¿Qué es UML? ¿Qué diagramas componen UML?

El **Lenguaje Unificado de Modelado** o UML ("Unified Modeling Language") es un lenguaje estandarizado de modelado. Está especialmente desarrollado para ayudar a todos los intervinientes en el desarrollo y modelado de un sistema o un producto software a describir, diseñar, especificar, visualizar, construir y documentar todos los artefactos que lo componen, sirviéndose de varios tipos de diagramas.

Estos diagramas contenidos en UML son la forma más común y más utilizada de modelado de software. Modelar consiste en hacer un diseño previo de una aplicación antes de proceder a su desarrollo e implementación. De forma similar que un arquitecto dibuja planos sobre la casa que va a construir, un analista de software (u otros perfiles) crea distintos diagramas UML que sirven de base para la posterior construcción/mantenimiento del sistema. El modelado es la principal forma de visualizar el diseño de una aplicación con la finalidad de compararla con los requisitos antes de que el equipo de desarrollo comience a codificar

El modelado es vital en todo tipo de proyectos, pero cobra especialmente importancia a medida que el proyecto crece de tamaño. Para que una aplicación funcione correctamente, debe ser diseñada para permitir la **escalabilidad, la seguridad y la ejecución**. Utilizando diagramas UML se consigue visualizar y verificar los diseños de sus sistemas de software antes de que la implementación del código haga que los cambios sean difíciles y demasiado costosos.

Estos **diagramas de UML** son representaciones gráficas que muestran de forma parcial un sistema de información, bien esté siendo desarrollado o ya lo haya sido. Suelen estar acompañados de documentación que les sirve de apoyo, adoptando estas múltiples formas. Además, UML no excluye la posibilidad de mezclar diagramas, algo que, de hecho, suele ser bastante común.

Como **principal desventaja** ampliamente mencionada de UML podemos nombrar el hecho de que se trata de un lenguaje muy amplio, haciendo, en ocasiones, complicado utilizar todas las posibilidades que ofrece. No obstante, los analistas tienden a utilizar los diagramas de forma sencilla, consiguiendo que sean entendidos fácilmente por cualquier persona que accedan a ellos.

Contenido

1 ¿Por qué UML?

Los modelos o diagramas de UML nos ayudan a trabajar a un mayor nivel de abstracción. Permite modelar cualquier tipo de aplicación corriendo en cualquier combinación de hardware y software, sistema operativo, lenguaje de programación y red, es decir, UML es independiente de la plataforma hardware sobre la que actúa el software. Su flexibilidad permite modelar cualquier tipo de aplicación e, incluso, otros tipos de proyecto que no son puramente software.

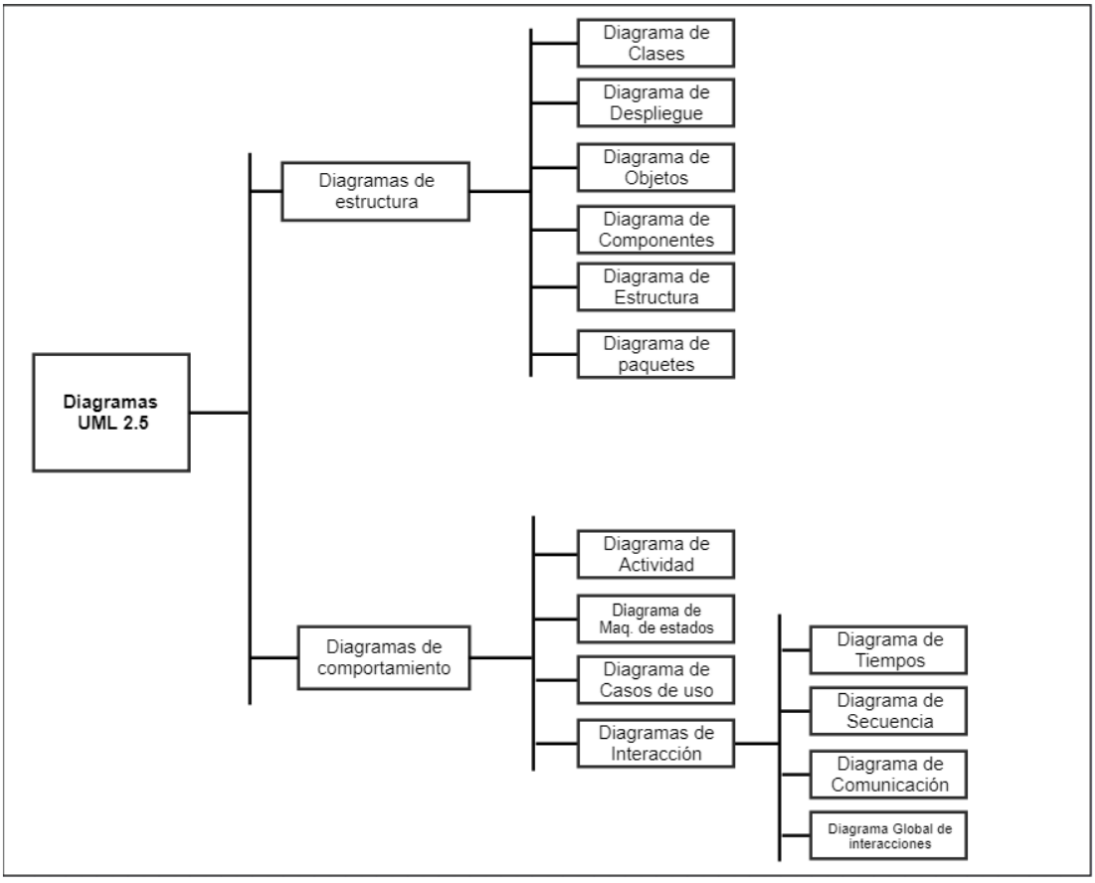
UML ofrece ese modelado utilizando diagramas y se denomina lenguaje por ser una forma común de expresarse por todos los analistas, desarrolladores y usuarios. Está desarrollado para ayudar a todos estos (y más) perfiles a especificar, visualizar, construir y documentar todos los componentes de un proyecto. A pesar de que cada diagrama UML en particular aporta su visión particular al modelado, el lenguaje en su conjunto tiene algunas características que interesa resaltar:

- Es muy **sencillo**. Pese a que si es usado de forma completa puede llegar a complicarse, lo normal es que se simplifique.
- Es capaz de modelar **todo tipo de sistemas**.
- Es un lenguaje **universal**, haciendo que todos los miembros del equipo se relacionen a través de sus diagramas sean del ámbito que sean.
- Es fácilmente **extensible**. Tiene mecanismos sencillos para especializar los conceptos fundamentales.
- Es **visual** y, por lo tanto, intuitivo.
- Es **independiente** del desarrollo, del lenguaje y de la plataforma.
- Bien ejecutado aporta un conjunto considerable de **buenas prácticas**.
- **No está completo**. Utilizando los distintos diagramas no podemos estar seguros de comprender con totalidad el sistema que va a desarrollarse. Los diagramas, para facilitar su comprensión pueden (y suelen) omitir información, pueden tener partes que se entienden de distintas maneras o, incluso, pueden tener conceptos que no pueden ser representados por ningún diagrama.

Este pequeño sitio web está dedicado a mostrar información sobre todos los tipos de diagramas que existen en UML de forma online, incluyendo teoría y ejemplos sobre los mismos.

2 Tipos de diagramas UML

A día de hoy, en la versión 2.5.1 de UML, existen dos clasificaciones de diagramas: Los **diagramas estructurales** y los **diagramas de comportamiento**. Todos los diagramas UML están contenidos en esta clasificación.



2.1.1 **Diagramas estructurales**

Los diagramas estructurales muestran la estructura estática del sistema y sus partes en diferentes niveles de abstracción. Existen un total de siete tipos de diagramas de estructura:

2.1.1.1 Diagrama de clases

Muestra la estructura del sistema, subsistema o componente utilizando clases con sus características, restricciones y relaciones: asociaciones, generalizaciones, dependencias, etc.

2.1.1.2 Diagrama de componentes

Muestra componentes y dependencias entre ellos. Este tipo de diagramas se utiliza para el desarrollo basado en componentes (CDB), para describir sistemas con arquitectura orientada a servicios (SOA).

2.1.1.3 Diagrama de despliegue

Muestra la arquitectura del sistema como despliegue (distribución) de artefactos de software.

2.1.1.4 ***Diagrama de objetos***

Un gráfico de instancias, incluyendo objetos y valores de datos. Un diagrama de objeto estático es una instancia de un diagrama de clase; muestra una instantánea del estado detallado de un sistema en un punto en el tiempo.

2.1.1.5 ***Diagrama de paquetes***

Muestra los paquetes y las relaciones entre los paquetes.

2.1.1.6 ***Diagrama de perfiles***

Diagrama UML auxiliar que permite definir estereotipos personalizados, valores etiquetados y restricciones como un mecanismo de extensión ligero al estándar UML. Los perfiles permiten adaptar el metamodelo UML para diferentes plataformas o dominios.

2.1.1.7 ***Diagrama de estructura compuesta***

Muestra la estructura interna (incluidas las partes y los conectores) de un clasificador estructurado.

2.1.2 **Diagramas de comportamiento**

A diferencia de los diagramas estructurales, muestran cómo se comporta un sistema de información de forma dinámica. Es decir, describe los cambios que sufre un sistema a través del tiempo cuando está en ejecución. Hay un total de siete diagramas de comportamiento, clasificados de la siguiente forma:

2.1.2.1 ***Diagrama de actividades***

Muestra la secuencia y las condiciones para coordinar los comportamientos de nivel inferior, en lugar de los clasificadores que poseen esos comportamientos. Estos son comúnmente llamados modelos de flujo de control y flujo de objetos.

2.1.2.2 ***Diagrama de casos de uso***

Describe un conjunto de acciones (casos de uso) que algunos sistemas o sistemas (sujetos) deben o pueden realizar en colaboración con uno o más usuarios externos del sistema (actores) para proporcionar algunos resultados observables y valiosos a los actores u otros interesados del sistema(s).

2.1.2.3 ***Diagrama de máquina de estados***

Se utiliza para modelar el comportamiento discreto a través de transiciones de estados finitos. Además de expresar el comportamiento de una parte del

sistema, las máquinas de estado también se pueden usar para expresar el protocolo de uso de parte de un sistema.

2.1.2.4 Diagramas de interacción.

Es un subconjunto de los diagramas de comportamiento. Comprende los siguientes diagramas:

2.1.2.5 ***Diagrama de secuencia***

Es el tipo más común de diagramas de interacción y se centra en el intercambio de mensajes entre líneas de vida (objetos).

2.1.2.6 ***Diagrama de comunicación***

Se enfoca en la interacción entre líneas de vida donde la arquitectura de la estructura interna y cómo esto se corresponde con el paso del mensaje es fundamental. La secuencia de mensajes se da a través de una numeración.

2.1.2.7 ***Diagrama de tiempos***

Se centran en las condiciones que cambian dentro y entre las líneas de vida a lo largo de un eje de tiempo lineal.

2.1.2.8 ***Diagrama global de interacciones***

Los diagramas global de interacciones brindan una descripción general del flujo de control donde los nodos del flujo son interacciones o usos de interacción.

3 ¿Qué versiones existen de UML?

La versión actual de UML es la 2.5.1 y fue publicada en Junio de 2015. UML es gestionada y actualizada por la OMG (Object Manabement Group).

Los creadores originales de UML son 3: Jim Rumbaugh, Grady Booch e Ivar Jacobson.

4 Breve historia de UML

Desde hace unos años, las tecnología de la información y comunicación ya han producido una enorme variedad de métodos y notaciones para llevar a cabo el modelado. Existen métodos y anotaciones para el diseño, la estructura, el procesamiento y el almacenamiento de información. De la misma manera también podemos encontrar métodos para la planificación, modelado, implementación, ensamblaje, prueba, documentación, ajuste, etc. de los

sistemas. Entre los conceptos que se utilizan existen algunos relativamente fundamentales y, debido a eso, se expanden más allá del ámbito en el que fueron creados en un principio.

Desde la concepción de la tecnología de la información hasta finales de 1970, los desarrolladores de software se tomaron el desarrollo del software como un arte. Pero estos sistemas fueron poco a poco haciéndose más complejos y por esta razón el mantenimiento y el desarrollo exigía otro tipo de visión, más allá del previamente descrito. Este hecho dio lugar a la ya famosa crisis del software.

Esta crisis lleva al enfoque de ingeniería (ingeniería de software) y la programación estructurada. Se desarrollaron métodos para la estructuración de sistemas y para los procesos de diseño, desarrollo y mantenimiento. Los enfoques orientados a procesos, por ejemplo, el método de salida de procesamiento de entrada de jerarquía, enfatizaron la funcionalidad de los sistemas. Con este método, el sistema total se divide en componentes más pequeños a través de la descomposición funcional.

La crisis del software

Al mismo tiempo, se desarrollaron enfoques orientados a la estructura de datos, como el método de Jackson, en el que la estructura del programa se deriva de la visualización gráfica de las estructuras de datos.

En todos estos métodos y notaciones, dividimos el sistema en dos partes: una sección de datos y una sección de procedimientos. Esto es claramente reconocible en lenguajes de programación más antiguos, como COBOL. Los diagramas de flujo de datos, los diagramas de estructura, los diagramas HIPO y los diagramas de Jackson se utilizan para ilustrar el rango de funciones. Naturalmente, estos primeros métodos enfatizaron el desarrollo de nuevos sistemas.

En la década de 1980, el análisis estructural clásico se desarrolló aún más. Los desarrolladores generaron diagramas de relaciones de entidades para el modelado de datos y redes de Petri para el modelado de procesos.

A medida que los sistemas se volvieron más complejos, ya no se podría diseñar cada sistema “desde cero”. Las propiedades, como la mantenibilidad y la reutilización, se hicieron cada vez más importantes. Se desarrollaron lenguajes de programación orientados a objetos, y con ellos, los primeros lenguajes de modelado orientados a objetos surgieron en los años 70 y 80. En la década de 1990, las primeras publicaciones sobre análisis orientado a objetos y diseño orientado a objetos se pusieron a disposición del público. A mediados de la década de 1990, ya existían más de 50 métodos orientados a objetos, así como muchos formatos de diseño. Un lenguaje de modelado unificado parecía indispensable.

A principios de la década de 1990, los métodos orientados a objetos de Grady Booch y James Rumbaugh se utilizaron ampliamente. En octubre de 1994, Rational Software Corporation (parte de IBM desde febrero de 2003) comenzó la creación de un lenguaje de modelado unificado. Primero, acordaron una estandarización de la notación (lenguaje), ya que esto parecía menos elaborado que la estandarización de los métodos. Al hacerlo, integraron el Método Booch de Grady Booch, la Técnica de modelado de objetos (OMT) de James Rumbaugh y la Ingeniería de software orientada a objetos (OOSE), de Ivar Jacobsen, con elementos de otros métodos y publicaron esta nueva notación bajo el nombre UML, versión 0.9 .

El objetivo no era formular una notación completamente nueva, sino adaptar, expandir y simplificar los tipos de diagramas existentes y aceptados de varios métodos orientados a objetos, como los diagramas de clase, los diagramas de casos de uso de Jacobson o los diagramas de gráficos de estado de Harel. Los medios de representación que se utilizaron en los métodos estructurados se aplicaron a UML. Por lo tanto, los diagramas de actividad de UML están, por ejemplo, influenciados por la composición de los diagramas de flujo de datos y las redes de Petri.

Lo que es sobresaliente y nuevo en UML no es su contenido, sino su estandarización a un solo lenguaje unificado con un significado definido formalmente.

Compañías conocidas, como IBM, Oracle, Microsoft, Digital, Hewlett-Packard y Unisys se incluyeron en el desarrollo posterior de UML. En 1997, la versión 1.1 de UML fue enviada y aprobada por la OMG. La versión 1.2 de UML, con adaptaciones editoriales, se lanzó en 1998, seguida de la versión 1.3 un año después, y la versión 1.5 de UML en marzo de 2003. Los desarrolladores ya habían estado trabajando en la versión 2.0 de UML desde el año 2000, y se aprobó como una Especificación final adoptada por OMG en junio de 2003. Cuando este libro se imprimió en junio de 2005, la etapa final de adopción por parte de OMG como una especificación disponible aún no se había completado.