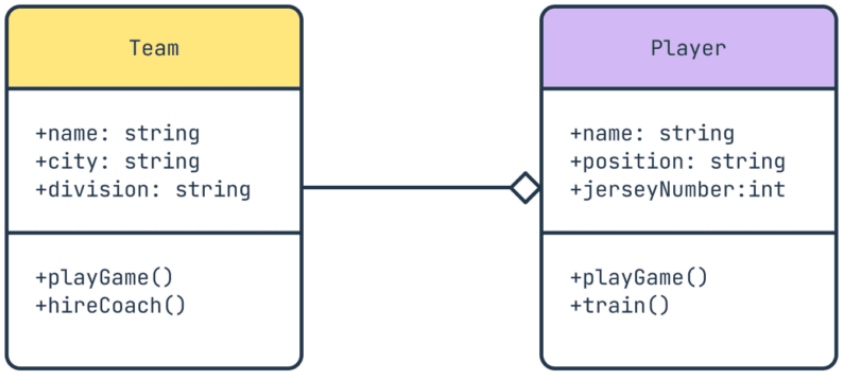
Aggregation

A special type of association, aggregation is where one class contains, or is composed of, other classes. A whole and its parts.

These parts can exist independently of the larger class and may be shared among multiple wholes.

With our football example, a team will aggregate players, but players can still exist on other teams or as free agents.

The shape of this connection is a hollow diamond attached to the "part" with a solid line going back to the "whole" or class.



Composition

On the flipside is composition, where a part cannot exist without the whole, or superclass.

Expanding the example further and adding a couple more classes; stadiums and snack bars. If the team's stadium is torn down, the snack bar can't exist.

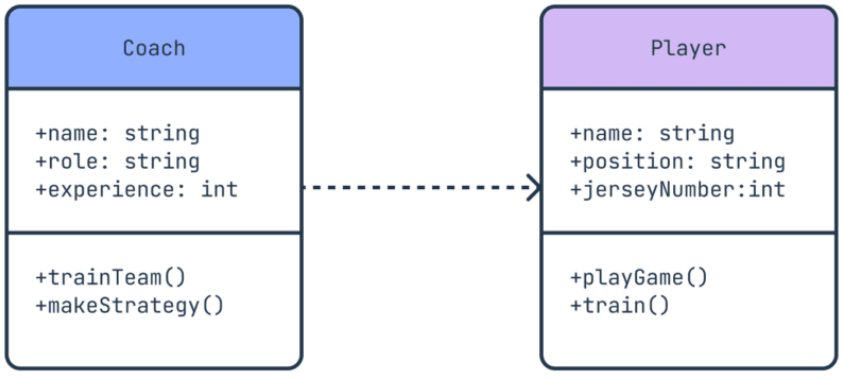
This is drawn with a solid line and filled in diamond at the superclass, or stadium in this case.

Dependency

This is a relationship where one class relies on another in some way, often through method parameters, return types or temporary associations.

Coaches and players are related in this way, where the coach depends on the performance data or behavior of players to make strategic decisions.

We'd show this with a dashed line and an open arrow.



6. Add multiplicity notation (if necessary)

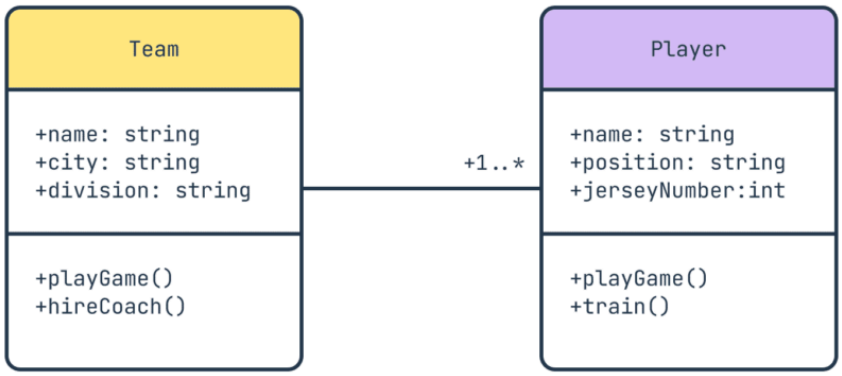
Multiplicity specifies how many instances of one class are associated with instances of another class in an association.

For our NFL example, the player and team association would have multiplicity notation to indicate there are multiple players on a team.

The way we show that on the structure diagram is as follows:

- 0..1 = zero to one
- 1 = exactly one
- 0..* = zero to many
- 1..* = one to many
- n..m = specific number range

Note: a real-world class diagram for the NFL would involve way more classes and relationships to capture the full complexity of the league, including divisions, conferences, stadiums, game schedules and oh so much more.



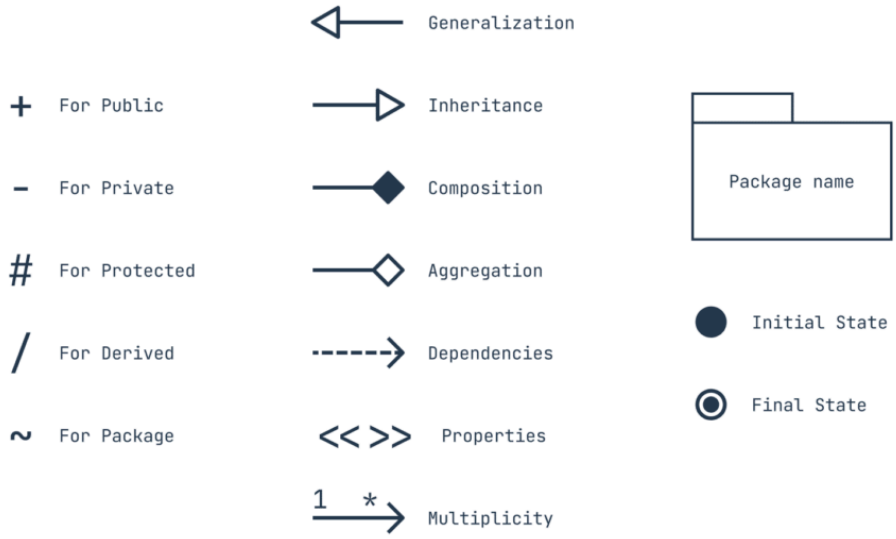
7. Review, Refine, Share, Iterate

Once you have everything in place, it’s time to do what you should do with every diagram, map or document you put together; review it and edit as needed.

The goal is to visualize your system as clearly and cleanly as possible and now you’re equipped to create your own class diagram.

UML class diagrams notations library

For reference and convenience, here's a guide of all the symbols and notation used to construct UML class diagrams:



UML diagram examples

The football example was helpful in highlighting relationships but let's look at some more concrete UML examples.

UML diagrams can be broken down into two main groups: structural diagrams and behavioral diagrams.

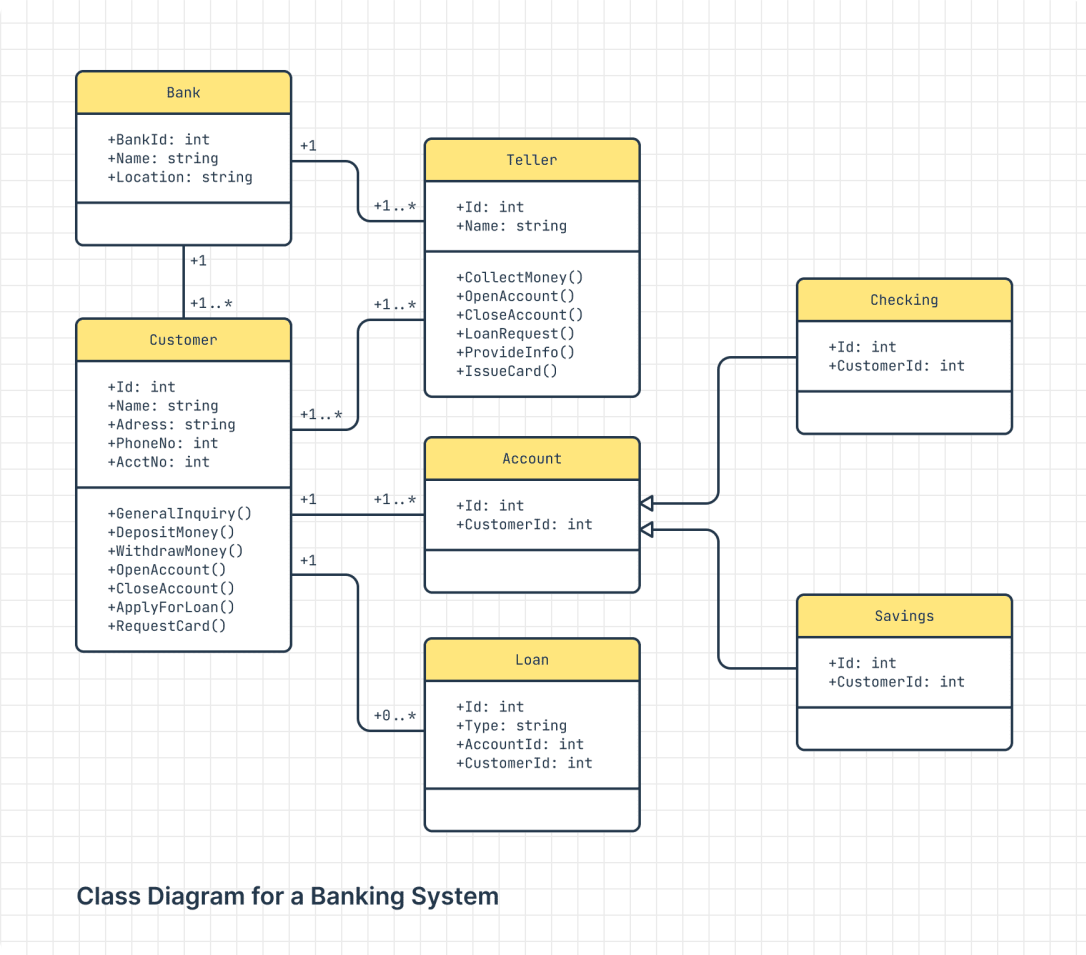
Structural diagrams

These focus on representing the static aspects of a system's architecture.

UML class diagram example

Of all the types of UML diagrams, we may as well start with a UML class

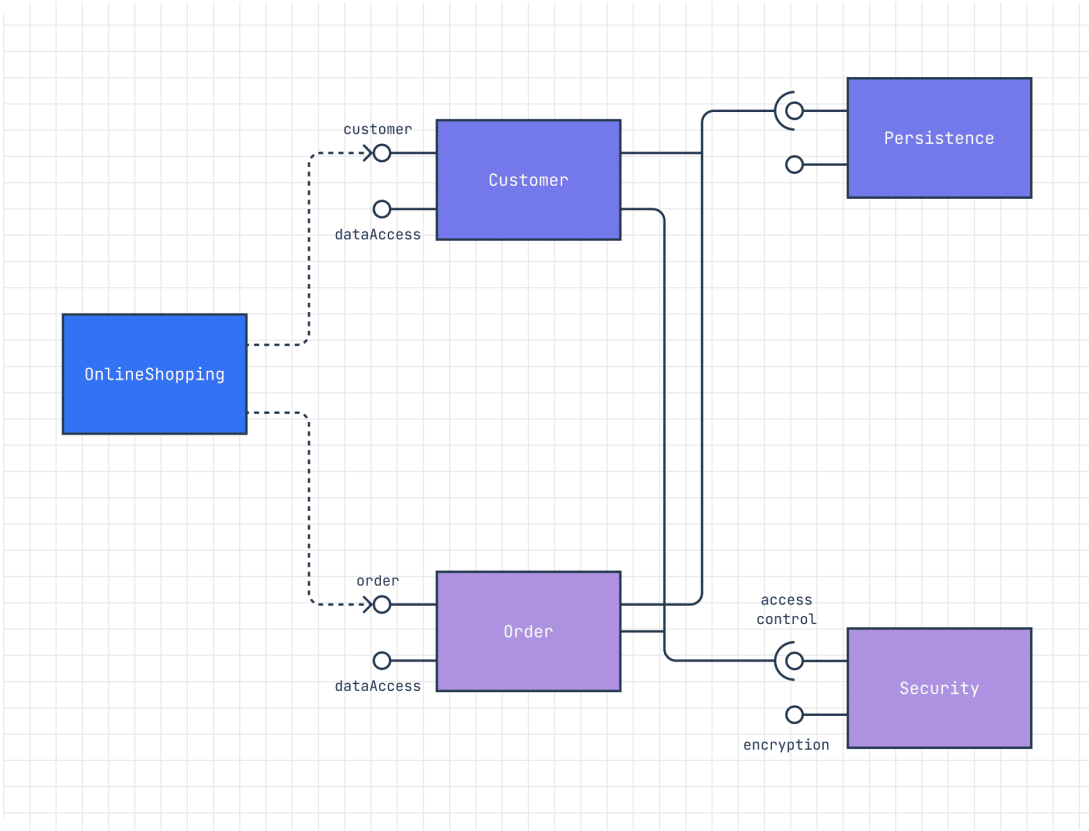
classes.



Class Diagram for a Banking System

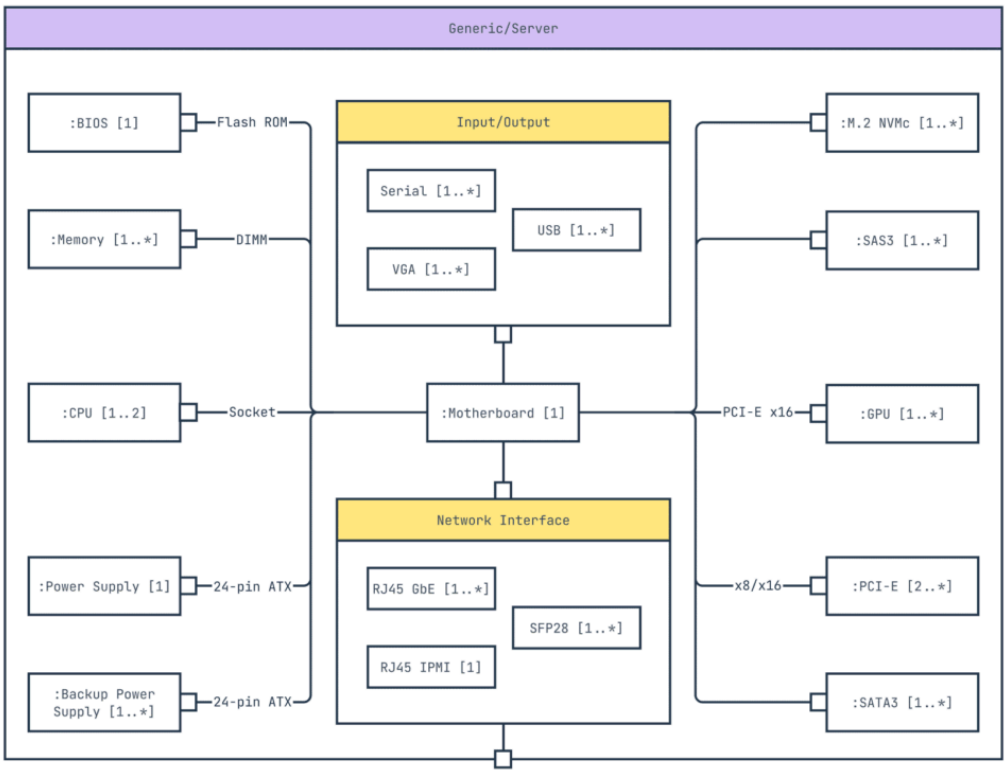
Component UML diagram example

A component diagram illustrates how physical components are wired and organized, along with their interfaces and relationships, to form a larger system. Given that, you'll note that this example UML diagram is a bit similar to the class diagram.



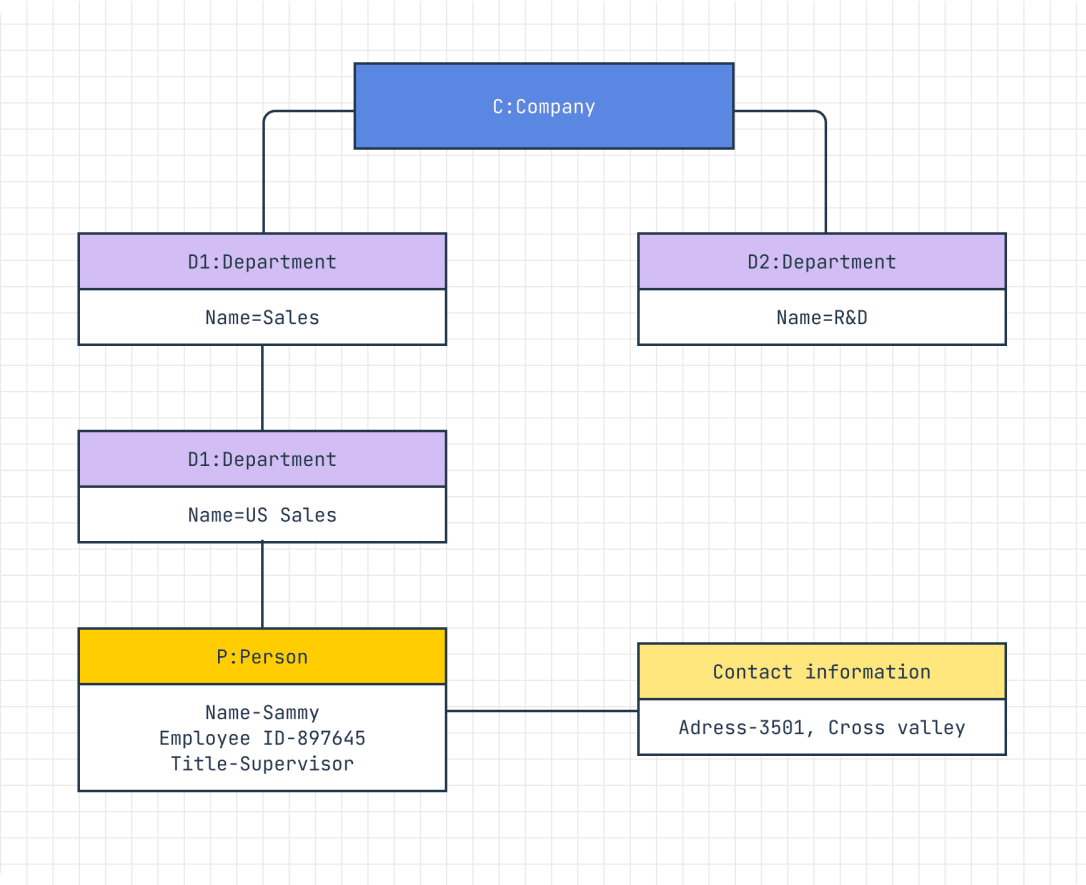
Composite structure diagram example

This diagram represents the internal structure of a class and how its parts collaborate to fulfill its responsibilities; especially useful for gaining an understanding of complex classes or components.



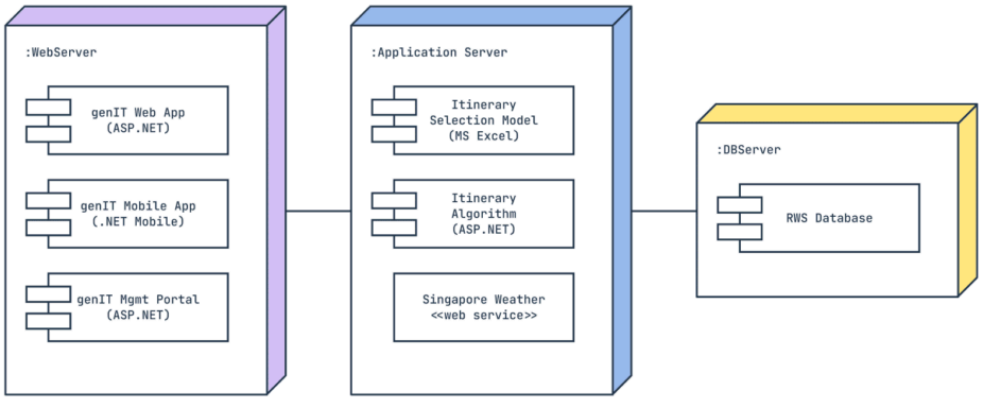
Object diagram example

An object diagram captures a snapshot of the instances of classes and their relationships at a specific point in time; great for visualizing how objects interact within a specific scenario.



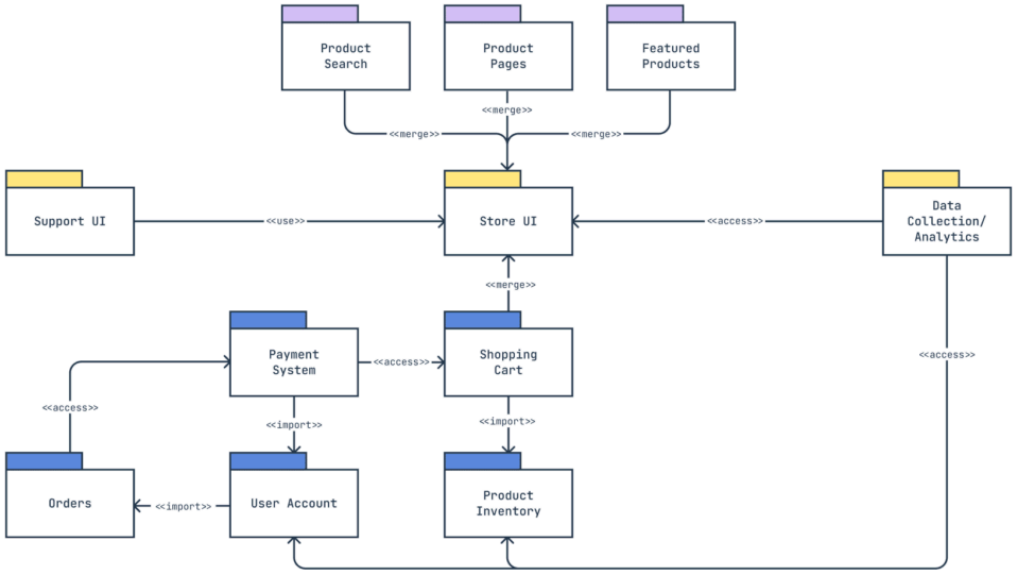
Deployment UML example

Deployment diagrams visualize the physical architecture of a system, including the hardware nodes, like servers or devices, and the software components deployed on them. Great for systems engineers or when software is deployed on many machines.



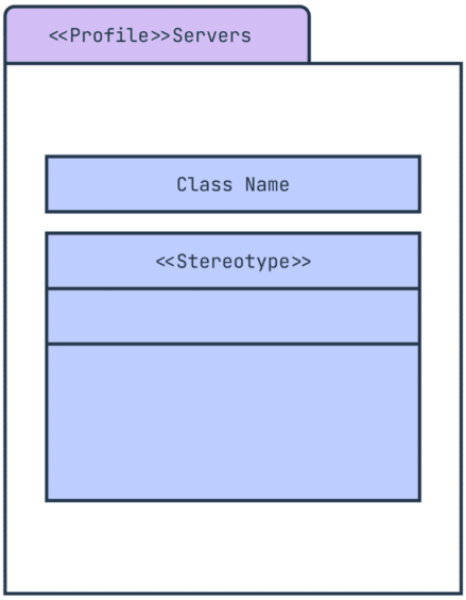
Package diagram example

Package diagrams organize elements into groups (packages), depicting the dependencies and links between packages. They provide a very high-level view of the system's organization and structure and within each package is where you'd find the classes.



Profile diagram example

Used to extend and customize UML to suit a specific domain, allowing you to define new stereotypes, tagged values and constraints.



Behavioral diagrams

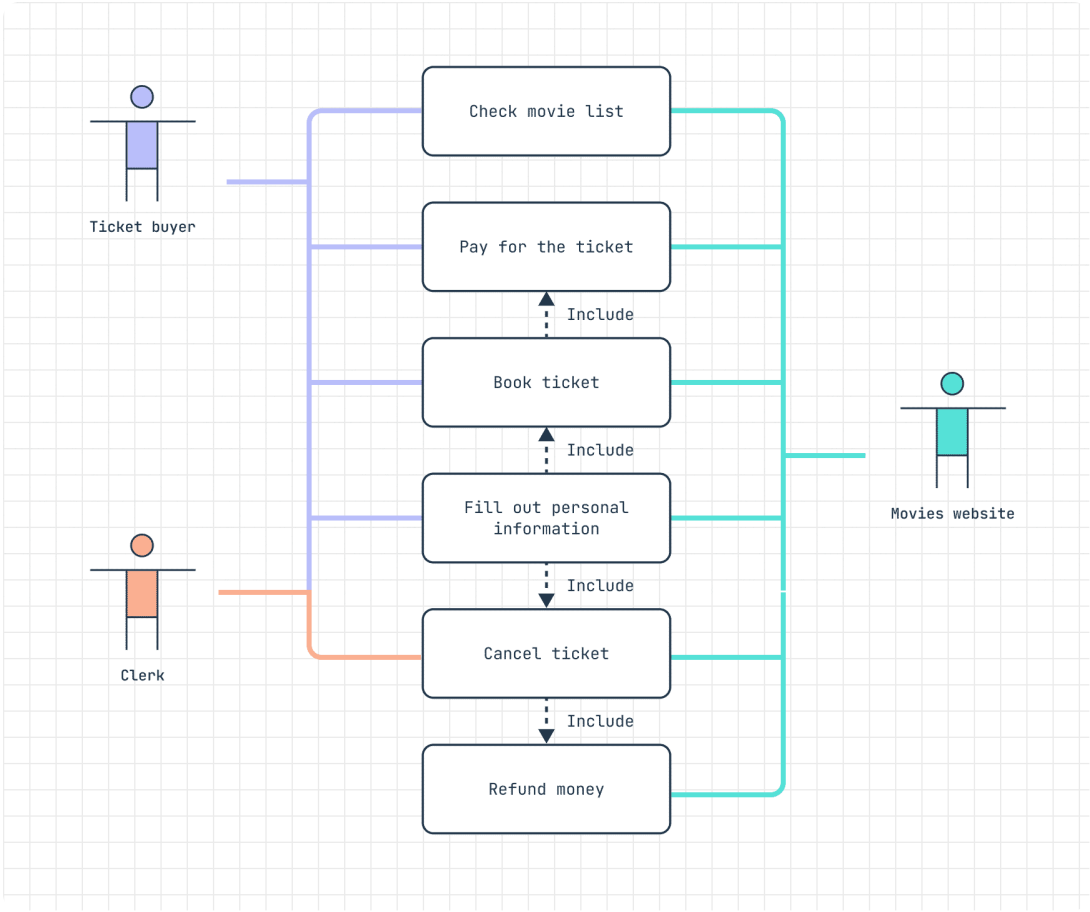
These emphasize the dynamic aspects of how a system behaves and interacts over time.

Use case diagram example

Use case diagrams depict the interactions between actors (neat little stick figures), either users or external systems, and a system. They highlight the various use cases or scenarios that the system supports, helping to define functional requirements.

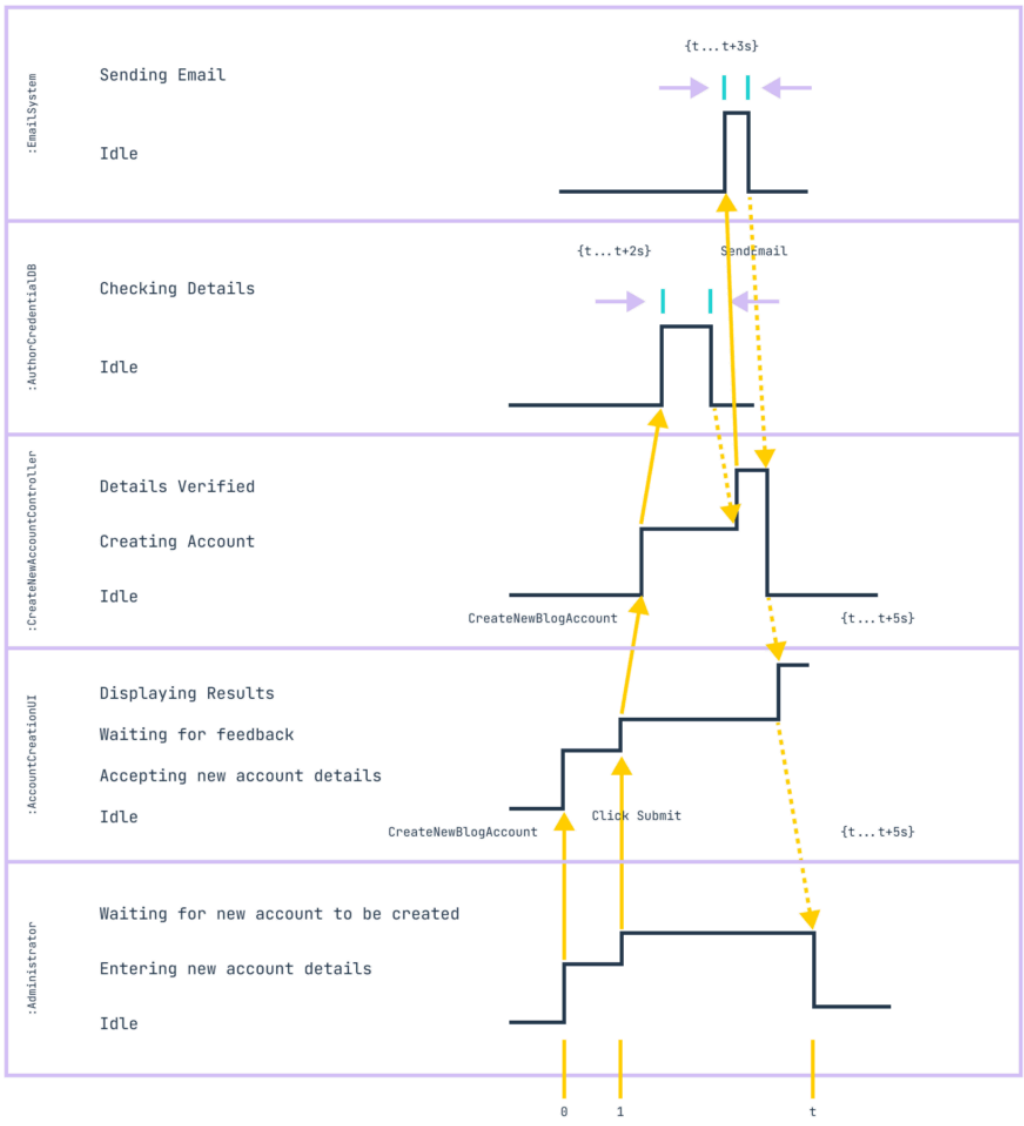
As the name suggests, being able to understand and depict use cases is huge.

You can start creating this sort of diagram now with our use case diagram template.



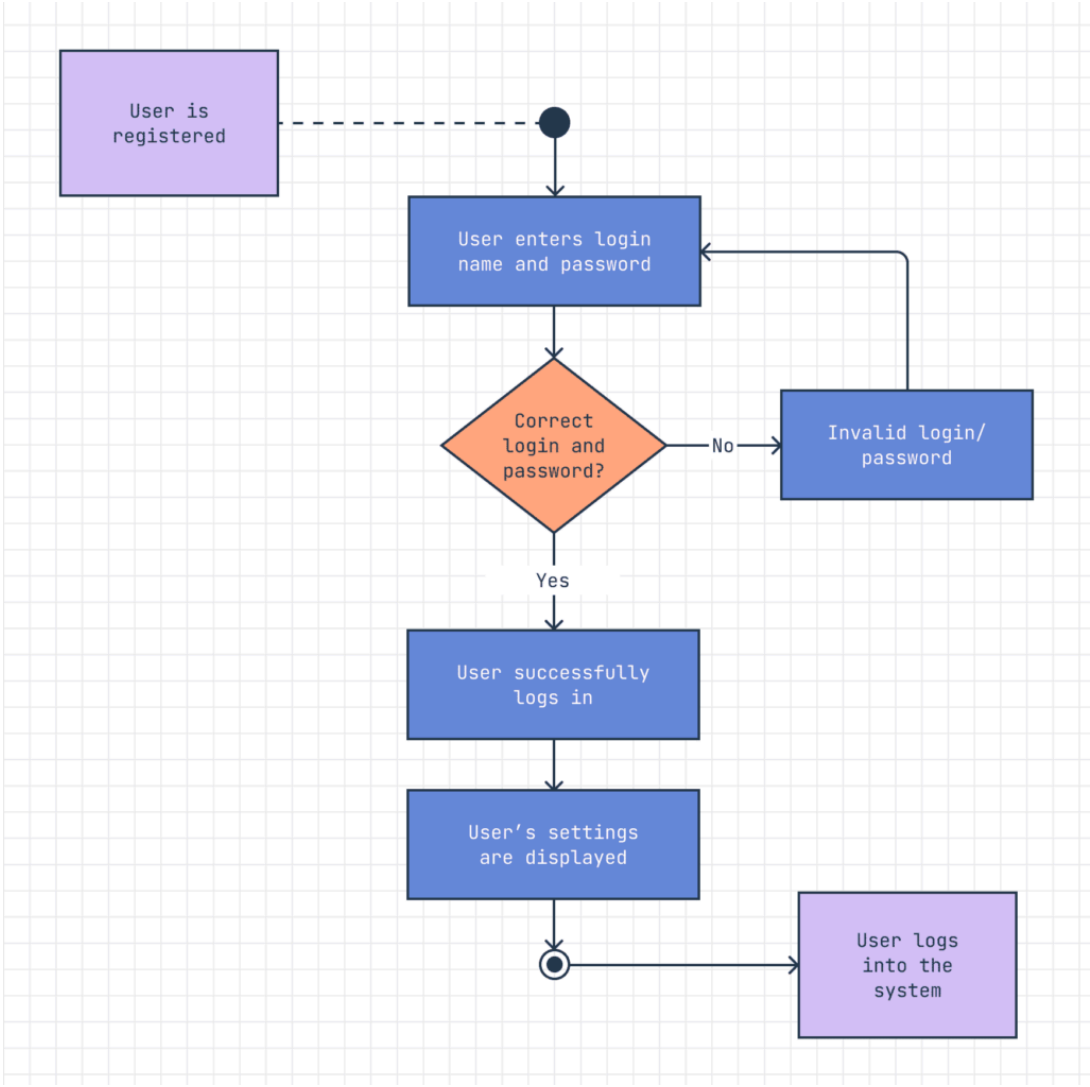
Timing diagram example

Timing diagrams show the behavior of objects in terms of sequences of states over time. They're used to model timing constraints and interactions in real-time systems. Great for pinpointing issues and areas to improve.



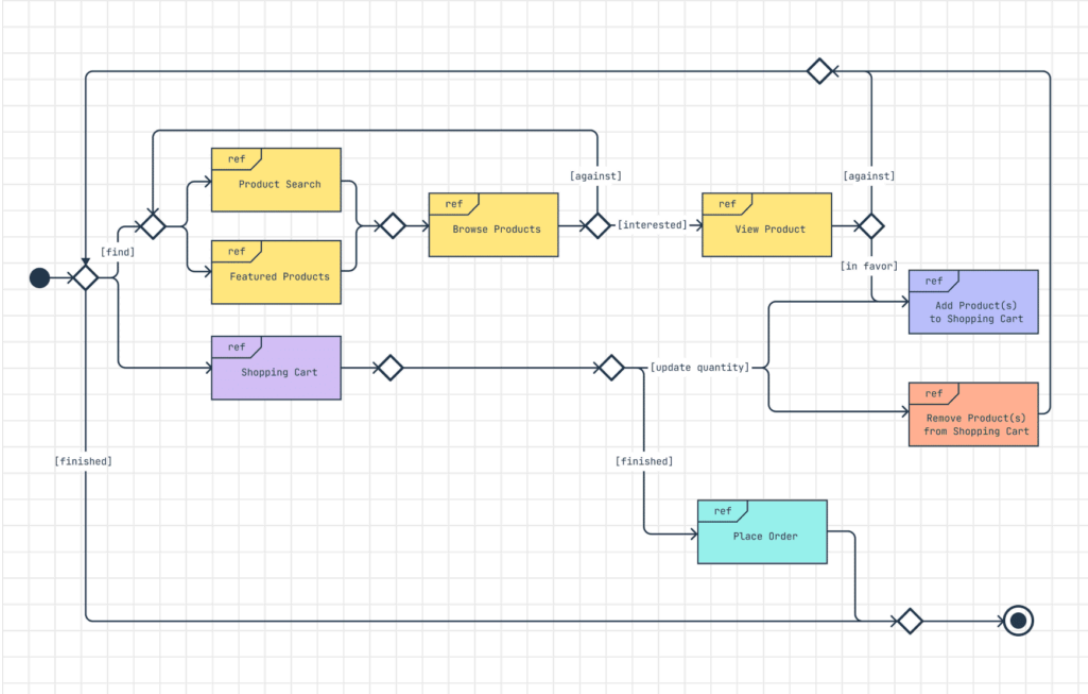
Activity diagram example

These are for visualizing workflows, processes or behavior within a system or use case. They represent the flow of activities, decisions and branching paths, helping to illustrate how tasks are performed.



State machine diagram example

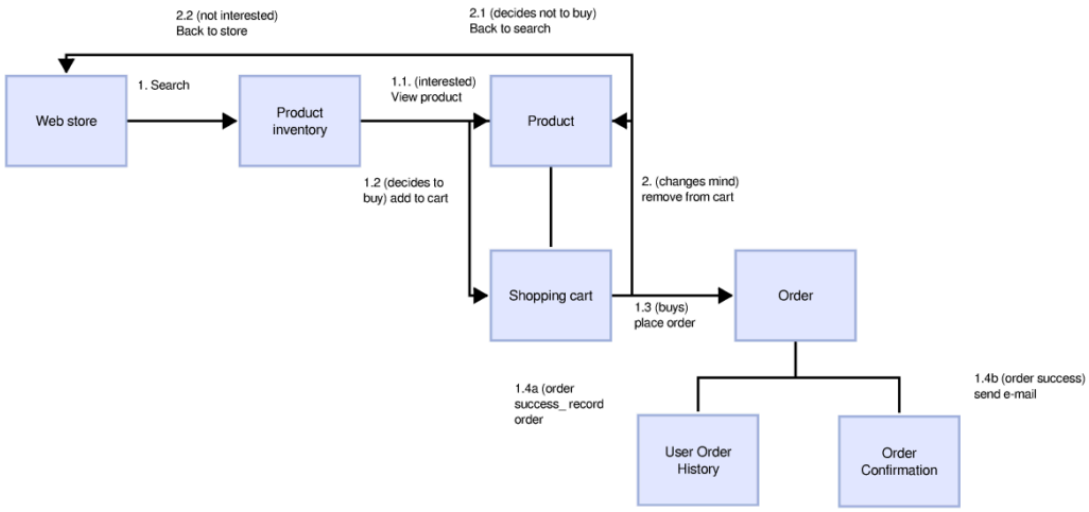
State machine diagrams depict the possible states of an object or system and the transitions between them given their state at the moment. They're useful for modeling dynamic behavior and representing state-based logic.



Communication diagram example

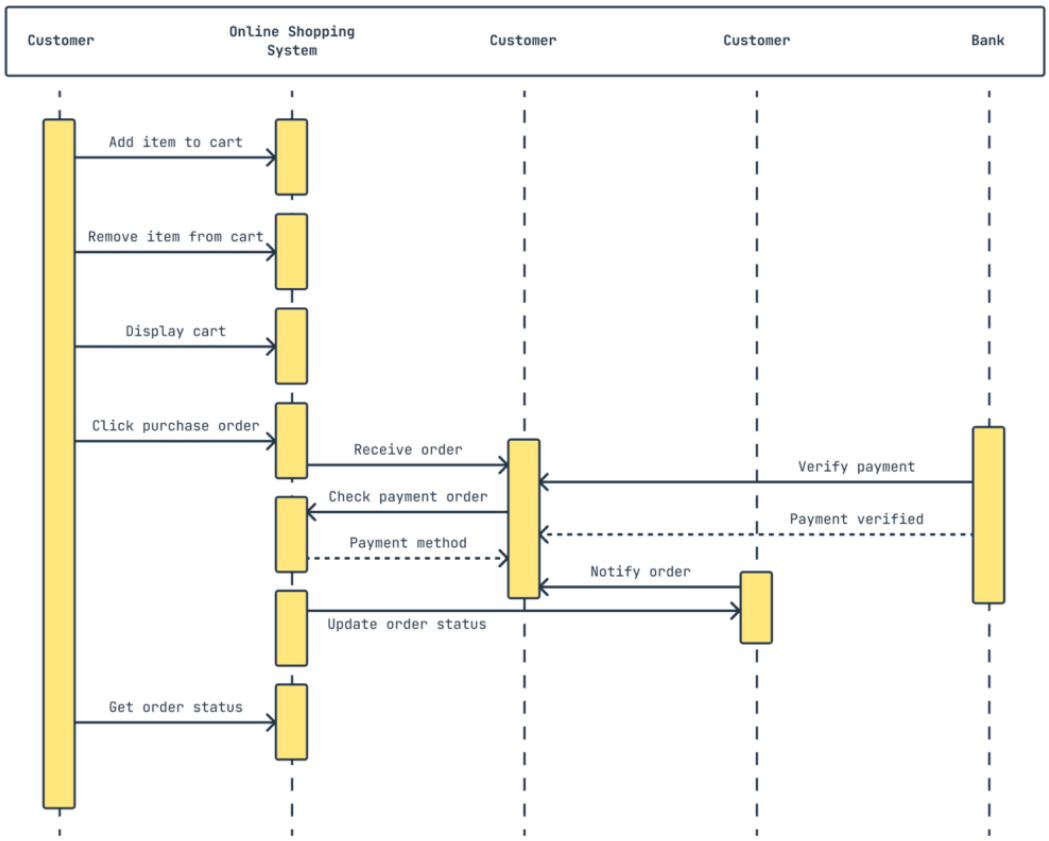
Communication diagrams focus on the interactions between objects or components, emphasizing the relationships and interactions of how a message moves rather than the specific time sequencing.

We have a UML diagram template built just for this.



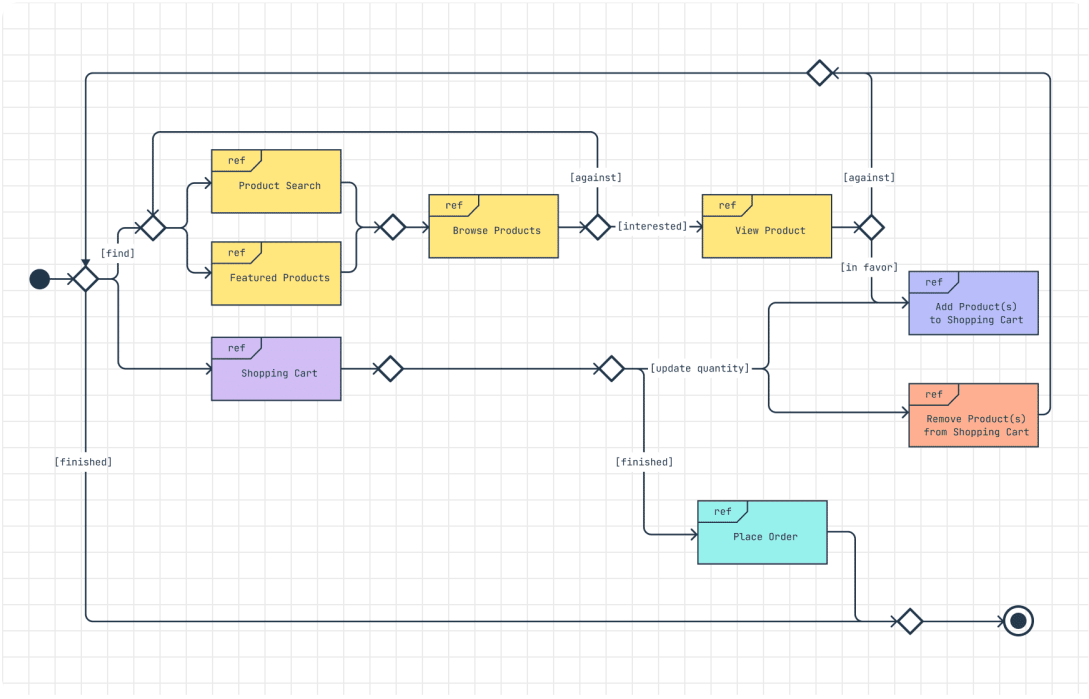
Sequence diagram example

Similar to the communication diagram, a sequence diagram is a graphical representation of the interactions between objects or components over time. They show the sequence of messages exchanged between objects and the precise order of their execution.



Interaction overview example

This one is exactly what it sounds like, Interaction overview diagrams combine elements from interaction diagrams (like sequence and activity diagrams) to provide a higher-level overview of the flow of interactions within a system.



Tips on how to draw UML diagrams that work harder for you

A few pro tips never hurt, particularly when it comes to a class of charts that can be somewhat tough.

Keep the diagram simple and focused

Avoid overcomplicating your UML diagram by including unnecessary details. Focus on the most important elements that are relevant to the system and remove any clutter or ambiguity.

Appropriate symbols and shapes

Use the standard UML symbols and shapes to represent the different elements in your diagram to ensure consistency and make it easier for others to understand your diagram.

Correct naming conventions for classes and objects

Use clear and descriptive names for your classes and objects to make it easier for readers to understand their purpose and functionality. Also, follow the naming conventions so your diagrams are universally understood.

How to make UML diagram images

After you're done designing your diagram, click Export at the top right corner of Slickplan, select Image and choose between PNG or EPS (vector), your preferred format, then add any additional options as needed.

The file is automatically downloaded to your computer and can then be shared to other devices, applications, internal servers, websites or libraries via file sharing.

How to create UML diagram PDFs

UML diagrams are an indispensable tool for understanding and explaining complex systems and processes with confidence. And no matter which type of UML diagram suits your needs, making them couldn't be easier with Slickplan's drag and drop diagram tool.

Sign up for a free 14-day trial and start creating clarity with our Diagram Maker.

Written by Steve Tsentserensky

Steve Tsentserensky is a tech copywriter focused on demystifying SaaS tools and creating clarity around complex topics like information architecture, SEO content and website planning. Find him on LinkedIn.

Browse our content:

Website Design

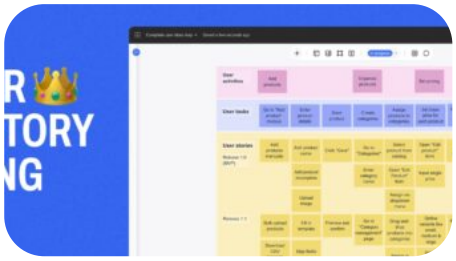
You might also like

User stories: templates, tips & how to write a user story (with examples)

User stories are short, simple descriptions of software features written from the user's perspective. They're a powerful tool for turning vague ideas into actionable plans, helping teams deliver real value…

DIAGRAMMING

USER FLOW



User story mapping: 5 steps to better product development

User story mapping is a way for Agile teams to work together visually, organizing simple descriptions of what users need, called user stories, into a map that follows the user's…

DIAGRAMMING

USER FLOW

View more

Communicate your vision with diagrams

Sign up — It's free!

Website planning simplified

Enter your email

Sign up — It's free!

14-day free trial

No credit card required

Features

Sitemap Builder

Diagram Maker

Content Planner

Design Mockups

Pricing

Company

About us

Blog

Customers

Our sitemap

Support

Help desk

Contact us

Terms & conditions

GDPR & CCPA

Developer API

Resources

Templates & examples

Information architecture

Pre-CMS planning

Integrations

Follow us

Facebook YouTube Twitter LinkedIn

Vimeo Codepen Dribbble Pinterest