Técnicas de Análisis Orientado a Objetos: Clasificación y Evaluación

Guillermo Bustos R.
Escuela de Ingeniería Industrial
gbustos@ucv.cl

Resumen

Hoy en día existen diversas técnicas propuestas de análisis de sistemas con orientación a objetos. Estas diferentes técnicas poseen diferentes historias (algunas se originan a partir de técnicas de diseño computacional orientado a objetos, otras surgieron autónomamente como técnicas de análisis, independiente de la forma de implementación) y diferentes enfoques (énfasis en el modelado de datos o el énfasis en el modelado de procesos, por ejemplo). Para permitir la comparación entre diferentes técnicas de análisis, en este trabajo se presentan diversos criterios que permiten distinguir y clasificar las técnicas. Con base en estos criterios, técnicas representativas de las diferentes clases son brevemente evaluadas, así como también el análisis orientado a objetos como técnica de modelado.

1 Introducción

El paradigma de la orientación a objetos ha logrado una gran difusión en el área de análisis de sistemas de información. Han sido propuestas diversas técnicas de *Análisis Orientado a Objetos* (AOO), muchas de ellas ya documentadas en la forma de libros. Para el lector no familiarizado con esta área le es tremendamente difícil distinguir, de entre la gran variedad de propuestas, cuáles son los aspectos efectivamente relevantes y diferenciadores de cada propuesta. Para permitir una comparación entre las técnicas de AOO es necesario establecer un sistema de clasificación que destaque propiedades comunes de las diferentes técnicas.

En la literatura es posible encontrar algunas formas muy generales de clasificar técnicas de AOO. En [Capretz&Lee92] se propone una clasificación que distingue dos direcciones en las técnicas de orientación a objetos. Una es la *adaptación*, donde técnicas establecidas y conocidas de desarrollo estructurado de sistemas son combinadas para formar una nueva técnica de AOO. La otra corresponde a la *asimilación*, donde la tendencia es modificar solamente las técnicas de análisis, diseño y implementación para la orientación a objetos, preservando eso sí las líneas básicas del ciclo de vida tradicional.

En [Monarchi&Puhr92] se describe una clasificación que utiliza como criterio básico la forma y el grado en que las técnicas orientadas a objetos incorporan otros paradigmas. Son identificados tres enfoques:

- *Enfoque Combinativo*: Es aquél en que diferentes técnicas son usadas para modelar diferentes perspectivas de la realidad (estática, funcional y de control).
- *Enfoque Adaptativo*: Es aquél en que las técnicas existentes son usadas de una nueva manera (orientada a objetos), o extendidas para incluir la orientación a objetos. En este enfoque es posible adaptar procedimientos y notaciones que ya probaron ser útiles, en la perspectiva del nuevo paradigma.
- *Enfoque Puro*: Es aquel en que nuevas técnicas son concebidas para modelar la multidimensionalidad de la orientación a objetos.

En el presente trabajo se define una clasificación más amplia y dirigida específicamente al problema de la comparación de técnicas de AOO.

2 Las perspectivas del AOO

En el contexto del desarrollo de sistemas de software con orientación a objetos, se entiende por *Análisis Orientado a Objetos* al proceso de construcción de modelos del dominio del problema, identificando y especificando un conjunto de objetos semánticos que interactúan y se comportan de acuerdo a los requerimientos del sistema. Los objetos semánticos son aquellos que poseen un significado específico en el dominio del problema, según [Monarchi&Puhr92].

De acuerdo a esta definición, el AOO es esencialmente basada en modelado. Es razonable esperar entonces, que la especificación resultante de la aplicación de técnicas de AOO resulte en múltiples modelos y múltiples notaciones. En esta perspectiva, el proceso de construcción de los modelos del dominio del problema debe considerar diferentes aspectos o puntos de vista. Estos aspectos constituyen las dimensiones del modelado orientado a objetos.

El modelado orientado a objetos comprende, como mínimo, dos aspectos relativamente ortogonales o dimensiones para describir un sistema complejo: la dimensión estructural de los objetos y la dimensión dinámica del comportamiento. Puede ser considerada también una dimensión adicional: la dimensión funcional de los requerimientos.

La dimensión estructural de los objetos se centra en el aspecto estático o pasivo. Está relacionada con la estructura estática de los objetos que forman parte del sistema. La estructura incluye la identidad de cada objeto, su clasificación, su encapsulamiento (sus atributos y sus operaciones) y sus relacionamientos estáticos (jerarquías de herencia, agregación, composición y asociaciones específicas).

La dimensión dinámica del comportamiento tiene que ver con el aspecto dinámico o activo, por esto describe el comportamiento y la colaboración de los objetos que constituyen el sistema. El comportamiento es reflejado por medio de estados (pasos dentro del ciclo de vida del objeto que caracterizan comportamientos diferentes del mismo), transiciones entre estos estados, eventos (hechos que ocurren y que producen las transiciones) y acciones (representadas por los métodos de los objetos, pudiendo ocurrir durante las transiciones o durante la permanencia en los estados). La colaboración es representada por modelos que muestran el flujo de eventos o mensajes entre los objetos. Así algunas acciones generadas en un objeto pueden gatillar transiciones, bajo la forma de eventos, en otros objetos.

Para el caso de la dimensión funcional de los requerimientos es considerado el aspecto relativo a la función de transformación global del sistema, es decir, a la conversión de entradas en salidas. Esta transformación global es reflejada por procesos o funciones (que transforman valores) y flujos de datos (entradas y salidas de estas funciones), configurando con esto redes funcionales, a través de un proceso de refinamiento sucesivo o *top-down*. Este modelado es claramente opuesto al concepto de encapsulamiento de métodos en los objetos, por esto existe una controversia en la literatura sobre la conveniencia o no de utilizarlo. Algunos autores sugieren el modelado de procesos denominado *end-to-end* como una alternativa para abordar este aspecto (ver [Bailin89], [Jalote89] y [Fichman&Kemerer92]).

La distinción anterior de las dimensiones no significa que necesariamente sean construidos modelos separados para cada aspecto. Puede ser modelado más de un aspecto simultáneamente, sin perjuicio de los elementos mencionados en cada dimensión. Por ejemplo en la figura 1, donde se muestra una analogía de la tridimensionalidad espacial aplicada a las dimensiones del modelado, una determinada técnica puede sugerir como procedimiento seguir la secuencia **a** (funcional), **b** (estático) y **c** (dinámico), así como otra puede sugerir la secuencia **1** (estático) y **2** (dinámico-funcional).

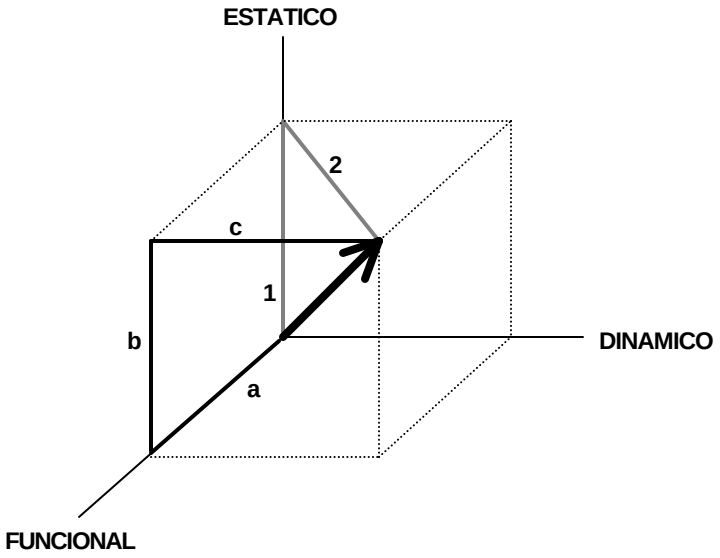

Figura 1 - La tridimensionalidad del modelado de sistemas.

Es posible afirmar que las tres dimensiones descritas muestran aspectos que son importantes para el modelado orientado a objetos y por tanto deberían ser consideradas por las técnicas que observan este paradigma.

3 Una Clasificación Para las Técnicas de AOO

La clasificación propuesta utiliza como criterio básico el origen de la técnica, es decir, el conjunto de conceptos a partir del cual se originó la técnica. Además, es considerado el énfasis que las téc-

nicas de AOO otorgan a los conceptos, aspectos, procedimientos o representaciones en cada una de las dimensiones del modelado.

Las categorías usadas para clasificar las técnicas son: textuales, evolutivas, integracionistas, reversas y comportamentales. La figura 2 muestra la estructura de esta clasificación.

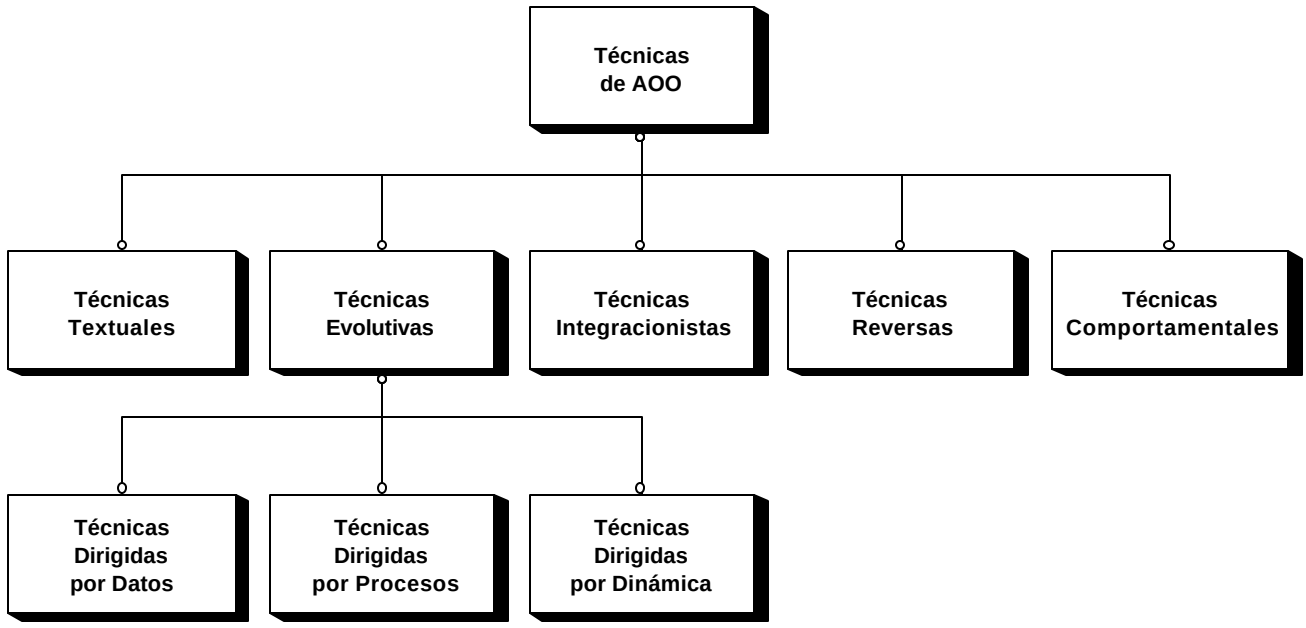

Figura 2 - La estructura de la clasificación propuesta para las técnicas de AOO.

3.1 Las Técnicas Textuales

Las técnicas denominadas textuales son aquellas que se basan en descripciones informales, pero precisas, escritas en lenguaje natural para identificar objetos, atributos y operaciones tanto del dominio del problema como del dominio de la solución, a través de un análisis sintáctico de sustantivos, adjetivos, verbos y adverbios.

Las técnicas de esta categoría tienen su origen fuera del paradigma de la orientación a objetos, específicamente en el trabajo de [Abbott83], que propone diseñar programa en *Ada* a partir de descripciones informales en inglés. Sin embargo, estas técnicas son insuficientes para abordar problemas más complejos y pueden ser consideradas como sobrepasadas. Se consideran aquí sólo por su relevancia histórica.

Como técnica representativa puede ser indicada la propuesta descrita por ejemplo en [Pressman87][1]. Otras técnicas textuales pueden ser encontradas en propuestas de [Booch83], de la empresa EVB Software Engineering [EVB86] y de [Mattoso&Blum88].

3.2 Las Técnicas Evolutivas

Las técnicas evolutivas son aquellas producto de la extensión o evolución de técnicas dirigidas por alguna de las dimensiones del modelado (estructural, dinámica y/o funcional) y su complementación con otros aspectos del modelado. Esta categoría de técnicas puede ser subdividida según la dimensión "dominante" en dirigidas por datos, dirigidas por procesos y dirigidas por dinámica.

Sin duda esta es la categoría más poblada por razones obvias: las técnicas originales son todas ampliamente conocidas y probadas en el desarrollo de sistema de software. Entonces parece natural intentar extenderlas para la orientación a objetos.

3.2.1 Las Técnicas Dirigidas por Datos

Esta subcategoría incluye las técnicas que utilizan extensiones semánticas de modelos de datos o el denominado modelado de información. El modelo de datos más ampliamente utilizado por su divulgación y carácter intuitivo es el modelo entidad-relacionamiento extendido.

Como la técnica más representativa en esta subcategoría puede ser indicada *Object-Oriented Analysis* (OOA) de [Coad&Yourdon92]. La técnica propone cinco actividades principales que resultan en un modelo multicamadas, donde cada camada es construida sobre la camada anterior. Las actividades son:

1) ubicación de clases y objetos;
2) identificación de estructuras;
3) identificación de asuntos;
4) definición de atributos y
5) definición de servicios.

Posteriormente a la divulgación de la técnica han sido propuestas algunas extensiones. Entre ellas cabe destacar la de [Schaschinger92] denominada ESA (*Expert Supported* OOA) que mejora el aspecto dinámico y la del propio [Yourdon94] en que también redefine el aspecto dinámico de esta técnica.

Otras técnicas que pueden ser clasificadas en esta subcategoría son: los modelos Entidad-Relacionamiento Orientados a Objetos (OOER) propuestos por [Navathe&Pillalamarri89] y [Manfredi89]. También las propuestas de [Kurtz89], [Coleman94], [Shlaer&Mellor90], [Dillon&Tan93] y [Martin& Odell92] pueden ser consideradas como dirigidas por datos.

---

[1] En realidad, en [Pressman87] no se propone la autoría de una técnica textual de AOO, sino que más bien se sintetiza las técnicas existentes hasta ese momento en un procedimiento y notaciones mínimas asociadas, que constituyen la técnica referida en este trabajo.

3.2.2 LAS TÉCNICAS DIRIGIDAS POR PROCESOS

Esta subcategoría incluye las técnicas que utilizan extensiones de modelos funcionales con descomposición funcional. El modelo funcional más ampliamente utilizado, también por su divulgación y carácter intuitivo, es el diagrama de flujo de datos (DFD).

Como la técnica más representativa en esta subcategoría puede ser indicada *Object-Oriented Requirements Specifications Method* de [Bailin89]. Este método de AOO se basa en los *Entity Data Flow Diagrams* que representan tanto entidades (objetos) como funciones (métodos). El método propone los siguientes pasos:

1) identificar las entidades (objetos) claves en el dominio del problema;
2) distinguir entre entidades activas y pasivas;
3) establecer flujos de datos entre las entidades activas;
4) descomponer entidades (o funciones) en subentidades y/o funciones;
5) buscar nuevas entidades;
6) agrupar las funciones bajo las nuevas entidades; y
7) definir dominios apropiados para las entidades.

Es importante destacar que la propuesta no utiliza una descomposición funcional pura, sino que particiona entidades en funciones y subentidades y las funciones en subfunciones. Así las funciones siempre pertenecen a una entidad de más alto nivel, sin contrariar el principio de encapsulamiento.

Otras técnicas que análisis y diseños orientados a objetos que utilizan DFD son las de [Booch86], [Seidewitz89], [Alabiso88], [Bulman89] y [Lee&Carver91].

3.2.3 LAS TÉCNICAS DIRIGIDAS POR DINÁMICA

Esta subcategoría incluye las técnicas que utilizan extensiones de modelos dinámicos de alguna especie. Los modelos dinámicos más utilizados son los diagramas de transición de estados, los *statecharts* de [Harel87] y las redes de Petri [Heuser90].

Como técnica representativa de esta subcategoría puede ser indicada la propuesta de [Kappel& Schrefl91], que corresponde a una variante de diagrama de transición de estado y redes predicado/transición adaptada para la orientación a objetos.

Otra técnica clasificable como evolutiva dirigida por dinámica es la de [Schiel&Mistrik90], denominada OKAY (*Object Oriented Knowledge Analysis and Design*).

3.3 Las Técnicas Integracionistas

Esta categoría representa a aquellas técnicas que integran modelos separados de las diferentes dimensiones.

Como técnica representativa de esta categoría se encuentra la de [Rumbaugh91]. Los autores proponen una técnica de desarrollo de software orientado a objetos denominada OMT (*Object Modeling Technique*), que incluye explícitamente el AOO como la construcción de tres modelos, uno para cada dimensión, que especifiquen el dominio del problema considerando los requerimientos. El procedimiento del AOO está íntimamente ligado a la construcción de modelos de estos tres aspectos: modelado estructural, modelado dinámico y modelado funcional. El orden en que debe ser realizado el modelado es: 1) objetos, 2) dinámica y 3) funcionalidad.

Para el modelado de objetos se utiliza una extensión del modelo entidad-relacionamiento. En el modelado dinámico es utilizada una variante de los *statecharts*. En el modelado funcional son usados DFD extendidos con flujos de control. La integración final de estos modelos, por ejemplo la asociación de las funciones (del modelo funcional) y las acciones (del modelo dinámico) a los objetos, es hecha en una fase posterior denominada diseño de objetos.

La técnica OMT se ha transformado en una de las técnicas más divulgadas, existiendo por ejemplo diferentes herramientas CASE (*Computer-Aided Software Engineering*) que incluyen su notación. También ha sido objeto de evaluaciones como por ejemplo la de [Hayes&Coleman91] y la de [Bruegge92].

Otra técnica en esta misma categoría es la de [Shlaer&Mellor89] y [Shlaer&Mellor92] que extienden la propuesta inicial de los mismos autores presentada en [Shlaer&Mellor90], referenciada en este trabajo como técnica evolutiva dirigida por datos. También pueden ser consideradas como técnicas integracionistas las propuestas de [Clyde92] y [Embley92].

3.4 Las Técnicas Reversas

Las técnicas reversas son aquellas originadas a partir de necesidades de implementación, como por ejemplo el soporte a conceptos de lenguajes de programación orientados a objetos específicos (por ejemplo *Smalltalk*, C++, *Eiffel* o *Ada*[2]). Esta categoría puede ser fácilmente confundida con otras, pues al soportar conceptos de un lenguaje de programación orientado a objetos puede ser apropiado utilizar enfoques, notaciones y procedimientos de otra naturaleza.

La técnica escogida como más representativa es la propuesta por [Nerson92], que define una técnica de análisis y diseño orientados a objetos para el lenguaje Eiffel [Meyer88], usando la notación de objetos mejorada (*Better Object Notation* o BON). Esta técnica consiste en tres pasos:

1) identificación, designación y *clustering* de clases;
2) identificación de eventos y protocolos de comunicación entre objetos; y
3) definición de clases y diseño preliminar de la arquitectura básica.

La técnica se basa en conceptos de lenguaje Eiffel (*cluster*, relacionamiento cliente/proveedor, clases diferidas y modelos estático/dinámico), la técnica *OBA* (presentada en la próxima sección) de [Rubin&Goldberg92] y los conceptos de tarjetas de clases de [Beck&Cunningham89].

---

[2] Este es un caso singular, ya que *Ada*, que no es un lenguaje de orientación a objetos pues carece de soporte para la herencia, es el lenguaje para el cual históricamente han sido propuestas más técnicas de análisis y diseño orientados a objetos.

Otra técnica que puede ser clasificada en esta categoría es la técnica más reciente de [Booch91] que ha evolucionado desde un método específico de diseño para el lenguaje *Ada* hasta una propuesta más general. También en la línea del lenguaje *Ada* se encuentran las propuestas de análisis de requerimientos orientados a objetos de [Freitas90] y de [Ladden88].

Una propuesta que toma una perspectiva diferente es la de [Reenskaug92], que propone la técnica *Object-Oriented Role Analysis Synthesis and Structuring* o OORASS, dirigida inicialmente al desarrollo para el lenguaje Smalltalk-80.

3.5 Las Técnicas Comportamentales

Las técnicas comportamentales reúnen técnicas en las cuales los objetos son derivados a partir del comportamiento externo que debe exhibir el sistema. Actuando de esta forma, en general se pospone el encapsulamiento de los atributos y/o métodos en los objetos hasta más adelante en el procedimiento, porque inicialmente interesa el comportamiento global del sistema y la interacción de componentes al interior del sistema que satisfacen este comportamiento global. Estos componentes serán potenciales objetos del sistema.

El mejor representante de esta categoría es *Object-Oriented Software Engineering* (OOSE) de [Jacobson92] (una simplificación de la metodología *Objectory* de los mismos autores) que contempla la fase de análisis. Para esta fase, OOSE sugiere los siguientes pasos:

1) identificación de actores;
2) construcción de los casos de uso (*use case model*);
3) descripción de las interfaces;
4) modelado de objetos del dominio del problema;
5) refinamiento del modelo de requerimientos; y
6) construcción del modelo de análisis.

El modelo de casos de uso es el modelo fundamental de esta técnica. Con los actores (papeles de usuarios del sistema) identificados se construye un modelo que describe un aspecto de la funcionalidad del sistema. Este modelo representa diversas maneras específicas de usar el sistema. Cada caso de uso describe un escenario completo de eventos iniciados por un actor y especifica la interacción que ocurre entre el actor y el sistema.

Este modelo está ganando cada vez mayor aceptación para describir el comportamiento esperado del sistema al ser visto externamente e incluso se sugiere su incorporación a otras técnicas de AOO no comportamentales (ver por ejemplo [Hills&Tornier95] y [Jacobson&Christerson95]).

También dentro de esta categoría está la técnica denominada *Object Behavior Analysis* (OBA) propuesta por [Rubin&Goldberg92], que tiene como antecedente la técnica propuesta por [Gibson90]. Otra técnica vinculada a esta categoría, pero que corresponde más a diseño que a análisis, es *Responsibility-Driven Design* de [Wirfs-Brock90].

4. Evaluación de las Técnicas de AOO

Inicialmente son evaluadas las técnicas por categoría de la clasificación propuesta. Después se discuten la fortaleza y las debilidades que las técnicas de AOO en general presentan.

4.1 Las Críticas a las Categorías de AOO

Las técnicas textuales están definitivamente obsoletas como técnicas de AOO por sí mismas, pero pueden auxiliar en los pasos iniciales de otras técnicas.

Las técnicas evolutivas dirigidas por datos son las más desarrolladas, gracias al bagaje que les proporciona el modelado semántico de datos. Los modelos son más estables y ya poseen un considerable consenso en los procedimientos y una relativa uniformidad en las notaciones. Estas técnicas son incluso recomendadas por autores de prestigio del área de análisis y diseño estructurados, tales como Edward Yourdon [Yourdon94], Meilir Page-Jones [Page-Jones90] y Larry Constantine [Constantine90]. Sin embargo, su aplicabilidad exige la construcción de modelos en los otros aspectos de modelado. Estas técnicas parecen ser más apropiadas para dominios de problemas que requieren un especial énfasis en el modelado de una base de datos.

Las técnicas evolutivas dirigidas por procesos se encuentran en algún punto de la transición entre las técnicas estructuradas y la técnicas orientadas a objetos. Parecen ser más útiles en dominios de problemas que presentan muchos cálculos (numerosas funciones de transformación y conversión) como por ejemplo aplicaciones matemáticas o científicas.

Las técnicas evolutivas dirigidas por dinámica tienen un gran campo para expandirse si se considera el hecho de que las aplicaciones actuales dan cada vez mayor énfasis a las interfaces gráficas del usuario (*Graphical User Interface* o GUI) y que los dominios son cada vez más enfocados a la dinámica de la interacción con agentes externos al sistema de software. Sin embargo, las propuestas revisadas aun carecen de la madurez y simplicidad que demanda la aplicación práctica.

Las técnicas integracionistas se proponen una mayor aplicación en dominios diferentes. Dependiendo del tipo de problema, el énfasis en determinados aspectos del modelado puede ser mayor o menor. La mayor dificultad aparece a la hora de integrar visiones que pueden ser muy independientes. Esto dificulta la transición a la fase de diseño e implementación.

Las técnicas reversas buscan una mayor eficiencia en el desarrollo usando lenguajes específicos de programación orientados a objetos, sacrificando la generalidad que las aplicaciones requieren. Pueden constituir alternativas válidas cuando la decisión sobre la implementación ya está tomada al momento de iniciar el desarrollo.

Finalmente, las técnicas comportamentales aportan un concepto interesante: la definición de las responsabilidades de los objetos al interior de los sistemas. La estrategia de identificar objetos, y por tanto sus responsabilidades, a partir del comportamiento esperado o deseado del sistema es muy impor-

tante en los pasos iniciales del análisis. Estas técnicas debieran tener cada vez una mayor aceptación en el concierto de técnicas de AOO[3].

4.2 Principal Fortaleza de las Técnicas de AOO

El aspecto más consolidado en la mayoría de las técnicas y que constituye su principal fortaleza es el modelado de la dimensión estructural de los objetos. Como fue indicado anteriormente la causa de esto es que la mayoría de las propuestas utilizan los conceptos del modelado semántico en base a modelos extendidos entidad-relacionamiento. Estas técnicas poseen un desarrollo de más de 20 años a partir del trabajo original de [Chen76].

En este sentido la identificación y especificación de objetos, clases, métodos, atributos, asociaciones dependientes del dominio y jerarquías de herencia es la principal fortaleza de las técnicas de AOO.

4.3 Debilidades de las Técnicas de AOO

Como en cualquier tecnología emergente las debilidades que presenta son mucho más numerosas que sus fortalezas. En este sentido el AOO no es una excepción.

Como principales debilidades pueden ser indicadas las siguientes:

- *Criterios consistentes de particionamiento de la complejidad:* Deben existir criterios objetivos y prácticos que permitan agregar clases o particionar sistemas. Las primeras preguntas que surgen son: ¿El particionamiento inicial debe ser en la perspectiva de la organización, de la cual el sistema de información forma parte (sistemas y subsistemas) o en la perspectiva del software orientado a objetos (*clusters*, clases y objetos)? ¿son incompatibles estas visiones? En este sentido también es válido cuestionar si el particionamiento debe ser realizado antes o después de la identificación de las clases, o en otras palabras, si el sistema es particionado o las clases son agregadas. Considerando esta última alternativa, existen en la literatura diversas propuestas entre las cuales pueden ser indicadas: los asuntos (*subjects*) de [Coad&Yourdon92], los *clusters* de [Nerson92], los dominios (*domains*) de [Bailin89], los subsistemas (*subsystems*) de [Shlaer&Mellor92] y de [Jacobson92], los módulos (*modules*) de la técnica OMT de [Rumbaugh91], los objetos compuestos (*composite objects*) de [Booch91], las visiones de alto nivel (*high-level views*) de [Embley92] y los *ensembles* de [deChampeaux93]. Este último concepto permite el agrupamiento de clases en estructuras de agregación y composición de manera análoga a los objetos, pudiendo presentar atributos, métodos, estados y transiciones, relacionamiento y clasificación, pero con la diferencia fundamental de que los *ensembles* pueden presentar paralelismo interno mientras que los objetos no. La única propuesta alternativa para particionar antes de la identificación completa de las clases y objetos son las áreas de interés (*areas of concern*) de la técnica OORASS de [Reenskaug92].

[3] Para mayores detalles sobre esta tendencia ver [Bustos97], donde la noción de una categoría de técnicas comportamentales de AOO es generalizada para una estrategia dirigida por comportamiento para el modelado orientado a objetos.

- *Reutilización de la especificación:* Esta cuestión aún no está debidamente tratada en la literatura. Las propuestas de AOO más aventuradas sólo recomiendan estudiar otras especificaciones en dominios iguales o semejantes como [Coad&Yourdon92], sin embargo no proveen ningún mecanismo concreto para identificar y evaluar las clases potenciales para la reutilización. Esta debilidad puede parecer paradojal si se considera el hecho de que una de las principales ventajas argumentadas en favor de la tecnología de objetos es su mayor de grado de reutilización. En las fases de diseño y particularmente de implementación, este concepto está más desarrollado que en el análisis. La reutilización en el AOO (también denominado reutilización *harvesting* [Fichman&Kemerer92]) puede tomar dos formas: como reutilización de especificaciones previas de análisis y diseño orientados a objetos o como abstracciones de componentes de programas ya implementados (ingeniería reversa). La reutilización a nivel de especificación parece particularmente difícil porque el AOO trata con conceptos de dominios de problemas y no con componentes de software (aunque ambos se expresen finalmente como objetos), por tanto la máxima reutilización ocurre al interior de dominios específicos de aplicación, como por ejemplo a través de los *patterns* (ver [Fowler97]).
- *Modelado funcional v/s modelado end-to-end:* La complejidad de las aplicaciones exige modelos funcionales para especificar los requerimientos, pero las técnicas consideran que esto puede influir negativamente en la orientabilidad de los sistemas. La cuestión clave parece ser el criterio de particionamiento en el aspecto funcional. Si el particionamiento es por descomposición funcional (estrategia *top-down* o refinamiento sucesivo) como propuesta en el análisis estructurado tradicional de [deMarco78], éste es daramente contrario a la orientación a objetos. Si las funciones son subordinadas a los objetos, en cuyo caso se volverían métodos, no hay problemas de encapsulamiento y la especificación correspondería a un modelado de procesos *end-to-end*, según [Fichman&Kemerer92], o un modelado funcional encapsulado.
- *Validación del usuario:* la mayoría de las técnicas no considera explícitamente la participación del usuario en el modelado del dominio del problema o en la definición de requerimientos para la solución. Sólo la propuesta de [Coad&Yourdon92] da alguna indicación al respecto. El usuario puede participar en la construcción y validación de los modelos de AOO o a través del uso de prototipos de la aplicación.
- *Estimación o dimensionamiento de sistemas:* la estimación del tamaño de los sistemas considerando alguna métrica es una actividad esencial en los proyectos de ingeniería de software, sin embargo esto aún está en bajo investigación para la tecnología de objetos. En general, en la literatura existe poca vinculación entre este tipo de estimaciones y el modelado y especificación de sistemas. Algunos trabajos recientes en esta área son los de [Chidamber&Kemerer94], [Lorenz&Kidd94] y [Henderson-Sellers97].

Otras debilidades que pueden ser indicadas son las siguientes:

- *Análisis de dominio y la reutilización*: Existe una fuerte vinculación entre el AOO y el concepto de *análisis de dominio* (ver por ejemplo [Arango89]). Las preguntas claves en relación a esta vinculación son: ¿Es válido estudiar el dominio de forma independiente del problema específico ? En este caso ¿Cómo podría ser lograda la reutilización dentro del dominio ? (ver [Pressman97], cap.20, para más detalles).
- *Interacción dinámica de clases*: Las diversas técnicas no son uniformes a este respecto. Algunas identifican las colaboraciones o interacciones directamente en el modelo estructural como [Coad& Yourdon92], mientras que otras proponen un modelo separado para tal efecto como por ejemplo

[Embley92]. La coordinación del comportamiento de las clases es fundamental para una especificación del aspecto dinámico de los sistemas de objetos.

- *Integración de modelos estáticos y dinámicos*: En la mayoría de las propuestas existen dificultades para integrar los aspectos estáticos y dinámicos de los sistemas de objetos. Esta debilidad es una herencia que antecede a la orientación a objetos y aún no está generalmente resuelta en la literatura.
- *Formalización de la especificación*: Puede ser necesario definir formalmente la sintaxis y la semántica de los modelos de software orientados a objetos para garantizar una comprensión y comunicación precisas. Sin embargo, la flexibilidad de la informalidad también es deseable en el ámbito del modelado. La mayoría absoluta de las técnicas más comerciales es informal.
- *Métodos para la evaluación de la calidad de los modelos*: Cuanto mayor sean los sistemas a ser desarrollados con tecnología de objetos, mayor será la necesidad de técnicas y métodos que permitan evaluar, tanto en términos cuantitativos como cualitativos, la calidad de los modelos obtenidos.

En [Askit&Bergmans92] se puede encontrar otras debilidades específicas de diferentes técnicas de AOO.

5 Conclusiones y Comentarios

Fue presentada una clasificación para las técnicas de AOO que considera el origen histórico de cada técnica conjuntamente con la forma de modelado de las diferentes dimensiones. Fueron identificadas las siguientes categorías: textuales, evolutivas dirigidas por datos, procesos y dinámica, integracionistas, reversas y comportamentales. Fue indicada una técnica específica de AOO como la más representativa de cada una de estas categorías.

En términos de evaluación, la fortaleza del AOO es el modelado estructural de objetos. Entre las principales debilidades están el particionamiento de la complejidad multidimensional, la reutilización de la especificación, el modelado funcional, la validación del usuario y el dimensionamiento de los sistemas de objetos.

En términos generales, las técnicas revisadas presentan una relativa falta de madurez, cuestión por de más natural dado el hecho que la idea de AOO es reciente. Por esto, aún no existen técnicas estandarizadas y de amplia aceptación que sustituyan definitivamente las metodologías más tradicionales tales como el análisis estructurado o la ingeniería de la información En relación a la estandarización, la OMG (*Object Management Group*) [OMG97] por medio del denominado UML (*Unified Modeling Language*) se propone unificar notaciones para los modelos orientados a objeto.

Considerando el estado del arte en AOO, las categorías de técnicas comportamentales y las evolutivas dirigidas por datos y por dinámica, en este orden, parecen ser las más prometedoras.

En relación a las técnicas comportamentales, éstas debieran tener un mayor auge en los próximos años debido a la idea de que la estructura del sistema de objetos es determinada por el comportamiento esperado de este sistema. Con esta concepción se evita encapsular prematuramente, asegurando un modelo más eficaz y de mejor calidad (ver [Bustos96] y [Bustos97] para más detalles).

Las técnicas evolutivas dirigidas por datos y por dinámica y las técnicas reversas, en este orden, poseen un buen sustento de modelado y constituyen una buena alternativa a la hora de enfrentar el análisis con la decisión tomada a priori de implementar orientado a objetos. En otras palabras, si al iniciar el desarrollo de una aplicación ya se sabe que será implementada en un lenguaje orientado a objetos, estas técnicas son las más convenientes. Si la aplicación es de tipo comercial, la categoría de técnicas dirigidas por datos es más apropiada. Si la aplicación es de tiempo-real o de tipo distribuido o aún si la interface de usuario es muy sofisticada, la categoría de técnicas dirigidas por dinámica es recomendada. Si se desarrolla software a pedido usando un lenguaje específico orientado a objetos, puede usarse las técnicas reversas.

Por otra parte y considerando la comprensible resistencia de la industria en adoptar nuevas tecnologías, para una primera fase de transición desde las metodologías convencionales hacia la tecnología de objetos, las técnicas integracionistas y evolutivas dirigidas por procesos, en este orden, son las más apropiadas. Las primeras presentan la flexibilidad de darle énfasis gradual a las dimensiones dependiendo del tipo de dominio en que se esté modelando. Las últimas sólo parecieran indicadas para aquellos modeladores que todavía se sienten muy apegados a modelos funcionales.

El AOO ofrece todo su potencial si las siguientes fases también son conducidas bajo el mismo paradigma. En caso contrario, los mapeamientos e traducciones entre las sucesivas fases, como por ejemplo los mapeamientos de modelos de objetos a modelos funcionales o viceversa, podrían transformar el desarrollo innecesariamente complejo. Sin embargo, para las aplicaciones más comerciales donde una base de datos es indispensable en la mayoría de los casos, las técnicas de AOO encuentran un obstáculo tecnológico: no existen ampliamente disponibles bases de datos orientadas a objetos en el mercado, lo cual significa un mapeamiento forzado de los modelos de objetos a modelos relacionales, con toda la pérdida de semántica que esto significa y en particular para la implementación de las operaciones o métodos.

Finalmente una técnica ideal de AOO debería:

- permitir un modelado y especificación precisos y multidimensionales de los sistemas de objetos;
- organizar los modelos en distintos niveles de abstracción en consistencia con todos los aspectos;
- de preferencia no debería forzar el encapsulamiento prematuramente;
- permitir una transición sin obstáculos a las fases de diseño e implementación orientados a objetos;
- facilitar tanto cuanto posible el diseño y/o la implementación no orientados a objetos;
- facilitar la reutilización de los modelos, minimizando el esfuerzo y permitiendo especificar por extensión;
- permitir validar los modelos a través de la construcción rápida de prototipos;
- facilitar la estimación del dimensionamiento de los sistemas de objetos; y
- proporcionar criterios para evaluar la calidad de los modelos resultantes de su aplicación.

Bibliografía

[Abbott83] Abbott, Rusell. Program Design by Informal English Descriptions. **Communications of the ACM**, New York, v.26, n.11, p.882-894, Nov. 1983.

[Alabiso88] Alabiso, B. Transformation of Data Flow Analysis Models to Object-Oriented Design. **ACM SIGPLAN Notices**, New York, v.23, n.11, p.335-353, Nov. 198. Trabajo presentado en la Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA '88, 3., 1988, San Diego.

[Arango89] Arango, Guillermo. Domain Analysis: From Art Form to Engineering Discipline. **ACM SIGSOFT Software Engineering Notes**, New York, v.14, n.3, p.152-159, May. 1989. Trabajo presentado en el International Workshop on Software Specification and Design, 5., 1989, Pittsburgh.

[Askit&Bergmans92] Askit, Mehmet & Lodewijk Bergmans. Obstacles in Object-Oriented Software Development. **ACM SIGPLAN Notices**, New York, v.27, n.10, p.341-358, Oct. 1992. Trabajo presentado en la Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA '92, 7., 1992, Vancouver.

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

[Beck&Cunningham89] Beck, Kent & Ward Cunningham. A Laboratory for Teaching Object-Oriented Thinking. **ACM SIGPLAN Notices**, New York, v.24, n.10, p.1-6, Oct. 1989. Trabajo presentado en la Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA '89, 4., 1989, New Orleans.

[Booch83] Booch, Grady. Object-Oriented Design. In: FREEMAN, Peter; WASERMAN, Anthony (eds.) **Tutorial on Software Design Techniques**, Los Alamitos: IEEE Computer Society Pres, 1983, p.420-436.

[Booch86] Booch, Grady. Object-Oriented Development. **IEEE Transactions on Software Engineering**, New York, v.12, n.2, p.211-221, Feb. 1986.

[Booch91] Booch, Grady. **Object Oriented Design with Applications**. Redwood City: Benjamin/Cummings, 1991.

[Bruegge92] Bruegge, Bernd et al. Object-Oriented System Modeling with OMT. **ACM SIGPLAN Notices**, New York, v.27, n.10, p.359-376, Oct. 1992. Trabajo presentado en la Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA '92, 7., 1992, Vancouver.

[Bulman89] Bulman, David. An Object-Based Development Model. **Computer Language**, v.6, n.8, p.49-59, Aug. 1989.

[Bustos96] Bustos, Guillermo. **Una Proposta de Modelagem Conceitual de Sistemas Dirigida por Comportamento**. Porto Alegre : CPGCC-UFGRS. Tesis para optar al grado de Doctor en Ciencias de Computación, 1996. (En português).

[Bustos97] Bustos, Guillermo. Modelado Orientado a Objetos: Una Evaluación Crítica. Trabajo sometido al **CLEI - Panel '97**, Valparaíso, 1997.

[Capretz&Lee92] Capretz, L. & P. Lee. A Classification of Object-Oriented Development Methodologies. In: SIMPOSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 6., 1992, Gramado. **Anales...** Gramado: II-UFRGS, 1992. p.129-141.

[Chen76] Chen, Peter. The Entity-Relationship Model: Toward an Unified View of Data. **ACM Transactions on Database Systems**, New York, v.1, n.1, p.9-36, Mar. 1976.

[Chidamber&Kemerer94] Chidamber, Shyam & Chris Kemerer. A Metric Suite for Object-oriented Design. **IEEE Transactions on Software Engineering**, New York, v.20, n.6, p.476-493, Jun. 1994.

[Clyde92] Clyde, Stephen et al. Tunable Formalism in Object-Oriented Systems Analysis: Meeting the Needs of Both Theoreticians and Practitioners. **ACM SIGPLAN Notices**, New York, v.27, n.10, p.452-465, Oct. 1992. Trabajo presentado en la Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA '92, 7., 1992, Vancouver.

[Coad&Yourdon92] Coad, Peter & Edward Yourdon. **Análise Baseada en Objetos**. Rio de Janeiro: Campus, 1992.

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

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

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

[deMarco78] deMarco, Tom. **Structured Analysis and System Specification**. New York: Yourdon Press, 1978.

[Dillon&Tan93] Dillon, Tharam & Poh Lee Tan. **Object-Oriented Conceptual Modeling**. Englewood Cliffs: Prentice-Hall, 1993.

[Embley92] Embley, David et al. **Object-Oriented System Analysis: A Model-Driven Approach**. Englewood Cliffs: Prentice-Hall, 1992.

[EVB86] EVB Software Engineering. **Object-Oriented Design Handbook**. Rockville: EVB Software Engineering Inc., 1986.

[Fichman&Kemerer92] Fichman, Robert & Chris Kemerer. Object-Oriented and Conventional Analysis and Design Methodologies: Comparison and Critique. **IEEE Computer**, Los Alamitos, v.25, n.10, p.22-39, Oct. 1992.

[Fowler97] Fowler, Martin. **Analysis Patterns**: Reusable Object Models. Menlo Park: Addison Wesley, 1997.

[Freitas90] Freitas, Maria et al. Object-Oriented Requirements Analysis in an Ada Project. **Ada Letters**, v.10, n.6, p.97-109, Jul./Aug. 1990.

[Gibson90] Gibson, Elizabeth. Objects - Born and Bred. **Byte**, Peterborough, v.15, n.10, p.245-254, Oct. 1990.

[Harel87] Harel, David. Statecharts: A Visual Formalism for Complex Systems. **Science of Computer Programming**, Amsterdam, v.8, n.3, Jun. 1987.

[Hayes&Coleman91] Hayes, Fiona & Derek Coleman. Coherent Models for Object-Oriented Analysis. **ACM SIGPLAN Notices**, New York, v.26, n.11, p.171-183, Nov. 1991. Trabajo presentado en la Annual Conference on Object-Oriented Programming, Systems, Languages, and Applications - OOPSLA '91, 6., 1991, Phoenix.

[Henderson-Sellers97] Henderson-Sellers, Brian. Corrigenda: Software Size Estimation of Object-Oriented Systems. **IEEE Transactions on Software Engineering**, Los Alamitos, v.23, n.4, p.269-276, Apr. 1997.

[Heuser90] Heuser, Carlos. **Modelagem Conceitual de Sistemas: Redes de Petri**. Kapelusz: Buenos Aires, 1990.

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

[Jacobson92] Jacobson, Ivar et al. **Object-Oriented Software Engineering:** An Use-Case Approach. Reading: Addison Wesley, 1992.

[Jacobson&Christerson95] Jacobson, Ivar & Magnus Christerson. A Growing Consensus on Use Cases. **Journal of Object-Oriented Programming**, New York, v.8, n.1, p.15-19, Jan. 1995.

[Jalote89] Jalote, Pankaj. 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.

[Kappel&Schrefl91] Kappel, Gerti & Michael Schrefl. Using an Object-Oriented Diagram Technique for the Design of Information Systems. In: Sol, H. & K. Van Hee (eds.). **Dynamic Modelling of Information Systems**, Amsterdam: North-Holland, 1991, p.121-164.

[Kurtz89] Kurtz, Barry et al. An Object-Oriented Methodology for Systems Analysis and Specification. **Hewlett-Packard Journal**, Palo Alto, v.40, n.2, p.86-90, Apr. 1989.

[Ladden88] Ladden, Richard. A Survey of Issues to be Considered in the Development of an Object-Oriented Development Methodology for Ada. **ACM SIGSOFT Software Engineering Notes**, New York, v.13, n.3, p.24-30, Jul. 1988.

[Lee&Carver91] Lee, Sangbum & Doris Carver. Object-Oriented Analysis and Specification: A Knowledge Base Approach. **Journal of Object-Oriented Programming**, New York, p.35-43, Jan. 1991.

[Lorenz&Kidd94] Lorenz, M. & J. Kidd. **Object-Oriented Software Metrics**. Englewood Cliffs: Prentice-Hall, 1994.

[Manfredi89] Manfredi, F. et al. An Object-Oriented Approach to the System Analysis. In: EUROPEAN SOFTWARE ENGINEERING CONFERENCE - ESEC '89, 2., 1989. **Proceedings...** Publicado en Lecture Notes in Computer Science, Berlin: Springer-Verlag, v.387, 1989. p.395-410.

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

[Mattoso&Blum88] Mattoso, Adriana & Hélcio Blum. Proposta de Desenvolvimento de Software com Orientação a Objetos. In: SIMPOSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE, 2., 1988, Canela. **Anales...** Canela, 1988. p.7-16.

[Meyer88] Meyer, Bertrand. **Object-Oriented Software Construction**. Hertfordshire: Prentice-Hall, 1988.

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

[Navathe&Pillalamarri89] Navathe, Shamkant & Mohan Pillalamarri. OOER: Toward Making the ER Approach Object-Oriented. In: Batini, C. (ed.) **Entity-Relationship Approach**, Amsterdam: North-Holland, 1989, p.185-206.

[Nerson92] Nerson, Jean-Marc. Applying Object-Oriented Analysis and Design. **Communications of the ACM**, New York, v.35, n.9, p.63-74, Sep. 1992.

[OMG97] Object Management Group. Página Web http://www.omg.org

[Page-Jones90] Page-Jones, Meilir et al. Modeling Object-Oriented Systems: The Uniform Object Notation. **Computer Language**, v.7, n.10, p.69-87, Oct. 1990.

[Pressman87] Pressman, Roger. **Software Engineering**: A Practitioner's Approach. 2nd ed., New York: McGraw-Hill, 1987.

[Pressman97] Pressman, Roger. **Software Engineering**: A Practitioner's Approach. 4th ed., New York: McGraw-Hill, 1997.

[Reenskaug92] Reenskaug, Trygve et al. OORASS: Seamless Support for the Creation and Maintenance of Object-Oriented Systems. **Journal of Object-Oriented Programming**, New York, v.5, n.6, p.27-41, Oct. 1992.

[Rubin&Goldberg92] Rubin, Kenneth & Adele Goldberg. Object Behavior Analysis. **Communications of the ACM**, New York, v.35, n.9, p.48-62, Sep. 1992.

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

[Schaschinger92] Schaschinger, Harald. ESA - An Expert Supported OOA Method and Tool. **ACM SIGSOFT Software Engineering Notes**, New York, v.17, n.2, p.50-56, Apr. 1992.

[Schiel&Mistrik90] Schiel, Ulrich & Ivan Mistrik. Using Object-Oriented Analysis and Design for Integrated Systems. **Arbeitspapiere der GMD**, v.449, Jun. 1990.

[Shlaer&Mellor89] Shlaer, Sally & Stephen Mellor. An Object-Oriented Approach to Domain Analysis. **ACM SIGSOFT Software Engineering Notes**, New York, v.14, n.5, p.66-77, Jul. 1989.

[Shlaer&Mellor 90] Shlaer, Sally & Stephen Mellor. **Análise de Sistemas Orientada para Objetos**. São Paulo: McGraw-Hill, 1990.

[Shlaer&Mellor 92] Shlaer, Sally & Stephen Mellor. **Object Life Cycles: Modeling the World in States**. Englewood Cliffs: Yourdon Pres, 1992.

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

[Yourdon94] Yourdon, Edward. **Object-Oriented Systems Design**. Englewood Cliffs: Prentice-Hall, 1994.

[Wirfs-Brock90] Wirfs-Brock, Rebecca et al. **Designing Object-Oriented Software**. Englewood Cliffs: Prentice Hall, 1990.