UML: Un Lenguaje de Modelo de Objetos

Luis Ignacio Lizcano Bueno
Universidad Francisco de Paula Santander
llizcano@yahoo.com

RESUMEN

Un Lenguaje Unificado de Modelado (UML: Unified Modeling Language) es una herramienta que permite modelar software orientado a objetos a través de un amplio vocabulario gráfico enfocado a la representación conceptual y física de los sistemas de software. Actualmente es un estándar adoptado por el grupo de desarrollo en objetos (OMG: Object Management Group).

En este trabajo se presenta una introducción a este lenguaje, se muestran los diferentes componentes que lo forman y se explican brevemente sus funciones.

1 INTRODUCCION

Los lenguajes de modelado orientados a objetos aparecieron en un momento entre la mitad de los setenta y finales de los ochenta cuando los metodologos, enfrentados a los nuevos lenguajes de programación orientados a objetos y a sus aplicaciones, cada vez más complejas, empezaron a experimentar con enfoques alternativos al análisis y el diseño. El número de métodos orientados a objetos se incrementa de más de 10 a 50 durante el período entre 1989 y 1994. Muchos usuarios de estos ambientes tenían el problema al intentar encontrar documentación de modelado que cubriera sus necesidades completamente, alimentando de esta manera la llamada guerra de metodologías, eran solo estudiados solo dentro de las universidades en principio como una nueva técnica de programación y luego de la Ingeniería del software. A partir de estas experiencias comenzaron a aparecer nuevas generaciones formales de metodologías, de entre las que se destacan sólo unas pocas, vale la pena mencionar la metodología de Booch [Booch 1994], la metodología OOSE (Object Oriented Software Engineering: Ingeniería del Software Orientada a Objetos) de Jacobson [Jacobson 1992] y la metodología OMT (Object Modeling Technique: Técnica de Modelado de Objetos) de Rumbaugh [Rumbaugh 1996]. Otras metodologías que merecen ser mencionadas son: Fusión, las ampliaciones a Shalaer-Merllor y de Coad-Yourdon entre otras

Cada método es completo dentro del contexto, tienen sus puntos fuertes y sus debilidades. Por ejemplo la metodología de Booch es particularmente expresiva durante las fases de diseño y construcción de proyectos, la OOSE proporcio-na un soporte excelente para los casos de uso como forma de dirigir la toma de requisitos, el análisis y el diseñó de alto nivel, y la OMT-2 es importante para el análisis y para los sistema de información con gran cantidad de datos.

Una gran cantidad de críticas comenzaron a formarse en la primera mitad de los noventa, cuando Grady Booch (de Rational Software Corporation), Ivan Jacobson (de Objectary) y James Rumbaugh (de General Electric) empezaron a adoptar ideas cada uno de las otras metodologías, las cuales habían sido reconocidas en conjunto como las tres principales metodologías de modelado de objetos a nivel mundial. Por está razón y otras razones como autores de metodologías se sintieron motivados para crear un UML:

Cuando comenzaron con la unificación, establecieron tres metas para su trabajo:

1. Que se pudieran modelaran sistemas, desde la descripción conceptual hasta los elementos ejecutables, utilizando técnicas orientadas a objetos.
2. Cubrir las cuestiones relacionadas con el tamaño inherente a los sistemas complejos y críticos.
3. Crear un lenguaje de modelado para que pueda ser utilizado tanto por las personas como por máquinas.

El esfuerzo de generar un UML comenzó en octubre de 1994, cuan-do Rumbaugh se unió a Booch en Rational. El objeto inicial del proyecto fue una unificación de los métodos de Booch y OMT. El borra-dor de la versión 0.8 (como fue llamada en ese entonces) se publi-có en octubre de 1995. En esa época Jacobson se unió a Rational y el proyecto se amplió al incorpo-rar OOSE. Los esfuerzos se plasma-ron en los documentos de la versión 0.9 en junio de 1996.

Durante 1996 se solicitó la opinión de la comunidad internacional relacionada con la Ingeniería del software. Durante este mismo tiempo se noto que muchas organizaciones de software veían en UML, como un punto estratégico a sus negocios. Se estableció un consorcio de UML con varias organizacio-

nes que querían dedicar recursos para trabajar hacia una definición consistente y completa de UML.

Las organizaciones que participaron en la versión 1.0 de UML fueron: Digital Equipment Corporation, Hewlett Packard, I-Logix, IntelliCorp, IBM, ICON Completing, MCI-Systemhome, Microsoft, Oracle, Rational, Texas Instruments y Unisys. Esta colaboración produjo un lenguaje de modelado expresivo, bien definido, potente y aplicable a un amplio espectro de dominios de problemas. UML 1.0 se ofreció para su estandarización a Object Manager Group (OMG) en enero de 1997 en respuesta a su solicitud de propuesta para un lenguaje estándar de modelado.

El grupo inicial de colaboración se amplió para incluir prácticamente el resto de organizaciones que habían enviado algunas propuestas o ideas a la propuesta inicial al OMG. UML 1.1 fue adoptado por el OMG el 14 de noviembre de 1997. La RTF (Revision Task Force) publicó una versión editorial, UML 1.2 en junio de 1998, versiones posteriores han aparecido según lo detalla el siguiente diagrama.

El siguiente diagrama muestra la evolución en el tiempo de UML.

08/01	UML 1.5	
05/01	UML 1.4	
10/98	UML 1.3	
06/98	UML 1.2	
11/97	UML 1.1	
01/97	UML 1.0	Remitido a OMG
06/96	UML 0.9	
10/95	UML 0.8	
10/95	Unified Method	UML Partner´s Expertise
90-94		Booch 93 OMT-2 OOSE Booch 91 OMT-1
80´s	Object Oriented Techniques	
70´s	Structured Analysis and Design	

Figura 1. Evolución de la Ingeniería de software

2 MODELADO DE SISTEMAS

Una empresa de software con éxito es aquella que produce de manera consistente software de calidad que satisface las necesidades de los usuarios. Una empresa que puede desarrollar ese software de forma probable y persistente con un uso eficiente y efectivo de recursos, tiene un negocio sostenible.

El modelado es una parte central de todas las actividades que conducen a la producción de un buen software. Se construyen modelos para comunicar la estructura deseada y el comportamiento de un sistema. Se construyen modelos para visualizar y controlar mejor el sistema a construir, muchas veces descubriendo oportunidades para la simplificación y la reutilización. Se construyen modelos para controlar el riesgo.

2.1 PRINCIPIOS DE MODELADO

El uso del modelado tiene una historia interesante en todas las disciplinas de Ingeniería. Esa experiencia sugiere cuatro principios básicos del modelado.

Primero: La elección de los modelos a crear tiene una profunda influencia sobre cómo se acomete un problema y cómo se le da forma a una solución.

Segundo: Todo modelo es expresado a distintos niveles de precisión.

Tercero: Los mejores modelos están ligados a la realidad.

Cuarto: Un único modelo no es suficiente. Cualquier sistema no trivial es abordado mejor a través de modelos independientes entre sí.

3 MODELADO ORIENTADO A OBJETOS

Los Ingenieros Civiles construyen muchos tipos de modelos. Lo más frecuente es que utilicen modelos estructurales que les ayuden a visualizar y especificarlas apartes de los sistemas y la forma en que esas estructuras se relacionan entre sí.

Dependiendo de las cuestiones más importantes del sistema o de la Ingeniería que les preocupan, los Ingenieros podrían construir mode-

los dinámicos. Por ejemplo, para ayudarle a estudiar el comportamiento de un estructura en presencia de un terremoto. Cada tipo de modelo se organiza de manera diferente y cada uno tiene su propio enfoque.

En el software hay varias formas de enfocar un modelo. Las formas más comunes son la perspectiva orientada a objetos y la perspectiva algorítmica.

La visión tradicional del desarrollo del software es la perspectiva algorítmica. En este enfoque, el bloque principal de construcción de todo el software es el procedimiento o función. Esta visión conduce a los desarrolladores a centrarse en las cuestiones de control y descomposición de algoritmos grandes en otros pequeños. No hay nada inherentemente malo en este punto de vista, salvo que tiende a producir sistemas frágiles y monolíticos. Cuando los requisitos cambien (¡y lo harán!) y el sistema crece (¡y lo hará!) los sistemas construidos con un enfoque algorítmico se vuelven muy difícil de mantener.

La visión actual de desarrollo del software tiene una perspectiva orientada a objetos. En este enfoque, el principal bloque de construcción de todos los sistemas software es el objeto o la clase. Un objeto es un elemento extraído del vocabulario del espacio del problema o del dominio concreto en cuestión; una clase es la descripción de un conjunto de objetos similares. Todo objeto tiene una identidad (puede nombrarse o distinguirse de otros objetos), estado (generalmente hay algunos datos que se asocian a él) y comportamiento (se le pueden hacer cosas al objeto y él a su vez hacer cosas a otros objetos).

El desarrollo orientado a objetos proporciona la base fundamental para ensamblar sistemas a partir de sus componentes, utilizando tecnologías como Java Beans, COM+ y otras plataformas de programación de actualidad. Visualizar, especificar, construir y documentar sistemas orientados a objetos es exactamente el propósito de UML.

4 UML

UML está pensado principalmente para sistemas con gran cantidad de software`[Booch y Otros 1999]. Ha sido utilizado de forma efectiva en domi-nios como:

- Sistemas de Información de empresa.
- Bancos y servicios financieros.
- Telecomunicaciones
- Transporte
- Defensa/Industria aeroespacial
- Comercio
- Electrónica médica
- Ambiente Científico
- Servicios distribuidos basados en la Web.

UML no está limitado al modelado de software. De hecho, es lo suficiente expresivo para moldear sistemas que no son software, como flujos de trabajo en el sistema jurídico, comportamiento de un sistema de vigilancia médica de un enfermo, protocolos sanitarios y el diseño de hardware.

4.1 MODELO ESTRUCTURAL BASICO

El vocabulario de UML [Rumbaugh y Otros 1999]. incluye tres clases de bloque de construcción:

- Elementos
- Relaciones
- Diagramas

4.1.1 Elementos en UML

Hay cuatro tipos de elementos.

- Elementos Estructurales
- Elementos de Comportamiento.
- Elementos de Agrupación.
- Elementos de Anotación

Los elementos estructurales son las partes estáticas de un modelo y representan objetos conceptuales o concretos del dominio, existen siete distintos tipos de elementos estructurales.

- Clase, descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.
- Interfaz, es una colección de operaciones que especifican un servicio de una clase o componente. Describe el comportamiento visible externamente de una clase.
- Colaboración, define una interacción y es una sociedad de roles y otros elementos que colaboran un comportamiento cooperativo mayor que la suma de los comportamientos de sus elementos.
- Caso de uso, es una descripción de un conjunto de seriación de acciones que un sistema ejecuta y que produce un resultado observable de interés para un actor particular.
- Clase activa, es una clase cuyos objetos tienen uno o más procesos o hilos de ejecución y por lo tanto pueden dar origen a actividades de control.
- Componente es una parte física y reemplazable de un sistema que forma un conjunto de interfaces y proporciona la implementación de dicho conjunto.
- Nodo, es un elemento físico que existe en tiempo de ejecución y representa un recurso computacional que por lo general dispone de algo de memoria y con frecuencia capacidad de procesamiento. Un conjunto de componentes puede residir en un nodo y pueden también migrar de un nodo a otros.

Existen otras variaciones de estos elementos, pero estos son los más importantes.

Los elementos de comportamiento son las partes dinámicas de los modelos UML.

- Interacción, es un comportamiento que comprende un conjunto de mensajes intercambiables entre un conjunto de objetos, dentro de un contexto particular para alcanzar un propósito específico.
- Máquina de estados, es un comportamiento que específica las secuencias de estados por los que pasa un objeto o una interacción durante su vida en respuesta a eventos, unto con sus reacciones a estos eventos.

Los elementos de agrupación son las partes organizativas de los modelos UML.

- Paquete, es el elemento de agrupación principal. Es un mecanismo de propósito general para organizar elementos en grupos.

Los elementos de anotación son las partes explicativas de los modelos UML. Son comentarios que se pueden aplicar para describir, clasificar y hacer observaciones sobre cualquier elemento de un modelo. El principal elemento de anotación es la nota.

4.1.2 Relaciones en UML

Hay cuatro tipos de relaciones

1. Dependencia
2. Asociación
3. Generación
4. Realización

Una dependencia es una relación semántica entre dos clases en la cual un cambio de un elemento (independiente) puede afectar la semántica de otro (dependiente). Una asociación es una relación estructural que describe un conjunto de ligas, las cuales representan conexiones a través de objetos. La agregación es una clase especial de asociación que representa una relación de estructura entre un conjunto y sus partes. La generalización es una relación de especialización/generalización en la cual los objetos de un elemento especializado (hijos) son consistentes con los objetos de un elemento generalizable (el padre). De esta forma, los hijos comparten la estructura y comportamiento del padre. Una realización es una relación semántica entre clasificadores, en donde un clasificador especifica un contrato que otro clasificador garantiza llevar a cabo. Se pueden encontrar realizaciones en dos partes: entre interfaces y las clases o componentes que las realizan, y entre casos de uso y las colaboraciones que los realizan.

4.1.3 Diagramas en UML

Un diagrama es la representación gráfica de un conjunto de elementos conectados entre sí. Estos diagramas son en forma de grafos conectados donde los vértices representan elementos y los arcos relaciones.

Los diagramas sirven para visualizar

un sistema desde diferentes perspectivas.

Un mismo elemento puede aparecer en varios diagramas, en sólo algunos o en ninguno. En teoría, un diagrama puede contener cualquier combinación de elementos y relaciones, sin embargo en la práctica es mejor tener un número reducido de combinaciones.

Estas relaciones son los bloques básicos de construcción para las relaciones en UML [Jacobson y Otros 1999].

1. Diagrama de clases
2. Diagrama de objetos
3. Diagrama de casos de uso
4. Diagrama de secuencias
5. Diagrama de colaboración
6. Diagrama de estados
7. Diagrama de actividades
8. Diagrama de componentes
9. Diagrama de despliegue

Los diagramas de clases muestran la vista estática de un sistema a través de un conjunto de clases, interfaces y colaboraciones junto con sus relaciones.

Un diagrama de objetos muestra un conjunto de objetos y sus relaciones. Representan un instante de la instancia de los elementos encontrados en el diagrama de clases.

Un diagrama de casos de uso muestra la vista estática de casos de uso a través de un conjunto de casos de uso, actores y sus relaciones.

Un diagrama de interacción permi-te visualizar como un conjunto de objetos interactúan entre sí mediante sus relaciones y mensajes. Estos diagramas direccionan la vista dinámica de un sistema. Existen dos tipos de diagramas de interacción, los de secuencias y los de colaboración; en el diagrama de secuencias las acciones entre objetos se ordenan de acuerdo al tiempo en que ocurren los mensajes y en los diagramas de colaboración el énfasis es en la organización estructural de los objetos que envían y reciben mensajes.

Un diagrama de estados también maneja la vista dinámica del sistema, y consiste en una máquina de estados formada por estados, transiciones, eventos y actividades. Estos diagramas permiten el modelado del comportamiento de una interfaz de clase o de colaboración.

Un diagrama de actividades es una clase especial del diagrama de estados y muestra el flujo desde una actividad a otra dentro del sistema y sirven para modelar las funciones del mismo.

Un diagrama de componentes muestra la organización y dependencias a lo largo de un conjunto de componentes mostrando la vista estática de implementación de un sistema.

Un diagrama de despliegue mues-tra la configuración de los procesos en tiempo de corrida y los compo-nentes que se encuentran en los nodos.

CONCLUSIONES

La búsqueda de nuevas técnicas y herramientas que ayuden al incremento de la calidad en los productos de software es una constante dentro de los grupos de investigación del área de Ingeniería de software. Como resultado de estas investigaciones constantemente surgen formas innovadoras de modelado de sistemas. UML nace de la necesidad de estandarizar la forma de aplicar la Ingeniería de software en un ambiente orientado a objetos, proporcionando un lenguaje común que apoya el modelado y diseño de sistemas mediante el cual se pueden obtener beneficios como: reducción del tiempo dedicado al análisis de requerimientos, posibilidad de involucrar en el proceso de diseño al usuario, habilidad para automatizar las tareas, mejor la documentación y el soporte para una buena administración del software, y una producción de software más robusto, entre otras.

La expresividad y potencia de UML es tal que el 80 % de los problemas se pueden modelar con el 20 % de UML.

BIBLIOGRAFIA

Booch G., Rumbaugh J. and Jacobson I., "The unified modeling language: User Guide", Addison Wesley, 1999.

Booch G., "Object Oriented A n a l y s i s a n d D e s i g n w i t h Applications", Benjamin/Cummins, 1994.

Jacobson, I., G. Booch G. and Rumbaugh J., "The unified software development process", Addison Wesley, 1999.

Jacobson I., "Object Oriented Software Engineering: A Use Case Driven Approach", Addison Wesley, 1992.

Rumbaugh, J., Jacobson I., and Booch G., "The unified modeling language: Reference Manual", Addison Wesley, 1999.

Rumbaugh J., "Modelado y diseño orientados a objetos: Metodología OMT", Prentice Hall, España, 1996.

www.rational.com
www.sdmagazine.com/uml/
www.researchitect.com
www.omg.org