Introducing the UML Unified Modeling Language

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 only a language so it is just one part of a software development method. The UML is process independent, although optimally it should be used in a process that is use case driven, architecture-centric, iterative and incremental.

The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting

The UML is a language – a language provides a vocabulary and the rules for combining words in that vocabulary for the purpose of communications.

Modeling yields an understanding of a system.

UML is a Language for Visualizing

The programmer is doing some modeling, albeit mentally. He might even sketch out a few ideas on a white board.

Writing models in the UML – an explicit model facilitates communication. Some things are best modeled textually; others are best modeled graphically.

In an interesting system, there are structures that transcend what can be represented in a programming language.

UML is a Language for Specifying.

Specifying means building models that are precise, unambiguous and complete.

UML is a Language for Constructing.

UML is not a visual programming language, but its models can be directly connected to a variety of programming languages.

- You can map from a model in the UML to a programming language such as Java, C++, Python or Visual Basic.

This mapping permits forward engineering – the generation of code from a UML model into a programming language.

UML is a Language for Documenting

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

The UML addresses the documentation of a system's architecture and all of its details

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

Some of these artifacts are treated more or less formally than others by organizations

A conceptual model of the UML. To understand the UML, you need to form a conceptual model of the language, and this requires learning three major elements:

1. The UML's basic building blocks
2. The rules that dictate how those building blocks may be put together
3. Some common mechanisms that apply throughout the UML.

Building Blocks of the UML

The vocabulary of the UML encompasses three kinds of building blocks:

1. Things
2. Relationships
3. Diagrams

Things in the UML

There are four kinds of Things in the UML

Things are abstractions that are first class citizens in a model. Relationships tie these things together. Diagrams group interesting collections of things.

Things:

1. Structure things
2. Behavioral things
3. Grouping things
4. Annotational things

Structured Things are the nouns of UML models. These are the mostly static parts of a model, representing elements that are either conceptual or physical. Collectively, structured things are called classifiers.

Class is a description of a set of objects that share the same attributes, operations,

relationships, and semantics. A class implements one or more interfaces.

Window
origin size
open( ) close( ) move( ) display( )

Interface is a collection of operations that specify a service of a class or component. An interface, therefore, describes the externally visible behavior of that element.

The declaration of an interface looks like a class with the keyword <<interface>> above the name; attributes are not relevant, except sometimes to show constants.

An interface rarely stands alone. However, an interface provided by a class to the outside world is shown as a small circle attached to the class box by a line. An interface required by a class from some other class is shown as a small semicircle attached to the class box by a line.

<<interface>> Iwindow
open( ) close( ) move( ) display( )