Modelado Visual con UML.

Carlos Alberto Fernández y Fernández

Instituto de Electrónica y Computación, Universidad Tecnológica de la Mixteca.
Km. 2.5 carretera Huajuapan – Acatlima. Huajuapan de León, Oaxaca, México.
caff@mixteco.utm.mx

Resumen.

El documento ofrece una perspectiva de UML, resaltando la importancia del modelado visual en el desarrollo de software, como una forma de mejorar la problemática de elaboración de software de calidad en las empresas. El modelado visual es independiente del lenguaje de programación y ofrece diferentes vistas dependiendo del el aspecto del sistema que se quiera resaltar.

Introducción.

En la década de los ochenta, la programación orientada a objetos comenzó a entrar a las empresas, pero es a hasta finales de la década de los 80 cuando surgen infinidad de metodologías de análisis y diseño orientado a objetos. Es decir, el análisis que existía anteriormente - análisis estructurado- no se adecuaba del todo a la POO.

Aparecieron algunas propuestas apoyadas por su correspondiente bibliografía. Citaremos las principales:

- Sally Shlaer y Steve Mello. Publican en 1989 y 1991 sobre análisis y diseño. Su enfoque se conoce como Diseño Recursivo.
- Peter Coad y Ed Yourdon en 1991 escribieron con un enfoque hacia los métodos ligeros orientados a prototipos de Coad.
- La comunidad de Smalltalk de Portland, Oregon, aportó el Diseño Guiado por la Responsabilidad (*Responsibility-Driven Design*) en 1990 y las tarjetas de Clase-Responsabilidad-Colaboración (*Class-Responsibility-Collaboration*) conocidas como tarjetas CRC, en 1989.

- Grady Booch trabajando para Rational Software, desarrollando sistemas en Ada. 1994 y 1995.
- Jim Rumbaugh trabajando en laboratorios de investigación de General Electric. Publica un libro de la Técnica de Modelado de Objetos (*Object Modeling Technique*)en 1991. Su metodología, conocida como OMT avanza hasta una segunda versión (OMT-2) en 1996.
- Ivar Jacobson trabajando para Ericcson en conmutadores telefónicos, introdujo el concepto de Casos de Uso (*Use Cases*) en 1994.

El caso es que existían múltiples metodologías, cada una con su grupo de seguidores. Muchas se parecen entre sí, con distintas notaciones o términos distintos.

Es en la OOPSLA '94 donde se conoce que Rumbaugh deja a General Electric para trabajar junto con Booch en Rational Software, con la intención de unificar sus métodos. En el mismo evento en 1995, se anuncia que Rational Software había comprado Objectory y que Jacobson se uniría a su equipo de trabajo.

En 1996 Grady Booch, Jim Rumbaugh e Ivar Jacobson ("los tres amigos") proponen su método al que le llaman *Unified Modeling Language* (**UML,** Lenguaje Unificado de Modelado).

En conclusión, **UML** unifica esencialmente los métodos de Booch, Rumbaugh (OMT) y Jacobson.

Modelado Visual.

El Modelado Visual es el modelado de una aplicación usando notaciones gráficas. Pero, ¿qué tan importante es construir el modelo de una aplicación? En este sentido comúnmente se hace la comparación hacia la arquitectura tradicional, en la construcción de casas. Aún cuando la construcción que se planee hacer sea una casa sencilla, el resultada será más satisfactorio si cuenta con todo un respaldo en un correcto diseño.

Booch compara la construcción de software con la construcción de una casa para un perro, de una casa para tu familia y de un gran edificio [Booch, 1999]. En el primer caso no será tan evidente la falta de un buen diseño, o al menos nuestro perro no se quejará demasiado. En el segundo caso, es humanamente posible hacer la construcción de una casa sin los planos adecuados, pero la casa resultante seguramente tendrá varias carencias, o en un peor caso, posiblemente no resistirá ciertas condiciones extremas como un temblor. En el caso del edificio, definitivamente sería un grave error comenzar la construcción sin los estudios y planos adecuados.

En el caso del software, es curioso como muchos proyectos del tamaño de un rascacielos, son construidos como si se tratara de la casa de un perro. El modelado visual del software ayuda a capturar las partes esenciales de un sistema:

- Se utiliza para capturar los procesos de negocios desde la perspectiva del usuario.
- El Modelado Visual se utiliza para analizar y diseñar una aplicación, distinguiendo entre los dominios del negocio y los dominios de la computadora.
- Ayuda a reducir la complejidad.
- El Modelado Visual se realiza de manera independiente al lenguaje de implementación.
- Promueve el reuso de componentes.

UML, Lenguaje de Modelado Unificado.

Es un **lenguaje de modelado**[1] y es independiente del proceso, por lo que no se considera una metodología. Un lenguaje de modelado define la notación que es utilizada por los métodos para representar los diseños.
El método o proceso sería los pasos a seguir para llevar a cabo el análisis y diseño. UML debe en parte su gran aceptación a que no incluye un proceso como parte de su propuesta.

En la figura 1, es posible apreciar el conjunto de diagramas que componen la notación de modelado visual de UML, cada una de ellas con su propia sintaxis; pero al mismo tiempo, compartiendo aquellos elementos que llegan a ser comunes a diversos diagramas. Por ejemplo, la notación general de una clase en cualquier diagrama de UML es un rectángulo, aunque puedan variar algunas características o el nivel de detalle dependiendo del contexto en que se este presentando.

[1] Un modelo es una descripción completa de un sistema desde una perspectiva particular.

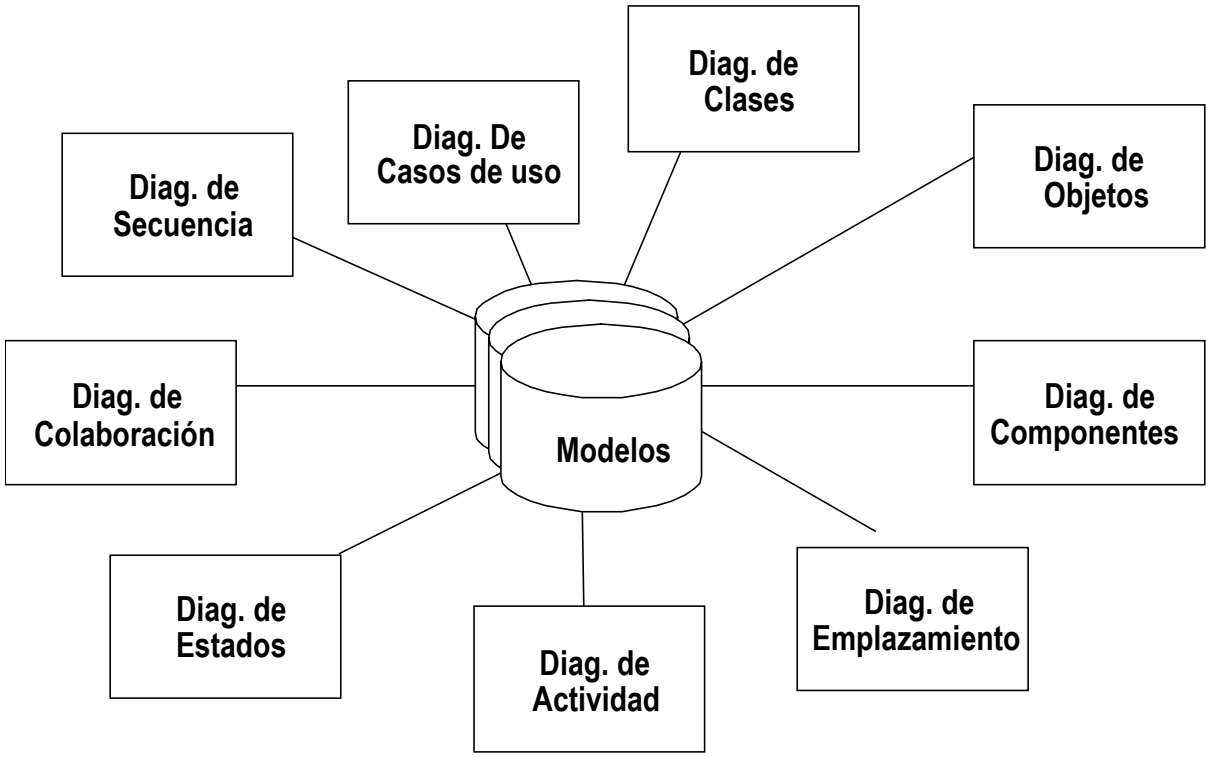

Figura 1. Diagramas de UML [Jacobson, 1999]

Podría decirse que UML es una combinación de [RATIONAL, 1997] :

- Conceptos de Modelado de Datos (Diagramas Entidad Relación).
- Modelado de Negocios.
- Modelado de Objetos.
- Modelado de Componentes.

Precisamente, el nombre de UML viene del esfuerzo de la compañía *Rational* de unificar todo el conjunto de métodos de análisis y diseño orientados a objetos. Las ideas principales pertenecen a Booch, Rumbaugh y Jacobson quienes actualmente trabajan para *Rational Software*. [FOWLER, 1999].

Además como complemento, también se propuso un proceso unificado, que sirva de complemento a UML, pero que no sería obligatorio utilizar. Es desarrollado también principalmente por Booch, Rumbaugh y Jacobson; y se conoce como RUP ó *Rational Unified Process*.

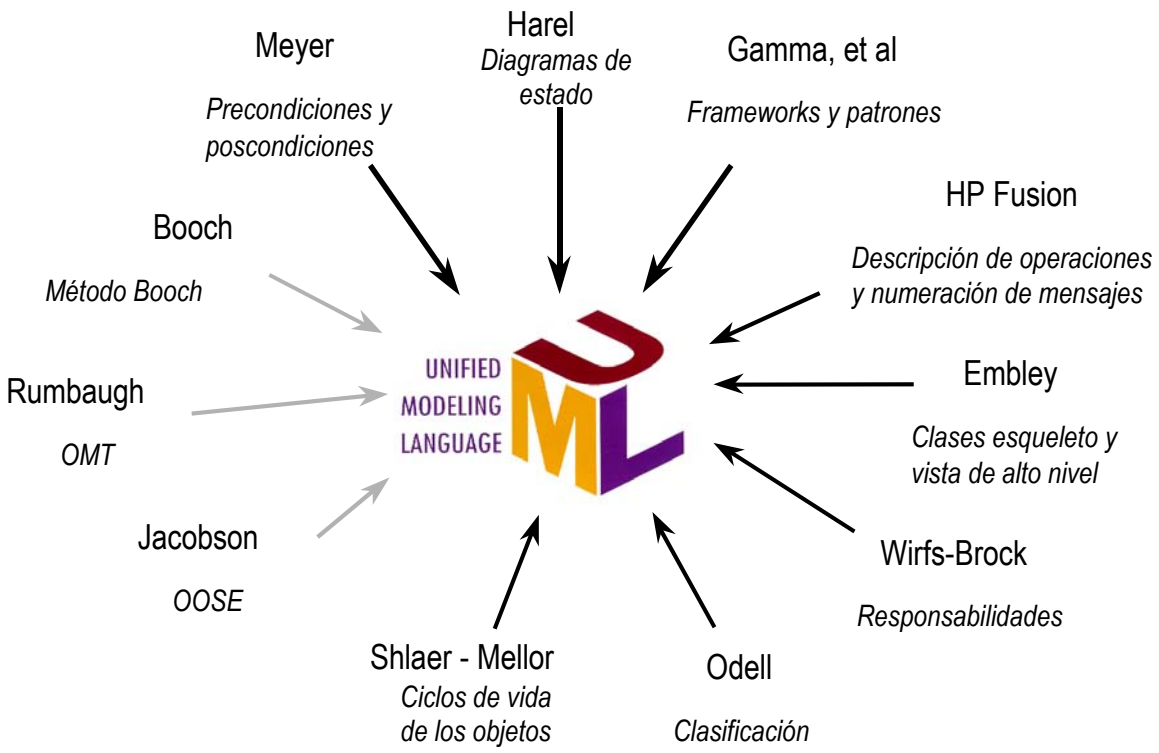

Figura 2. UML es una combinación de diferentes enfoques, pero esencialmente de los conceptos de Rumbaugh, Booch y Jacobson.

Pero volviendo a UML, se puede decir que su utilización es una ventaja; pues actualmente ya no es otra nueva propuesta que pueda confundir más el modelado de sistemas orientados a objetos, al ser aceptado como un estándar de **OMG** (*Object Management Group*) a partir de noviembre de 1997: para la visualización, especificación, construcción y documentación de sistemas de software[RATIONAL, 1999]. Además, en la figura dos se puede ver la diversidad de técnicas de modelado que fueron consideras e incluidas en mayor o menor grado para proponer UML.

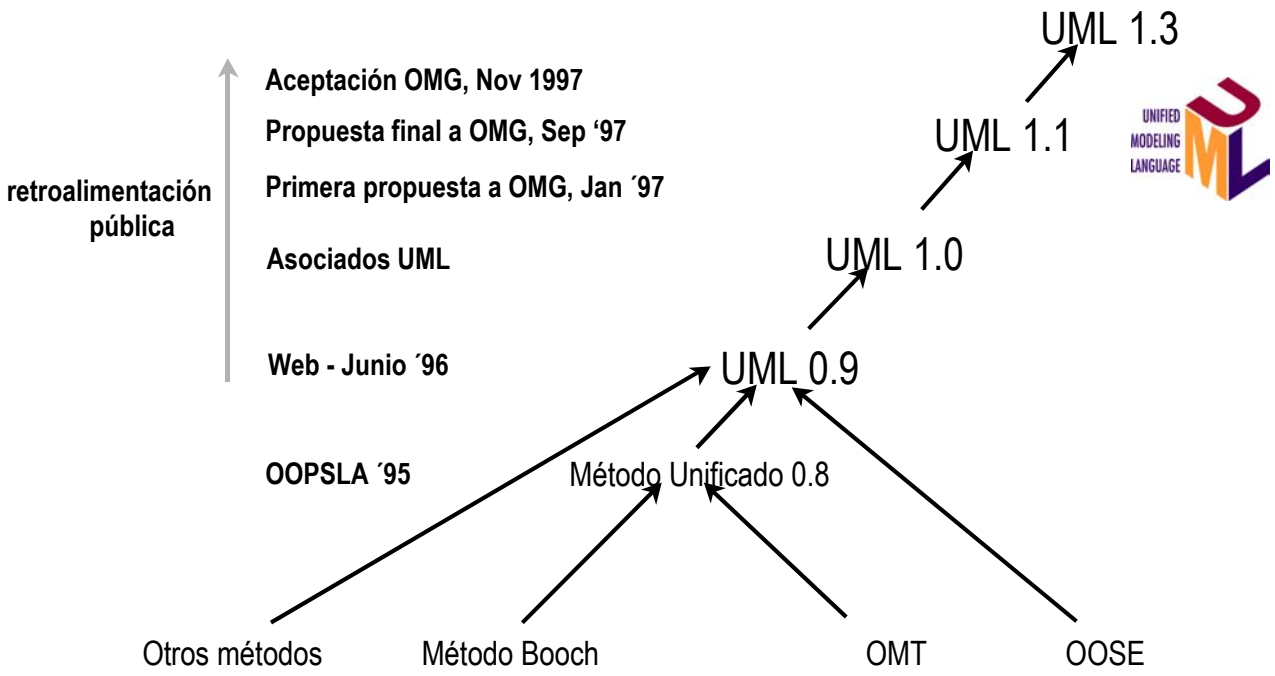

Figura 3. Evolución de UML. [BOOCH, 1999]

Puede ser usado con cualquier proceso, a lo largo del desarrollo del ciclo de vida, y de manera independiente de la tecnología de implementación. Por ejemplo:

- Desplegar los **límites de un sistema** sus principales funciones mediante casos de uso y actores.
- Representar la **estructura estática** de un sistema usando diagramas de clases.
- Modelar los **límites de un objeto** con diagramas de estados.
- Mostrar la **arquitectura de la implementación** física con diagramas de componentes y de emplazamiento o despliegue.

UML se encuentra actualmente en la versión 1.3 (ver figura 3); y, aunque ya se considera un estándar maduro, es natural que siga evolucionando. Otro aspecto importante es que existen algunas variaciones que se deben considerar dependiendo del tipo de modelado que se quiera realizar, por ejemplo en el caso de diseño de sistemas de tiempo real o de páginas de *web*.

Otra ventaja de UML, es que es apoyado por las más importantes compañías de desarrollo de software. Presentamos una lista de las principales compañías involucradas en el desarrollo de UML:

- Rational Software Corporation
- Hewlett-Packard
- I-Logix
- Microsoft
- ObjecTime
- Oracle

- IBM
- ICON Computing
- Intellicorp
- MCI Systemhouse
- Platinum Technology
- Taskon
- Texas Instruments/Sterling Software
- Unisys

Conclusiones.

UML ha probado ser un buen conjunto de herramientas para el modelado visual, separado del proceso de desarrollo. Es posible usar UML con el Proceso Unificado, pero no depende del mismo. En la medida de que, unido al método de preferencia de las empresas, sea utilizado para modelar sistemas de cómputo, es posible mejorar la calidad y disminuir los tiempos de desarrollo paulatinamente.

Además, la adopción de UML como estándar para el modelado de software no ofrece una curva de aprendizaje muy grande si ya se usa en la organización alguna otra notación basada en objetos.

Referencias.

[Booch, 1999] BOOCH, Grady. **SOFTWARE ARCHITECTURE AND THE UML.** Rational Software Corporation. 1999.

[Fowler, 1999] FOWLER, Martin. **UML GOTA A GOTA.** México. Ed. Addison Wesley. 1999.

[Jacobson, 1999] JACOBSON, Ivar. **APPLYING UML IN THE UNIFIED PROCESS**. Rational Software Corporation. 1999.

[Rational, 1997] **ANALYSIS AND DESIGN WITH UML**. Rational Software Corporation. USA. 1997.

[Rational, 1999] **INSIDE THE UNIFIED MODELING LANGUAGE**. Rational Software Corporation. USA. 1999.