

[white paper]

UML 2.0

En este artículo nos introduciremos en el mundo del diseño de aplicaciones de software a través de su puerta más novedosa: UML 2.0.

A lo largo de estas páginas trataremos de irnos introduciendo en detalle en UML 2.0. En este caso, nos enfocaremos en el lenguaje propiamente dicho: su historia, estructura, cambios y objetivos de mayor influencia en esta nueva y radical versión.

Objetivos de UML 2.0

Al momento de desarrollar el nuevo estándar 2.0 de UML, la OMG (ver recuadro: ¿Qué es la OMG?) se propuso, entre otros, dos objetivos que podríamos considerar principales debido a su influencia en la versión final del estándar. Ellos son:

- Hacer el lenguaje de modelado mucho más extensible de lo que era.
- Permitir la validación y la ejecución de modelos creados mediante UML.

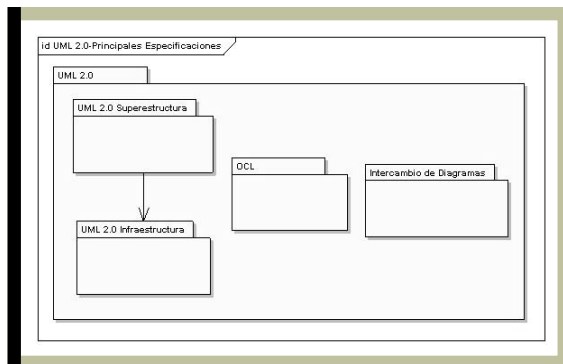
Sobre la base de estos dos objetivos, se desarrolla UML 2.0, que produce un quiebre con respecto a versiones anteriores. Para entender por qué se genera esta ruptura y por qué se da esta evolución tan marcada, profundizaremos un poco en la historia y en la definición misma de UML.

UML y la industria del software

UML se ha vuelto el estándar de facto (impuesto por la industria y los usuarios) para el modelado de aplicaciones de software. En los últimos años, su popularidad trascendió el desarrollo de software y, en la actualidad, es utilizado para modelar muchos otros dominios, como los procesos de negocios.

Conceptos básicos sobre UML

UML es la sigla de *Unified Modeling Language*, o lenguaje de modelado unificado. Para comprender de qué se trata, basta con analizar cada una de las palabras que componen su nombre:



[Figura 1] Especificaciones principales de UML 2.0.

- **Lenguaje:** UML es, precisamente, un lenguaje, lo que implica que cuenta con una sintaxis y una semántica. Por lo tanto, al modelar un concepto en UML, existen reglas sobre cómo deben agruparse los elementos del lenguaje y el significado de esta agrupación.
- **Modelado:** UML es visual. Mediante su sintaxis, se modelan distintos aspectos del mundo real, que permiten su mejor interpretación y entendimiento.
- **Unificado:** Significa que se realiza la unión entre varias técnicas de modelado en una sola.

Dado que UML proviene de técnicas orientadas a objetos, se creó con la fuerte intención de que permitiera un correcto modelado orientado a objetos.

(Muy) breve reseña histórica

Las raíces de UML están en tres métodos distintos: el de Grady Booch; la técnica de modelado de objetos, de James Rumbaugh; y “Objetory”, de Ivar Jacobson; tres personalidades conocidas como “los tres amigos”. En 1994, Booch, Rumbaugh y Jacobson dieron forma a la primera versión de UML.

En 1997, el lenguaje fue aceptado por la OMG y se lanzó la versión v1.1. Desde entonces, UML pasó por varias revisiones y refinamientos, hasta llegar a la versión actual: UML 2.0.

El nuevo enfoque de UML 2.0

En las versiones previas, se hacía un fuerte hincapié en que UML no era un lenguaje de programación, ya que un modelo creado con él no podía ejecutarse.

En UML 2.0, esta asunción cambió en forma drástica, y el lenguaje se modificó de manera tal, que permitiera capturar mucho más comportamiento (Behavior), con el fin de permitir la creación de herramientas que soportaran la automatización y la generación de código ejecutable a partir de modelos UML.

Reestructuración del lenguaje

Para lograr los objetivos enunciados, varios aspectos del lenguaje fueron reestructurados y/o modificados. La especificación se separó en cuatro partes (paquetes) bien definidas, tal como se muestra en la [Figura 1].

Es interesante destacar que UML 2.0 puede definirse a sí mismo; es decir, su estructura y organización son modelables utilizando el propio UML 2.0. De esta manera, puede usarse en un dominio distinto del de desarrollo de software. En este caso, cada paquete del diagrama representa cada una de las cuatro especificaciones que componen el lenguaje.

Veamos a continuación un poco más en detalle cada una de las principales especificaciones que componen UML 2.0. No nos explicaremos demasiado, debido a que, en futuras ediciones, habrá oportunidad de profundizar en ellas.

VALERIO ADRIAN ANACLETO
Lic. en Ciencias de la
Computación. Docente de la
FCEyN – UBA.



OCL

Es la sigla en inglés de *Object Constraint Language*, o lenguaje de restricciones de objetos. Define un lenguaje simple para escribir restricciones y expresiones sobre elementos de un modelo. Suele ser útil cuando se está especificando un dominio particular mediante UML y es necesario restringir los valores permitidos para los objetos del dominio. OCL brinda la posibilidad de definir invariantes, precondiciones, poscondiciones y restricciones en los elementos de un diagrama. Fue incorporado a UML en la versión 1.1. Originalmente, fue especificado por IBM y es un ejemplo más de las muchas herramientas agregadas a UML.

Especificación para el intercambio de diagramas

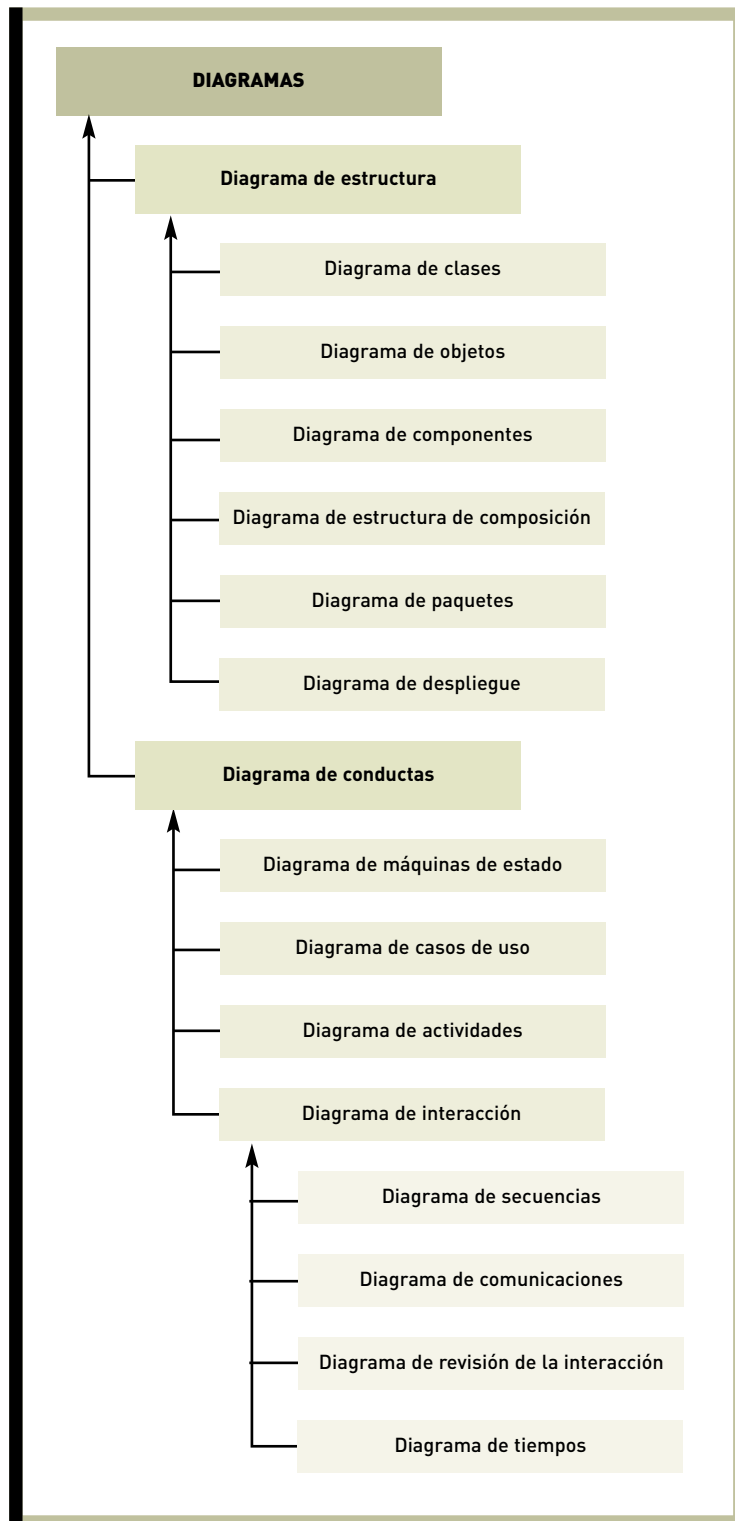
La especificación para el intercambio de diagramas fue escrita para poder compartir modelos realizados mediante UML entre diferentes herramientas de modelado.

En versiones anteriores de UML se utilizaba un Schema XML para capturar los elementos utilizados en el diagrama; pero este Schema no decía nada acerca de cómo debía graficarse el modelo. Para solucionar este problema, la nueva especificación para el intercambio de diagramas fue desarrollada utilizando un nuevo Schema XML, que permite construir una representación SVG (*Scalable Vector Graphics*).

Esta especificación se denomina XMI (*XML Metadata Interchange*) o XML de intercambio de metadata (datos que representan datos). Típicamente, es utilizada sólo por quienes desarrollan herramientas de modelado UML.

¿Qué es la OMG?

La OMG (*Object Management Group*) es una asociación sin fines de lucro formada por grandes corporaciones, muchas de ellas, de la industria del software, como IBM, Apple Computer, Sun Microsystems Inc y Hewlett Packard. Se encarga de la definición y el mantenimiento de estándares para aplicaciones de la industria de la computación. Otro de los estándares definidos por la OMG, además de UML, es CORBA, que ofrece interoperabilidad multiplataforma a nivel de objetos de negocio.



[Figura 2] Estructura taxonómica de UML 2.0.

[white paper - UML 2.0]

Descripción de los distintos tipos de diagramas		
DIAGRAMA	DESCRIPCION	PRIORIDAD
Diagrama de clases	Muestra una colección de elementos de modelado declarativo (estáticos), tales como clases, tipos, y sus contenidos y relaciones.	Alta
Diagrama de componentes	Representa los componentes de una aplicación, sistema o empresa. Además, sus relaciones, interacciones e interfaces públicas.	Media
Diagrama de estructura de composición	Representa la estructura interna de un clasificador (tal como una clase, un componente o un caso de uso), incluyendo los puntos de interacción de clasificador con otras partes del sistema.	Baja
Diagrama de despliegue físico	Muestra cómo y dónde se desplegará el sistema. Las máquinas físicas y los procesadores se representan como nodos, y la construcción interna, como nodos o artefactos embebidos. Dado que los artefactos se localizan en los nodos para modelar el despliegue del sistema, la ubicación es guiada por el uso de las especificaciones de despliegue.	Media
Diagrama de objetos	Presenta los objetos y sus relaciones en un punto del tiempo. Puede considerarse como un caso especial de un diagrama de clases o de uno de comunicaciones.	Baja
Diagrama de paquetes	Indica cómo se organizan los elementos de modelado en paquetes y las dependencias entre ellos, incluyendo importaciones y extensiones de paquetes.	Baja
Diagrama de actividades	Representa los procesos de negocios de alto nivel, incluido el flujo de datos. También puede utilizarse para modelar lógica compleja y/o paralela dentro de un sistema.	Alta
Diagrama de comunicaciones (antes: de colaboraciones)	Enfoca la interacción entre líneas de vida, donde es central la arquitectura de la estructura interna y la manera en que ella se corresponde con el pasaje de mensajes. La secuencia de los mensajes se da a través de un esquema de numerado de la secuencia.	Baja
Diagrama de revisión de la interacción	Enfoca la revisión del flujo de control, donde los nodos son interacciones u ocurrencias de interacciones. Las líneas de vida de los mensajes no aparecen en este nivel de revisión	Baja
Diagrama de secuencias	Representa una interacción poniendo el foco en la secuencia de los mensajes que se intercambian, junto con sus correspondientes ocurrencias de eventos en las líneas de vida.	Alta
Diagrama de máquinas de estado	Ilustra cómo un elemento (muchas veces, una clase) puede moverse entre estados que clasifican su comportamiento, de acuerdo con disparadores de transiciones, guardias de restricciones y otros aspectos de los diagramas de máquinas de estados que representan y explican el movimiento y el comportamiento.	Media
Diagrama de tiempos	Su propósito primario es mostrar los cambios en el estado o en la condición de una línea de vida (representando una instancia de un clasificador o un rol de un clasificador) a lo largo del tiempo lineal. El uso más común es mostrar el cambio de estado de un objeto a lo largo del tiempo, en respuesta a los eventos o a estímulos aceptados. Los eventos que se reciben se anotan a medida que indican cuándo se desea mostrar el evento que causa el cambio en la condición o en el estado.	Baja
Diagrama de casos de uso	Muestra las relaciones entre los actores y el sujeto (sistema), y los casos de uso.	Media

[Tabla1] Para quien quiera seguir profundizando en UML 2, es muy importante conocer todos estos diagramas.

Infraestructura

En la infraestructura de UML se definen los conceptos centrales y de más bajo nivel. La infraestructura es un meta-modelo (un modelo de modelos), medianamente el que se modela el resto de UML.

No es empleada por usuarios finales de UML, pero ofrece la piedra fundamental sobre la cual se define la superestructura, que es la utilizada por el común de los usuarios. La infraestructura brinda, también, varios mecanismos de extensión que hacen de UML un lenguaje configurable.

Para los usuarios normales de UML, basta con saber que la infraestructura existe y con conocer cuáles son sus objetivos.

Superestructura

La superestructura de UML es la definición formal de los elementos que lo componen. Esta definición sola contiene más de 640 páginas, y es utilizada por los desarrolladores de aplicación. Es aquella sobre la que hablan los libros y la que la mayoría conoce de versiones anteriores del lenguaje.

La superestructura de UML

Es aquí donde se definen los diagramas y los elementos que los componen. La superestructura se encuentra dividida en niveles, que se conocen como:

- **Básico (L1):** Contiene los elementos básicos de UML 2.0, entre ellos, diagramas de clases, diagramas de actividades, diagramas de interacciones y diagramas de casos de uso.
- **Intermedio (L2):** Contiene diagramas de estado, perfiles, diagramas de componentes y diagramas de despliegue.
- **Completo (L3):** Representa la especificación de UML 2.0 completa, como por ejemplo, acciones, características avanzadas y PowerTypes, entre otros.

Es importante destacar que basta con que una herramienta implemente el nivel de conformidad Básico (L1) para que se considere UML 2.0 compatible. Por eso, es normal ver una disparidad de características (*features*) bastante amplia entre dos herramientas distintas, aunque éstas sean UML 2.0 compatibles.

Organización de la superestructura

El bloque de construcción básico de UML es un diagrama. La estructura de los diagramas está reflejada por el de la [Figura 2], de acuerdo con la especificación de UML 2.0 del *Object Development Group*. Los detalles sobre estos diagramas específicos se organizan siguiendo esta estructura taxonómica, que da la perspectiva a los diagramas y a sus interrelaciones. Los diagramas de interacción comparten propiedades y atributos similares, como lo hacen los estructurales y de comportamiento. En color azul se distinguen aquellos que aparecen en esta versión de UML.

Diagramas de estructura y diagramas de comportamiento

Los diagramas estructurales representan elementos que componen un sistema o una función. Pueden reflejar las relaciones estáticas de una estructura –como lo hacen los diagramas de clases o de paquetes–, o arquitecturas en tiempo de ejecución, tales como diagramas de objetos o de estructura de composición.

Los diagramas de comportamiento representan las características de comportamiento de un sistema o proceso de negocios. Incluyen los diagramas de: actividades, casos de uso, máquinas de estados, tiempos, secuencias, repaso de interacciones y comunicaciones.

Breve descripción de los diagramas

En beneficio de quienes quieran seguir investigando dentro del mundo UML, en la [Tabla 1] se muestra la importancia que tiene para un desarrollador conocer cada una de las nuevas características de los diagramas UML 2.0. Sobre esta premisa recomendamos investigar cada tipo de diagrama, dando más importancia a los diagramas que figuran con prioridad más alta en el cuadro.

Conclusión

A lo largo del artículo hemos analizado los objetivos y el impacto de la versión 2.0 de UML. Analizamos la estructura de UML 2.0, cómo se conforma internamente y la división taxonómica de sus diagramas. ●

Bibliografía adicional

Hemos revisado varios libros dedicados a la superestructura de UML 2.0. Entre ellos, los más destacados son:

→ UML 2.0 in a Nutshell

de Dan Pilone y Neil Pitman

Excelente referencia para el trabajo diario. La introducción teórica es un poco superficial debido a que el foco del libro es la superestructura. Puede ser útil tanto para expertos de UML como para quienes recién se inician.

→ UML 2 for Dummies

de Michael J. Chonoles y James A. Schardt

Libro muy fácil de leer, con una introducción teórica adecuada. El tratamiento de los temas podría ser un poco más amplio, ya que apunta a personas que no tienen conocimiento de UML.

→ Fast Track UML 2.0

de Kendall Scott y Apress

Libro totalmente orientado a la superestructura.

La introducción conceptual teórica a los nuevos aspectos de UML 2.0 es bastante superficial y carece de un hilo conductor en cuanto a la organización de los temas. A pesar de esto, los distintos diagramas están bien explicados.