Una experiencia docente en la enseñanza del desarrollo orientado a objetos de *software* utilizando herramientas visuales

José A. Pérez Castellanos, Rafael Corchuelo Gil y Miguel Toro Bonilla
Departamento de Lenguajes y Sistemas Informáticos
Universidad de Sevilla
Tlf: (95) 455.27.70, Fax: (95) 455.71.39
E-mail: {jpcast, corcho, mtoro}@cica.es

Resumen: *En este trabajo ofrecemos una visión general de nuestra experiencia impartiendo la asignatura "Desarrollo orientado a objetos de software utilizando herramientas visuales" que se ha comenzado a ofrecer este año en la Facultad de Informática y Estadística de Sevilla. El curso se oferta como asignatura convalidable por créditos de libre configuración y tiene como objetivo fundamental ofrecer a los alumnos una formación de marcado carácter práctico en herramientas y técnicas que son de aplicación habitual dentro de nuestro entorno empresarial.*

1. Introducción

Una proporción muy considerable del *software* que se desarrolla en la empresa es *software* de gestión, caracterizado normalmente por ser muy interactivo, fácil de manejar (el usuario habitual no tiene por qué tener conocimientos de informática) y por hacer uso intensivo de bases de datos. A pesar de ser *software* simple desde el punto de vista funcional, estas aplicaciones suelen resultar caras en su desarrollo, debido fundamentalmente a la fuerte interactividad a la que están obligadas para ser fácilmente manejadas por cualquier usuario.

Este hecho, unido al auge de los entornos gráficos (X-Window en sistemas UNIX, Presentation Manager en OS/2, MS Windows, FlexOS, etcétera), ha propiciado la proliferación en el mercado de las llamadas herramientas RAD (*Rapid Application Development*) para el desarrollo de *software*: Visual Basic, Delphi, Visual C++, etcétera. Estas herramientas, que analizaremos con más detalle en el siguiente apartado, suelen constar de un entorno de diseño visual de aplicaciones, una biblioteca de clases o componentes y un compilador de un lenguaje de programación sobre el cual se basan.

A continuación se enumeran sus principales características:

- Son herramientas visuales. El programador diseña gráficamente el *interface* de la aplicación. Dicho *interface* produce una serie de eventos, a cada uno de los cuales se les asigna posteriormente el código que debe ejecutar.

- Son herramientas orientadas a objetos. La forma en que estas herramientas reducen el coste de desarrollo del *software* es mediante la reutilización del código, bien sea del código de la propia biblioteca de la herramienta o del código escrito por el usuario. Es por ello que todas estas herramientas se basan en un lenguaje que da soporte en cierto grado a la programación orientada a objetos (Visual Basic, Object Pascal, C++...).
- A pesar de ser herramientas orientadas a objetos, en cuanto al manejo de las bases de datos suelen reaprovechar los motores de bases de datos relacionales existentes actualmente en el mercado.
- Otra característica común a todas ellas es su intuitividad y facilidad de manejo. Esto conlleva que cualquier persona, aunque cuente con unos conocimientos muy básicos en programación y ningún conocimiento de orientación a objetos, puede sentarse delante del ordenador y desarrollar en muy poco tiempo un programa de aspecto profesional, pero de ínfima calidad ya que ha sido desarrollado desconociendo la metodología en la que se basa la herramienta utilizada.

Lo que en cualquier caso es un hecho innegable es que estas herramientas RAD disminuyen drásticamente el tiempo de desarrollo del *software* que típicamente necesitan las empresas a la vez que mejoran la presencia y facilidad de uso de las aplicaciones desarrolladas. Esto ha provocado una gran demanda de programadores con conocimientos en el manejo de estas herramientas y dada su facilidad de uso, dicha demanda ha encontrado respuesta. Eso sí, en la mayoría de los casos por parte de programadores que no hacen un uso correcto de las posibilidades que ofrece la herramienta para desarrollar *software* de calidad.

En este artículo mostramos cuál ha sido nuestra experiencia impartiendo la asignatura "Desarrollo Orientado a Objetos de Software Utilizando Herramientas Visuales", que ha comenzado a impartirse este año en la Facultad de Informática y Estadística de la Universidad de Sevilla. El curso ha sido motivado por nuestra experiencia de trabajo con distintas empresas de nuestro entorno, durante la cual hemos podido comprobar el mal uso que se suele hacer de este tipo de herramientas, y de nuestro deseo de ofrecer a nuestros alumnos una formación de buen nivel académico complementada siempre con una buena formación de carácter práctico que les resultará de aplicación inmediata en su vida profesional. Para poder fijar los contenidos con precisión hemos realizado varias experiencias piloto previas en las que hemos ofrecido este curso a profesionales de nuestro entorno que trabajan en empresas dedicadas a la construcción de *software* de gestión.

2. Nuevas herramientas RAD

Tal vez la herramienta RAD de uso más extendido, en parte por ser la primera en esta categoría y en parte por haber sido desarrollada por MicroSoft, sea Visual Basic [1]. Esta herramienta se basa en una evolución del lenguaje Basic dotada con la posibilidad de manejar objetos (los componentes visuales encapsulan datos y código), pero sin la posibilidad de definir nuevas clases de objetos. A pesar de ello, y gracias a estar basada en Basic, lenguaje de gran difusión entre los profanos en la materia, es la herramienta que con más aceptación ha contado. Su principal característica es la facilidad de manejo, especialmente para programadores no profesionales.

También proporcionado por MicroSoft, existe en el mercado Visual C++ [9]. Como su nombre indica, esta herramienta se basa en el lenguaje C++. Su punto débil es que no se trata de una herramienta de desarrollo 100% visual, ya que se mueve a un nivel más bajo y está basada en las conocidas MFC (*MicroSoft Foundation Classes*). Esta herramienta permite realizar tareas concretas que no pueden ser directamente implementadas en Visual Basic, en el cual el *interface* del programa se

puede definir de una forma completamente visual. Esta es la causa de que esta herramienta haya venido siendo utilizada típicamente por programadores profesionales. Ahora bien, el tiempo de aprendizaje es considerablemente mayor y la dificultad de manejo es también sensiblemente mayor, por lo cual hay profesionales que dudan de que realmente se trate de una herramienta RAD.

En un punto intermedio está Delphi [5], herramienta aportada por Borland y basada en el lenguaje Object Pascal 7.0. A pesar de utilizar como lenguaje base uno que no es tan popular como Basic, su tiempo de aprendizaje es ínfimo si se compara con Visual C++ y su potencia es sensiblemente superior a la de Visual Basic. Además, y a diferencia de éste, el lenguaje Object Pascal es un lenguaje orientado a objetos en el que se pueden definir nuevas clases de objetos, reutilizarlas mediante herencia, etcétera.

Al igual que MicroSoft proporciona Visual C++ para aquellas aplicaciones en que la potencia de Visual Basic no es suficiente, Borland acaba de lanzar al mercado su nueva herramienta Borland C++ Builder. De una forma sencilla podríamos definirla como Delphi basado en C++, en lugar de en Object Pascal. Las principales deficiencias de Delphi están en su lenguaje base, que no permite el uso de herencia múltiple ni tampoco el concepto de tipo de dato genérico, por lo que Borland ha corregido estas deficiencias sustituyendo el lenguaje base por C++. A pesar de estar basado en dicho lenguaje, dado que conserva la filosofía de diseño de Delphi, esta herramienta es sensiblemente más fácil de manejar que su homóloga de MicroSoft, Visual C++.

Mención especial merece el lenguaje TCL [7] (*Tool Command Language*), normalmente asociado al *toolkit* TK (TCL/TK). El lenguaje TCL es básicamente otra *shell* más del sistema operativo UNIX, aunque cuenta con una semántica propia. La novedad de esta *shell* es que está asociada al *toolkit* TK para desarrollo de aplicaciones visuales en ambientes X-Window, por lo que proporciona una potencialidad enorme para el desarrollo de sistemas gráficos. Desde que empezó a utilizarse de forma generalizada en 1.993, siempre se le ha atribuido cierta similitud con Visual Basic. Debido al gran éxito que ha tenido la versión mas reciente (la 7.5 de TCL y 4.1 de TK), se han hecho versiones "oficiales" para otras plataformas. Es por esto que resulta posible llevar una aplicación TCL/TK que originalmente fue programada sobre X-Window a Windows 3.1, Windows 95, MAC, o viceversa. Esta portabilidad es una característica de la que carecen las aplicaciones desarrolladas con las demás herramientas visuales.

Por último, cabría mencionar herramientas con menor difusión como Optima++ o Visual Eiffel. Optima++ [2] ha sido desarrollada por Watcom y usa como lenguaje base C++. Está fuertemente inspirada en Delphi, del cual toma prestado el aspecto de su *interface*. Visual Eiffel [11] ha sido desarrollado por SiG Computer y pese a su nombre no es una herramienta de desarrollo visual tan avanzada como Delphi. Básicamente se trata de un compilador de Eiffel para Windows 95 que incorpora entre las herramientas de su entorno un diseñador gráfico de *interfaces* visuales. En cualquier caso la integración entre esta herramienta y el entorno de desarrollo no alcanza el grado de acabado que tiene Delphi en la actualidad.

Finalizamos esta sección haciendo referencia a otras herramientas visuales como Visual FoxPro, Visual dBase, etcétera... No obstante, estas últimas son aplicaciones típicamente de oficina, con escasa aceptación por parte del colectivo de programadores.

3. Necesidad del curso

Los alumnos se quejan habitualmente de la diferencia conceptual existente entre la preparación académica que reciben y la realidad laboral en las empresas en las que acabarán desarrollando su vida profesional.

La formación académica que se imparte pretende adiestrar a los alumnos en el uso de métodos formales de análisis y programación a un nivel mucho más alto de lo que normalmente se precisa en una pequeña o mediana empresa. Paradójicamente, se echa en falta el adiestramiento en el uso de herramientas concretas de desarrollo que son frecuentemente empleadas por profesionales con un nivel de preparación académica muy inferior. Esta es la razón fundamental por la que se hace un uso poco adecuado, o al menos poco ortodoxo, de estas herramientas, obteniéndose así aplicaciones de baja calidad, poco robustas y difícilmente modificables o reutilizables.

Nuestro objetivo fundamental al abordar este curso es proporcionar al alumno una visión general del panorama actual del uso de las herramientas visuales en las empresas de nuestro entorno y ofrecerles una formación práctica de inmediata aplicabilidad.

4. Enfoque del curso

El curso se caracteriza por un marcado enfoque práctico en el que en varias ocasiones hemos sacrificado la calidad en términos estrictamente académicos en aras de una mejor formación práctica. Hemos intentado que los alumnos cuenten en el curso con información útil y actualizada acerca de qué herramientas se emplean en las empresas de nuestro entorno y acerca del grado de utilización que en ellas se hace de las mismas.

El curso se ha dividido en tres partes que intentan abordar de una forma sistemática las tres etapas habituales en cualquier ciclo de vida de un producto *software*: análisis, diseño e implementación. No tratamos el tema del análisis de requisitos, el del mantenimiento de las aplicaciones o el de la reingeniería puesto que estos aspectos son tratados con suficiente nivel de detalle en la asignatura "Ingeniería del *Software*" de quinto curso de la Licenciatura en Informática.

En las secciones siguientes comentaremos con detalle los contenidos de cada una de las partes del curso y los conocimientos que se espera obtengan los alumnos de cada una de ellas.

4.1 Primera parte: Análisis

Está plenamente aceptado que el desarrollo de una aplicación debe comenzar con un conjunto de entrevistas con el cliente en las que se desarrolla un catálogo de requisitos en el que se describe de una manera informal a qué debe dar respuesta la aplicación informática que vamos a desarrollar. A partir de este documento se comenzará a trabajar en un modelo semiformal del producto en el que analizaremos cada una de las necesidades del cliente, pero sin entrar en los detalles concretos de cómo se les da solución en una máquina o con unas herramientas concretas. El resultado debe ser siempre un documento fácil de entender en el que todos los elementos del problema hayan sido estudiados y desmenuzados hasta sus últimas consecuencias. Un análisis riguroso del problema permite al equipo de desarrollo enfrentarse a errores de comprensión en una fase temprana en la que su corrección aún resulta sencilla y poco costosa.

En un análisis orientado a objetos los sistemas se modelan como un conjunto de objetos relacionados que concurrentemente en el tiempo desarrollan ciertas actividades e interaccionan entre sí.

No estamos inicialmente preocupados por las funciones que va a desarrollar el sistema, ni por qué almacenamientos vamos a utilizar o cuáles van a ser los flujos de información entre los distintos componentes. Nuestro interés principal es estudiar las entidades que aparecen dentro del dominio del problema y las interrelaciones que se establecen entre ellas. Posteriormente estudiaremos la dinámica y la funcionalidad de cada una por separado.

Hemos seleccionado la metodología OMT [3] para abordar esta parte del curso puesto que es una de las más asentadas y de más amplia aceptación dentro de nuestro entorno empresarial. Por otra parte los alumnos pueden encontrar abundante bibliografía y también diversas herramientas visuales para poder desarrollar sus modelos: Object Domain, Meta Case, Meta Class, Rational Rose, etcétera.

Quizá la característica que ha hecho de OMT una de las técnicas más utilizadas sea su intuitividad, puesto que los modelos desarrollados con ella resultan fáciles de entender incluso con unos conocimientos básicos de la técnica, lo que nos permite convertir a estos modelos en un documento fundamental de trabajo entre la empresa desarrolladora y el cliente. Los modelos desarrollados con OMT no son formales, lo que significa que no contamos con una semántica formal para los mismos y por lo tanto desde el punto de vista académico carecen del interés con el que técnicas como OOZE [4] o Z^{--} [10] cuentan. Pero insistimos en que son una herramienta adecuada para la comunicación con el cliente y con el equipo de desarrollo de la aplicación. En ningún momento del curso presentamos OMT como una panacea universal con la que podremos dar solución a todos los problemas, ni tampoco como la mejor técnica que existe.

El objetivo de esta parte del curso es que los alumnos conozcan la técnica OMT y sepan aplicarla en el análisis de cualquier aplicación habitual de gestión. Para conseguir este objetivo se realizan multitud de ejemplo prácticos en los que se persigue que el alumno sea capaz de identificar objetos, clases, relaciones entre las mismas y adquiera soltura en el manejo de los diversos elementos de OMT. También insistimos mucho en la importancia del documento de análisis y en que cuando en el mismo quedan ideas vagas lo único que vamos a conseguir es posponer problemas a etapas más avanzadas del desarrollo en las que su corrección va a resultar muchísimo más costosa.

4.2 Segunda parte: Diseño

En la parte de diseño de aplicaciones volvemos a poner de manifiesto que no pretendemos presentar en este curso las metodologías orientadas a objetos como las únicas utilizables para obtener buenas aplicaciones informáticas. Y esto queda aún más patente en esta fase en la que proponemos la implementación de los modelos orientados a objetos sobre herramientas estructuradas clásicas. La razón es que la inmensa mayoría de las empresas interesadas en el desarrollo de *software* con métodos orientados a objetos no puede abordar la renovación completa de toda su infraestructura, que actualmente está formada por herramientas de desarrollo basadas en lenguajes estructurados clásicos o de cuarta generación y sistemas de gestión de bases de datos relacionales.

El método de diseño que mostramos en el curso sigue básicamente las recomendaciones de [3], pero desarrollamos con mucho mas detalle el tema de la administración de los almacenes de datos. Básicamente nos hemos decidido por utilizar los sistemas clásicos de bases de datos relacionales a los que la herramienta de implementación que hemos elegido ofrece un buen soporte. El método para implementar los objetos y las relaciones identificadas durante la fase de análisis sigue las líneas expuestas en [6]. Resulta un método sencillo de aprender e intuitivo para todos aquellos que tienen experiencia en el manejo de bases de datos relacionales y por lo tanto es muy adecuado.

Un objetivo importante de esta segunda parte del curso es que el alumno aprecie la dificultad que las empresas tienen para renovar por completo su infraestructura y la necesidad, por tanto, de aprovechar para la implementación las herramientas estructuradas con las que cuentan en el momento actual. También se realizan diversos ejercicios que le permitirán adquirir destreza en el método de diseño que se enseña.

4.3 Tercera parte: Implementación

Si bien la adquisición de modernos sistemas de bases de datos orientados a objetos resulta difícil para las empresas de nuestro entorno, lo cierto es que las herramientas de programación visuales orientadas a objetos resultan bastante asequibles y además suelen ofrecer un adecuado soporte para la comunicación con sistemas de bases de datos relacionales. Esta es la razón de que para la implementación hayamos elegido la herramienta Delphi [8] de Borland.

Esta herramienta presenta una excelente relación de compromiso entre la calidad del *software* que se pueda desarrollar con ella, (ya que es 100% orienta a objetos, excepto en lo que al manejo de bases de datos se refiere) y la dificultad que entraña su aprendizaje. Otro punto a su favor es que es capaz de reaprovechar componentes desarrollados para otras herramientas visuales (Visual Basic) y que genera código ejecutable nativo, muy eficiente en tiempo de ejecución. Las aplicaciones generadas son fácilmente escalables, y la creación y reutilización de componentes es también sencilla. Además, hay disponible una gran cantidad de componentes de libre distribución para resolver problemas tan diversos como representación de gráficos, control de *módems*, comunicaciones en internet o en redes locales, etcétera.

5. Diseño de la asignatura

La asignatura es cuatrimestral y a los alumnos se les exigen como requisitos antes de la matriculación contar con conocimientos de técnicas de desarrollo estructurado, de algún lenguaje de programación imperativo y del sistema MS Windows.

El tiempo disponible se divide en cuatro bloques dedicados a enseñar la metodología de análisis (40 %), el método de diseño (10 %), el lenguaje de programación Object Pascal (10 %) y el entorno de desarrollo Delphi (40 %).

La evaluación de los alumnos se lleva a cabo en función al trabajo desarrollado en la construcción en grupos de tres personas de una aplicación típica de gestión empresarial.

El material con el que cuentan durante el curso son las notas de clase redactadas por los profesores que imparten la asignatura. Para el año próximo esperamos tener terminado un libro de texto complementario.

6. Conclusiones

La valoración del curso por parte del alumnado ha resultado bastante satisfactoria, pues en nuestras conversaciones mantenidas con los mismos nos han indicado que les resulta muy atractiva la idea de una asignatura tan práctica como esta. El número de matriculados, por limitaciones físicas, es de treinta, pero se recibieron unas ciento veinticinco solicitudes de admisión de alumnos de cuarto y quinto curso de Informática.

Por otra parte, haber ofrecido con anterioridad este curso a profesionales de nuestro entorno nos ha permitido conocer qué esperan de la nueva tecnología orientada a objetos, de forma que hemos podido ofrecer a nuestros alumnos un curso en el que han adquirido conocimientos listos para ser aplicados en su vida profesional.

En años posteriores seguiremos ofreciendo el curso actual, pero iremos permanentemente adaptando el contenido conforme las necesidades del mercado laboral cambien y a la evolución de las herramientas RAD.

7. Agradecimientos

Sabemos que un curso de estas características es un verdadero fracaso si no cuenta con el apoyo de unos alumnos que gracias a sus comentarios y continua crítica a nuestra labor nos han impulsado a dotarlo de unas características eminentemente prácticas y absolutamente orientadas a prepararlos en herramientas que les serán de uso habitual en su vida profesional. Nuestro sincero agradecimiento a todos ellos.

8. Bibliografía

[1] "Enciclopedia del Visual Basic"
F. Ceballo
Ra—Ma, 1.994

[2] "Herramientas de Desarrollo: Optima++"
F. de la Villa
Sólo Programadores, N° 28, pp. 53—57, 1.997

[3] "Modelado y diseño orientado a objetos. Metodología OMT"
J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen
Prentice Hall, 1.996

[4] "OOZE: An object oriented Z environment"
A. Alencar
Actas del ECOOP'91 vol. 512, pp. 180—199, 1.991

[5] "Programación con Delphi"
F. Charte
Anaya Multimedia, 1.996

[6] "Requisitos para un generador automático de mecanismos de persistencia objeto-relacionales"
A. Durán y M. Toro
Actas III Jornadas de Informática, 1.996

[7] "TCL and the TK Toolkit"
J. Osterhout
Addison Wesley, 1.994

[8] "Visual Basic o Delphi, la gran pregunta"
L. Caballero
Sólo Programadores, N° 13, pp. 12—16, 1.997

[9] "Visual C++"
D. Kruclinski
McGraw Hill, 1.994

[10] "Z++: An object oriented extension to Z"
K. Lano
Z user meeting workshops in computing, Springer-Verlag, 1.991

[11] "Visual Eiffel"
SiG Computer GmbH
http://www.sigco.com