Anexo UD4. Cosas que no sabías (o sí) de UML

La historia de UML

El lenguaje UML comenzó a gestarse en octubre de 1994, cuando Rumbaugh se unió a la compañía Rational fundada por Booch (dos reputados investigadores en el área de metodología del software).

El objetivo de ambos era unificar dos métodos que habían desarrollado: el método Booch y el OMT (Object Modelling Tool). El primer borrador apareció en octubre de 1995. En esa misma época otro reputado investigador, Jacobson, se unió a Rational y se incluyeron ideas suyas. Estas tres personas son conocidas como los "tres amigos". Además, este lenguaje se abrió a la colaboración de otras empresas para que aportaran sus ideas. Todas estas colaboraciones condujeron a la definición de la primera versión de UML.

De las tres metodologías de partida, las de Bco. y Rumbaugh pueden ser descritas como centradas en objetos, ya que sus aproximaciones se enfocan hacia el modelado de los objetos que componen el sistema, su relación y colaboración. Por otro lado, la metodología de Jacobson es más centrada a usuario, ya que todo en su método se deriva de los escenarios de uso. UML se ha ido fomentando y aceptando como estándar desde el OMG, que es también el origen de CORBA, el estándar líder en la industria para la programación de objetos distribuidos.

En 1997 UML 1.1 fue aprobada por la OMG convirtiéndose en la notación estándar de facto para el análisis y el diseño orientado a objetos. UML es el primer método en publicar un metamodelo en su propia notación, incluyendo la notación para la mayoría de la información de requisitos, análisis y diseño. Se trata pues de un metamodelo auto-referencial (cualquier lenguaje de modelado de propósito general debería ser capaz de modelarse a sí mismo).

Objetivos de UML

1. Proporcionar una notación y semánticas suficientes para poder alcanzar una gran cantidad de aspectos del modelado contemporáneo de una forma directa y económica.
2. Proporcionar las semánticas suficientes para alcanzar aspectos del modelado que son de esperar en un futuro, como por ejemplo aspectos relacionados con la tecnología de componentes, el cómputo distribuido, etc.
3. Proporcionar mecanismos de extensión de forma que proyectos concretos puedan extender el metamodelo a un coste bajo.
4. Proporcionar mecanismos de extensión de forma que aproximaciones de modelado futuras podrían desarrollarse encima del UML.
5. Proporcionar semánticas suficientes para especificar las interfaces a bibliotecas para la comparación y el almacenamiento de componentes del modelo.

6. Ser tan simple como sea posible, pero manteniendo la capacidad de modelar toda la gama de sistemas que se necesita construir.
7. UML es un lenguaje de modelado de propósito general que pueden usar todos los modeladores.
8. Debe ser un lenguaje universal, como cualquier lenguaje de propósito general.
   Imponer un estándar mundial.
9. Ser independiente del proceso de desarrollo y de los lenguajes de programación.

Los principales beneficios de UML son:

- Mejores tiempos totales de desarrollo (de 50 % o más).
- Modelar sistemas (y no sólo de software) utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.
- Alta reutilización y minimización de costos

¿Qué es una vista en UML?

Las vistas muestran diferentes aspectos del sistema modelado. Una vista no es una gráfica, pero sí una abstracción que consiste en un número de diagramas y todos esos diagramas juntos muestran una "fotografía" completa del sistema. Las vistas también ligan el lenguaje de modelado a los métodos o procesos elegidos para el desarrollo. Las diferentes vistas que UML tiene son:

- Vista Use-Case: Una vista que muestra la funcionalidad del sistema como la perciben los actores externos.

- Vista Lógica: Muestra cómo se diseña la funcionalidad dentro del sistema, en términos de la estructura estática y la conducta dinámica del sistema.

- Vista de Componentes: Muestra la organización de los componentes de código.

- Vista Concurrente: Muestra la concurrencia en el sistema, direccionando los problemas con la comunicación y sincronización que están presentes en un sistema concurrente.

- Vista de Distribución: muestra la distribución del sistema en la arquitectura física con computadoras y dispositivos llamados *nodos*.

Por si quieres investigar y saber un poco más:

1. ¿Qué es OMG (Object Management Group)?

2. ¿Quién fue James Rumbaugh y cuáles fueron sus aportaciones al terreno de la informática?