Modelado Orientado a Objetos: Una Evaluación Crítica

Guillermo Bustos Reinoso
gbustos@ucv.cl
Escuela de Ingeniería Industrial - Universidad Católica de Valparaíso
Av. Brasil 2147 - Casilla 4059 Valparaíso - Chile

Resumen

Este artículo presenta una evaluación crítica de lo que se conoce como modelado dentro del análisis orientado a objetos. Se muestran algunos problemas con relación a la pretensa naturalidad de este paradigma, al encapsulamiento en el modelado, a los costos de la continuidad estructural y a la transición hacia el diseño. También se indican algunas consideraciones sobre las estrategias y dimensiones utilizadas en el modelado orientado a objetos.

Introducción

El análisis orientado a objetos, o más precisamente, el Modelado Orientado a Objetos (MOO) es un área relativamente reciente. Durante estos años 90 aparecieron en la literatura algunas propuestas más elaboradas (principalmente en forma de libros) que apuntan a esta fase del desarrollo de software bajo el paradigma de la orientación a objetos. Existe actualmente una diversidad de técnicas, lo cual sólo aumenta la dificultad para identificar, entre otros, cuáles son las reales ventajas de abordar el análisis de sistemas con orientación a objetos, cuáles son las actividades generalmente aceptadas que deben realizarse en el MOO, y cuáles son las estrategias alternativas más apropiadas para conducir el MOO. Sin embargo, esta situación está cambiando producto del esfuerzo emprendido por la OMG (***O****bject* ***M****anagement* ***G****roup*) con su propuesta UML (***U****nified* ***M****odeling* ***L****anguage*) de estandarización notacional para los sistemas orientados a objetos (ver [Rational99], [Rumbaugh99] y [Booch99]). Más recientemente, se ha propuesto además un proceso unificado de desarrollo de software orientado a objetos [Jacobson99], que utiliza la notación UML, sin embargo aún está por verse si logrará transformase en estándar.

Este trabajo se propone una evaluación crítica de lo que se conoce como MOO. Para esto, en la sección 1 de este trabajo se describe brevemente las dimensiones y las actividades del MOO. En la sección 2 se muestra las estrategias posibles para conducir el MOO. En la sección 3 se evalúa el MOO desde el punto de vista de su naturalidad, encapsulamiento, continuidad estructural y su transición al diseño orientado a objetos. Finalmente, la sección 4 entrega algunas consideraciones con relación a las dimensiones y estrategias del MOO.

1 Modelado Orientado a Objetos (MOO)

En el contexto del desarrollo de sistemas de software con orientación a objetos, el MOO es la construcción de modelos de un sistema por medio de la identificación y especificación de un conjunto de objetos relacionados, que se comportan y colaboran entre sí de acuerdo a los requerimientos establecidos para el sistema de objetos (definición adaptada de [Monarchi&Puhr92]).

La definición anterior ya sugiere la distinción, dentro del proceso de MOO, de tres dimensiones o perspectivas relativamente ortogonales para describir un sistema de objetos:

- *Dimensión estructural de los objetos*: Se centra en las propiedades estáticas o pasivas de los sistemas. Está relacionada con la estructura estática del sistema de objetos.
- *Dimensión dinámica del comportamiento*: Se centra en las propiedades activas y describe el comportamiento individual y la colaboración entre los objetos que constituyen el sistema.
- *Dimensión funcional de los requerimientos*: Son consideradas las propiedades relativas a la función de transformación del sistema de objetos, es decir, los procesos de conversión de entradas en salidas.

El proceso de MOO puede ser dividido en un conjunto mínimo de actividades. La lista a seguir muestra estas actividades sin ninguna secuencia específica:

- *Identificar las clases, objetos y atributos:* Se determinan cuáles son las clases, objetos y atributos que deben incluirse en el modelo.
- *Asociar estáticamente los objetos:* Es la configuración de una estructura estática que exprese relaciones dependientes del dominio del problema.
- *Describir el comportamiento de los objetos:* Es la especificación del comportamiento de los objetos sobre la base de los conceptos básicos de estado, regla de transición, evento y acción.
- *Definir la colaboración del comportamiento de los objetos:* Busca reflejar la interacción o colaboración entre los objetos, considerando el flujo de eventos o mensajes entre los mismos.
- *Organizar las clases en jerarquías de herencia:* Se propone organizar las clases de tal forma a maximizar la compartición de propiedades comunes entre ellas.
- *Agregar y/o particionar las clases por niveles de abstracción:* Jerarquiza el modelo por niveles de complejidad haciendo uso de mecanismos de particionamiento o de agregación.

2 Estrategias del MOO

Las seis actividades descritas en la sección anterior pueden ser ordenadas de diversas maneras, de acuerdo a los diferentes procedimientos propuestos por las técnicas de MOO. Sin embargo, estos procedimientos pueden ser categorizados en tres estrategias generales asociadas a las dimensiones del MOO. Estas estrategias son:

- *Dirigida por datos*: En esta estrategia el MOO es dirigido principalmente por la información estática sobre el dominio del problema. La idea central que sustenta esta estrategia es que la estructura de los objetos del sistema es determinante para el comportamiento que éste presenta. La gran mayoría de las técnicas de MOO siguen esta estrategia, con algunas pequeñas variaciones. Entre estas técnicas se encuentran las de [Coad&Yourdon 91], [Coleman94], [de-Champeaux93], [Henderson-Sellers&Edwards94], [Martin&Odell 92], [Rumbaugh91] y [Shlaer&Mellor92].
- *Dirigida por comportamiento*: La estrategia dirigida por comportamiento se basa en información sobre el comportamiento esperado del sistema de objetos. El MOO bajo esta estrategia comienza con la definición del comportamiento global del sistema, para seguir después con la definición de la colaboración de algún tipo de componentes al interior del sistema. Estos componentes o entidades serán candidatos a objetos. La hipótesis básica es que la estructura de los objetos del sistema es derivada del comportamiento que éste debe presentar. La cantidad de procedimientos de propuestas que siguen esta estrategia es bastante menor que la que adopta la estrategia dirigida por datos. Algunas técnicas son las de [Rubin&Goldberg92], [Jacobson92] y [Jacobson99].
- *Dirigida por procesos*: En esta estrategia el MOO es dirigido por la información sobre las tareas (procesos, responsabilidades o funciones) que deben ser desempeñadas por el sistema y sus componentes. Esencialmente esta estrategia utiliza las facilidades de los modelos funcionales (o de procesos) para derivar (o asociar) los objetos y su estructura. Los procedimientos de las propuestas que siguen esta estrategia están entre los primeros que fueron publicados. Entre las propuestas que utilizan modelos funcionales están las de [Bailin89], [Jalote89], [Schiel&Mistrik90], [Seidewitz89] y [Ward89].

3 Críticas y Consideraciones sobre el MOO

A continuación se presentan algunas consideraciones respecto a cómo el MOO es presentado mayoritariamente en la literatura. Muchos autores afirman incluso la conveniencia de usar orientación a objetos en el modelado conceptual. Estas afirmaciones serán analizadas críticamente en las siguientes secciones.

3.1 Naturalidad

La esencia del paradigma de la orientación a objetos y más específicamente, del MOO, es la visión de la realidad compuesta por objetos. Frecuentemente se afirma que esta visión es “más natural” que una visión funcional o comportamental. Por ejemplo, en el ámbito de la ingeniería de negocios, algunos autores, como [Montgomery94] y [Taylor95], sugieren utilizar el modelo de objetos para diseñar negocios dada esta aparente naturalidad que facilitaría la comunicación.

Es razonable preguntar si los objetos (como entes estáticos) son realmente una visión más

natural. Una afirmación a esta pregunta es en verdad una ingenuidad, dadas las diversas indicaciones en contrario desde las siguientes perspectivas:

- *Filosófica*: Es un hecho sustentado hace mucho tiempo en las filosofías orientales y, en menor grado, en algunas escuelas de la filosofía occidental, que la realidad es antes de todo cambio (visión dinámica).
- *Física*: La visión de la realidad surgida de la física moderna (ver por ejemplo [Heisenberg85], [Capra92] y [Bohm92]) sugiere una "tela" de relaciones dinámicas en su nivel más fundamental.
- *Lingüística*: En el lenguaje natural y más específicamente en la lengua española, el verbo ("... cuyo papel semántico es expresar la acción que realiza o padece el sujeto, su existencia o estado...", [Larousse96], p.1016) es la clase de palabra más sofisticada existente, con variadas formas de número, persona, tiempo y modo. Usualmente todas las otras palabras de la oración se relacionan con él de alguna forma con el propósito de complementarlo. No está claro porque se ha de dar un mayor énfasis a los nombres (objetos) en vez de a los verbos (procesos).
- *Sistémica*: Del punto de vista de la teoría de sistemas, los sistemas de información son subsistemas de las organizaciones sociales, y éstas son sistemas dinámicos por excelencia. Modelar inicialmente el comportamiento es más adecuado en este contexto.

Hace 10 años ya en [Constantine89] se afirmaba que no existen *clases de objetos* en el universo físico real. Son construcciones que existen en la mente de los observadores; los *objetos* no son más reales o naturales que las funciones, concluía. A lo anterior, en [Loy90] se agregaba que esta variación de la controversia *forma vs. función* no está resuelta y probablemente nunca lo será. Por lo tanto no es posible afirmar que las clases de objetos sean la forma *más natural* por la cual las personas ven la realidad. No obstante ser éste un argumento a favor de la orientación a objetos muy frecuente en la literatura.

Así cualquier abstracción que pueda ser entendida como un objeto es subjetiva, arbitraria y fuertemente dependiente de la concepción y experiencia del modelador.

3.2 Encapsulamiento

En Informática existe una fuerte tendencia a forzar la realidad modelada a una conceptualización originada de la construcción de software. Por ejemplo, el MOO se sustenta en el encapsulamiento que propone separar los aspectos externos de los internos. Los aspectos externos son conocidos por otros objetos, en cambio los aspectos internos son de conocimiento exclusivo del objeto propietario. De esta forma, los objetos son auto-contenidos y no comparten ningún tipo de dato que no sea solicitado en las colaboraciones con otros objetos. La cuestión es la siguiente: ¿tiene sentido encapsular en el nivel de abstracción requerido en el modelado conceptual? Dado que la mayor ventaja del encapsulamiento se revela en la fase de implementación, ¿qué beneficios se podrían esperar

del encapsulamiento en el modelado conceptual? ¿encapsular en el modelado conceptual no representaría introducir prematuramente rasgos de implementación?

3.3 Continuidad Estructural

La continuidad estructural, según [deChampeaux92], en las sucesivas fases del ciclo de vida orientado a objetos, es comúnmente indicada como una ventaja. El *gap semántico*[1] se vería disminuido: por ejemplo, los objetos identificados en el dominio del problema serían mapeados directamente en los objetos especificados en el dominio de la solución durante el diseño. Lo mismo sucedería con estos objetos al ser mapeados en los objetos codificados de la implementación. Se puede afirmar entonces, que el MOO sería ventajoso si tanto el diseño como la implementación fueran también conducidos bajo este mismo paradigma. Sin embargo, según [Opdahl&Sindre93], el beneficio de la continuidad estructural no debería tener el costo de colocar el MOO más cerca del diseño orientado a objetos[2]. El modelado conceptual, en la perspectiva de la ingeniería de requerimientos, debe ser orientado al problema (realidad) y no a la solución concreta (implementación orientada a objetos).

3.4 Transición al Diseño

La confusión existente entre lo que es el análisis y el diseño orientados a objetos, es decir, la poca claridad en los límites entre estas dos fases es indicada en la literatura (ver por ejemplo [Shlaer&Mellor93] y [Jacobson&Christerson95a]). El principio de la ingeniería de software de que el análisis debe ser declarativo con relación al diseño, es decir, que describe *qué* debe hacer el sistema en contraste con el diseño que establece *cómo* ha de hacerse, no ocurre en la orientación a objetos. En la práctica, los modelos orientados a objetos del análisis son sólo menos detallados que los del diseño o sólo constituyen el "centro" del modelo de diseño; así la pretensa declarativid no es alcanzada. De esta forma, el MOO puede entenderse como un diseño preliminar, según [Hoydahlsvik&Sindre93], porque una vez que los requerimientos son entendidos, éstos se organizan en un modelo que sirve como estructura interna del sistema a ser diseñado.

La orientación a objetos debe ser considerada entonces como una decisión de diseño e implementación y no como una decisión de modelado conceptual o de análisis de requerimientos.

Por lo anterior, los términos *análisis* y *orientación a objetos* en realidad se contraponen. La noción de *análisis* como descomposición y examen crítico con el fin de aumentar la comprensión no

---

[1] Como gap semántico se define la "distancia conceptual" entre el dominio del problema en el mundo real y el dominio de la solución en la implementación computacional.

[2] Por ejemplo, en el modelado de sistemas de información nuevos, que no vayan a usar tecnología de la información como un todo o en alguno de sus subsistemas, no se justificaría utilizar orientación a objetos.

calza con las ideas de síntesis (encapsulamiento de atributos y operaciones en los objetos) y de alternativa de diseño e implementación de la *orientación a objetos*.

Una forma de resolver este problema es creando mecanismos que permitan postergar el encapsulamiento tanto cuanto sea posible. En otras palabras, modelar el dominio del problema fuera del paradigma de la orientación a objetos, con suficiente poder de expresión y flexibilidad para poder derivar modelos orientados a objetos con relativa facilidad, si la implementación desea ser realizada bajo este paradigma. De esta forma, el modelado concluye antes de introducirse el encapsulamiento, punto en el cual comienza el diseño orientado a objetos[3].

En este sentido, el modelo de casos de uso (*use case*), propuesto en [Jacobson92] y más desarrollado en [Schneider&Winters98], puede utilizarse previo a la construcción de modelos de objeto (ver [Jacobson99] para detalles sobre el proceso de desarrollo). Un modelo de casos de uso representa una manera específica de usar el sistema: a través de la ejecución de alguna parte de su funcionalidad. Cada caso de uso describe un escenario completo de eventos iniciados por un actor (o papel de usuario) y especifica la interacción que ocurre entre el actor y el sistema. Corresponde así a una visión externa del sistema, que no se construye bajo el paradigma de la orientación a objetos.

4 Sobre las Dimensiones y Estrategias del MOO

Históricamente las aplicaciones de procesamiento de datos de negocios han presentado los *datos* como una característica destacada. Un modelado según la estrategia dirigida por datos es la más adecuada en estos casos. Es un camino más explorado porque ha evolucionado a partir del modelado semántico de datos.

Para las aplicaciones de tiempo real o de sistemas distribuidos, la dimensión más compleja es la dinámica, sugiriendo la estrategia dirigida por comportamiento para el MOO.

De acuerdo con [Fichman&Kemerer92], los dominios de problemas que contienen procesos globales que afectan varios tipos de objetos (e involucran la ejecución secuencial y/o paralela de numerosos pasos intermediarios entre inicio y fin) indican la dimensión funcional como más apropiada. Ejemplos de este tipo de problemas constituyen los procesos de pedidos de manufactura, conciliación diaria de cuentas bancarias y traductores de lenguajes. En este tipo de problemas parece de mayor utilidad una estrategia dirigida por procesos para el MOO. Sin embargo, las técnicas de MOO le han asignado poca importancia al modelado funcional, argumentando que el concepto de un proceso de transformación global no subordinado a cualquier objeto individual va contra los principios de la orientación a objetos, como indican [Fichman& Kemerer92]. Algunos autores tales como

---

[3] Para una propuesta en este sentido ver [Bustos96].

[Booch89] y [deChampeaux91] recomendaban incluso no utilizar ningún tipo de modelo de este tipo para no introducir una orientación funcional en el MOO. Esto se relaciona principalmente con la inconveniencia de utilizar una descomposición funcional clásica en el sentido del análisis estructurado tradicional de [deMarco78], por ejemplo.

Esta discusión puede resolverse introduciendo el concepto de visión funcional de un sistema de objetos, es decir, la construcción de un modelo de procesos encapsulados en los objetos que representa la secuencia, ejecución condicional e ideas relacionadas con procesos globales de transformación del sistema. Este modelado se denomina *end-to-end* y ha sido previamente sugerida en otros contextos por [Bailin89] y [Henderson-Sellers&Constantine91]. Por ejemplo, la técnica MOSES de [Henderson-Sellers&Edwards94] recomienda el desarrollo de un modelo de estructura de servicios para esta visión funcional. Asimismo, también el modelo de casos de uso constituye una visión funcional aunque generalmente se construye antes de los modelos orientados a objetos.

Con relación a las estrategias del MOO, aquella dirigida por procesos fue la primera a surgir (a partir de mediados de los 80). Las técnicas propuestas en esa época aprovecharon los modelos funcionales del análisis estructurado de forma de conseguir una transición del análisis funcional al diseño orientado a objetos. De acuerdo a la figura 1, esta estrategia tiende a desaparecer.

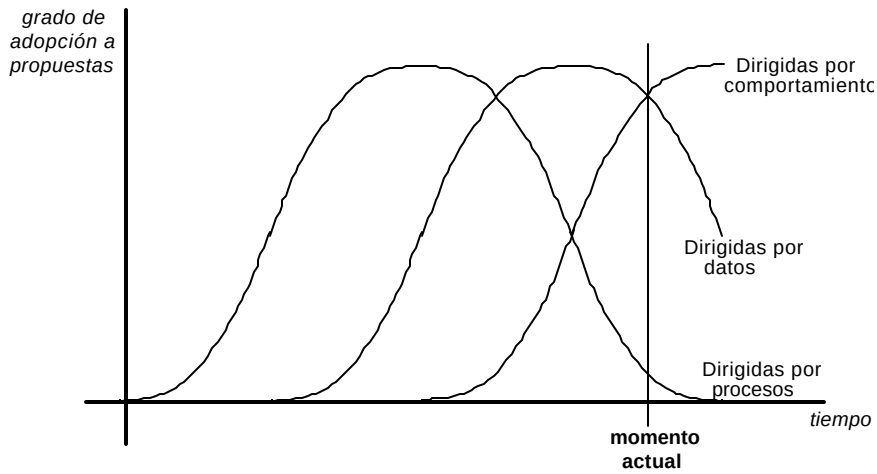

Figura 1 - Evolución histórica de las estrategias del MOO.

Después surge la estrategia dirigida por datos (a partir de fines de los 80). La mayoría de las propuestas vigentes de análisis orientado a objetos opta por esta estrategia. El rasgo principal es la mayor importancia relativa dada a la identificación de clases y sus asociaciones estáticas y a la organización de clases en jerarquías de herencia. Actuando de esta forma, las técnicas permiten capturar primero la estática y luego la dinámica del sistema (subordinando ésta a aquélla) con lo cual la dimensión dinámica recibe generalmente un menor énfasis que la dimensión estructural.

Con relación a esta sobrevaloración de la dimensión estática, y por tanto de la estrategia dirigida por datos, existe una tendencia al cambio. El principio de que el comportamiento del sistema de objetos depende de la estructura del mismo ha sido cuestionado. El aspecto comportamental del sistema es tanto o más importante que el aspecto estático. La mayoría de las críticas atribuidas al MOO se deben en gran medida, al uso generalizado de esta estrategia, que era tomada como *la forma* de modelar orientado a objetos. Esto también se refleja en la literatura, donde es común encontrar términos generales como "modelo de objetos", "modelo orientado a objetos" o "modelo del análisis orientado a objetos" refiriéndose en realidad sólo al modelo estructural orientado a objetos. Sin embargo, su adopción se está dejando de lado en favor de la estrategia dirigida por comportamiento, como puede apreciarse en la figura 1.

La aparición de la estrategia dirigida por comportamiento es relativamente más reciente (a partir del inicio de los 90). Cabe destacar el modelo de casos de uso de Jacobson, como una contribución significativa a esta estrategia, ya que éste también puede ser entendido en una perspectiva dinámica, porque un caso de uso se describe en términos de escenarios de interacción entre el actor y el sistema.

La idea de que la estructura del sistema de objetos es determinada por el comportamiento esperado del sistema está logrando una mayor aceptación. Muchos investigadores y consultores sugieren usar este tipo de modelos de interacción global antes de desarrollar la estructura del sistema de objetos, como se indica, por ejemplo, en [Hills&Tornier95], [Jacobson99] y [Booch99]. La curva que representa el desarrollo de esta estrategia está en crecimiento [Jacobson&Christerson95b], de acuerdo a la figura 1.

Conclusiones

Bajo los argumentos aquí presentados, el MOO debe ser considerado como una alternativa más para hacer frente al análisis de sistemas, con una fuerte restricción impuesta a priori de implementar orientado a objetos. Con el fin de obviar esta restricción se sugiere modelar usando técnicas que posterguen el encapsulamiento, es decir, que permitan modelar fuera del paradigma de la orientación a objetos y que, eventualmente y si así se desea, sea posible realizar un mapeamiento a un modelo de objetos, momento en el cual se iniciaría el diseño orientado a objetos. Esta evolución del pensamiento sobre el MOO, se ve reflejada en la reciente propuesta de utilizar el modelo de casos de uso como conductor del proceso de desarrollo de software orientado a objetos [Jacobson99], es decir, un modelo no orientado a objetos sirve como guía o marco para el resto de los modelos orientados a objetos.

En el caso de optar por un análisis orientado a objetos, se debe estar consciente de que en realidad se está construyendo un modelo de diseño inicial o preliminar del sistema. Otros términos posibles para el MOO en esta perspectiva son los mostrados en la tabla 1. Los términos para el análisis orientado a objetos se contraponen a los del diseño orientado a objetos, reconociendo en ambos el carácter de diseño.

Tabla 1 – Términos alternativos para denominar al Análisis y Diseño Orientados a Objetos.

ANÁLISIS ORIENTADO A OBJETOS	DISEÑO ORIENTADO A OBJETOS
Diseño lógico	Diseño físico
Diseño ideal	Diseño real
Diseño esencial	Diseño de implementación
Diseño de alto nivel	Diseño de bajo nivel

También se debe mencionar la importancia de contar con modelos que proporcionen una visión funcional del sistema de objetos. Este modelo funcional (modelado *end-to-end*) es consecuente con el encapsulamiento. Permite mejorar la comprensión de algunas secuencias más complejas de transformación.

Finalmente, y a modo de síntesis de estas conclusiones, se sugiere optar por técnicas de modelado que no introduzcan prematuramente el encapsulamiento, pero que sin embargo permitan una transición sin tropiezos al diseño e implementación orientados a objetos, faciliten la construcción de visiones funcionales de los sistemas y que valoricen debidamente el comportamiento con relación a la estructura de los sistemas. Las técnicas que mejor se aproximan a estos requerimientos son las que se proponen bajo la estrategia dirigida por comportamiento.

Bibliografía

[Bailin89] Bailin, S. An Object-Oriented Requirements Specification Method. **Communications of the ACM**, New York, v.32, n.5, p.608-623, May 1989.

[Bohm92] Bohm, D. **A Totalidade e a Ordem Implicada**. São Paulo: Cultrix, 1992.

[Booch99] Booch, G. et al. **The Unified Modeling Language**. Reading: Addison-Wesley, 1999.

[Bustos96] Bustos, G. **Uma Proposta de Modelagem Conceitual de Sistemas Dirigida por Comportamento**. Tesis para optar al grado de Doctor en Ciencia de Computación. Porto Alegre: CPGCC, 1996. (en portugués)

[Capra92] Capra, F. **O Ponto de Mutação**. São Paulo: Cultrix, 1992.

[Coleman94] Coleman, D. et al. **Object-Oriented Development:** The Fusion Method. Englewood Cliffs: Prentice-Hall, 1994.

[Constantine89] Constantine, L. Object-Oriented and Structured Methods: Toward Integration. **American Programmer**, New York, v.2, n.7/8, Aug. 1989.

[deChampeaux91] de Champeaux, D. Object-Oriented Analysis and Top-Down Software Development. In: America P. (Ed.). **European Conference On Object-Oriented Programming**

(ECOOP '91), 1991, Geneva. Proceedings... Geneva: Springer-Verlag, 1991. p.360-376.

[deChampeaux92] de Champeaux, D. et al. The Process of Object-Oriented Design. **ACM SIGPLAN Notices**, New York, v.27, n.10, p.45-62, Oct. 1992.

[deChampeaux93] de Champeaux, D. et al. **Object-Oriented System Development**. Reading: Addison-Wesley, 1993.

[deMarco78] de Marco, T. **Structured Analysis and System Specification**. New York: Yourdon Press, 1978.

[Heisenberg85] Heisenberg, W. **La Imagen de la Naturaleza en la Física Actual**. Madrid: Orbis, 1985.

[Henderson-Sellers&Edwards94] Henderson-Sellers, B.; Edwards, J. **BOOKTWO of Object-Oriented Knowledge**: The Working Object. Sydney: Prentice-Hall, 1994.

[Hills&Tornier95] Hills, N.; Tornier, P. An Object-Oriented Approach to "Architecting" Open Client/Server Applications. **Object Magazine**, New York, v.4, n.9, p.51-56, Feb. 1995.

[Hoydahlsvik&Sindre93] Hoydahlsvik, G.; Sindre, G. On the Purpose of Object-Oriented Analysis. **ACM SIGPLAN Notices**, New York, v.28, n.10, Oct. 1993.

[Jacobson92] Jacobson, I. et al. **Object-Oriented Software Engineering**: A Use Case Driven Approach. Wokingham: Addison-Wesley, 1992.

[Jacobson&Christerson95a] Jacobson, I.; Christerson, M. A Growing Consensus on Use Cases. **Journal of Object-Oriented Programming**, New York, v.8, n.1, p.15-19, Jan. 1995.

[Jacobson&Christerson95b] Jacobson, I.; Christerson, M. Modeling with Use Cases: A Confused World of OOA and OOD. **Journal of Object-Oriented Programming**, New York, v.8, n.5, p.15-20, Sept. 1995.

[Jacobson99] Jacobson, I. et al. **The Unified Software Development Process**. Reading: Addison-Wesley, 1999.

[Jalote89] Jalote, P. Functional Refinement and Nested Objects for Object-Oriented Design. **IEEE Transactions on Software Engineering**, New York, v.15, n.3, p.264-270, Mar. 1989.

[Larousse96] **El Pequeño Larousse**. México: Larousse, 1996.

[Loy90] Loy, P. A Comparison of Object-Oriented and Structured Development Methods. In: Thayer, R.; Dorfman, M. (eds.) **System and Software Requirements Engineering**, Los Alamitos: IEEE Computer Society Press, 1990.

[Martin&Odell92] Martin, J.; Odell, J. **Object-Oriented Analysis and Design**. Englewood Cliffs: Prentice-Hall, 1992.

[Monarchi&Puhr92] Monarchi, D.; Puhr, G. A Research Typology for Object-Oriented Analysis and Design. **Communications of the ACM**, New York, v.35, n.9, p.35-47, Sept. 1992.

[Montgomery94] Montgomery, S. **Object-Oriented Information Engineering**. Cambridge: AP Professional, 1994.

[Opdahl&Sindre93] Opdahl, A.; Sindre, G. Concepts for Real-World Modelling. In: Colette, R. et al. (eds.). **Lecture Notes in Computer Science Nº 685**, Springer-Verlag: Berlin, p.309-327, 1993.

[Rubin&Goldberg92] Rubin, K.; Goldberg, A. Object Behavior Analysis. **Communications of the ACM**, New York, v.35, n.9, p.48-62, Sept. 1992.

[Rumbaugh91] Rumbaugh, J. et al. **Object-Oriented Modeling and Design**. Englewood Cliffs: Prentice-Hall, 1991.

[Rumbaugh99] Rumbaugh, J. et al. **The Unified Modeling Language Reference Manual**. Reading: Addison-Wesley, 1999.

[Schiel&Mistrik90] Schiel, U.; Mistrik, I. Using Object-Oriented Analysis and Design for Integrated Systems. **Arbeitspapeiere der GMD**, v.449, Jun. 1990.

[Schneider&Winters98] Schneider, G.; Winters, J. **Applying Use Cases**: A Practical Guide. Reading: Addison-Wesley, 1998.

[Seidewitz89] Seidewitz, E. General Object-Oriented Software Development: Background and Experience. **The Journal of Systems and Software**, v.9, n.2, p.95-108, Feb. 1989.

[Shlaer&Mellor92] Shlaer, S.; Mellor, S. **Object Life Cycles**: Modeling the World in States. Englewood Cliffs: Prentice-Hall, 1992.

[Shlaer&Mellor93] Shlaer, S.; Mellor, S. A Deeper Look... at the Transition from Analysis to Design. **Journal of Object-Oriented Programming**, New York, v.6, n.1, p.16-19, Jan. 1993.

[Taylor95] Taylor, D. **Business Engineering with Object Technology**. New York: Wiley, 1995.

[Rational99] Rational Corporation. **Unified Modeling Language**. 1999. (Disponible vía web en http://www.rational.com/uml/index.jtmpl)

[Ward89] Ward, P. How to Integrate Object Orientation with Structured Analysis and Design. **IEEE Software**, Los Alamitos, v.6, n.3, Mar. 1989.