Aprendiendo los conceptos básicos de la programación orientada a objetos a través del enunciado

Carlos A. Lázaro Carrascosa

Departamento de Lenguajes y Sistemas Informáticos, Universidad Rey Juan Carlos
carlos.lazaro@urjc.es

Resumen El paradigma de programación orientada a objetos (POO) se ha convertido en un referente a la hora de afrontar proyectos "software", tanto en el mundo empresarial como en el académico. Es, por tanto, una labor para los docentes de la informática encontrar diferentes técnicas y enfoques pedagógicos para enseñarlo con eficacia. La taxonomía de Bloom, o la división del aprendizaje en diferentes niveles ofrece un marco idóneo para afrontar nuestra tarea con éxito. Este trabajo presenta una propuesta de división de los conceptos relacionados con la POO con el fin de desarrollar herramientas relacionadas con cada uno de estos conceptos, siempre bajo el marco de la citada taxonomía. Además, de una forma más concreta, ofrece la especificación de una de estas herramientas, llamada TextOO, dedicada a la enseñanza de los conceptos básicos de la POO enfocados desde el punto de vista del diseño. TextOO se basa en el uso de las especificaciones iniciales de los problemas en lenguaje natural, es decir, de los enunciados. Permite al alumnado seleccionar partes del enunciado, asociándolas con los elementos básicos de la POO (clases, objetos, atributos, valores, métodos y mensajes) a través de su representación en diagramas UML de clases, objetos y secuencia. Además, TextOO incluye un pequeño corrector que, a través de la comparación de la solución del alumnado con la del profesorado permite ofrecer cierta realimentación a los estudiantes. Hemos realizado una evaluación previa de TextOO con alumnado de ingeniería del software; aunque no es el grupo de estudiantes más adecuado, los resultados obtenidos concluyen que nuestra herramienta tiene cierta utilidad para el modelado, y, sobre todo, que al alumnado le resulta práctica y fácil de usar.

1 Introducción

La taxonomía de Bloom (Ver Tabla 1) es ampliamente utilizada en los ámbitos educativos y sirve de marco generalmente aceptado para medir o clasificar el aprendizaje de los conocimientos. Sin embargo, su uso presenta ciertas dificultades cuando intentamos adaptarla de un modo claro y eficaz al aprendizaje de la programación.

Tabla 1. Taxonomía de Bloom

Nivel de la taxonomía	Descripción
1 ó de conocimiento	El alumnado es capaz de reconocer o recordar información sin que sea necesario ninguna comprensión o razonamiento sobre lo que hay tras dicha información.
2 ó de comprensión.	El alumnado es capaz de entender y explicar el significado de la información recibida.
3 ó de aplicación.	El alumnado es capaz de seleccionar y usar datos y métodos para resolver una nueva tarea o un problema.
4 ó de análisis	El alumnado es capaz de distinguir, clasificar y relacionar hipótesis y evidencias de la información dada, así como descomponer un problema en sus partes.
5 ó de síntesis.	El alumnado es capaz de generalizar ideas y de integrarlas para resolver o realizar algún problema que es nuevo para él.
6 ó de evaluación	El alumnado está capacitado para comparar, criticar y evaluar métodos o soluciones para resolver un problema o para discernir la mejor entre varias soluciones.

Uno de los principales obstáculos aparece cuando el dominio objeto del aprendizaje es complejo. En estos casos, parece razonable descomponer el problema en diferentes partes, cada una de las cuales puede a su vez ser categorizada mediante los niveles de Bloom. Cabe pensar que el avance progresivo por cada uno de los subdominios implique la mejora general del problema global.

Sabemos, por ejemplo, que los *problets* de Kumar, pequeñas aplicaciones escritas en Java que abordan diferentes aspectos de la semántica de la programación estructurada, alcanzan el nivel de aplicación dentro del aprendizaje de conceptos concretos, como los bucles o las sentencias de selección, pero surgen dudas si nuestro objetivo es más grande: no sabemos qué incidencia exacta tienen estas herramientas si pretendemos alcanzar un nivel dado dentro de la programación estructurada completa.

Nuestro grupo de trabajo tiene interés en profundizar en la programación orientada a objetos (POO). Es éste un claro ejemplo de dominio de aprendizaje complejo. Para alcanzar altos niveles en su aprendizaje será necesario, por lo tanto, alcanzar altos

niveles en las partes que lo componen. ¿Cuáles son esas partes? A nuestro juicio, distinguimos varias:

- Conceptos básicos estáticos: clase, método, atributo.
- Conceptos básicos dinámicos: objeto, mensaje, estado.
- Relaciones básicas: composición, asociación, agregación, uso.
- Herencia.
- Polimorfismo.
- Construcciones sintácticas.
- Manejo de APIs existentes.

Por otra parte, distinguiremos el hecho de diseñar un programa orientado a objetos con el hecho de implementarlo, son dos tareas necesarias y complementarias, que sin embargo mantienen cierto grado de autonomía.

La POO se caracteriza de manera esencial por la fusión entre datos y procesos. Antes de que existiera la POO, los datos y los procesos se enseñaban por separado. Curiosamente, a la hora de enseñar conceptos relacionados con los procesos (programación estructurada, algoritmia), es común comenzar explicando aspectos relacionados con la implementación, como la sintaxis o las estructuras de datos para terminar con conceptos relacionados con el diseño y la ingeniería del *software* (ver tabla 2).

Tabla 2. Asignaturas de programación de Ingeniería Informática de la URJC

Curso	**Asignatura**
Primero	Introducción a la programación
Segundo	Estructura de datos
Segundo	Metodología de la Programación
Segundo	Estructuras de la información
Segundo	Programación orientada a objetos
Tercero	Diseño y análisis de algoritmos
Tercero	Sistemas de información
Cuarto	Ingeniería del *software* I
Quinto	Ingeniería del *software* II

Sin embargo, este orden es alterado cuando se trata de explicar conceptos relacionados con los datos. Es frecuente que en asignaturas de Bases de Datos se expliquen en primer lugar los modelos (Relacional, Entidad-Relación) para después explicar los aspectos de la implementación, como el paso a tablas y el lenguaje SQL (ver tabla 3).

Tabla 3. Temario de la asignatura "Bases de datos" de Ingeniería Informática de la UPM

Módulo I: Introducción a las Bases de Datos (2h)
UD 1: Presentación de la Asignatura
UD 2: Definiciones y Arquitectura de Base de Datos
Módulo II: Diseño Conceptual (12h)
UD 3: Modelo Entidad/Relación Básico
UD 4: Modelo Entidad/Relación Extendido
Módulo III: Paso del Diseño Conceptual al Diseño Lógico (20h)
UD 5: Modelo Relacional. Conceptos básicos
UD 6: Paso del M. Entidad/Relación al M. Relacional
UD 7: Integridad Referencial
UD 8: Introducción a SQL
Módulo IV: Diseño Relacional (20h)
UD 9: Álgebra Relacional
UD 10: Diseño de Bases de Datos Relacionales

Es razonable, por tanto, que en un paradigma en el que los datos y los procesos se funden sean válidos los dos enfoques. En resumen, la propuesta de esquema jerárquico de aprendizaje para la POO se ve reflejada en el siguiente gráfico:

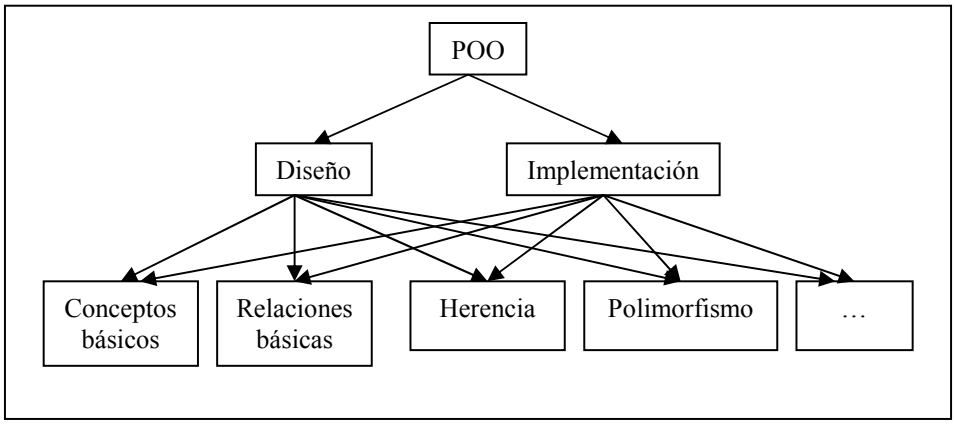


Fig. 1. Descomposición de conceptos relacionados con la Programación Orientada a Objetos

Cada uno de los nodos del árbol mostrado admite la descomposición de su aprendizaje de acuerdo con la taxonomía de Bloom, pero pensamos, sin embargo, que el avance de los niveles en los nodos hoja está estrechamente relacionado con el avance en los nodos superiores. Es por esto por lo que pretendemos centrar nuestros esfuerzos en los aspectos relacionados en los nodos hoja, tanto desde la perspectiva del diseño como desde la perspectiva de la implementación, así como, en su caso, desde la relación entre ambas.

Para ello pretendemos desarrollar, con esta base, una plataforma de aprendizaje compuesta de varios módulos, que aborden todos y cada uno de los aspectos arriba mencionados. Este capítulo está centrado en la implementación inicial de esta idea: la herramienta TextOO, orientada al aprendizaje de los conceptos y relaciones básicos de la POO desde el punto de vista del diseño, a través del modelado.

En el resto del capítulo describiremos, en primer lugar, la herramienta propuesta: interacción, aspectos pedagógicos, módulo corrector y aspectos técnicos. Después analizaremos las principales características de algunos trabajos relacionados con el nuestro, desde la óptica de la enseñanza de la POO y también desde la Ingeniería del Software, por ser ambas ramas afines a nuestros propósitos (sobre todo la primera de ellas). A continuación, describiremos la experimentación realizada con el alumnado de Ingeniería del Software I de la Universidad Rey Juan Carlos, para terminar citando las conclusiones alcanzadas y el trabajo futuro.

2. TextOO

TextOO es una aplicación diseñada para facilitar el aprendizaje de los conceptos básicos de la Programación Orientada a Objetos a través del modelado. Parte de un enunciado preparado por el profesorado, que los estudiantes tendrán que modelar para representar un problema.

Tradicionalmente, estos enunciados, expresados en lenguaje natural, son bastante informales, ocultando en ocasiones las relaciones existentes entre los elementos clave

y dificultando las tareas de diseño. Nosotros pretendemos abordar este problema para que el alumnado identifique los conceptos básicos presentes y, mediante la comparación con la propuesta del profesorado, descubra estas relaciones implícitas.

2.1. Interacción con TextOO

La herramienta cuenta con dos perfiles distintos. Desde el punto de vista del docente, ofrece la posibilidad de elaborar los enunciados y de resolverlos a través de diferentes diagramas, expresados en notación UML. En concreto, los diagramas escogidos son el de clases, el de objetos y el de secuencia, porque permiten representar los conceptos básicos de la POO, además de reflejar bien la diferencia entre la parte estática y la dinámica de un problema. Por otra parte, en este perfil es posible añadir comentarios para justificar o explicar las decisiones de diseño tomadas.

Desde el punto de vista del alumnado, la aplicación proporciona el enunciado de un problema para que el estudiante lo modele. Para ello, al igual que hizo el docente cuando preparó el problema, es necesario seleccionar, en el propio texto, las palabras consideradas relevantes, que se traducirán en las representaciones gráficas de los conceptos escogidos a través de diagramas UML.

Tanto en un perfil como en el otro es posible añadir elementos adicionales que no estén presentes en el enunciado. Esto permite dotar de mayor flexibilidad a la aplicación.

La figura 2 muestra una captura de TextOO. Es destacable que la ventana está dividida en dos partes. En la zona superior aparece el enunciado, junto con unas opciones de menú generales. En la zona inferior es posible seleccionar tres pestañas distintas, que se corresponden respectivamente con los diagramas de clases, objetos y secuencia. En el ejemplo mostrado, las palabras inglesas *figures* y *pictures* están destacadas en la primera línea del enunciado. El diagrama de clases contiene, por lo tanto, en la parte superior las clases *Figure* y *Picture*, porque el estudiante (o el docente) las marcó como tales en el enunciado.

Las tres pestañas se corresponden con tres grupos de conceptos presentes en un enunciado típico:

- Conceptos asociados a las clases. Describen la parte estática de un problema.
- Conceptos asociados a los objetos. Describen la parte dinámica de un problema.
- Conceptos asociados a la ejecución. Describen el comportamiento de los objetos ante un ejemplo concreto.

La selección de cada pestaña muestra el diagrama UML correspondiente, que incluye diferentes botones de acción:

- Diagrama de clases: el usuario podrá seleccionar clases, atributos, métodos y relaciones (herencia, asociación o composición), como puede verse en la figura 2.
- Diagrama de objetos: el usuario podrá seleccionar objetos, valores de atributos y relaciones (figura 3).

- Diagrama de secuencia: el usuario podrá seleccionar objetos y mensajes (figura 4).

El modo de crear un elemento gráfico es sencillo: el usuario selecciona un término en el enunciado y pulsa el botón correspondiente (también se puede utilizar el botón derecho del ratón); se abre un cuadro de diálogo para añadir la información necesaria, por ejemplo, la clase donde un método se incorporará. También se pide cierta información relacionada con la sintaxis UML de los métodos y atributos, en su caso. Los términos seleccionados se marcan en el enunciado con un código de colores: clases en rojo, métodos en verde, atributos en azul, objetos en rosa y valores de los atributos en verde claro.

Es importante advertir que las relaciones entre clases y objetos se manejan de manera distinta. Típicamente, no suelen aparecer de forma explícita en los enunciados, por lo que deben crearse de manera independiente a éstos.

La interacción se completa con la posibilidad de realizar diferentes operaciones sobre los elementos gráficos: es posible agruparlos, desplazarlos e incluso borrarlos, actualizando en este último caso de forma automática la información presente en el enunciado.

Por último, TextOO permite generar diagramas UML completos de forma independiente al enunciado, no sólo para las antes mencionadas relaciones, sino para cualquier elemento. Esto permite crear diseños más flexibles, e incluso da la posibilidad de usar la herramienta como si fuera un editor visual.

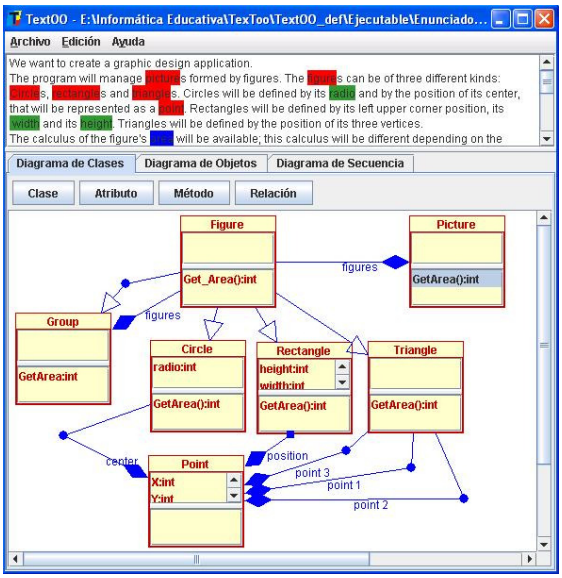


Fig. 2. Captura de la herramienta: Diagrama de clases

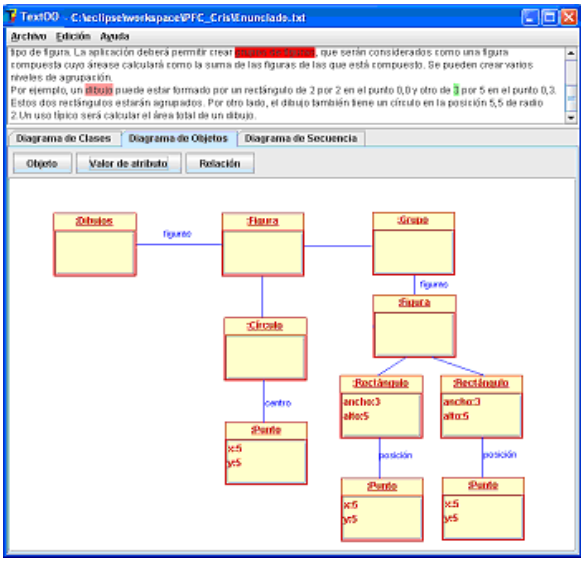


Fig. 3. Captura de la herramienta: Diagrama de objetos

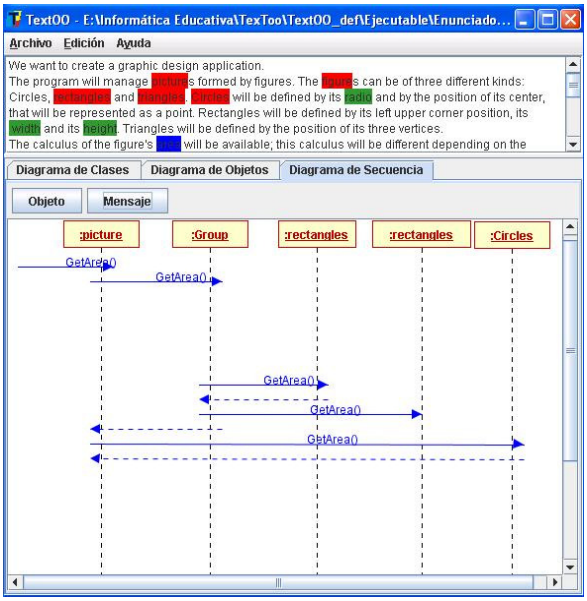


Fig. 4. Captura de la herramienta: Diagrama de secuencia

2.2. Metas pedagógicas

TexTOO es una aplicación pensada para el aprendizaje de los conceptos básicos de la Programación Orientada a Objetos a través del modelado; en menor medida, creemos que también puede ser de cierta utilidad a los estudiantes de Ingeniería del Software.

En general, el alumnado debe conocer (en términos de la Taxonomía de Bloom) estos conceptos; la herramienta reforzará la comprensión y, sobre todo, trabajará el nivel de aplicación, cuando sea preciso resolver problemas sencillos. El nivel de análisis también esta contemplado, ya que el alumnado puede añadir, borrar y modificar elementos propios de un diseño, descomponiéndolo de diferentes maneras y experimentando con alternativas. En problemas más complicados, los estudiantes tendrán que escoger alternativas personales, alcanzando incluso el nivel de síntesis.

Por otra parte, la herramienta será útil para aprender a distinguir la parte estática (clases, atributos y métodos) de la parte dinámica (objetos, estados y mensajes) de un programa orientado a objetos.

2.3. Corrección de los diseños

Los estudiantes pueden consultar en cualquier momento la solución ofrecida por el profesorado. Esta solución está en modo de *sólo lectura*, y pudiera incluir comentarios distribuidos sobre los elementos gráficos de cualquiera de las tres vistas, a través de *hints*, o etiquetas contextuales.

Cuando el alumnado considere que su trabajo ha concluido, puede corregir su propuesta. TextOO incluye un sistema de evaluación rígido que corrige la solución en tres fases distintas:

- Enunciado: Se comprueba si los términos elegidos por el alumnado coinciden con los del profesorado, tanto en posición como en tipo de selección (clase, método, etc.).
- Elementos. Se comprueba si los componentes gráficos definidos por profesor/a y alumno/a de forma independiente del enunciado coinciden.
- Relaciones. Se comprueba si las relaciones definidas por alumno/a y profesor/a coinciden. Las relaciones incluyen aquellas entre clases, entre objetos, y los mensajes en el diagrama de secuencia. Se comprueba el tipo de la relación, el nombre y los elementos relacionados.

TextOO proporciona un informe en forma de tabla, distinguiendo las tres fases antes mencionadas. Para cada una de ellas existen tres posibles resultados: buenas elecciones (si las decisiones de alumno/a y profesor/a coinciden), diferencias (cuando no coinciden), y por último, omisiones (cuando el estudiante olvida alguna decisión tomada por el docente). De manera inicial, la herramienta proporciona un resultado cuantitativo (figura 5). Para cada resultado y fase, es posible acceder a la información detallada correspondiente a cada elemento (figura 6).

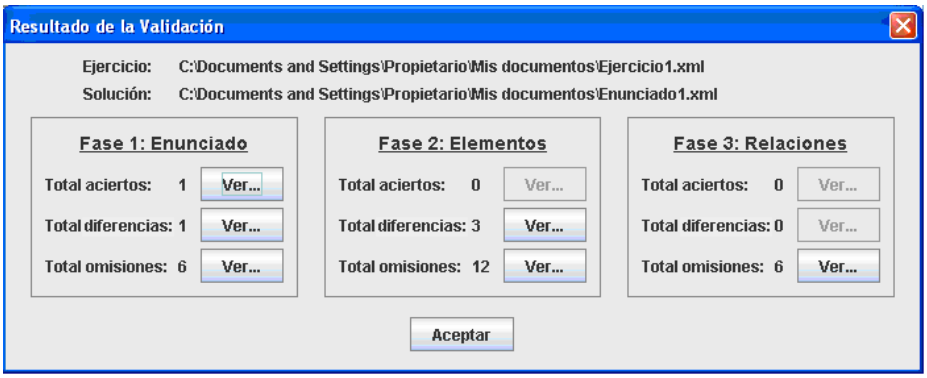


Fig. 5. Captura de la herramienta: Validación global

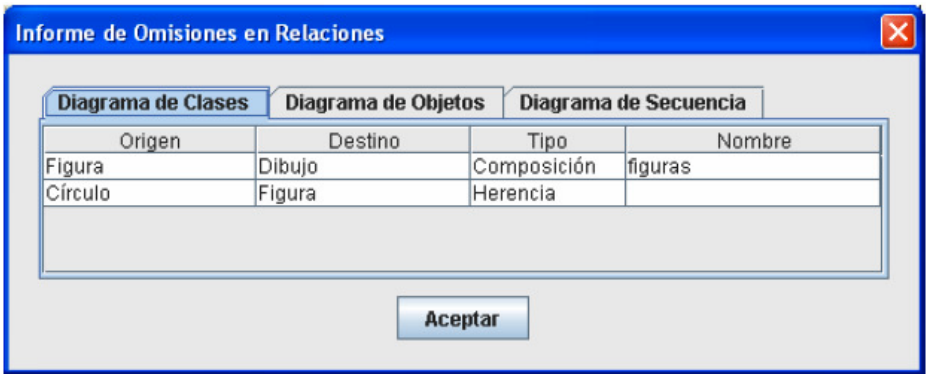


Fig. 6. Captura de la herramienta: Detalle de la validación

2.4. Aspectos técnicos

TextOO ha sido programado en Java. Toda la parte gráfica asociada se ha elaborado utilizando la API JSD (http://vido.escet.urjc.es/JSD). Los ficheros de la aplicación, donde se almacenan los enunciados y sus soluciones, son ficheros XML. Se han utilizado Esquemas XML para estructurar la información.

Hemos publicado TextOO como "software" libre bajo la licencia GPL. Está disponible en: http://vido.escet.urjc.es/textoo. Para acceder a la aplicación hay dos contraseñas: 'p' para los profesores y 'a' para los alumnos.

3 Trabajos relacionados

Hemos analizado diferentes entornos de programación educativos; la mayoría de ellos nos han inspirado en mayor o menor medida en la confección de nuestra herramienta. También nos hemos detenido en algunas herramientas procedentes del mundo de la ingeniería del software, por la relación existente entre ellas y nuestro enfoque.

Los entornos educativos analizados poseen características interesantes. Podemos ver un resumen de estas características en la figura 7. La programación visual orientada al diseño, que aparece en *BlueJ* y en *FUJABA Life*, es una de ellas. TextOO también incluye diagramas de diseño, y aunque no genera código asociado con estos diagramas, ofrece algunas facilidades para la implementación relacionadas con la sintaxis de los atributos y de los métodos.

Características / Herramientas	*Programación visual*	*Visualización en ejecución de código*	*Ayuda al alumno*	*División en niveles del aprendizaje*	*Tratamiento del error*	*Analisis textual*
BlueJ	✘					
FUJABA life	✘					
Jeliot 2003		✘				
jGRASP		✘				
Josh on Line			✘			
Teach Yourself Java			✘			
Profesor J				✘		
DeepTest					✘	✘
Visual Paradigm						✘

Fig. 7. Herramientas analizadas y características destacadas

También es frecuente la visualización de la ejecución del código (*Jeliot 2003* y *JGrasp*). TextOO es una herramienta orientada al modelado, basada esencialmente en la descripción de los problemas (enunciados), y no en las soluciones (código). Aún así, el diseño de la herramienta favorece la visualización de diagramas relacionados con la ejecución de los programas.

Las herramientas profesionales asociadas a la ingeniería del software no son especialmente fáciles de usar. Generalmente, incluyen una relación entre el diseño y la implementación, asociando los cambios que se producen en ambos sentidos. Sin embargo, suelen adolecer de facilidades que favorezcan el aprendizaje. Por ejemplo, *Visual-Paradigm* dispone de un análisis de texto que permite la interacción directa con los enunciados. Sin embargo, no considera las relaciones entre las distintas partes de los mismos. Otras herramientas de éxito, como *IBM Rational Rose* o *System Architect* carecen también de esta componente educativa.

Algunas herramientas educativas han sido desarrolladas para enseñar al alumnado cierta metodología de construcción de programas. Destacamos *ECoDe*, una aplicación que incluye el uso de tarjetas CRC en la etapa de diseño.

Nuestra herramienta, TextOO, incluye algunas características que son relevantes en otras aplicaciones educativas. Por ejemplo, la distinción entre diferentes niveles que permiten clasificar el grado de aprendizaje alcanzado aparece en *ProfessorJ*. Tal y como explicamos arriba, TextOO comparte esta idea al estar inspirada en la Taxonomía de Bloom, aunque en nuestro caso esta división no es explícita.

Mencionaremos también el hecho de integrar las herramientas en tutoriales (*Josh On Line*, *Teach Yourself Java*). La última incluye también explicaciones acerca del código. Este enfoque es complementario al nuestro: Estas aplicaciones comienzan explicando la teoría y utilizan las aplicaciones interactivas para los ejemplos, en lugar de comenzar directamente con los problemas.

Por último, destacaremos que los errores que comete el alumnado forman parte de su proceso de aprendizaje, siendo fundamental sacar partido de ellos. Algunas herramientas han trabajado este aspecto, sustituyendo los complejos mensajes de error que los compiladores ofrecen por otros más pedagógicos, u ofreciendo pistas que lleven a los estudiantes a una solución adecuada. En esta línea, TextOO permite al alumnado comprobar sus avances mediante la comparación de su solución con la propuesta por el profesorado. Éste último, además, puede incluir comentarios que justifiquen sus decisiones de diseño para que los estudiantes los examinen en el momento de la corrección.

3. Experimentación

Hemos realizado una evaluación preliminar de la herramienta en la asignatura de cuarto curso "Ingeniería del Software I", incluida en el plan de estudios de la titulación "Ingeniería Informática" de la Universidad Rey Juan Carlos.

Los estudiantes fueron divididos en dos grupos, uno de ellos experimental y el otro de control. En primer lugar, ambos grupos realizaron un ejercicio inicial (figura 8) sobre conceptos básicos de Ingeniería del Software, con el fin de comprobar que los estudiantes fueron asignados a los grupos de manera efectivamente aleatoria. Esta duda es razonable, puesto que el grupo de control correspondió a alumnado del grupo de horario de mañana, mientras que el grupo experimental lo hizo al de horario de tarde. En general, cabe pensar que el perfil de ambos turnos pueda variar, porque los alumnos de tarde suelen tener una mayor edad, en un mayor número de casos están incorporados a la vida laboral y el absentismo es, con frecuencia, mayor.

Los resultados del ejercicio (resumen en tabla 4) fueron sometidos, en primer lugar, a las hipótesis de normalidad correspondientes; una vez superadas éstas se utilizó el test t para muestras independientes (tabla 5), concluyendo que ambas poblaciones coinciden, por lo que la aleatoriedad de la elección está garantizada.

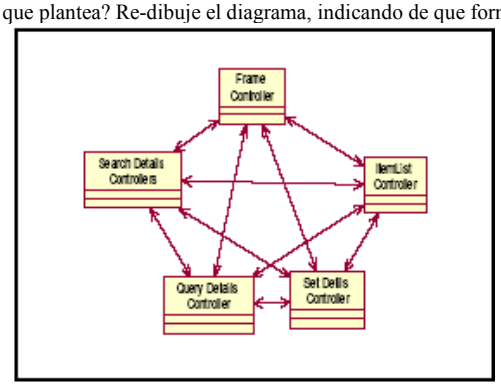
TEST INICIAL (PUNTUADO DE 0 A 10)

Conteste Verdadero o Falso a estas cuestiones:

1. Un caso de uso es un conjunto de actividades que modela el comportamiento de un objeto
2. Los actores modelan el entorno del sistema
3. Los casos de uso modelan como interactúan los objetos del sistema
4. Cuando un caso de uso tiene secuencias que se ejecutan excepcionalmente, se utiliza la relación de extensión
5. Un caso de uso puede tener varios caminos de ejecución que se modelan por separado
6. Un caso de uso puede ser realizado por los objetos del sistema o por actores humanos.

Ejercicio

Dado el siguiente diagrama donde cada flecha representa la comunicación entre las clases. Responder: ¿Cuál es el inconveniente que plantea? Re-dibuje el diagrama, indicando de que forma quedaría.



Pregunta: En el siguiente fragmento de caso de uso, indique si es correcto o incorrecto. En caso de ser incorrecto señale cual es el error, indicando cual/es es la línea/s en que se produce/n y por que.

Caso1:

1. El usuario ingresa su código de identificación personal
2. El sistema valida el código del usuario
3. Si el código es inválido se muestra un mensaje de error
4. Si el código es válido el sistema muestra el menú principal

Caso 2:

1. El usuario selecciona la opción de 'Transferencia' en el menú.
2. El objeto Menú envía el mensaje Validarautorización() al objeto cliente para verificar si está habilitado
3. El sistema muestra la pantalla para seleccionar cuenta origen
4. El usuario selecciona cuenta origen y destino, e importe a transferir
5. El usuario confirma la transferencia

Fig. 8. Ejercicio inicial de la experimentación

Tabla 4. Resumen descriptivo de los resultados del ejercicio inicial.

	N	Minimum	Maximum	Mean	Std. Deviation
Experimental	8	,00	7,00	4,0000	2,32993
Control	29	1,00	6,00	3,5000	1,28869

Tabla 5. Test t para los datos del ejercicio inicial.

Levene's Test for Equality of Variances		t-test for Equality of Means						
F	Sig.	t	df	Sig. (2-tailed)	Mean Difference	Std. Error Difference	95% Confidence Interval of the Difference	
3,879	**,057**	~~,806~~	~~35~~	~~,426~~	~~,50000~~	~~,62051~~	~~-,7597~~	~~1,7597~~
		,583	8,217	**,576**	,50000	,85781	-1,4690	2,4690

En esta prueba estadística, en primer lugar se comprueba si existe igualdad de varianzas, a través del test de Levene. En nuestro caso no es así, por lo que los datos interesantes son los de la segunda fila. En ellos podemos comprobar, a través del valor destacado, que las dos muestras pertenecen a la misma población, lo que garantiza la hipótesis perseguida.

A continuación, a los dos grupos se les planteó un ejercicio de modelado orientado a objetos clásico, la resolución de un problema geométrico (figura 9). El objetivo era modelarlo utilizando diagramas de clases, de objetos y de secuencia. El alumnado del grupo de control lo resolvió o bien utilizando la herramienta Rational Rose (13 personas) o bien sin ninguna aplicación informática (15 personas). En el grupo experimental se utilizó TextOO (7 personas).

La figura 10 muestra los resultados obtenidos. Los valores oscilan entre el 0 (mínimo) y el 3 (máximo). El valor medio obtenido para el grupo que no utilizó ninguna herramienta fue de 2.0; el grupo que utilizó Rational, en promedio, consiguió una puntuación de 1.77; finalmente, el grupo que usó TextOO alcanzó un valor medio de 2.21.

A continuación se aplicó de nuevo el test t para muestras independientes, de tres formas diferentes: Uso de TextOO frente a otros usos, Uso de TextOO frente a uso de Rational Rose y, por último, uso de TextOO frente a uso de ninguna herramienta. Ninguno de los resultados obtenidos fue estadísticamente significativo.

Se desea crear una aplicación de diseño gráfico.
La aplicación gestionará dibujos formados por figuras. La figuras pueden ser de diferentes tipos: círculos, rectángulos y triángulos. Los círculos estarán caracterizados por su radio y posición del centro, que será representado por un punto. Los rectángulos estarán caracterizados por la posición de la esquina superior izquierda y el ancho y alto. Los triángulos estarán caracterizados por la posición de los tres vértices.
La aplicación deberá permitir el cálculo del área de las figuras, que será determinado de forma diferente dependiendo del tipo de figura. La aplicación deberá permitir crear grupos de figuras, que serán considerados como una figura compuesta cuya área se calculará como la suma de las figuras de las que está compuesto. Se pueden crear varios niveles de agrupación.
Por ejemplo, un dibujo puede estar formado por un rectángulo de 2 por 2 en el punto 0,0 y otro de 3 por 5 en el punto 0,3. Estos dos rectángulos estarán agrupados. Por otro lado, el dibujo también tiene un círculo en la posición 5,5 de radio 2.
Un uso típico será calcular el área total de un dibujo

Fig. 9. Enunciado propuesto

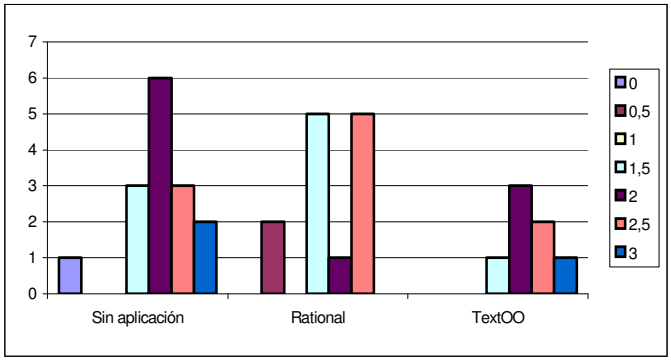


Fig. 10. Resultados del ejercicio

Por último, los estudiantes que utilizaron TextOO rellenaron un cuestionario acerca de su utilidad y de su facilidad de uso. La tabla 6 incluye las preguntas y el resumen de sus respuestas, expresado a través de la media y de la desviación típica. La escala utilizada incluía valores entre 1 (totalmente en desacuerdo) y 5 (totalmente de acuerdo). La figura 11 incluye el detalle de las respuestas.

Tabla 6. Cuestionario y resumen de los resultados

	Pregunta	Media	Desviación típica
P1	La herramienta es fácil de usar	3.71	0.49
P2	Considero útil el hecho de disponer del enunciado mientras realizo el modelado	4.29	0.76
P3	Me parece útil el hecho de poder interactuar con el enunciado	3.71	0.95
P4	El corrector me parece útil	4.00	0.82
P5	En general, creo que la herramienta puede contribuir al mejor aprendizaje de los conceptos básicos de modelado	3.86	0.69

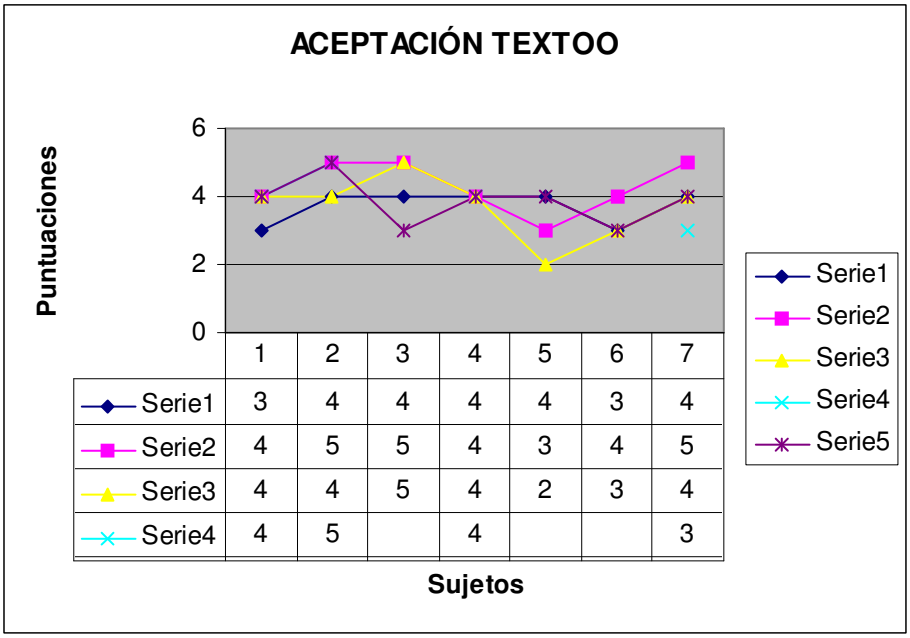


Fig. 11. Resultados detallados del cuestionario

En general, la reacción del alumnado fue positiva, lo cual es muy destacable en este tipo de experimentos. En particular, nos gustaría subrayar los resultados de la segunda pregunta, relacionada con el enunciado, ya que constituye uno de los elementos clave en el diseño de nuestra aplicación.

4. Conclusiones y trabajos futuros

En este capítulo hemos planteado una propuesta general para construir aplicaciones destinadas al aprendizaje de la programación orientada a objetos, para pasar a describir TextOO, una aplicación educativa concreta diseñada para ayudar a los estudiantes

en su aprendizaje de los conceptos y relaciones básicos, desde el punto de vista del modelado. La herramienta está enfocada hacia asignaturas de Programación Orientada a Objetos, y por extensión, aunque en un grado menor, hacia asignaturas de Ingeniería del Software.

TextOO se basa en el uso del lenguaje natural utilizado para describir los problemas de un modo inicial, y utiliza diagramas de clases, de objetos y de secuencia. Además, incluye un sencillo corrector, basado en la comparación directa de las soluciones propuestas por el alumnado y el profesorado.

En función de la naturaleza y dificultad del problema planteado, el alumnado trabajará los niveles de comprensión, aplicación, análisis o síntesis de la taxonomía de Bloom.

Hemos evaluado TextOO con un grupo de estudiantes de Ingeniería del Software, obteniendo resultados prometedores, pero no estadísticamente significativos. Por otra parte, los estudiantes rellenaron un cuestionario acerca de la usabilidad y utilidad de la herramienta, concluyendo de manera muy satisfactoria.

En cualquier caso, estos resultados deben ser analizados con precaución, porque el experimento no se realizó con el grupo más apropiado. Pensamos que la herramienta puede ser de cierta utilidad para los estudiantes de Ingeniería del Software, pero las ideas generales y la base pedagógica de la herramienta la orientan claramente hacia estudiantes primerizos de programación orientada a objetos. Tenemos la intención de realizar evaluaciones con esta clase de alumnos, tan pronto como el calendario escolar nos sea propicio.

Estamos trabajando en diferentes mejoras. En primer lugar, algunos profesores de nuestra universidad tienen la intención de utilizar TextOO con diferentes enunciados, para identificar necesidades o carencias de la aplicación, y también para reforzar sus posibles puntos fuertes. En particular, tenemos especial interés en poner a prueba a la aplicación con enunciados grandes.

En segundo lugar, queremos añadir funcionalidades adicionales; sería interesante incluir más elementos relacionados con la POO (polimorfismo), y también generar código asociado a los diagramas.

Por ultimo, nos gustaría flexibilizar el corrector, tarea que puede ser realizada en tres direcciones: teniendo en cuenta la sintaxis y semántica de los enunciados, permitiendo al profesor incluir varias soluciones y ampliando el informe del corrector.

5. Agradecimientos

Me gustaría agradecer a Ángel Velázquez y a Isidoro Hernán sus fundamentales consejos y aportaciones. También a Francisco Gortázar, a Micael Gallego y a Raquel Hijón su apoyo técnico. De forma especial, gracias a Cristina Garrigós, por su esfuerzo y dedicación. Este trabajo está incluido en la tarea investigadora realizada gracias al proyecto TIN2004-07568 del Ministerio de Educación y Ciencia de España.

Referencias

- Bloom, B., and Krathwohl, D. R. *Taxonomy of Educational Objectives: Handbook I, The Cognitive Domain*. Addison-Wesley: New York, 1956.
- Bridgeman, S. et al. PILOT: An interactive tool for learning and grading. In *Proc. Thirty-first SIGCSE Technical Symp. on Computer Science Education (SIGCSE 2000)*, ACM Press, New York, 2000, 139-143.
- Cross II, J.H., et al. Using the debugger as an integral part of teaching CS1. In *Proc. 32th ASEE/IEEE Frontiers in Education Conf. (FIE 2002)*, 2002.
- Diehl, S., and Bieg, C. A new approach for implementing stand-alone and Web-based interpreters for Java. In *Proc. 2nd International Conf. on Principles and Practice of Programming in Java (PPPJ 2003)*, ACM Press, 2003, 31-34.
- Flatt, M., and Gray, K.E. ProfessorJ: A gradual introduction to Java through language level. In *Proc. 18th Annual ACM SIGPLAN Conf. on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, ACM Press.
- Gray, K.A., Guzdial, M., and Rugaber, S. Extending CRC cards into a complete design process. In Proc. 8th *Annual Conf. on Innovation and Technology in Computer Science Education (ITiCSE 2003)*, ACM Press, 2003, 226.
- Hernán-Losada, I., Lázaro-Carrascosa, C., and Velázquez-Iturbide, J.Á. Una aplicación educativa basada en la jerarquía de Bloom para el aprendizaje de la herencia de POO. In *Proc. VII Simposio Internacional de Informática Educativa (SIIE 2005)*, 2005.
- Hernán-Losada, I., Lázaro-Carrascosa, C., and Velázquez-Iturbide, J.Á. On the use of Bloom's taxonomy as a basis to design educational software on programming. In *Proc. World Conf. on Engineering and Technology Education (WCETE 2004)*, 2004, 351-355.
- Hoggarth, G., and Lockyer, M. An automated student diagram assessment system. In Proc. 3rd Annual *Conf. on Innovation and Technology in Computer Science Education (ITiCSE'98)*, ACM Press, 1998, 122-124.
- Jackson, D., and Usher, M. Grading student programs using ASSYST. In *Proc. Twenty-Eight SIGCSE Technical Symp. on Computer Science Education (SIGCSE'97)*, ACM Press, New York, 1997, 335-339.
- Kostadinov, R., and Kumar, A. A tutor for learning encapsulation in C++ classes. In *Proc. World Conf. on Educational Multimedia, Hypermedia and Telecommunications (ED-MEDIA 2003)*, AACE Press.
- Moreno, A. et al. Visualizing programs with Jeliot 3. In *Proc. International Working Conf. on Advanced Visual Interfaces (AVI 2004)*, ACM Press, 2004, 273-276.
- Nickel et al. The FUJABA environment. In *Proc. 22nd International Conf. on Software Engineering*, ACM Press, 2000, 742-745.
- *Teach Yourself Java* web site: http://www.duke.edu/~trc7/cps/.
- Thomas, P., Waugh, K., and Smith, N. Experiments in the automatic marking of E-R diagrams. In *Proc. 10th Annual Conf. on Innovation and Technology in Computer Science Education (ITiCSE 2005)*, ACM Press, 2005, 158-162.
- Van Haaster, K., and Hagan, D. Teaching and learning with BlueJ: An evaluation of a pedagogical tool. In *Information Science + Information Technology Education Joint Conf.*, Rockhampton, QLD, Australia, 2004.
- Visual Paradigm Co. *Visual Paradigm for the UML 2.0 User's Guide*