



INTRODUCCIÓN

A pesar de que la orientación a objetos cuenta hoy con una existencia de más de tres décadas, posee una difusión moderada y una aplicación práctica no acorde con esta realidad. Como muy bien predijera Rentsch hace alrededor de 30 años, "la programación orientada a objetos va a ser en los ochenta lo que fue la programación estructurada en los setenta. Todo el mundo va a estar a favor de ella. Todos los fabricantes van a promocionar sus productos afirmando que la soportan. Todos los administradores hablarán bien de ella. Todos los programadores la practicarán (de forma diferente). Y nadie va a saber exactamente qué es."¹. Es lamentable que habiendo transcurrido algunos años del siglo XXI todavía esas palabras tengan vigencia. Incluso Kenneth Rubin, entonces Manager of Methodology Development de ParcPlace Systems, Inc., en noviembre de 1994, en su visita a Buenos Aires, me contestó acerca de su opinión sobre cuánto tiempo faltaba para que la orientación a objetos se difundiera en forma masiva, en empresas, universidades, etc. de manera categórica: "-Diez años.". Creo que se quedó corto con su apreciación.

Probablemente se pueda argumentar que muchas corrientes precisaron tanto tiempo como ésta para su establecimiento y su aceptación definitiva, pero no me convence admitir que se trata sólo de una cuestión de tiempo.

Tampoco sería posible establecer una única razón de por qué su difusión y aplicación real demandó tanto tiempo, porque la orientación a objetos abarca muchos ámbitos, y cada uno de ellos debe considerarse en forma individual. Es así que, entonces, es preciso realizar una distinción entre el empleo de la orientación a objetos en la etapa de análisis de sistemas y su aplicación en las fases de diseño o implementación. En cada una de ellas su grado de penetración es diferente.

Cuando hoy se habla de análisis de sistemas, no cabe la menor duda de que se hace referencia al análisis orientado a objetos. No se juzga si el paradigma es adecuado, a lo sumo se lo hace con relación a los métodos en particular que podrían ser utilizados dentro de esta corriente; y menos aún con la aparición del UML como un estándar mundial. A lo sumo se podrá discutir si las empresas actualmente utilizan métodos formales para el modelado de sistemas. Pero, insisto, hoy nadie discute hoy la importancia y la necesidad de un enfoque orientado a objetos en la fase del análisis de sistemas.

¹ RENTSCH, T. Object Oriented Programming. SIGPLAN Notices vol.17(12), p.51. Cit. por BOOCH, Grady. **Análisis y Diseño Orientado a Objetos con Aplicaciones**. 2º ed. Wilmington, Addison-Wesley Iberoamericana, 1996. p.31.



Análisis de Sistemas con UML para Estudiantes
Versión 1.1 – Diciembre de 2004

Con respecto al diseño sucede otro tanto. Si analizamos la realidad del mercado informático, podremos observar que la aparición de Windows no fue sólo la posibilidad de administrar de una forma simpática nuestros utilitarios en la PC. Cada ícono en la pantalla está representando un concepto, está reflejando de una manera concreta y con su presencia, las características y el comportamiento del programa al que simboliza. La mayoría de los programas incluyen una filosofía de diseño orientada a objetos y sus beneficios son irrefutables. La reutilización de componentes y la programación orientada a eventos, hacen uso de objetos y los usuarios están acostumbrados a "instanciarlos" al usar numerosas aplicaciones sin darse cuenta siquiera de que conviven a diario con objetos. Los patrones de diseño de la "Banda de los Cuatro" –The Gang Of Four, como se denomina a los autores de más famoso libro de diseño de la historia²- son el recurso de diseño más difundido, aplicado y útil que se conoce desde siempre.

Seguramente es en el ámbito de la programación donde la orientación a objetos todavía no logró imponerse con tanta fuerza. Además de las aplicaciones existentes hechas en lenguajes tradicionales de tercera generación, existen desarrollos con lenguajes de cuarta generación, cuya productividad es el argumento más valioso para justificar su empleo -aunque esto hoy también está puesto en tela de juicio-. Claro que existen trabajos realizados con lenguajes orientados a objetos, pero no se observa en la programación lo que apunté para el análisis y el diseño. La programación con Java es el camino más concreto en esta dirección, aunque como predijera Rentsch, cada programador aplica la orientación a objetos a su manera y cree estar haciéndolo como corresponde.

Además de los lenguajes de programación verdaderamente orientados a objetos, existen otros cuyos fabricantes dicen que soportan el paradigma de objetos. No sólo cumplen fielmente la profecía de Rentsch, sino que también hacen ciertas las palabras de Roger King: "Tengo un gato que se llama Trash... Si estuviera intentando venderlo (por lo menos, si intentara vendérselo a un científico de la computación), no haría énfasis en que es cariñoso con los seres humanos y en que es autosuficiente, por cuanto vive fundamentalmente de los ratones que caza. Más bien, yo diría que es orientado a objetos"³. En otras palabras, promocionan un soporte del paradigma de objetos, aunque ese soporte no sea completo ni completamente adecuado.

Las aplicaciones que se construyen con estos lenguajes que dicen ser orientados a objetos, y que los usuarios se llevan en el bolsillo, por lo general no son otra cosa que una base de datos relacional para administrar los datos y un conjunto de funciones que se ejecutan de acuerdo a un enfoque muchas veces estructurado.

² GAMMA, E. y otros. **Design Patterns. Elements of reusable object-oriented software.** Addison-Wesley, 1995.

³ KING, Roger. Mi gato es Orientado a Objetos. Cit. por GRAHAM, Ian. **Métodos Orientados a Objetos.** 2ª ed. Wilmington, Addison-Wesley Iberoamericana, 1996, p. 159.



Análisis de Sistemas con UML para Estudiantes
Versión 1.1 – Diciembre de 2004

Más allá de esta confusión de conceptos, no cabe duda de que la orientación a objetos está comenzando a hacerse presente en la mayoría de los lenguajes de programación. Todavía no se observa su empleo masivo en las aplicaciones, pero sí se lo puede encontrar en las herramientas de desarrollo actuales.

En el transcurso de este trabajo intentaré justificar el empleo de la orientación a objetos para comprender la realidad a la que se le debe brindar una solución, y la necesidad de contar con un buen conjunto de herramientas a tal efecto. Estableceré las características que debe poseer un método adecuado de análisis, las diferencias existentes entre los métodos tradicionales y los orientados a objetos en base a las características mencionadas y, dentro de la orientación a objetos, las corrientes existentes, los métodos que las integran, y las herramientas y técnicas que se proponen, haciendo el mayor hincapié en el UML desde una perspectiva tanto teórica como práctica, y siempre dentro del marco del enfoque de ingeniería de software que hoy predomina en el mundo⁴, que se describe en la guía SWEBOK.

Por último, saldremos trataré de sondear el futuro del análisis orientado a objetos en base a los adelantos que presentan los metodologistas. De esta manera, obtendremos los criterios que nos permitan establecer la forma más conveniente de realizar el análisis de sistemas.

En cuanto a la exposición de este trabajo, cada vez que mencione un nuevo concepto que no se haya difundido aún en español, lo presentaré traducido a esa lengua y destacado en bastardilla, aunque inmediatamente al lado de él, y entre paréntesis, incluiré el término en su lengua original, a fin de no confundir al lector y permitirle ajustar la traducción en la medida que considere que no existe la precisión adecuada. Durante el resto del trabajo emplearé la traducción que yo haya propuesto. De todas formas, existen ciertas palabras que, por su difusión, las mantendré sin traducir.

Quiero hacer un último comentario acerca de este material que hoy presento. Parte de él había sido producido para conformar el capítulo central de un libro cooperativo que sería publicado por la Universidad de Vigo, España, en 1997. Luego lo fui ampliando y actualizando, hasta ser lo que hoy encuentran los lectores en sus manos.

Quiero finalizar esta introducción con las palabras que mi amigo Kenneth Rubin había escrito como prólogo de aquel intento fallido de publicación, pero que valoro enormemente y deseo compartir con Uds.:

Antes del uso extenso de las computadoras, los negocios a cabo su trabajo mediante personas interactuando directamente con otras personas. Con el tiempo, las computadoras y los sistemas de información comenzaron a jugar un rol crítico en los procesos de negocio y como intermediarios entre las personas de negocio. Al punto que, los sistemas de información diseñados apropiadamente para cubrir las necesidades de los usuarios de negocio, proporcionaron un valor agregado. Sin

⁴ IEEE. **Guide to the Software Engineering Body of Knowledge**. Versión 1.00, 2001.



Análisis de Sistemas con UML para Estudiantes
Versión 1.1 – Diciembre de 2004

embargo, muy a menudo, los sistemas de información comenzaron a interferir con las operaciones del día a día de los negocios. Con frecuencia, los usuarios del se veían forzados a modificar su forma de trabajo para adaptarse a un sistema de información inadecuado.

Esta situación disfuncional era y es intolerable. En lugar de interferir con los negocios de la empresa, los sistemas de información deben apoyarlos y mejorarlos. En lugar de rezagarse bajo los requerimientos de la empresa, los sistemas de información deben ser la fuente de nuevas oportunidades de negocio. El diseño apropiado de los sistemas de información puede permitir una ventaja competitiva significativa.

Para desarrollar sistemas de información adecuados, superiores, los profesionales de software deben poseer un conocimiento práctico muchos tipos de disciplinas de desarrollo de negocios y de software. Los usuarios de negocio esperan que los sistemas de información sean poderosos y fáciles de usar. Normalmente, tales sistemas son complicados y difíciles de crear.

Dentro de IBM, dirijo un grupo de profesionales de software que desarrollan grandes sistemas de información de negocios de misión crítica. Nuestra experiencia se centra en la tecnología orientada a objetos. Usamos herramientas y métodos sofisticados orientados a objetos para construir soluciones de negocio que deleiten a nuestros clientes. Sin embargo, frecuentemente usamos muchas otras disciplinas, además de la orientación a objetos, en nuestros compromisos. Actividades que involucran re-ingeniería de procesos empresariales (BPR), trabajo en grupos (groupware) y, flujo de trabajos (workflow), son frecuentemente empleadas para proporcionar una solución de negocio completa y coherente. Los profesionales del software deben familiarizarse con estas disciplinas, y otras, para desarrollar sistemas de información capaces de cubrir las necesidades de los usuarios de negocio que las solicitan.

Hay un enorme cuerpo de literatura técnica y administrativa disponible para los profesionales de software. Con la existencia de tanta información, es imprescindible determinar lo que realmente es importante. Los autores de este libro han invertido tiempo en clasificar y organizar el material disponible para proporcionar una discusión clara y completa de los aspectos principales que influyen el desarrollo de los sistemas de información. No sólo proporcionan detalles de muchas disciplinas relevantes, sino que también discuten cómo tales disciplinas calzan entre sí. El resultado es un libro de lectura esencial para aquellos que deseen una visión global de los aspectos críticos que sirven de base al desarrollo de sistemas de información.

Particularmente estoy impresionado con la calidad de los individuos que han colaborado en este libro. El lector se beneficiará de la diversa experiencia profesional y académica de los autores, que les permite proporcionar un tratamiento bien balanceado, tanto de aspectos teóricos como prácticos. Las experiencias ofrecidas por muchos de los autores, desde proyectos reales en diversos países, aumentan la utilidad y la diversidad de este trabajo.



Análisis de Sistemas con UML para Estudiantes
Versión 1.1 – Diciembre de 2004

Los estudiantes de informática, así como los profesionales del software encontrarán en este libro un útil compendio de disciplinas relevantes para el desarrollo de software.

*Kenneth S. Rubin⁵
Director and Managing Principal
IBM North American Object Technology Services*

⁵ Kenneth Rubin fue uno de los principales metodologistas de la década de los '90. Trabajó en ParcPlace Systems junto a Adele Goldberg (participó en la creación del paradigma de objetos), y con ella crearon el método Object Behavior Analysis and Design (OBA/D) que fuera presentado en OOPSLA '94 y cuyas ideas tomara posteriormente Rebecca Wirfs-Brock para complementar con las de su clásico método. Es coautor también con Adele Goldberg del libro **Succeeding with Objects: Decision Frameworks for Project Management** y es posible encontrar su nombre entre los colaboradores que participaron activamente con el aporte de ideas para el UML en el prólogo de todos los documentos y libros que los autores del UML publicaron. Luego de Parc-Palce pasó por IBM y luego fundó su propia empresa consultora Innolution.