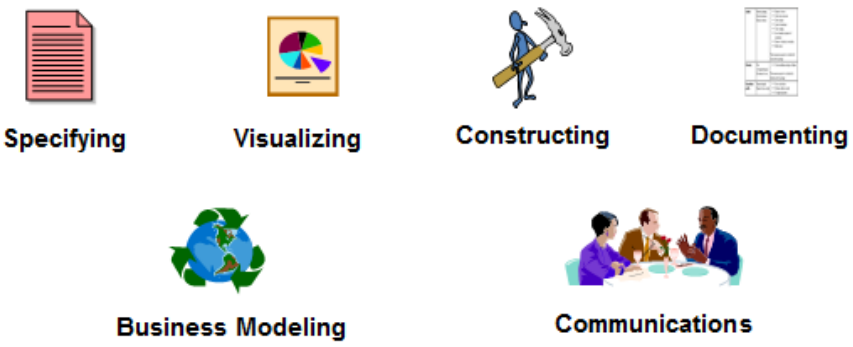
Introduction to UML
What is UML?
Motivations for UML
Types of UML diagrams
UML syntax
Descriptions of the various diagram types

What is UML?
Unified Modeling Language
UML is a modeling language to express and design documents, software

- Particularly useful for OO design
- Not a process, but some have been proposed using UML
- Independent of implementation language

Motivations for UML
UML is Open Standard, Graphical notation for
Specifying, visualizing, constructing, and documenting software systems
Language can be used from general initial design to very specific detailed design across the entire software development lifecycle
We need a modeling language to:
help develop efficient, effective and correct designs, particularly Object Oriented designs.
communicate clearly with project stakeholders (concerned parties: developers, customer, etc).
give us the "big picture" view of the project.
UML is intended to ease the task of communicating software designs.
Typical uses of UML:
conceptual component diagrams in the concept document.
use cases and class diagrams in the requirements document.
class, sequence, state, package and deployment diagrams in the architecture document.

UML Diagram
The Unified Modeling Language (UML) is a standard language for



Systems, Models and Views

A ***model*** is an abstraction describing a subset of a system

A ***view*** depicts selected aspects of a model

A ***notation*** is a set of graphical or textual rules for depicting views

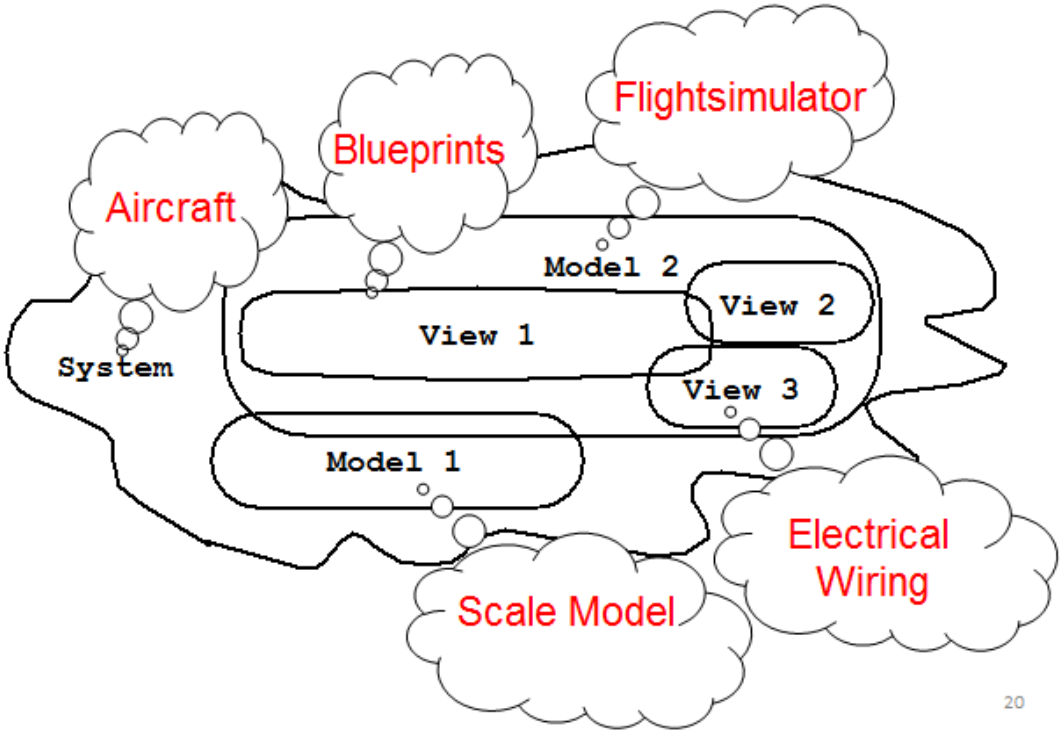
Views and models of a single system may overlap each other

Examples:

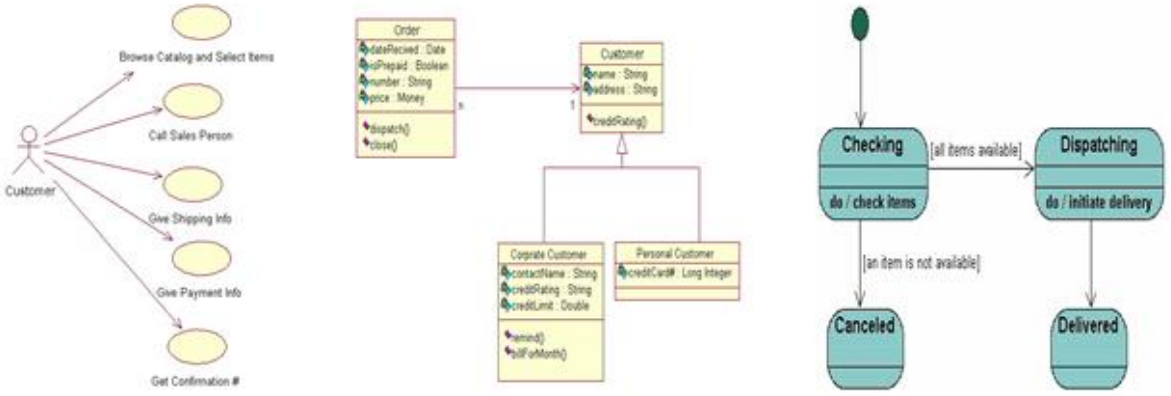
System: Aircraft

Models: Flight simulator, scale model

Views: All blueprints, electrical wiring, fuel system



Different Views



Users Designers Analyzers

UML Models, Views, Diagrams

UML is a multi-diagrammatic language

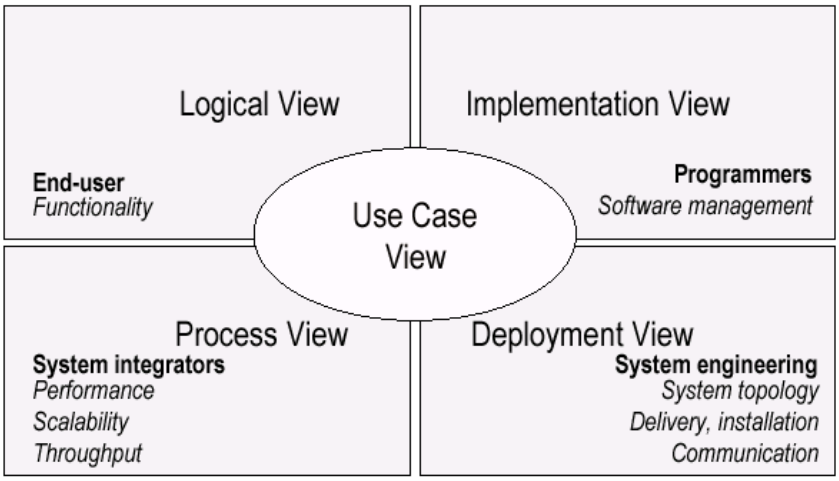
Each diagram is a view into a model

Diagram presented from the aspect of a particular stakeholder

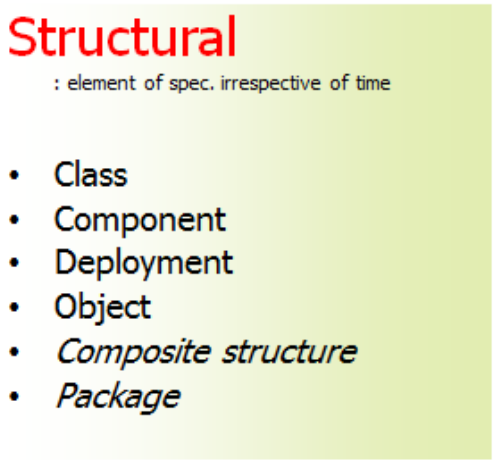
Provides a partial representation of the system

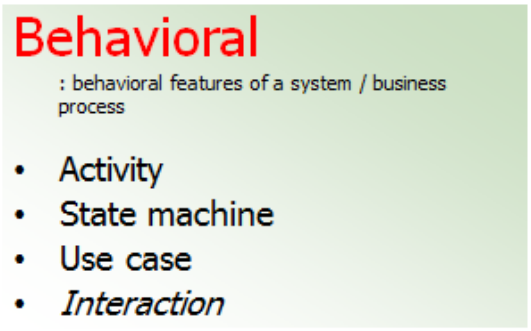
Is semantically consistent with other views

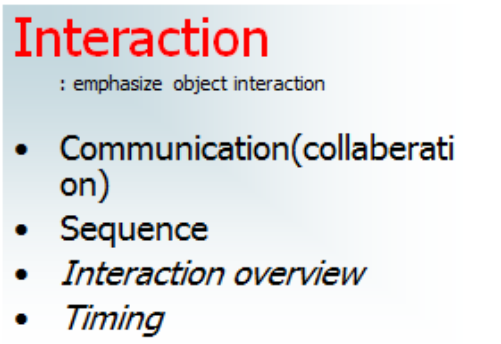
Example views



Overview of UML Diagrams







23

Types of UML diagrams

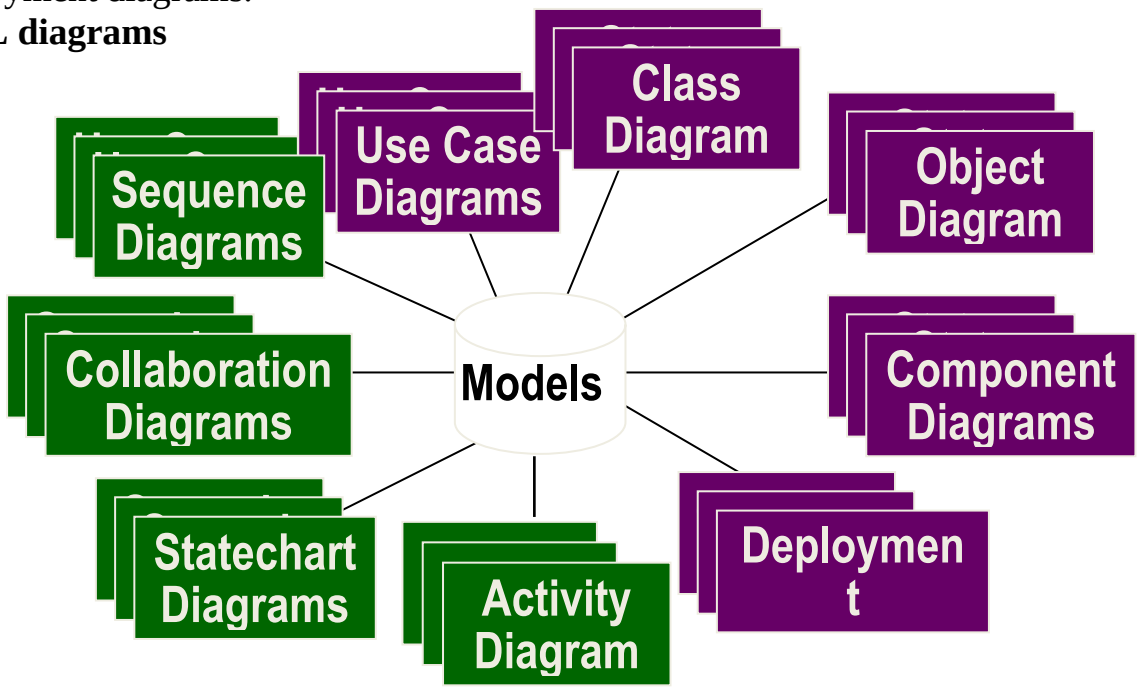
There are different types of UML diagram, each with slightly different syntax rules:

use cases.

class diagrams.

sequence diagrams.
package diagrams.
state diagrams
activity diagrams
deployment diagrams.

UML diagrams



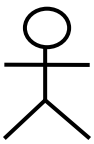
Enterprise Architect

- *Enterprise Architect is an intuitive, flexible and powerful UML analysis and design tool for building robust and maintainable software. From requirements gathering, through analysis, modelling, implementation and testing to deployment and maintenance.*
- *Enterprise Architect is a fast, feature-rich, multi-user UML modelling tool, driving the long-term success of your software project.*
- *Enterprise Architect is a complete UML-based solution for analysing, designing, managing, sharing and building software systems.*

UML syntax, 1

Actors: a UML actor indicates an external entity which communicates with the system:

User
External system
Physical environment



An actor has a unique name and an optional description.

Examples:

Passenger: A person in the train
GPS satellite: Provides the system with GPS coordinates

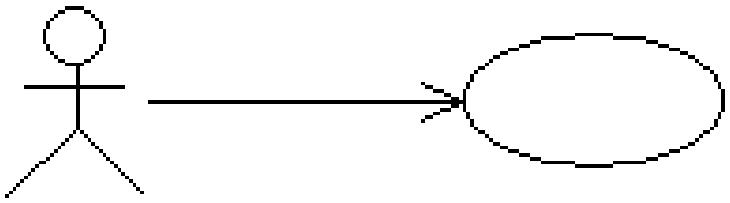
Boxes: boxes are used variously throughout UML to indicate discrete elements, groupings and containment.

Arrows: arrows indicate all manner of things, depending on which particular type of UML diagram they're in. Usually, arrows indicate flow, dependency, association or generalization.
Cardinality: applied to arrows, cardinalities show relative numerical relationships between elements in a model: 1 to 1, 1 to many, etc.
Constraints: allow notation of arbitrary constraints on model elements. Used, for example, to constrain the value of a class attribute (a piece of data).
Stereotypes: allow us to extend the semantics of UML with English. A stereotype is usually a word or short phrase that describes what a diagram element does. That is, we mark an element with a word that will remind us of a common (stereotypical) role for that sort of thing. Stereotypes should always be applied consistently (with the same intended meaning in all instances).

UML diagrams: use cases

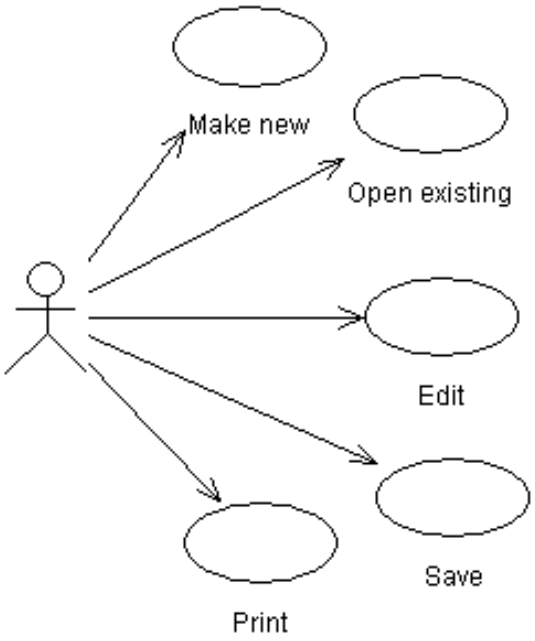
Use cases represent a sequence of interaction for a type of functionality; summary of scenarios
The use case model is the set of all use cases. It is a complete description of the functionality of the system and its environment
A use case encodes a typical user interaction with the system. In particular, it:
captures some user-visible function.
achieves some concrete goal for the user.
A complete set of use cases largely defines the requirements for your system: everything the user can see, and would like to do.
The granularity of your use cases determines the number of them (for you system). A clear design depends on showing the right level of detail.
A use case maps actors to functions. The actors need not be people.

Use case examples, 1
(High-level use case for powerpoint)

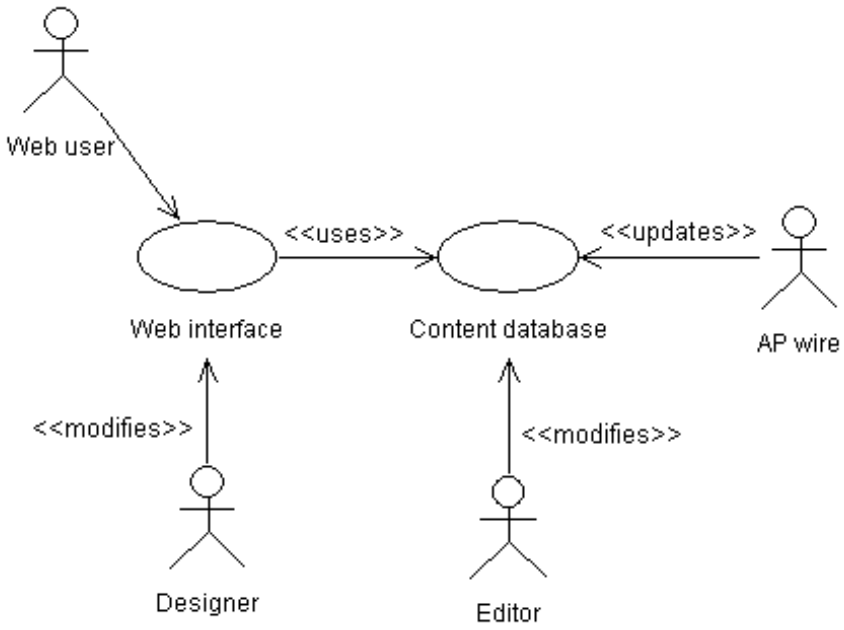


Although this is a valid use case for powerpoint, and it completely captures user interaction with powerpoint, it's too vague to be useful.

Use case examples, 2
(Finer-grained use cases for powerpoint.)



The last example gives a more useful view of powerpoint (or any similar application).
The cases are vague, but they focus your attention the key features, and would help in developing a more detailed requirements specification.
It still doesn't give enough information to characterize powerpoint, which could be specified with tens or hundreds of use cases (though doing so might not be very useful either).



The last is more complicated and realistic use case diagram. It captures several key use cases for the system.

Note the multiple actors. In particular, ‘AP wire’ is an actor, with an important interaction with the system, but is not a person (or even a computer system, necessarily).
The notes between << >> marks are *stereotypes:* identifiers added to make the diagram more informative. Here they differentiate between different roles (ie, different meanings of an arrow in this diagram).

Use Cases are useful to…

Determining requirements
New use cases often generate new requirements as the system is analyzed and the design takes shape.
Communicating with clients
Their notational simplicity makes use case diagrams a good way for developers to communicate with clients.
Generating test cases
The collection of scenarios for a use case may suggest a suite of test cases for those scenarios.

Use Case Diagrams: Summary

Use case diagrams represent external behavior
Use case diagrams are useful as an index into the use cases
Use case descriptions provide meat of model, not the use case diagrams.
All use cases need to be described for the model to be useful.