Tecnología

Tendencias de la Tecnología de Objetos

J. Cervera*, E. Marcos**
GIT, DYCSA*; GO!, Grupo de Objetos, Univ. Rey Juan Carlos*

jcervera@dycsa.es
cuca@escet.urjc.es

Resumen: *en este artículo presentamos una visión general del estado actual de la Tecnología de Objetos, así como de lo que, en nuestra opinión, constituirán los puntos clave de dicha tecnología en los próximos años.*

1. Introducción

Desde el nacimiento de los primeros lenguajes de programación orientados a objetos, en la década de los 60, la Tecnología de Objetos (en adelante TO) se ha presentado como la solución a muchos de los problemas del desarrollo de software, consolidándose, en la década de los 90, como el paradigma de desarrollo para sistemas de complejidad media o alta (aplicaciones CAD/CAM, multimedia, aplicaciones distribuidas, etc.). Además, proporciona otra serie de ventajas como facilitar la reutilización de código o permitir obtener aplicaciones más fácilmente escalables, (Heinckiens, 1998). En la actualidad, y especialmente desde la aparición de Java, la TO parece ser también la que mejor se adapta a los nuevos tipos de desarrollos hipermedia y Web (Schawe y Rossi, 1995).

Hablar de TO lleva consigo la necesidad de hablar de objetos, de componentes, de patrones, de lenguajes, de *frameworks*, de arquitecturas, de persistencia, etc. Por otra parte, la TO hoy pasa por la integración de cada una de las técnicas y herramientas de desarrollo, tanto de análisis y diseño, como de programación, persistencia o distribución. Cada una de estas **palabras clave**, así como la problemática de la integración de las mismas, constituye un área de interés que podría, sin duda alguna, dar lugar a un articulo como éste. Por ello, no pretendemos profundizar en ninguna de estas áreas sino, más bien, ofrecer una visión general de la TO en la actualidad así como de lo que, en nuestra opinión y haciendo una arriesgada previsión de futuro, constituyen las principales tendencias de la misma.

2. El estado actual y las tendencias de la OO

En cuanto a tendencias, hay que separar claramente dos aspectos: tendencias en cuanto a *métodos para el desarrollo y gestión*, que ayudan en la concepción, diseño, codificación y pruebas así como la gestión de un sistema OO (Orientación a Objetos); y tendencias en cuanto a la *tecnología o infraestructura de desarrollo y explotación* propiamente dicha.

2.1. Procesos de desarrollo y gestión

Como consecuencia del nacimiento de los primeros lenguajes de programación OO así como de la complejidad creciente de las aplicaciones, comienzan a surgir, en la década de los 80, los primeros métodos de análisis y diseño OO. Surgen así diversas metodologías, entre las que podemos destacar el método de Booch (centrado en las etapas de diseño y construcción), OOSE de Jacobson (que proporcionaba un buen soporte para los casos de uso) y OMT de Rumbaugh (más enfocada al análisis de sistemas de información con grandes volúmenes de datos). Estos tres autores se unieron para definir **UML**, Lenguaje de Modelado Unificado (Booch et al., 1999), un conjunto de técnicas y un lenguaje de definición para el modelado de sistemas OO. UML 1.1 fue aprobada por el OMG como notación estándar de modelado en septiembre de 1997. En 1998 se aprobó la versión 1.2 y en 1999 la 1.3. En la actualidad se espera UML 1.4, que incluirá revisiones menores al estándar (Kobrin, 1999) y UML 2.0 (OMG, 1999).

Sin embargo, y pese a toda la funcionalidad que parece poseer UML y de la clara aceptación que ha tenido en el mercado, existen otras propuestas relevantes como OPEN (Henderson-Sellers, 1996), Catalysis (D'Souza y Cameron, 1999) para el desarrollo basado en componentes y frameworks con UML, o OHDM (Schawe y Rossi, 1995), para el desarrollo OO en Web.

2.2. Tecnología

Llamamos tecnología a la infraestructura que nos permite desarrollar sistemas OO: servidores de transacciones, *middleware*, Sistemas de Gestión de Bases de Datos (SGBD), Lenguajes de Programación (LP), etc.

· En cuanto a *middleware*, CORBA es probablemente la solución a corto plazo para arquitecturas distribuidas, aunque DCOM también mantendrá un puesto importante. Actualmente existen productos que soportan el estándar de CORBA 2.3 (p.e Visibroquer 4.0, Orbix 2000, MICO, etc.) esperándose, en menos de un año, productos que soporten el estándar de CORBA 3.0. Además, se seguirá evolucionando y ampliando las funcionalidades de los ORBs (*Object Request Brokers*), mejorando los rendimientos y dando soporte a aspectos tales como mejoras en la seguridad, tolerancia a fallos, manejo de transacciones, etc.
· Las ventajas de los monitores transaccionales (p.e. TUXEDO) y los ORB se están fundiendo en los CTM (*Component Transaction Monitors*) o servidores de aplicación (p.e. Net Dinamycs o MTS - *Microsoft Transaction Server*). Estos servidores de aplicación combinan la fluidez y facilidad de acceso de los sistemas de objetos distribuidos basados en ORB con un monitor transaccional. Los CTM proporcionan un entorno de componentes en el servidor que gestiona la concurrencia, transacciones, distribución de

objetos, balance de carga, seguridad, y gestión de recursos automáticamente. Actualmente esta tecnología, que podría imponerse a medio y largo plazo, está en desarrollo centrándose casi exclusivamente en Java. Existen ya varios CTM que soportan el modelo de *Enterprise JavaBeans* (EJB).

· Actualmente, los **Sistemas de Gestión de Bases de Objetos** (SGBO), como ObjectStore o POET, no tienen la madurez suficiente para soportar aplicaciones con grandes volúmenes de datos que requieren rápidos tiempos de respuesta. Para este tipo de aplicaciones se utilizarán las nuevas versiones de los productos objeto-relacionales (p.e. Oracle 8i, Universal Server de Informix). Sin embargo, los SGBO pueden ofrecer buenas prestaciones para BD, de tamaño pequeño o medio, que estén orientadas a Internet y que sean programadas en Java.

· En cuanto a los **lenguajes de programación**, se puede pronosticar, pese a los problemas de rendimiento que presenta todavía este joven lenguaje (Wiebe, 1999), una mayor utilización de Java para sistemas de gestión y de JavaBeans para el desarrollo basado en componentes. Asistiremos también en los próximos años a una evolución en los LP visuales (p.e Visual Basic), así como a un retroceso del C++ que quedaría más para software de sistemas y menos para sistemas de gestión. Hay que tener en cuenta que este lenguaje no está pensado para un desarrollo basado en componentes. Un avance importante para el desarrollo de aplicaciones distribuidas en la Web es, tal y como señala Sutherland (1999), la estrategia de sinergia entre Java y ActiveX basada en OLE/COM, lo que facilita la construcción de componentes COM usando Java del mismo modo que se usaba Visual Basic o C++.

· Las ventajas presupuestas inicialmente a la OO en cuanto a reutilización de software, no han sido, en la práctica, las esperadas. Sin embargo, la OO es el punto de partida para lo que se conoce como **desarrollo basado en componentes** y que no constituye una nueva tecnología, sino una evolución natural de la TO. La reutilización de componentes ha comenzado con la reutilización de componentes ActiveX, algo que es práctica habitual en la actualidad. Sin embargo, para la reutilización de componentes de negocio queda mucho camino por recorrer. A medio plazo podrán existir componentes funcionales en diferentes dominios de aplicación. Se hace también necesaria la aparición de herramientas que faciliten el desarrollo basado en componentes (Allen y Frost, 1998).

· Una evolución de los objetos son los **agentes**. Los mismos principios de la TO, lógicamente adaptados y ampliados para este nuevo tipo de objeto, serán igualmente aplicables a los agentes. Así, tal y como señala Odell (2000), podemos tener jerarquías de agregación de agentes, colecciones de agentes, patrones de agentes, componentes de agentes, etc.

· El estándar de OMG permite la **movilidad** transparente de servicios de objetos a través de una red. Lo que en realidad se puede transferir de un punto a otro de la red son referencias de objetos, pero no objetos completos. La serialización de objetos de Java, según afirma Szyperki (1999), es una de las aproximaciones más cercanas para solucionar este problema, pero sin embargo continúa siendo una solución muy cara en tiempo de ejecución.

· En un futuro no muy lejano, la TO podría evolucionar hacia la manipulación de **objetos físicos distribuidos** en una red. La evolución de los dispositivos electrónicos tiende a que cada uno de ellos lleve un pequeño ordenador que se pueda conectar a una red de datos, publicando sus servicios. En esta situación será posible que cada dispositivo electrónico (doméstico o industrial) se presente como un objeto distribuido, en una aplicación central que permitiría que interactuaran entre sí.

3. Algunas predicciones de futuro

Tal y como afirma Fishman (1997), la tecnología de objetos y componentes en la presente década estaría basada en la utilización de CORBA, COM y *JavaBeans* para la implementación de sistemas de información distribuidos. Sin embargo, está predicción es no decir mucho más de lo que todos podemos intuir. Para terminar, y haciendo un previsión un poco más arriesgada, vamos a resumir lo que podrían ser los puntos claves de la TO en los próximos años:

1. Ampliación del estándar de UML para el modelado de nuevas aplicaciones (web, WAP, etc.) así como para el desarrollo basado en componentes.
2. Desaparición casi total de C++ para sistemas de gestión a favor de una mayor utilización de Java.
3. Utilización progresiva de los servidores de aplicaciones para sistemas de gestión.
4. Las BD reducirán sus funcionalidades, ocupándose casi exclusivamente de la persistencia de objetos y de un modo totalmente transparente.
5. Aparición del «Jwap», una extensión de Java para aplicaciones WAP.
6. Serán habituales los entornos de desarrollo basados en componentes que permitirán construir sistemas «prefabricados».
7. Cibertiendas de aplicaciones «hágalo usted mismo», donde el usuario, por menor coste, se lleva a casa un sistema desmontado que el mismo tendrá que componer.
8. Conexión de objetos físicos a la red, que permitirán, por ejemplo, poner a calentar la comida antes de salir del trabajo, o leer el contador de la luz desde un teléfono movil.

4. Referencias

Allen, P. y Frost, S. (1998), *Component-Based Development for Enterprise Systems. Applying The SELECT Perspective™*. Cambridge University Press.

Booch, G. Rumbaugh, J. y Jacobson, I. (1999), *El Lenguaje Unificado de Modelado*. Addison Wesley Iberoamericana, Madrid.

D'Souza, F. y Cameron, A. (1999), *Objects, Componentes, and Frameworks with UML. The Catalysis℠ Approach*. Addison-Weley.

Fishman, D. (1997), «Object-Oriented Information Systems in the 21st. Century». Conferencia invitada en: *International Conference on Object Oriented Information Systems*. Ed. M. E. Orlowska y R. Zicari, Springer.

Heinckiens (1998), *Building Scalable Database Applications. Object-Oriented Design, Architectures, and Implementations*. Peter M. Heinckiens. Addison-Wesley, 1998.

Henderson-Sellers (1996) «The OPEN-MeNtOR Methodology». Brian Henderson-Sellers. *Object Magazin*. Noviembre, pp.56-59.

Kobrin, C. (1999), «UML 2001: A Standarization Odyssey». *Communications of the ACM*, Vol. 42, No. 10, pp.29-37.

OMG (1999), *UML 2.0 Request for Information V1.0.* Analysis & Design Platform Task Force. OMG, Agosto 1999. En: *http:/www.omg.org.*

Odell, J., 2000, Agents and Emergence. *Journal on Object Oriented Programming*. Febrero, pp 34-36.

Schawe y Rossi (1995), «The Object-Oriented Hypermedia Design Model». *Communications ACM*, 58(8), pp. 45-46.

Sutherland (1999), *Business and the Evolution of the Internet*. Accedido el 13 de marzo de 2000 en: *http://www.onemind.com/hpj/917622885/index.html*

Szypersky, C. (1999), *Componnt Software. Beyond Object-Oriented Programming*. Addison-Wesley.

Wiebe, J. (1999), *Poly-wha? Understanding Business Object Terminology and Acronyms*. Accedido el 13 de marzo de 2000 en: *http://www.onemind.com/hpj/919200664/index.html.*