UML 1.1. Un lenguaje de modelado estándar para los métodos de ADOO

Francisco José García Peñalvo
*Licenciado en Informática. Profesor del Área de Lenguajes y Sistemas Informáticos de la **Universidad de Burgos**.*
fgarcia@.ubu.es

Carlos Pardo Aguilar
*Licenciado en Informática. Profesor del Área de Lenguajes y Sistemas Informáticos de la **Universidad de Burgos***
cpardo@.ubu.es

Desde la aparición en escena de UML (*Unified Modeling Language - Lenguaje Unificado de Modelado*), el interés de las empresas de desarrollo de software orientado al objeto por los métodos de análisis y diseño orientados al objeto (ADOO) se ha visto de nuevo incentivado. UML representa la apuesta de tres de los más prestigiosos expertos en metodologías orientadas al objeto, Grady Booch, James Rumbaugh e Ivar Jacobson, por el control de la parcela del ADOO dentro de la tecnología de objetos bajo la idea de la unificación. Pero UML no es más que una pieza dentro del complejo engranaje que se está construyendo entorno a la idea de la convergencia de los métodos orientados al objeto, y que intentaremos desgranar en el presente artículo.

Introducción

Una de las principales barreras con la que se han estado topando las empresas y departamentos de informática para la adopción de la tecnología de objetos es la multitud de métodos, metodologías, notaciones que han surgido entorno a este paradigma, dando lugar a lo que popularmente se conoce con el nombre de *guerra de los métodos*. Cada autor de relevancia sacaba su proyecto de método orientado al objeto, con sus propios procesos y su notación particular. Esta disparidad tiene como consecuencia inmediata la confusión de los usuarios de la tecnología de objetos, una mayor complicación en la formación del personal y un muro, muchas veces infranqueable, para el intercambio de información y de personal entre equipos de producción de software.

Ante este caos, se hacía cada vez más necesario la definición de un estándar que aunara todo el trasfondo común de los métodos existentes, presentando una única alternativa universalmente aceptada. Un primer esbozo de este proceso de unificación queda

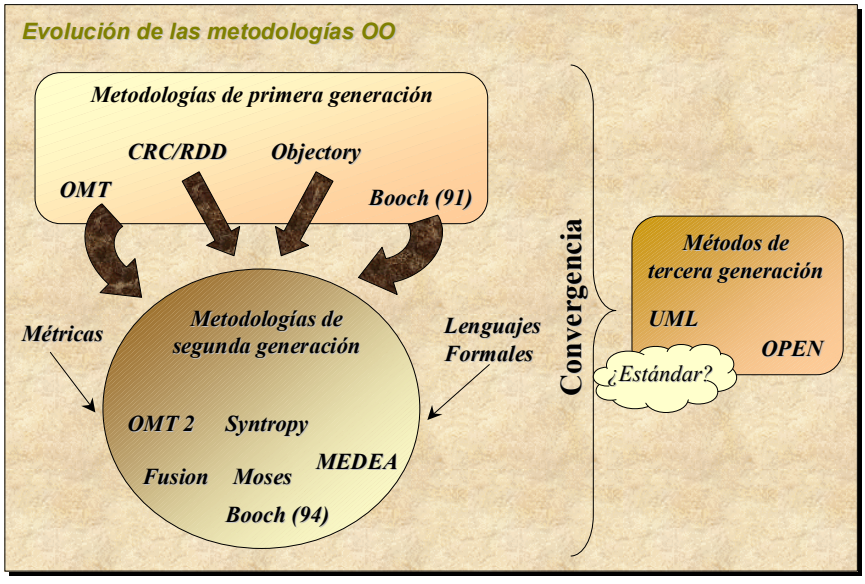

Figura A. Evolución de las metodologías OO

patente de forma explícita en lo que se denomina metodologías de segunda generación, que intentan reunir las ventajas de las metodologías de primera generación con añadidos de métricas y formalismos. De todas formas, estas metodologías de segunda generación son más una evolución que un proceso de convergencia, el cual realmente se da en lo que se ha dado por llamar métodos de tercera generación y cuyo objetivo es presentarse como un método estándar que deje fuera de la parcela al resto de los métodos. Esta evolución queda reflejada en la **Figura A**.

No obstante, la tendencia actual de convergencia hacia un estándar se debe a varios factores entre los que destacaremos dos de ellos: la labor de OMG (*Object Management Group*) en favor de la consecución de un estándar dentro de la orientación al objeto en general y más concretamente en el área de ADOO, y el trabajo y visión de futuro del equipo que se encuentra detrás de UML en Rational Software Corporation.

Con idea de sintonizar al lector dentro de la onda de la convergencia dentro del área de ADOO, vamos a presentar la evolución histórica de los acontecimientos, que en esta parcela de la tecnología de objetos se han ido sucediendo, pero siempre vistos desde el prisma de UML.

Evolución histórica de UML

UML ha sido desarrollado en el seno de Rational Software Corporation con el apoyo de diversos colaboradores a lo largo de su historia, convirtiéndose en el sucesor de los lenguajes de modelado de los diversos métodos de análisis y diseño orientados al objeto aparecidos desde finales de los años ochenta, y en especial UML se convierte en el elemento unificador de los lenguajes de modelado de los métodos Booch-93[1] (Grady Booch), OMT-2 (James Rumbaugh) y OOSE (Ivar Jacobson).

Los antecedentes históricos de UML nos llevan hasta el final de la década de los ochenta cuando varios expertos en metodologías presentan diferentes soluciones dentro del ADOO influenciadas por diferentes técnicas entre las que cabe destacar el modelo Entidad-Interrelación. El número de lenguajes de modelado aumenta de menos de 10 a más de 50 en el período comprendido entre 1989 y 1994. Esta amplia gama de soluciones confunde a los usuarios, que no se decantan claramente por ninguna de las tendencias, dando lugar a la llamada *guerra de los métodos*. Hacia la mitad de la década de los noventa aparecen en escena las nuevas versiones de los métodos más consolidados, así como nuevas incorporaciones que surgen como una acumulación de las características más destacadas de los métodos tradicionales junto con algunos añadidos.

Ante esta situación, se empiezan a escuchar los primeros rumores sobre la necesidad de estandarización dentro del ADOO, pero nadie parece querer dar el primer paso. Así un equipo de OMG lanza una primera propuesta de estandarización que recibe como respuesta una carta de protesta por parte de la mayoría de los expertos en el tema.

Grady Booch, director científico de Rational Software Corporation desde prácticamente su creación en 1980, empieza a planificar su estrategia a favor de la unificación de métodos. Así, para la comunidad de los métodos orientados al objeto la gran noticia en el OOPSLA'94 fue que James Rumbaugh había abandonado General Electric para unirse a Grady Booch en Rational Software Corporation para fusionar sus métodos. De esta manera el desarrollo de UML comienza en octubre de 1994, cuando Booch y

[1] La notación del método de Booch ya fue presentada en esta revista en los números 22,23 y 24, para más información sobre estos artículos consultar las referencias [3], [4] y [5] de la bibliografía.

Rumbaugh empiezan a trabajar para unificar los dos métodos que más repercusión habían alcanzado en la escena del ADOO, Booch'93 y OMT. Como primer fruto de esta colaboración aparece en octubre de 1995 la primera versión pública de la descripción de la unión de sus métodos. Esta versión se presenta en el OOPSLA'95 con el nombre de *Método Unificado versión 0.8*.

La siguiente bomba en el proceso de unificación se produce a finales de 1995 cuando Rational compra a la empresa Objectory, y su fundador, el prestigioso Ivar Jacobson, se une a Rational como vicepresidente de ingeniería de negocio para acoplar su metodología OOSE al método unificado de Booch y Rumbaugh. Objectory será el nombre que reciba el proceso unificado de los *tres amigos*, nombre por el que se conoce a Booch, Rumbaugh y Jacobson a partir de la unión de este último al grupo.

Como primer paso a dar después de la incorporación de Jacobson se proponen crear un lenguaje de modelado unificado debido principalmente a dos razones. En primer lugar el lenguaje de modelado permite que los tres métodos puedan evolucionar hacia un punto común de forma independiente. Por otro lado, la unificación de la semántica y de la notación les permite colocarse en un lugar de privilegio en el mercado de los métodos orientados al objeto, debe tenerse en cuenta que la mayoría de la gente que dice utilizar un método orientado al objeto realmente lo que usa es la notación asociada a dicho método, olvidándose de los procesos asociados al método, así pues, contar con una notación gráfica universal era fundamental para la unificación de sus métodos, y de hecho la notación de UML se ha convertido en el estándar de facto de las notaciones en tecnología de objetos.

La primera versión de este lenguaje de modelado aparece en junio de 1996 con el nombre de UML 0.9. En octubre de este mismo año aparece la versión 0.91 de UML.

Durante 1996 invitan a otros expertos a colaborar con ellos, entre los colaboradores se encuentran nombres muy ilustres y de reconocido prestigio dentro de la comunidad de la ingeniería del software y más particularmente de la orientación al objeto. Citarlos a todos sería largo y correríamos el riesgo de olvidarnos a alguno, pero como muestra podemos mencionar a personajes de tanto renombre como *Peter Coad, Derek Coleman, Ward Cunningham, David Ambly, Eric Gamma, David Harel, Richard Helm, Ralph Johnson, Stephen Mellor, Bertrand Meyer, Jim Odell, Kenny Rubin, sally Shlaer, JohnVlissides, Paul Ward, Rebbeca Wirfs-Brock, Edward Yourdon* entre otros.

Además, como foro de discusión se creó OTUG (*Object Technology Users Group*), una lista de correo en la que se discute y se comparten ideas sobre la tecnología de objetos, teniendo como trasfondo a UML.

Tras la salida a la luz de UML la comunidad de ADOO queda divida en dos grupos principales, aquellos que se unen a la estela de UML y aquellos que forma una coalición anti UML. Esta situación la aprovecha OMG para en 1996 crear el OOAD SIG (*Object-Oriented Analysis and Design Special Interest Group*) que se encargara de canalizar los esfuerzos para conseguir un estándar. De esta forma se organiza la OMG Task Force RFP (*Request For Proposal*), es decir una petición de propuestas para la creación de un estándar del lenguaje de modelado para los métodos de ADOO. Esta petición de propuestas sugería un lenguaje de modelado que contara con un metamodelo[2], una notación, una sintaxis y una semántica.

Rational va a responder a esta petición de propuesta con la versión 1.0 de UML en enero de 1997. Esta versión está avalada por un conjunto de empresas de mucho

[2] Un modelo que define el lenguaje para expresar un modelo.

prestigio dentro del mundo de la informática: Digital Equipment Corporation, HP, i-Logix, IntelliCorp, IBM, ICON Computing, MCI Systemhouse, Microsoft, Oracle, Rational Software, TI, y Unisys.

Como era de esperar, la respuesta de Rational no fue la única, OMG recibió en enero de 1997 las respuestas de IBM & ObjecTime, Platinum Technology, Ptech, Taskon & Reich Technologies y Softeam.

Durante los primeros meses de 1997 se hicieron predicciones de todo tipo y para todos los gustos. La notación de UML se había convertido en un estándar de facto apoyado por empresas de especial relevancia en el sector informático, lo cual colocaba a UML en una posición de privilegio, pero su oscura semántica y su metamodelo parecían no estar a la altura de otras propuestas, lo cual hacía peligrar su adopción como estándar por OMG en detrimento del resto de las propuestas. Así todo apuntaba a dos posibles soluciones. La primera de ellas, y quizás la peor, daba como resultado dos estándares: UML de Rational y OML de Platinum Technology. La segunda de ella apuntaba por una solución mixta que contemplase aspectos de todas las propuestas.

La solución final puede acercarse a la segunda opción, y para constatarlo se han unido al equipo liderado por Rational el resto de los equipos que enviaron propuestas, dando lugar a la versión 1.1 de UML, que ha sido enviada a OMG el 1 de septiembre de 1997, contribuyendo todos con sus ideas a la generación de una respuesta única.

El camino seguido hasta UML 1.1 se puede resumir en la **Figura B**.

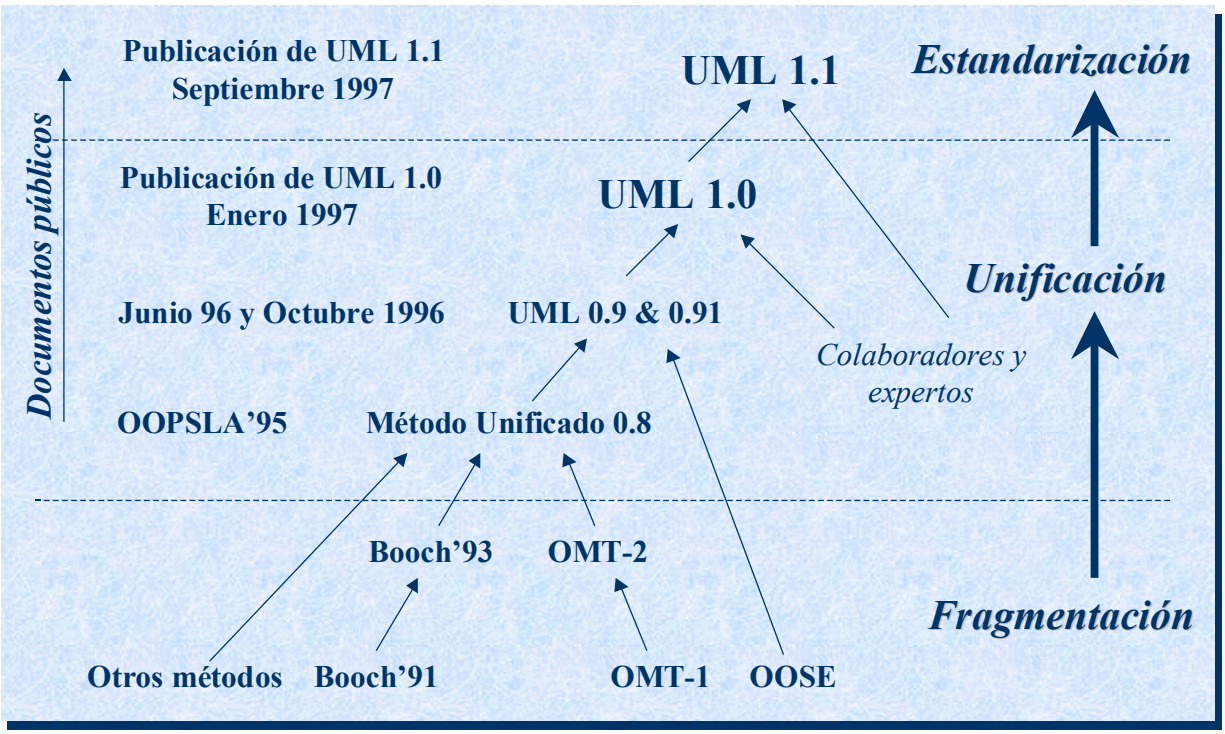

Figura B. Evolución de UML.

Qué es UML

Una vez vistos los orígenes y evolución de UML, vamos a centrarnos en lo que realmente es UML.

UML es un lenguaje para especificar, visualizar, construir y documentar los componentes de los sistemas software, también modelos de negocio y otros sistemas diferentes al software.

Es importante incidir en el hecho de que UML se trata de un lenguaje de modelado, no un método de ADOO. Un método está compuesto de un lenguaje de modelado y de un proceso. El lenguaje de modelado aporta los elementos de modelado, es decir los conceptos y la semántica, la notación (normalmente gráfica) y una serie de recomendaciones de cómo usarlo. Por su parte la parte de proceso de un método de ADOO indica los pasos a seguir para realizar un diseño.

UML al ser un lenguaje de modelado no pretende forzarnos a utilizar una determinada metodología, porque en general distintos dominios de un problema nos llevan a distintos procesos de análisis y diseño. Por lo tanto, lo que nos propone UML es la utilización de una semántica y una notación universal.

La construcción de modelos es una faceta esencial y prioritaria en la construcción de sistemas software. Los modelos serán esenciales para la comunicación entre equipos de proyectos, y su importancia crece cuanto mayor sea la complejidad y el tamaño del proyecto a abordar. Por este motivo, es vital contar con un lenguaje de modelado capaz de captar la semántica de cualquier sistema software para llevar a cabo con éxito la realización de un proyecto software de una cierta complejidad. Si además el lenguaje de modelado con el que se cuenta es estándar, obviamente dicho modelo tiene un valor añadido, tanto por la facilidad de comprensión por parte de personas ajenas a las que lo realizaron, como por el tratamiento que del modelo pueden llegar a hacer diferentes herramientas.

Puede sorprender al lector que teniendo UML los orígenes en unas metodologías con unos procesos tan bien definidos, todos los esfuerzos caminen por conseguir un lenguaje de modelado estándar. La respuesta a esta cuestión no es muy difícil de explicar, UML no tiene como objetivo ser un lenguaje de modelado líder dentro de los métodos de ADOO, sino que pretende convertirse en "*el lenguaje*" de modelado de los métodos de ADOO. Esto conduce a que los fabricantes de herramientas no tengan que soportar un número elevado de lenguajes de modelado, todos muy similares, pero ligeramente diferentes a la vez, lo cual es especialmente perjudicial para los pequeños fabricantes de herramientas.

Objetivos de UML

Como resumen de lo visto hasta el momento se van a presentar los principales objetivos que se persiguen con el diseño de UML:

- *Ofrece a los usuarios un lenguaje visual de modelado expresivo y listo para ser utilizado:* De esta forma los usuarios pueden desarrollar e intercambiar modelos con significado. UML presenta un conjunto de conceptos clave de modelado que son aceptados de forma general por la mayoría de los métodos y herramientas de modelado.
- *Ofrece mecanismos de extensión y especialización para ampliar el conjunto de conceptos clave*. UML puede ser extendido para cubrir nuevas necesidades o dominios específicos, sin que por ello se tengan que redefinir los conceptos clave, siguiendo el propio lenguaje de modelado las pautas de la tecnología de objetos.

- *Es independiente de los lenguajes de programación y los procesos de desarrollo*. De esta forma UML da soporte a varios métodos y procesos de construcción de modelos sin ninguna dificultad añadida.
- *Ofrece unos mecanismos para entender el lenguaje.* UML ofrece una definición del formato estático del modelo utilizando un metamodelo expresado en diagramas de clase de UML, con un añadido de restricciones en lenguaje natural y a partir de la versión 1.1 se introducen expresiones OCL (*Object Constraint Language*). OCL es una de las nuevas caracteríticas de UML 1.1, siendo una aportación de IBM, ya que fue desarrollado por IBM como lenguaje de modelado de reglas de negocio, aunque se deriva del método Syntropy.
- *Contribuye al crecimiento del mercado de herramientas:* La herramienta que soporta el lenguaje de modelado más utilizado y considerado como estándar sale beneficiada al tener más cuota de mercado.
- *Da soporte a conceptos de alto nivel de desarrollo:* Tales como patterns, frameworks o componentes. La inclusión de estos conceptos beneficia a la orientación al objeto y en especial a la reutilización del software.

Rational Objectory Process

Como se ha reflejado en el apartado anterior, UML es intencionadamente independiente del proceso, de forma que las empresas puedan utilizar los elementos que ofrece UML con independencia de los procesos que requiera el contexto en que se produce el desarrollo. Además, la definición de un proceso estándar de desarrollo para el ADOO estaba fuera de los objetivos marcados por la petición de propuestas de OMG.

La independencia de UML con respecto a los procesos de desarrollo no implica en absoluto que en el seno de Rational Software Corporation se olviden de la importancia del proceso, especialmente cuando los *tres amigos* copaban con sus métodos una importante parcela de mercado de la tecnología de objetos. Así, Booch, Rumbaugh y Jacobson están trabajando en la convergencia de sus procesos. El resultado tendrá el nombre de *Rational Objectory Process*, y está previsto que esté listo para los primeros meses de 1998.

Como grandes expertos en el tema, los *tres amigos* tienen presente que no se puede tener un único proceso. Habrá una serie de factores que lleven a un tipo de proceso o a otro. Un factor es el tipo de software que se desarrolla (*tiempo real, distribuido...*), otro es la escala (*desarrollos individualizados, equipos pequeños...*). Así, lo que están desarrollando es un framework de procesos, esto es, algo que capture los elementos comunes pero permita a las personas aplicar las técnicas más apropiadas al tipo de desarrollo.

La descripción del proceso marco que se encuentra tras Rational Objectory, además de pretencioso por no haber sido publicado todavía, escapa al alcance de este artículo. Pero lo que sí se puede adelantar es que el proceso definido en Rational Objectory es un proceso de desarrollo iterativo e incremental, en el que el software no se realiza al final del proyecto, sino que se desarrolla y se entrega por partes. Un esquema general del proceso se muestra en la **Figura C**.

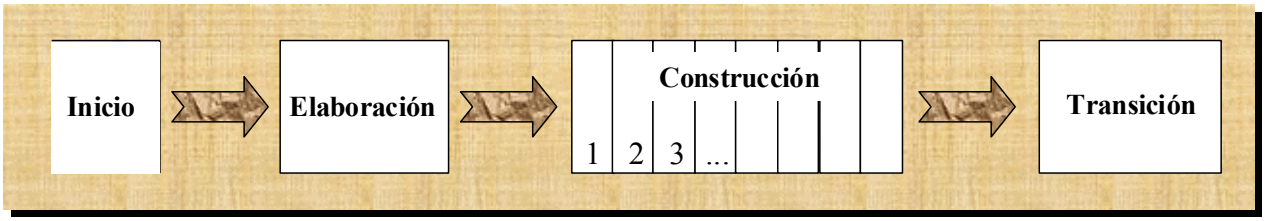

Figura C. Esquema general de Rational Objectory Process.

Donde la fase de **inicio** se estudia la viabilidad del proyecto, el ámbito en el que se va a desarrollar y un análisis inicial. En la fase de **elaboración** se recogen los requisitos de una forma más detallada y se realiza un análisis y diseño de alto nivel para generar un plan de construcción. La fase de **construcción** consiste en varias iteraciones en las que se van construyendo componentes software de calidad, probados e integrados que cumplen un subconjunto de los requisitos del proyecto. Por último, la fase de **transición** se encarga de las tareas que han de dejarse para el final del desarrollo, por ejemplo actividades de optimización, pruebas de uso y aceptación o formación de los usuarios.

Conclusiones

UML se presenta como una de las revoluciones en la tecnología de objetos que puede sentar unas bases de uniformidad, convergencia y estandarización dentro de la caótica escena de los métodos ADOO.

Las repercusiones más inmediatas de UML serán una mayor facilidad para la difusión y aceptación de la tecnología orientada a objetos, y una mayor homogeneidad en las herramientas CASE que tendrán que soportar un único lenguaje de modelado universalmente aceptado. La mayoría de las herramientas CASE OO ya soportan UML, como ejemplos representativos se pueden citar Rational Rose 4.0 o Microsoft Visual Modeler 1.0.

La independencia del proceso es otra característica a destacar, pues una empresa puede adoptar UML sin tener que cambiar todos sus procesos de desarrollo.

Como puntos desfavorables que se pueden mencionar, se tiene que UML es un estándar de facto, pero todavía falta que OMG lo acepte como estándar oficial, aunque con la versión 1.1 se ha dado un paso muy importante hacia la convergencia y la estandarización. Aunque UML 1.1 es una propuesta completa, todavía debe evolucionar más, especialmente en la definición de su semántica. Además desde un punto de vista académico, UML es *poco* formal en la descripción de su metamodelo, pues se basa en diagramas de clases y en el lenguaje natural en lugar de en una lógica formal.

Referencias bibliográficas

[1] **Booch, Grady.** "*Object-Oriented Analysis and Design with Applications*". 2nd Ed. Benjamin Cummnings, 1994.

[2] **Fowler, Martin and Scott, Kendall**. "*UML Distilled. Applying the Standard Object Modeling Language*". Addison-Wesley. http://www.awl.com/cp/uml/uml.html. 1997.

[3] **Hidalgo, Juan.** "*La Notación Booch (I). Análsis y Diseño Orientados a Objetos*". RPP, Nº 22, pp. 67-72. Diciembre 1996.

[4] **Hidalgo, Juan.** "*La Notación Booch (II). Análsis y Diseño Orientados a Objetos*". RPP, Nº 23, pp. 67-72, Enero 1997.

[5] **Hidalgo, Juan.** "*La Notación Booch (y III). Análsis y Diseño Orientados a Objetos*". RPP, Nº 24, pp. 63-68, Febrero 1997.

[6] **Jacobson, Ivar, Christerson, M., Jonsson, P., Overgaard, G.** "*Object-Oriented Software Engineering: A Use Case Driven Approach*". Addison-Wesley. 1992.

[7] **Rational Software Corporation et al**. "*UML 1.1 Documentation Set*". http://www.rational.com/uml. 1 September 1997.

[8] **Rumbaugh, James, Blaha, Michael, Premerlani, William, Eddy, Frederick and Lorensen, William**. "*Object-Oriented Modeling and Design*". Prentice-Hall, 1991.