UML - Lenguaje unificado de modelado

- Lenguaje de modelado para especificar o describir métodos o procesos.
- Se puede aplicar en el desarrollo de software. No solo en POO.
- No es programación, solo se diagrama la realidad de una utilización en un requerimiento.
- Desde 2004 es un estándar aprobado por la ISO. En 2012 se actualizó la norma a la versión definitiva.

Lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Ofrece un estándar para escribir un “plano” (modelo). Incluye:

- procesos
- funciones del sistema
- aspectos concretos como expresiones del lenguaje de programación, esquemas de bases de datos y compuestos reciclados.

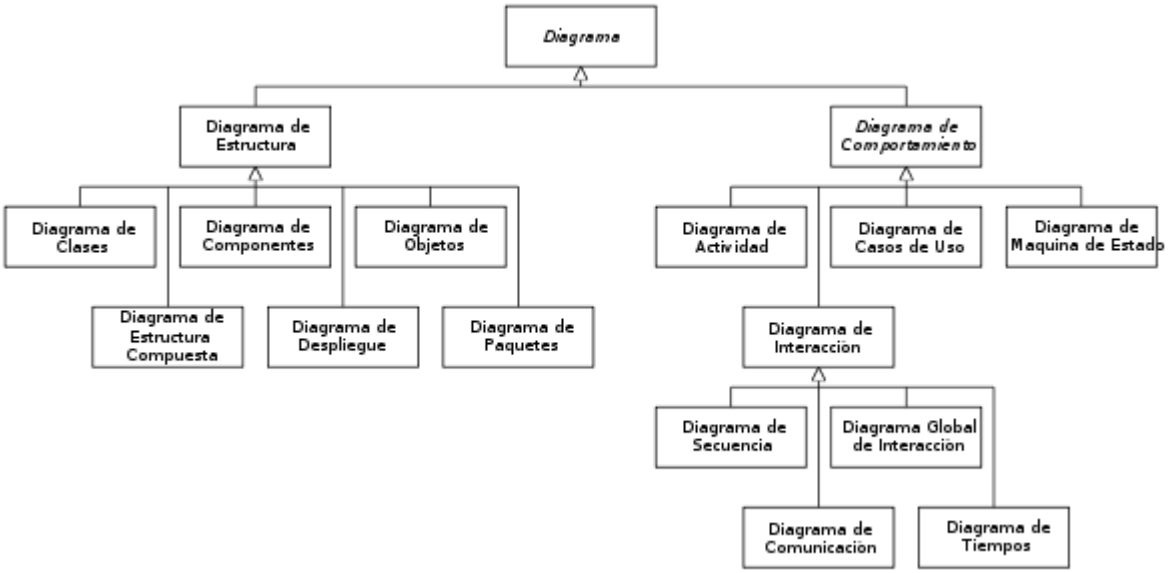
Historia: Existía OMT de Rumbaugh, Método Booch, Método de Ingeniería de Software Orientado a Objetos → Tres Amigos. En 1996 Rational concluyó que la abundancia de lenguajes de modelado estaba alentando la adopción de la tecnología de objetos, encargaron a Tres Amigos desarrollar un lenguaje unificado de modelado. El UML 1.1 fue adoptado por la OMG en 1997.

UML 1.x

- rectángulos para clases y objetos
- capacidad de especificar detalles de diseño en los niveles inferiores
- notación de casos de uso y componentes
- la integración semántica era relativamente débil

UML 2.x

- diagramas estructurales
- diagramas de comportamiento (interacciones)



TIPOS DE DIAGRAMA

Estructurales UML 1 → diagramas de **clases**: POO. Muestra las clases en un sistema, atributos y operaciones de cada clase y la relación entre cada clase. Nombre en la parte superior, atributos en el centro y operaciones o métodos en la parte inferior. Las relaciones se muestran por diferentes tipos de flechas.

Estructurales UML 2 → diagramas de **componentes**: Muestra la relación estructural de los componentes de un sistema. En sistemas complejos con muchos componentes. Los componentes se comunican entre sí mediante interfaces. Las interfaces se enlazan con conectores.

Estructurales UML 1 → diagramas de **despliegue/implementación**: Muestra el hardware de sus sistema y el software de ese hardware. ütiles cuando la solución se despliega en varios equipos.

Estructurales UML 1 → diagramas de **objetos/instancia**: muestran la relación entre los objetos como los de clase, pero usan ejemplos reales. Muestra cómo se verá un sistema en un momento dado. Para explicar relaciones complejas entre objetos.

Estructurales UML 1 → diagramas de **paquetes**: representa las dependencias entre los paquetes que componen un modelo. Agrupaciones y dependencias, suministran una descomposición de la jerarquía lógica de un sistema.

Estructurales UML 2→ diagramas de **perfiles**: se utiliza muy raramente.

Estructurales UML 2→ diagramas de **estructuras compuestas**: muestra la estructura interna de una clase.

Comportamiento UML 1 → diagrama de **actividades**: Representan flujos de trabajo de forma gráfica. Procedural and parallel behavior. Se usan como alternativa a los diagramas de máquina de estado.

Comportamiento UML 1 → diagrama de **casos de uso**: Ofrecen una visión general de los actores involucrados en un sistema, las funciones y cómo interactúan en el sistema. útil para identificar los principales actores y procesos del sistema.

Comportamiento UML 1 → diagrama de **estados**: Describir el comportamiento de los objetos que actúan de manera diferente de acuerdo con el estado en que se encuentran en el momento.

Comportamiento → diagramas de **interacción**:

- de **secuencia** (UML 1): Muestran cómo los objetos interactúan entre sí y el orden en que se producen esas interacciones en un escenario. Procesos verticalmente y flechas interacciones.
- de **comunicación/colaboracion**(UML1): similar a secuencia, pero el foco está en los mensajes pasados entre objetos.
- de **tiempos/sincronización** (UML 2): Representan el comportamiento de los objetos en un tiempo dado.

- diagrama **global de interacciones** (UML 2): Muestran una secuencia de diagramas de interacción y el orden en que suceden.

UML DISTILLED
The fundamental driver behind them all is that programming languages are not at a high enough level of abstraction to facilitate discussions about design.

Ways of Using the UML :
- sketch (communicate): highly important information. forward engineering UML → code. rough out some issues in code you are about to write. – reverse engineering code → UML: to explain how some part of a system works. Whiteboard.
- as blueprint (completeness): details. CASE and round-trip tools (forward and reverse), tripless tools (use source code). sketches are explorative, while blueprints are definitive .
- UML as programming language: when it gets sophisticated.

Understanding Legacy Code
The UML can help you figure out a gnarly bunch of unfamiliar code in a couple of ways .
Building a sketch of key facts can act as a graphical note-taking mechanism that helps you capture important information as you learn about it . Sketches of key classes in a package and their key interactions can help clarify what's going on .
A particularly nice capability is that of generating a sequence diagram to see how multiple objects collaborate in handling a complex method .