XV Jornadas de Ingeniería del Software y Bases de Datos
JISBD 2006
José Riquelme - Pere Botella (Eds)

SOBRE LA DIFERENCIA ENTRE ANÁLISIS Y DISEÑO, Y POR QUÉ ES RELEVANTE PARA LA TRANSFORMACIÓN DE MODELOS

Gonzalo Génova[1]*, María C. Valiente[1] y Mónica Marrero[1]

1: Information Engineering Group, Departamento de Informática
Escuela Politécnica Superior
Universidad Carlos III de Madrid
Avda. Universidad 30, 28911 Leganés (Madrid), España
e-mail: {ggenova, mcvalien, mmarrero}@inf.uc3m.es, web: http://www.ie.inf.uc3m.es

Palabras clave: Modelo, Análisis, Diseño, Dimensiones de modelado, Espacio de modelado, Transformación de modelos

Resumen. *En este artículo intentamos clarificar las confusiones que encontramos en torno a los términos "modelo de análisis" y "modelo de diseño", ampliamente usados en ingeniería del software. En nuestra experiencia, estas confusiones son la raíz de algunas dificultades que los profesionales encuentran al modelar, y que en ocasiones conducen a malas prácticas de ingeniería. Nuestro enfoque consiste en desplegar el binomio análisis-diseño en un espacio tridimensional de modelado, que a la vez facilite la comprensión de la transformación de modelos como un desplazamiento en este espacio.*

1. INTRODUCCIÓN

En ingeniería del software existen diferentes clases de modelos: modelos de análisis y diseño, modelos de estructura y de comportamiento, etc., de modo que entender el significado de cada tipo de modelo, cómo se relacionan entre sí, y cómo evolucionan, es de la mayor importancia [9]. Nuestro objetivo en este trabajo consiste específicamente en investigar el lugar que ocupan los modelos de análisis y diseño en el desarrollo de software dirigido por modelos. En una primera aproximación, "análisis" designa la comprensión de un *problema* o situación, mientras que "diseño" se relaciona con la creación de una *solución* para el problema analizado; un "modelo" es una *simplificación* utilizada para comprender mejor el problema ("modelo de análisis") o la solución ("modelo de diseño") [21].

Pero profundicemos en el significado de estos términos. *Análisis* es una palabra griega que significa lo mismo que *descomposición*, de raíz latina: "romper un todo en sus partes componentes (con el fin de comprenderlo mejor)". Se opone a *síntesis* (del griego), o *composición* (del latín): "construir un todo a partir de sus partes componentes". La palabra *diseño*, en cambio, está relacionada con la tarea de dibujar los planos de algo antes de construirlo. El diseño anticipa y guía el proceso de producción, la "síntesis".

El binomio análisis-diseño ha sido tradicionalmente utilizado en ingeniería del software para estructurar las fases (o mejor, las *actividades*, para evitar connotaciones temporales) del proceso de desarrollo de software. Como expondremos en este artículo, esta única dimensión resulta insuficiente para definir el espacio de modelado, es decir, el sistema de coordenadas en

el que podemos situar los distintos modelos empleados en un proyecto software. Otros autores han estudiado ya algunas dimensiones de este espacio, poniendo de manifiesto que se trata de un campo aún poco investigado [15]. Nosotros vamos a presentar tres dimensiones que consideramos particularmente útiles, sin pretender que sean las únicas que pueden definirse.

El resto del artículo se estructura como sigue: las Secciones 2-3-4 exponen de modo progresivo cada una de las tres dimensiones, desplegando en ellas el binomio análisis-diseño. La Sección 5 aplica este espacio tridimensional a la mejor comprensión del proceso de transformación de modelos. Finalmente, la Sección 6 contiene las conclusiones.

2. PRIMERA DIMENSIÓN: ENTORNO O SISTEMA

La primera dimensión de modelado que vamos a estudiar tiene que ver con la respuesta a esta pregunta básica: ¿qué representa el modelo, cuál es la realidad representada en él?

En ingeniería del software los modelos se utilizan para representar básicamente dos tipos distintos de realidades [6]. En primer lugar, el modelo puede utilizarse para representar un *sistema informático*, o una parte de un sistema informático. Puede tratarse de un sistema existente o de un sistema en desarrollo; así mismo, el sistema informático puede representarse con diferentes propósitos y desde distintas perspectivas. Estos dos aspectos los trataremos más adelante en la segunda y tercera dimensiones. Por el momento vamos a centrarnos en que la realidad representada en el modelo es un sistema informático. Con toda propiedad, por tanto, puede hablarse en este caso de "modelo del sistema", o "modelo del software". Nosotros utilizaremos preferentemente el primer término para referirnos a este tipo de modelos.

En segundo lugar, y por oposición al "modelo del sistema", con frecuencia los modelos que usan los ingenieros de software no se refieren propiamente a un sistema informático, sino más bien al *entorno del sistema informático*. Es decir, esa parte de la realidad que rodea al sistema informático y con la que éste interactúa para proporcionarle determinados servicios. En este caso se suelen utilizar términos tales como "modelo del mundo real", "modelo del universo del discurso", "modelo de negocio" o incluso "modelo de dominio", con diversos matices que hacen que estos términos no sean perfectamente sinónimos[1]. Resulta particularmente inadecuado el término "modelo del mundo real", ya que, en definitiva, el sistema informático también es parte del mundo real. En MDA *(Model Driven Architecture)* [16][18], el término CIM (*Computation Independent Model*) parece referirse a esto mismo, aunque su significado no es aceptado de modo unánime. Para evitar posibles confusiones, nosotros utilizaremos el término "modelo del entorno".

De hecho, el "entorno del sistema informático" puede considerarse a su vez como un sistema, típicamente un *sistema social* (un "negocio") en el que trabajan seres humanos, y que interactúa con el sistema informático, o incluso lo contiene como una de sus partes[2]. Así pues,

[1] El "modelo de negocio" representa más bien los *procesos*, mientras que el "modelo de dominio" suele referirse al *vocabulario*, aunque esta terminología no puede decirse que esté perfectamente asentada.

[2] Es más, un mismo sistema social puede contener varios sistemas informáticos que interactúen o no directamente entre sí, o a través de los agentes humanos. En este trabajo vamos a limitarnos a considerar un único sistema informático dentro del sistema social, ya que el caso de varios sistemas informáticos no es significativo para nuestra discusión.

puede considerarse que el entorno incluye o no el sistema informático, y consecuentemente que el modelo del entorno incluye o no el modelo del sistema.

A su vez, el sistema puede contener algún grado de *simulación del entorno* al que presta servicio, hasta tal punto que algunos autores y metodologías [2][3][21] proponen explícitamente incluir el modelo del entorno como parte del modelo del sistema. No obstante, en general no debemos esperar una correspondencia perfecta entre ambos modelos, ya que la finalidad del sistema informático habitualmente va mucho más allá de proporcionar una simulación de su entorno, o incluso puede no pretenderlo en absoluto [20].

A menudo se utilizan también los términos *problema* y *solución* (o "modelo del problema" y "modelo de la solución", "dominio del problema" y "dominio de la solución", "espacio del problema" y "espacio de la solución", etc.) para referirse a esta distinción. No obstante, estos términos tienen un carácter relativo que a nuestro modo de ver los hacen inadecuados para caracterizar la primera dimensión de modelado que estamos estudiando. En efecto, el sistema informático que se desea desarrollar puede entenderse no sólo como la solución a un problema de gestión de información en el negocio considerado, sino que puede verse él mismo como un problema que se desea resolver, por ejemplo cuando el sistema está ya especificado pero aún no implementado. Así mismo, el negocio considerado puede entenderse también como una solución empresarial a un problema en el mundo humano circundante. En definitiva, el binomio problema-solución contiene matices adicionales acerca del papel de los modelos en el desarrollo de software, más allá de la pura referencia al *tipo de realidad representada*, que es la dimensión que nos interesa en este momento. Por lo tanto, y para ganar en precisión, los términos que utilizaremos son "modelo del entorno" y "modelo del sistema". La Figura 1 representa gráficamente esta primera dimensión.

modelo del entorno	modelo del sistema
Entorno	Sistema

Figura 1. Primera dimensión de modelado: entorno o sistema

3. SEGUNDA DIMENSIÓN: DESCRIPCIÓN O ESPECIFICACIÓN

La segunda dimensión de modelado responde a esta pregunta: ¿para qué se usa el modelo?

En ingeniería del software los modelos pueden usarse principalmente de dos maneras distintas, tradicionalmente conocidas como *ingeniería inversa* e *ingeniería directa*. Los modelos serán, respectivamente, *descriptivos* o *especificativos* [22], es decir, descripción de *lo que hay* o especificación de *lo que debe haber*. Centrándonos en primer lugar en el modelo del sistema, en la ingeniería directa el modelo es utilizado como anticipación o *especificación* del sistema informático que se desea construir; el modelo puede usarse como plantilla para guiar la construcción del sistema, como plataforma para simular el comportamiento del sistema antes de construirlo de verdad, incluso como punto de partida para la generación (semi)automática del sistema, etc. En la ingeniería inversa, en cambio, el modelo es utilizado como herramienta conceptual o *descripción* para entender un sistema existente que hay que

mantener o mejorar. Como podemos observar, estos dos usos de los "modelos a escala" en la ingeniería del software son muy similares a los que encontramos en otras ramas de la ingeniería, tales como arquitectura, electrónica, mecánica, etc. Además, estos dos usos están relacionados con la útil distinción entre *modelo-original* y *modelo-copia* [17]: en la ingeniería directa el modelo a escala es usado como original a partir del cual se construye un artefacto software; en la ingeniería inversa el modelo a escala es una copia simplificada del artefacto software que representa, usada para entenderlo mejor. Finalmente, podemos poner todas estas nociones en relación con el binomio análisis-síntesis: la ingeniería directa es un *proceso de síntesis* en el que un sistema es construido a partir de un modelo-original, mientras que la ingeniería inversa es un *proceso de análisis* en el que un sistema existente es entendido por medio de un modelo-copia (ver Figura 2).

Ingeniería directa	**Ingeniería inversa**
modelo → sistema	sistema → modelo
especificación	descripción
modelo-original	modelo-copia
proceso de síntesis	proceso de análisis

Figura 2. Dos usos diferentes de los modelos en la ingeniería del software

No obstante, y tal vez de modo paradójico, este sentido perfectamente legítimo del término "análisis", muy cercano al significado original de la palabra, no es el más habitual entre los ingenieros de software, como mostraremos en la siguiente Sección.

Naturalmente, esta distinción puede aplicarse no sólo al modelo del sistema, sino también al modelo del entorno, por tanto se trata de *dos dimensiones ortogonales*, tal como se representa en la Figura 3. Efectivamente, los ingenieros software a menudo trabajan con modelos descriptivos del entorno como ayuda para entender los requisitos que debe satisfacer el sistema informático y construir un vocabulario adecuado para representar los elementos del sistema (que suele denominarse "modelo conceptual"). Igualmente, aunque tal vez sea menos frecuente, si el entorno no se considera una realidad inmutable, sino que su modificación entra dentro de los objetivos de un proyecto de gestión de la información, entonces será necesario crear un modelo de especificación del nuevo entorno deseado[3].

Especificación	modelo especificativo del entorno	modelo especificativo del sistema
Descripción	modelo descriptivo del entorno	modelo descriptivo del sistema
	Entorno	Sistema

Figura 3. Segunda dimensión de modelado: descripción o especificación

[3] De hecho, es difícilmente imaginable un entorno que no se vea afectado en absoluto por la introducción de un sistema informático que soporte parte de las tareas que previamente se realizaban de modo manual, o mediante otros sistemas informáticos que se desea sustituir. No obstante, este aspecto es a menudo soslayado en los proyectos informáticos.

Un típico proyecto informático incluirá un modelo descriptivo del entorno (ingeniería inversa) a partir del cual se construye un modelo especificativo del sistema (ingeniería directa). Nótese que en este caso no sólo la *realidad representada* es diferente, sino que el *propósito del modelo* también lo es. No obstante, con demasiada frecuencia la diferencia entre ambos modelos no es tratada con suficiente rigor, lo que supone un riesgo para el proyecto.

No puede negarse que entender el mundo real, es decir, analizarlo (en el sentido original de la palabra), es utilísimo para obtener una buena especificación de los requisitos, y por tanto un sistema software práctico que resuelva las necesidades de los usuarios. Esto se traduce a menudo en que el modelo especificativo del sistema (*modelo conceptual*) utiliza los conceptos encontrados en el modelo descriptivo del entorno (*modelo de dominio*). La diferencia es sutil, pero real: utilizando el *mismo vocabulario*, estos dos tipos de modelos representan dos realidades distintas, con un propósito distinto también. Tal vez sea éste el origen de la confusión que tratamos de evitar. Elaborar un modelo-copia del mundo real para entenderlo mejor es una práctica perfectamente legítima y útil, aun cuando deba distinguirse cuidadosamente de elaborar un modelo-original del sistema futuro.

Los usos contradictorios del término "análisis" favorecen a menudo esta confusión, ya que para algunos análisis, por oposición a síntesis, significa "comprensión del mundo real", es decir, modelado descriptivo del entorno; mientras que para otros significa, por oposición a diseño, "especificación de alto nivel del sistema", es decir, modelado especificativo del sistema. En la siguiente Sección trataremos con más detalle esta última distinción entre análisis y diseño.

4. TERCERA DIMENSIÓN: VISTA EXTERNA O VISTA INTERNA

La tercera dimensión de modelado considerada responde finalmente a esta pregunta: ¿cómo se representa la realidad en el modelo, qué tipo de relación hay entre el modelo y la realidad?

Como ya hemos comentado, cuando los ingenieros de software dicen "análisis", pueden referirse principalmente a dos tipos distintos de actividades de modelado, o incluso pueden mezclarlas descuidadamente: construcción de modelos especificativos del software (ingeniería directa) o construcción de modelos descriptivos del "mundo real" (ingeniería inversa).

Es muy habitual caracterizar el análisis como la tarea de especificar el *qué*, mientras que el diseño es especificar el *cómo*: qué se supone que el sistema debe hacer para sus usuarios, cómo va a hacerlo (en realidad, esto no expresa otra cosa que un principio clásico de la ingeniería del software: separar la especificación de la implementación). El análisis debe, pues, *capturar los requisitos de usuario* sin adoptar prematuramente decisiones de implementación, es decir, omitiendo detalles dependientes de la tecnología [14], y utilizando conceptos extraídos únicamente del dominio del problema. Por el contrario, el diseño debe *definir una solución software* que satisfaga de modo efectivo y eficiente los requisitos especificados en el análisis, y al hacer esto el modelo incorporará nuevos artefactos (nuevas clases, nuevos atributos y operaciones para las clases, etc.) y tendrá en cuenta la plataforma tecnológica concreta sobre la que debe construirse el sistema informático. Esta distinción se ha expresado a menudo como *diseño lógico-diseño físico* [8], y más modernamente, según

algunos autores, es paralela a la que existe en MDA entre PIM (*Platform Independent Model*) y PSM (*Platform Specific Model*) [16][18]. En cambio, otros autores consideran la correspondencia PIM-PSM como un *refinamiento del diseño* [15], es decir, ambos modelos deben considerarse fruto de la actividad de diseño. Como puede observarse, la confusión terminológica está servida.

El punto clave es que ambos tipos de modelos, análisis y diseño, o como los queramos llamar, representan el mismo sistema: ambos son modelos del sistema, no del entorno, y son utilizados en el contexto de un proceso de ingeniería directa, aun cuando tengan una diferencia importante en cuanto a perspectiva [12], y por tanto en la forma en que representan el sistema que se desea construir. La *forma de representar*, la relación con la realidad representada, es diferente en cada caso: el modelo de análisis representa la *vista externa* del sistema (la especificación), mientras que el modelo de diseño representa la *vista interna* (la implementación). También podemos denominarlos respectivamente "modelo de caja negra" y "modelo de caja blanca". En cada uno se adopta una *perspectiva diferente*, lo que produce un tipo diferente de modelo del mismo sistema, de modo que éste es representado también de forma distinta: el diseño (vista interna o caja blanca) omite más o menos detalles de implementación, mientras que el análisis (vista externa o caja negra) omite la implementación misma; el análisis especifica lo que el diseño realiza. El efecto visible puede ser un diferente nivel de complejidad en los modelos, pero el punto clave no es ése, sino la diferente perspectiva que adoptan[4]. Esta noción de análisis, que en realidad equivale a un *primer paso en la síntesis*, es fácilmente reconocible en multitud de prácticas de ingeniería y libros de texto, aun cuando raramente sea reconocida como tal de modo explícito (hay excepciones, por supuesto [13]).

El *modelo de casos de uso* es un buen ejemplo de esto[5]. Debido a su carácter de "alto nivel", más cercano al usuario que a la implementación, los casos de uso habitualmente se definen dentro de la actividad de análisis de un proyecto. Ahora bien, un caso de uso es la especificación de la funcionalidad esperada que un sistema informático debe proporcionar, descrita mediante interacciones típicas usuario-sistema [7]: debería ser obvio para cualquiera que un caso de uso está modelando el sistema software que se desea construir, y no el "mundo real" tal como es antes de que exista el sistema, o tal como será después [10]. Es decir, el modelado de casos de uso es una *actividad de síntesis* [1]. Lo mismo se aplica a los *requisitos de usuario* en general (expresados o no mediante casos de uso): son especificaciones del sistema informático deseado, no describen el mundo fuera del ordenador. Dado que muchos conciben el análisis como una actividad en la que se averiguan y escriben claramente los

[4] Por tanto, no estamos pretendiendo que el análisis sea meramente un primer borrador del diseño, un diseño poco detallado o de alto nivel: esto sería errar el tiro. Nótese, también, que la expresión que se usa habitualmente para distinguirlos, "diferente *nivel* de abstracción", puede resultar engañosa en este sentido.

[5] Aunque el modelo de casos de uso incluye tanto a los casos de uso como a los actores, su finalidad principal es modelar el sistema en tanto que interacciona con el exterior (los casos de uso); los actores representan meros agentes externos sin contenido. Puede discutirse si el modelo de casos de uso ocupa más bien un lugar intermedio entre el modelo del entorno y el modelo del sistema, pero en todo caso es seguro que no es un modelo del entorno, ya que no representa la interacción de los actores entre sí o con otros sistemas, es decir, no es un modelo completo del negocio.

requisitos de usuario [4][14], está claro que estos mismos están considerando el análisis como parte de un proceso de ingeniería directa, es decir, de síntesis.

Vistas así las cosas, la transición desde el modelo de análisis al modelo de diseño (de la vista externa a la vista interna, o de la caja negra a la caja blanca) puede ser difícil o no, pero no supone un cambio en la *realidad* representada (primera dimensión) ni en el *propósito* del modelo (segunda dimensión): ambos son, dentro de un proceso de síntesis [13], modelos especificativos del mismo sistema software, si bien desde una *perspectiva* diferente. Nuestro énfasis aquí no es que la transición sea fácil o difícil. De hecho, el diseño debe proporcionar una solución creativa para el problema especificado en el análisis, y esto raramente será fácil [12][14]. Más aún, un buen modelo de diseño que tenga en cuenta requisitos no funcionales tales como rendimiento, reutilización, mantenibilidad, etc., puede parecerse apenas al modelo de análisis que especifica el mismo sistema [11][12][19]. Pero esto no niega que ambos sean modelos del mismo sistema software, que es el punto que queremos subrayar: tanto el modelo de análisis como el modelo de diseño son *modelos del sistema, no del entorno*.

Hasta ahora hemos considerado la dimensión análisis-diseño (o mejor, vista externa-vista interna) en el contexto de ingeniería directa de un sistema informático, es decir, referida a los modelos especificativos del sistema. Sin embargo, *esta dimensión es también ortogonal* a las otras dos estudiadas anteriormente, entorno-sistema y descripción-especificación. Por tanto, podemos considerar con sentido cualquier combinación de "coordenadas", aunque de hecho algunas sean menos usadas en los procesos típicos de la ingeniería del software. Por ejemplo, podríamos hablar de un modelo de ingeniería inversa del sistema, que lo represente en su vista externa (sistema-descripción-externa), o de un modelo de ingeniería directa del entorno, que lo represente en su vista interna con la idea de realizar una modificación en el negocio (entorno-especificación-interna), etc. Es decir, tiene sentido definir un *espacio tridimensional de modelos*, con tres coordenadas ortogonales y dos valores distintos en cada coordenada, con ocho combinaciones posibles en total, como se representa en la Figura 4.

Especificación	Vista externa	modelo especificativo externo del entorno	modelo especificativo externo del sistema
	Vista interna	modelo especificativo interno del entorno	modelo especificativo interno del sistema
Descripción	Vista externa	modelo descriptivo externo del entorno	modelo descriptivo externo del sistema
	Vista interna	modelo descriptivo interno del entorno	modelo descriptivo interno del sistema
		Entorno	Sistema

Figura 4. Tercera dimensión de modelado: vista externa o vista interna

Los términos "análisis" y "diseño" tienen habitualmente connotaciones adicionales que los ligan al contexto de la ingeniería directa de sistemas informáticos, lo que hace extraño el uso de expresiones como "modelo de diseño descriptivo del entorno" (entorno-descripción-interna). Por este motivo consideramos más adecuados los términos "vista externa" y "vista interna" para caracterizar esta dimensión.

5. LA TRANSFORMACIÓN DE MODELOS EN EL ESPACIO DE MODELADO

A las tres dimensiones expuestas se pueden añadir otras (más o menos ortogonales) para enriquecer la definición del espacio de modelos. Particularmente, es típico considerar el *nivel de abstracción* como una más de estas dimensiones, entendida como nivel de detalle o complejidad con que el modelo representa la realidad. A diferencia de las dimensiones anteriores, que representan importantes saltos cualitativos en el significado del modelo, ésta presenta una marcada gradualidad, hasta el punto de que pueda considerarse una *dimensión cuasi-continua*. La adición o supresión de detalles en el modelo equivale a un desplazamiento a lo largo de esta dimensión, que puede expresarse como una transformación de modelos. Se trata del tipo de transformación más sencillo que podemos concebir y, en el caso de la supresión de detalles es incluso, con algunos matices, fácilmente automatizable. Esta cuarta dimensión es *ortogonal a las tres anteriores*, es decir, cualquiera de los ocho tipos de modelos de la Figura 4 puede representarse con mayor o menor nivel de detalle.

En el marco conceptual de la ingeniería de modelos, el proceso de desarrollo de software puede entenderse como una *trayectoria de transformaciones en el espacio de modelado*. Una trayectoria típica sería, por ejemplo, comenzar por un modelo descriptivo externo del entorno (lo que con frecuencia se denomina "modelo del mundo real", o incluso "modelo de análisis"), para pasar, saltando a la vez en la primera y segunda dimensiones, a un modelo especificativo externo del sistema ("análisis de requisitos", también denominado a menudo "modelo de análisis", pero con un sentido radicalmente distinto al anterior, que puede resultar en peligrosos equívocos), y saltar finalmente en la tercera dimensión para llegar a un modelo especificativo interno del sistema ("modelo de diseño"). Otra trayectoria típica la encontramos en el proceso de sustitución de un sistema heredado (*legacy system*) por uno tecnológicamente más moderno, aunque proporcione básicamente la misma funcionalidad: modelo descriptivo interno del sistema heredado (éste tal vez pueda omitirse), modelo descriptivo externo del sistema heredado, modelo especificativo externo del nuevo sistema, y modelo especificativo interno del nuevo sistema. En este caso nos hemos movido sólo en la segunda y tercera dimensiones, aunque es cierto que el sistema representado ha sido cambiado en el segundo salto.

A diferencia del cambio en el nivel de abstracción, el desplazamiento a lo largo de las otras tres dimensiones *no es conceptualmente sencillo*, ni debería ser presentado como tal. No es fácil establecer qué relación hay entre un modelo descriptivo externo del entorno y un modelo especificativo externo del sistema. Aunque la distinción pueda parecer obvia según nuestra argumentación, la realidad es que demasiado a menudo ambos modelos son confundidos, en buena medida debido a la ambigüedad del término "modelo de análisis", y posiblemente también debido al hecho de emplear *una notación uniforme* para representar realidades muy distintas [14]. Cuando se dice, por ejemplo, que las clases de análisis representan conceptos del mundo real, mientras que las clases de diseño representan fragmentos de código [5][12][14], aquí "modelo de análisis" significa modelo descriptivo externo del entorno. En cambio, cuando se recomienda comenzar con un modelo de análisis, para luego incluir este modelo dentro de un modelo de diseño que es completado con otros artefactos requeridos para

proporcionar la solución [2][3][21], entonces "modelo de análisis" significa modelo especificativo externo del sistema. Es un error común confundir ambos y suponer que el modelo de análisis describe el mundo real (modelo del entorno), para utilizarlo seguidamente como especificación del sistema que se desea construir (modelo del sistema). Como el sistema no es generalmente una copia o simulación del mundo exterior [20], *un modelo de ingeniería inversa del entorno no puede servir como modelo de ingeniería directa del sistema* [6]. Tomando una analogía de la ingeniería civil, un modelo de coches y ríos no puede ser un modelo del puente que los coches necesitan para cruzar por encima del río.

Tampoco es sencilla la relación entre la vista externa y la vista interna. Insistimos en que no debe confundirse el *nivel de abstracción* o cuarta dimensión (más o menos detalles) con la *perspectiva adoptada* (vista externa o vista interna), que corresponde a la tercera dimensión explicada. En efecto, el paso del análisis al diseño no puede concebirse como un mero cambio en el nivel de abstracción, porque esto implicaría que basta con añadir detalles al problema para alcanzar la solución, lo cual es absurdo [14]. Aunque el marco conceptual de la orientación a objetos y un lenguaje de modelado como UML hagan más fácil la transición del análisis al diseño, no debemos pretender que se trata de una transición cuasi-automática y sin costuras (*seamless*), porque esto no sería más que simplificar engañosamente la cuestión.

6. CONCLUSIONES

Hemos examinado los *dos significados diferentes de los modelos de análisis* que podemos encontrar entre los ingenieros software: un modelo que especifica la vista externa del sistema software, o bien un modelo descriptivo del "mundo real" que constituye el contexto del sistema software deseado. Es bastante frecuente confundir estas dos concepciones. La práctica real corresponde habitualmente al uso del modelo especificativo del sistema, aun cuando las explicaciones teóricas apuntan con frecuencia hacia el modelo descriptivo del entorno. Un peligro moderado de la confusión es pensar que estamos modelando el mundo real, cuando en realidad estamos elaborando una especificación de alto nivel del sistema software. Un peligro mucho más serio es construir un sistema que imite sin necesidad la estructura del mundo real. En cambio, sería legítimo *concebir la tarea de análisis como aquélla en la que se construyen ambos modelos*, estrechamente relacionados y a la vez adecuadamente distinguidos. Si tiene que utilizar la palabra "análisis" en el contexto de la ingeniería del software, escoja el sentido que prefiera, pero sea consciente de su elección y de los posibles malentendidos con su audiencia.

Confiamos en que el *despliegue del binomio análisis-diseño* en el espacio tridimensional de modelado que hemos presentado contribuya a deshacer los frecuentes malentendidos que obstaculizan el uso de los modelos en la ingeniería del software, y sea útil para definir mejor la transformación de modelos como un desplazamiento en este espacio.

REFERENCIAS

[1] K. Bittner, I. Spence. *Use Case Modeling.* Addison-Wesley, 2003.

[2] E. Braude. *Software Engineering. An Object-Oriented Perspective.* John Wiley &

Sons, 2001.
[3] P. Coad, E. Yourdon. *Object-Oriented Analysis*. Yourdon Press, 1991.
[4] European Space Agency. Board for Software Standardisation and Control. *Guide to the Software Requirements Definition Phase*. PSS-05-03, March 1995.
[5] M. Fowler, K. Scott. *UML Distilled.* Addison-Wesley, 2004.
[6] G. Génova, M. C. Valiente, J. Nubiola. "A Semiotic Approach to UML Models". *First Intern. Workshop on Phil. Found. of Information Systems Engineering-PHISE 2005*, 13 June 2005, Porto, Portugal. *Proc. of the CAiSE'05 Workshops*, vol. 2, pp. 547-557.
[7] G. Génova, J. Llorens. "The Emperor's New Use Case", *Journal of Object Technology*, 4(6): 81-94, Aug 2005 (http://www.jot.fm/issues/issue_2005_08/article7).
[8] I. Graham, B. Henderson-Sellers, H. Younessi. *The OPEN Process Specification.* Addison-Wesley, 1998.
[9] D. Harel, B. Rumpe. "Meaningful Modeling: What's the Semantics of 'Semantics'?". *IEEE Computer*, October 2004:64-72.
[10] D. Hay. "There Is No Object-Oriented Analysis". *Data to Knowledge Newsletter*, 27(1), January-February 1999.
[11] W. Haythorn. "What Is Object-Oriented Design?". *Journal of Object-Oriented Programming* 7(1):67-78, 1994.
[12] G. M. Høydalsvik, G. Sindre. "On the Purpose of Object-Oriented Analysis". *VIII Conf. on Object-Oriented Prog., Syst., Lang., and Appl. (OOPSLA-93)*, September 26 - October 1 1993, Washington, DC, USA. *ACM SIGPLAN Notices* 28(10):240-255.
[13] I. Jacobson. "A Confused World of OOA and OOD". *Journal of Object-Oriented Programming* 8(5):15-20, 1995.
[14] H. Kaindl. "Difficulties in the Transition from OO Analysis to Design". *IEEE Software*, 16(5):94-102, 1999.
[15] S. Kent. Model Driven Engineering. In *Proc. of the Third Intern. Conf. on Integrated Formal Methods–IFM 2002*, May 15-17, 2002, Turku, Finland, pp. 286-298.
[16] A. Kleppe, J. Warmer, W. Bast. *MDA Explained. The Model Driven Architecture: Practice and Promise.* Addison-Wesley, 2003.
[17] E. Marcos, A. Marcos. "A Philosophical Approach to the Concept of Data Model: Is a Data Model, in Fact, a Model?". *Information Systems Frontiers*, 3(2):267-274, 2001.
[18] S. J. Mellor, K. Scott, A. Uhl, D. Weise. *MDA Distilled. Principles of Model-Driven Architecture*. Addison-Wesley, 2004.
[19] D. L. Parnas. "On the Criteria to Be Used in Decomposing Systems into Modules". *Communications of the ACM*, 15(12):1053-1058, December 1972.
[20] C. Potts. "Modes of Correspondence between Information System and World". *Second Intern. Workshop on Phil. Found. of Information Systems Engineering-PHISE 2006*, 5 June 2006, Luxembourg. *Proc. of the CAiSE'06 Workshops*, pp. 745-756.
[21] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen. *Object-Oriented Modeling and Design*. Prentice Hall, 1991.
[22] E. Seidewitz. "What Models Mean". *IEEE Software* 20(5): 26 - 32, Sep-Oct 2003.