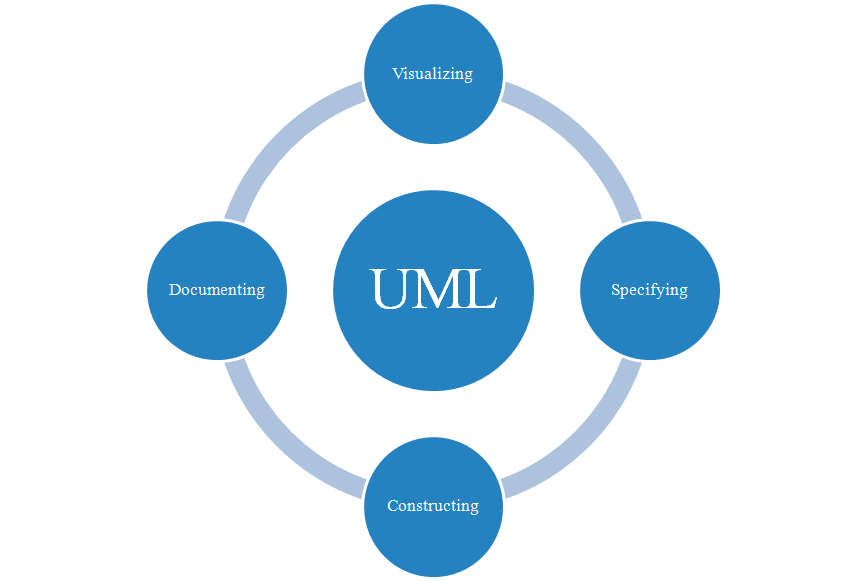
# Overview of UML:

The Unified Modeling Language (UML) is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software intensive system.

The UML is a language for

* Visualizing
* Specifying
* Constructing
* Documenting the artifacts of a software intensive system.

[](http://www.startertutorials.com/uml/wp-content/uploads/2013/08/overview-of-uml.gif)

## The UML is a Language

A language provides a vocabulary and the rules for combining words in that vocabulary for the purpose of communication. **A modeling language is a language whose vocabulary and rules focus on the conceptual and physical representation of the system.** A modeling language like UML is thus a standard language for software blueprints.

The vocabulary and rules of a language such as the UML tell you how to create and read well-formed models. But they don’t tell you what models you should create and when you should create them. That’s the role of software development process. A well-defined process will guide you in deciding what artifacts to produce, what activities and what workers to use to create them and manage them.

## UML is a Language for Visualizing

For many programmers, the distance between thinking of an implementation and then converting it out into code is close to zero. You think it, you code it. In fact, some things are best represented in code. Text is a wonderfully minimal and direct way to write expressions and algorithms.

The programmer is still doing some modeling. He or she may even sketch out a few ideas on a white board or paper. However, there are several problems with this.

*First, communicating those conceptual models to others is error-prone unless everyone involved speaks the same language.*

*Second, there are some things about a software system you can’t understand unless you build models.*

For example, the meaning of a class hierarchy can be inferred, but not directly grasped, by looking at the code for all the classes in the hierarchy.

*Third, if the developer who cut the code never wrote down the models that are in his or her head, that information would be lost forever or, at best, only partially re-creatable from the implementation, once that developer moved on.*

Writing models in UML addresses the third issue: An explicit model facilitates communication. Some things are best modeled textually; others are best modeled graphically. Indeed, in all software systems there are structures that transcend what can be represented in a programming language. The UML is such a graphical language. This addresses the second problem described earlier.

The UML is more than just a bunch of graphical symbols. Rather, behind each symbol in the UML notation is a well-defined semantics. In this manner, one developer can write a model in the UML, and another developer, can interpret that model without any errors or confusion. This addresses the first issue described earlier.

## UML is a Language for Specifying

Specifying means building models that are precise, unambiguous and complete. In particular, the UML addresses the specification of all the important analysis, design and implementation decisions that must be made in developing and deploying a software-intensive system.

**UML is a Language for Constructing**

The UML is not a visual programming language, but its models can be directly connected to a variety of programming languages. This means that it is possible to map from a model in the UML to a programming language such as Java, C++ or VB, or even to tables in a relational database or object-oriented database.

  This mapping permits forward engineering: The generation of code from a UML model into a programming language. The reverse is also possible: you can reconstruct a model from an implementation back into the UML which is known as reverse engineering.

Combining these two paths of forward code generation and reverse engineering yields round-trip engineering, meaning the ability to work with in either a graphical or a textual view, while the tools keep the two views consistent.

## UML is a Language for Documenting

A healthy software organization produces all sorts of artifacts in addition to raw executable code. These artifacts include:

* Requirements
* Architecture
* Design
* Source Code
* Project Plans
* Tests
* Prototypes
* Releases

The UML addresses the documentation of a system’s architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management.

**Where can UML be used? (Application domains of UML)**

UML can be used in domains such as:

* Enterprise information systems
* Banking and Financial services
* Telecommunications
* Transportation
* Defense
* Retail
* Medical electronics
* Scientific
* Distributed web-based services

UML is not limited to modeling software. In fact, it is expressive enough to model non-software systems, such as workflow in the legal system, the structure and behavior of a patient healthcare system and the design of hardware.