Computación y Sistemas Vol. 1 No. 3 pp. 179-182
© 1998, CIC - IPN. *ISSN 1405-5546* Impreso en México

La Red, los Libros y las Editoriales

El dilema de las publicaciones electrónicas

Thinking in Java
Bruce Eckel
Ed. Prentice-Hall, 1998

Por: Bárbaro Ferro Castro
Centro de Investigación en Computación
Instituto Politécnico Nacional

La revisión crítica de un libro siempre representa un desafío para el revisor Además de ser conocedor del tema, tiene que considerar, entre otras cosas, los factores subjetivos a la hora de escribir y leer un texto: el enfoque que el autor usa para explicar lo que desea con el texto y la forma de asimilar ese contenido por parte de los lectores.

Si a estos factores añadimos que el libro que en este trabajo se revisa, ya ha sido revisado por cientos o miles de lectores, inclusive antes de la publicación del mismo, entonces el reto de aportar algo nuevo en esta crítica es mayor.

Los tiempos actuales están modificando casi todas las actividades del ser humano y exigen quizá, de todos nosotros, un análisis de nuestro proceder en el trabajo diario. Aunque sin ánimo alguno de restar importancia a innumerables factores que influyen en el cambio de nuestro modo de pensar y actuar, sin duda alguna que la computación y las comunicaciones han provocado un cambio decisivo en nuestras vidas y en la manera en que hacemos ciencia y tecnología.

Cada día nos percatamos del acercamiento a fuentes de conocimientos en campos de nuestro interés que nos obligan a colectivizar nuestro trabajo científico, compartiéndolo con otras personas que ni conocemos y que quizá nunca conoceremos personalmente.

La originalidad de nuestro trabajo y la frase que muchos científicos utilizan "los métodos que estoy ideando no se han reportado en la literatura", tienen hoy en día validez solamente si se ha seguido una búsqueda bibliográfica —que puede resultar interminable— en esa gran biblioteca virtual altamente desorganizada que es *Internet*.

Para aquellos que hacen uso de ese gran potencial de conocimiento en su quehacer diario, lo anterior puede resultar trivial. Sin embargo, aún abundan aquellos que, enfrascados en su labor de llevar a cabo sus investigaciones, solamente consultan aquellas revistas científicas de interés de su área, que pueden ser muy pocas o muchas, pero que seguramente no contienen todo lo relevante que se publica en el campo, ni los progresos corrientes de esas investigaciones.

Las publicaciones electrónicas, que existen desde hace mucho tiempo, y que cada día florecen y crecen, contienen tal riqueza de información científica que es imposible proyectarlas hacia sus correspondientes publicaciones en papel, pues por una parte no alcanzarían las publicaciones seriadas existentes y por otra se perdería el deseo del autor de poner a la disposición inmediata los resultados de su trabajo.

En muchos foros científicos, dedicados a la diseminación y divulgación de los resultados de la investigación, se discute mucho el lema "publish or perish", asociado únicamente con publicaciones impresas, para medir la productividad de los investigadores, sin considerar las publicaciones electrónicas.

No resulta fácil la creación de los mecanismos (que desde hace cientos de años se usan para medir lo que es o no publicable en revistas científicas de prestigio), para evaluar la calidad y originalidad de las publicaciones electrónicas. Que esto no exista, en ningún momento implica que no existen en la actualidad miles de artículos, reportes de investigación, etc., publicables, que pueden accederse en línea, bien sea a través del Web u otro servicio de comunicación que brinda la red.

Algunos investigadores, cuando evalúan la factibilidad de una propuesta de investigación presentada por un nuevo investigador, revisan de manera minuciosa las referencias bibliográficas actuales en revistas periódicas y nunca preguntan si ese estudio del estado del arte también ha considerado los miles de sitios de Universidades y Centros de Investigación que trabajan en campos afines. El estudio del estado del arte de cualquier campo científico y técnico, modelado unicamente mediante la revisión de publicaciones impresas resulta obsoleto, si se tiene en consideración que para la mayoría de esas revistas, el tiempo de publicación de un artículo es mayor de un año. Durante ese período, surgen nuevas ideas que se diseminan a través de la red.

Muchos factores influyen en querer mantener un sistema de publicaciones impresas que para algunas cosas ya resulta obsoleto. Siempre es más agradable leer un "paper" en una revista, llena de toda la tipografía elegante que se logra en una imprenta, que leer ese mismo artículo en una copia de impresora, aún cuando la tipografía electrónica permite una elegancia similar. Los autores sienten satisfacción al ver sus publicaciones reales en lugar de bytes circulando a través de canales electrónicos, y por supuesto, las publicaciones científicas impresas también constituyen un negocio que emplea a decenas de miles de personas en todo el mundo. A esto se añade el alcance aún limitado de *Internet* para todos.

Sin embargo, ya existen muchos partidarios, incluyendo a científicos muy reconocidos en su campo, que abogan por la proliferación de las publicaciones electrónicas como un medio más idóneo de hacer que el conocimiento se convierta realmente en un patrimonio de la humanidad. Proyectos como NCSTRL, el Proyecto de Bibliotecas Digitales que llevan a cabo varias Universidades Estadounidenses y algunos proyectos locales en otros países, están demostrando las ventajas de la diseminación electrónica del conocimiento.

Thinking in *Java*

La publicación de libros, editados por casas editorialés de prestigio, no ha sido ajena a la influencia de *Internet*. No nos referimos aquí a los servicios que todas esas casas brindan a través de la red para el comercio electrónico, sino a la publicación de libros impresos (o a punto de editar) disponibles en su totalidad en la red. En estos momentos, existen las versiones electrónicas de algunos textos, en diferentes campos del saber. Para algunos de ellos, las ventas del libro impreso han aumentado, después de su publicación en la red. El libro que se revisa en este artículo que será publicado por Prentice-Hall en el mes de marzo de 1998, es uno que merece especial atención, por la manera en que surgió y maduró, hasta convertirse en uno de los mejores libros que se han publicado sobre el lenguaje de programación *Java*.

Thinking in Java, del autor ***Bruce Eckel*** es quizá el libro sobre computación que más revisores ha tenido antes de salir al mercado. **Bruce Eckel** es un autor conocido en el mundo de los lenguajes de programación orientados a objetos, con textos sobre C y C++ de mucha aceptación. Su libro sobre *Java* se escribe con un experimento poco usual entre los escritores de libros en general: el de poner en el dominio público ese experimento con el propósito de estudiar la aceptación del mismo desde sus inicios. Se trata de un texto que ha estado disponible en *Internet* y que ha sido leído por miles de lectores, quienes han tenido la posibilidad de criticarlo y de brindar al autor una información muy valiosa acerca del contenido del mismo. Cada nueva versión del libro hasta su versión final, disponible en el sitio ***www.eckelobjects.com*** ha considerado múltiples recomendaciones. Esta realimentación de futuros lectores ha logrado su objetivo: el de obtener un libro que satisface la mayoría de las necesidades de los que desean aprender *Java*.

La programación orientada a objetos es la metodología de programación, que a pesar de presentar un esquema o modelo superior a los paradigmas anteriores, tiene sin embargo una curva costosa de aprendizaje, donde casi nada de lo que este paradigma presenta es trivial. Se trata de un método de programación que mezcla muchos conceptos de diseño ajenos a otros lenguajes y por tanto la comprensión del mismo exige el entendimiento no solamente de conceptos de programación, sino también de análisis y diseño. La programación orientada a objetos se caracteriza fundamentalmente por la virtud de poder describir los programas en el espacio del problema, pudiendo representar las abstracciones o conceptos fundamentales de ese espacio con objetos datos en la computadora. Se trata de lenguajes con una semántica más rica, y por ende más compleja, que aquellos lenguajes procedurales y lenguajes de dominios particulares como LISP y PROLOG para el campo de la Inteligencia Artificial.

El reto de escribir un libro sobre lenguajes orientados a objetos radica en mezclar adecuadamente la explicación de esos conceptos de diseño con la sintaxis y aquellas cosas presentes en cualquier lenguaje de programación. *Java*, como lenguaje apropiado para la enseñanza de las técnicas

orientadas a objetos por el modelo de objetos que implementa, también introduce retos adicionales.

En menos de tres años de existencia pública de este lenguaje, se han escrito mucho más libros sobre él, que sobre el resto de los lenguajes de programación existentes. Añadir un nuevo libro a esa colección que brinde algo novedoso y que constituya un éxito, no parece ser una tarea fácil. Si a eso se añade todo lo que rodea al lenguaje *Java*, que aunque relacionado con el lenguaje no es parte de él, entonces las cosas se complican un poco más. Toda persona que compra un libro sobre *Java*, desea encontrar capítulos en el mismo que expliquen los paquetes de clases, el modelo de programación concurrente del lenguaje, la arquitectura de la máquina virtual, la programación centrada a red, el modelo de objetos distribuidos que soporta el lenguaje, etc., lo cual resulta imposible para cualquier proyecto práctico. Sin embargo, esa es la expectativa que se crea al presentar a este lenguaje con todo el entorno que le rodea.

Imagine el caso de alguien que espera encontrar en un libro sobre el lenguaje C, todas las cosas que se han realizado sobre ese lenguaje y todos los esquemas y arquitecturas de cómputo que se basan en el mismo. El libro sería interminable y por lo tanto incompleto.

Al escribir un libro sobre algún lenguaje procedural, como puede ser el caso del lenguaje C, el autor puede obviar todo lo concerniente al diseño de sistemas de software, pues lenguajes de ese tipo no incluyen una semántica que permite la descripción en el espacio del problema. Esto no sucede con los lenguajes orientados a objetos y mucho menos con *Java*, que al heredar las características de varios lenguajes como *C++, Smalltalk y Cedar/Mesa*, incorpora en sus construcciones más consideraciones de diseño que los que le sirvieron de fuente. No basta por ejemplo explicar el concepto de paquetes en *Java* como un mero agrupamiento de clases, si no se abunda en el concepto de categorías de clases como mecanismo de diseño para la clasificación de las entidades del dominio del problema. Si el concepto de paquete se ilustra solamente como un mecanismo de agrupamiento de implementaciones relacionadas, entonces el lector perderá la proyección hacia los conceptos de diseño que los paquetes involucran.

Casi todas las cosas que están presentes en *Java*, con excepción de aquellas típicas de cualquier lenguaje imperativo como sentencias de control de flujo, tipos incorporados, etc., tienen una componente de diseño implícita que es necesario hacer explícita para poder entender la potencia del lenguaje.

Eckel en su libro, trata de satisfacer ambos requerimientos, aunque en ocasiones se desvía un poco hacia aspectos puramente de programación que no tienen que ver mucho con los conceptos de la programación orientada a objetos. Esas ocasiones son raras y para nada le restan valor a los temas tratados. Hay que considerar también que se trata de un libro escrito fundamentalmente para diseñadores que ya conocen los fundamentos de la programación orientada a objetos con el modelo de objetos de C++. A pesar de eso, el libro puede ser usado por aquellos que se enfrentan a este estilo de programación por primera vez, aunque para este tipo de lector, acompañar el estudio con alguno de los libros clásicos sobre sistemas orientados a objetos puede resultar muy útil.

Thinking in Java, en su versión electrónica (versión completa del texto que publicará Prentice Hall) consta de 843 páginas en formato PDF. El paginado varía para otros formatos que se ofrecen. Consta de 17 capítulos y 6 anexos.

Los primeros 8 capítulos explican los conceptos básicos de la programación orientada a objetos y la implementación que hace *Java* de esos conceptos. Los ejemplos que se utilizan, a pesar de parecer "extraños" al principio son muy ilustrativos pues reflejan actividades naturales donde se presentan esos conceptos básicos. Todos los ejemplos han sido ampliamente probados y exigen muy pocos recursos computacionales para poder ejecutarlos. De hecho, todo el código que se incluye hace uso de la versión pública de *Java*: el JDK de *Sun MicroSystems*, disponible en la red desde la aparición de lenguaje.

El modelo de objetos de *Java*, basado en el acceso a éstos a través de referencias, permite la eliminación de apuntadores del contexto de la programación y presenta un enfoque homogéneo para la gestión y administración de objetos. Esa uniformidad le permite incluir un subsistema de recolección de basura eficiente.

Eckel, como representante del lenguaje C++ no deja pasar por alto el dilema de referencias y apuntadores y discute los primeros como apuntadores seguros que no permiten aritmética. Esa comparación, si bien ilustra las diferencias con C++, no parece apropiada, pues únicamente se refiere a aspectos de implementación del lenguaje. La presencia del pensamiento de C++ se manifiesta también en el uso del término "handle" en lugar del de

referencias para nombrar los identificadores de objetos.

Que las referencias se codifiquen internamente como apuntadores, es un asunto que no concierne al diseñador que persigue un modelado con objetos. A fin de cuentas, los apuntadores o cosas internas relacionadas con direcciones, están presentes en muchos de los métodos nativos del lenguaje y en esquemas tales como el de la identidad de los objetos persistentes en memoria.

El libro es extenso y cubre todos los aspectos que un libro intermedio sobre *Java* debe incluir. Los capítulos dedicados al modelo distribuido de objetos y el que presenta los patrones de diseño, sintetizan de forma excelente lo que subyace bajo el diseño y la programación basada en la experiencia.

Los anexos logran presentar un resumen cohesivo de un extenso material que en otros textos sobre el entorno de *Java* aparece de forma dispersa. ***Thinking in Java*** tendrá su "versión clásica"

dentro de muy pocos días. La casa Prentice Hall ha aceptado el experimento de vender un producto que ya es gratis. ¿Cuántos de aquellos que ya disponen de la versión electrónica lo comprarán? Tal experimento puede ser muy valioso para lograr libros de excelente calidad como el que aquí se ha reseñado.