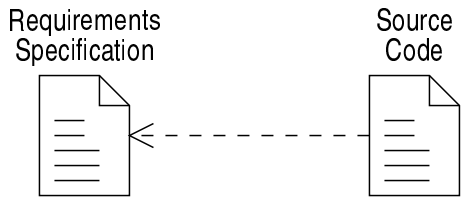
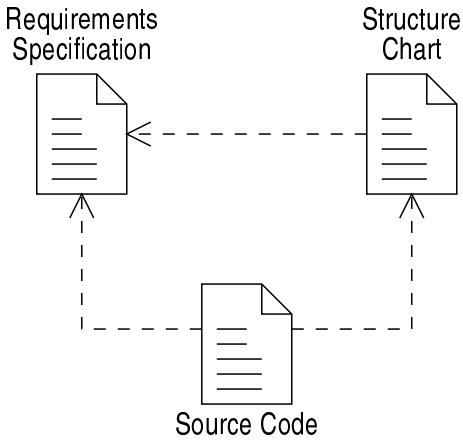
Software Development

Some simple software developments just involve developing code from a statement of requirements :



but in most cases some planning is necessary:



A structure chart gives an abstract overview of certain aspects of the structure of the system.

Documents that play this sort of a role in software development are known as *models*.

Models

Models provide:

- an abstract view of important aspects of the system
- documentation of system architecture
- a means of communication between team members

Models are used at different times in the development process:

Analysis models present information about a given application area.

Design models present information about the software system being developed.

Object-oriented concepts can be used to construct both types of model: this is often claimed to be an advantage of the object-oriented approach.

Methodologies

A methodology offers advice on how to manage the development of software. It will define:

Process The steps you should go through when developing a system.

Language The different types of models used to document the system's structure, and the diagrams used to express them.

Methodologies can be classified depending on the view they take of the basic architecture of software.

Structured Methods assume there is a global data repository manipulated by many functions; sharply differentiate analysis and design. Example: SSADM.

Object-oriented Methods unify data and associated functions in objects; recommend iterative, incremental development. Examples: UML, Shlaer-Mellor.

UML: the Unified Modeling Language

A unification of three prominent OO methods:

1. OMT (James Rumbaugh)
2. The Booch method (Grady Booch)
3. OOSE (Ivar Jacobson)

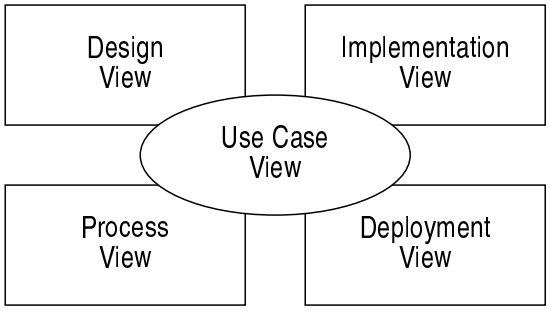
Initially developed at the Rational corporation, but submitted to the OMG (Object Management Group) for approval as a standard. UML 1.3 released in 1999, and described in "definitive" publications by the three authors above.

Unlike its predecessors, UML is *not* a methodology:

- UML is a *language*.
- It does not define a process for development.
- The "Unified Software Development Process" contains the views of the three on a development process, but does not form part of UML.

Views

The top-level structure of UML is organized around a number of *views* each of which describes a system from a different perspective.



Use Case View Describes system requirements; constrains other views.

Design View Defines logical structures supporting required functionality.

Implementation View Describes physical components of system.

Process View Deals with system's concurrency.

Deployment View Deals with physical location of components, for example across the nodes of a network.

Basic terminology

Models An abstract view of a complete system.

Model elements The ‘atoms’ out of which models are built. Things like classes, operations and relationships are model elements in UML.

Diagrams A graphical presentation of some or all of the information in a model.

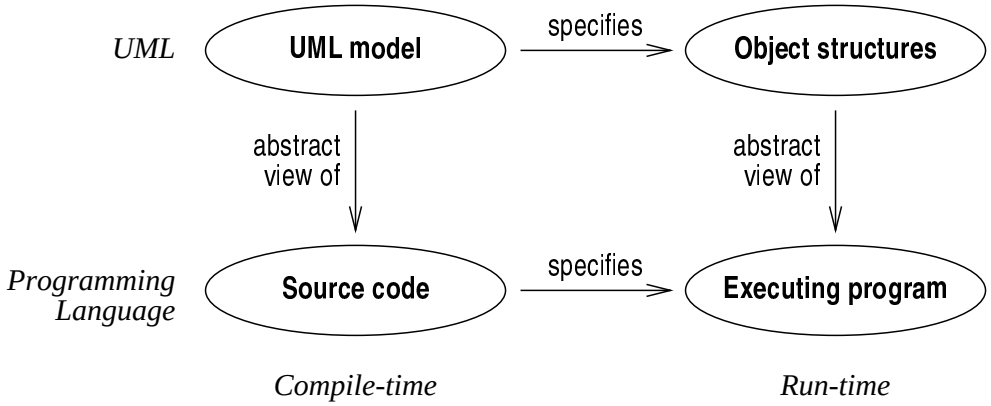
UML defines 9 types of diagram:

	Diagram	Associated view
1	Use case diagram	Use case view
2	Object diagram	Use case and design views
3	Sequence diagram	Use case and design views
4	Collaboration diagram	Use case and design views
5	Class diagram	Design view
6	Statechart diagram	Design view
7	Activity diagram	Design view
8	Component diagram	Implementation view
9	Deployment diagram	Deployment view

Design and code

There is a close relationship between UML and object-oriented programming languages.

In particular, both view a running program as consisting of a collection of *objects*. This is an abstract view of what happens when a program runs. In the same way, UML diagrams provide an abstract view of certain aspects of programs.



Making explicit the link with programs can provide a helpful way of thinking about what UML diagrams actually mean.

The Software Development Process

The following terms are often used to describe the software development process.

Linear Describes a process where a set of activities is carried out in a fixed order.

Iterative Describes a process where a set of activities is repeatedly carried out until development is complete.

Incremental Describes a process where a system's functionality is implemented in stages, or *increments*, rather than all in one go.

It is often stated that an iterative and incremental process is appropriate for object-oriented developments.

It has also been stated that a system's documentation should be presented as if a linear development had been carried out.