Universidad Rey Juan Carlos

Escuela Técnica Superior de Ingeniería Informática

Departamento de Lenguajes y Sistemas Informáticos I

Diseño de software educativo para la enseñanza de la programación orientada a objetos basado en la taxonomía de Bloom

TESIS DOCTORAL

Isidoro Hernán Losada

2012

Universidad Rey Juan Carlos

Escuela Técnica Superior de Ingeniería Informática

Departamento de Lenguajes y Sistemas Informáticos I

Diseño de software educativo para la enseñanza de la programación orientada a objetos basado en la taxonomía de Bloom

TESIS DOCTORAL

Autor: Isidoro Hernán Losada

Director: J. Ángel Velázquez Iturbide

Móstoles, 2012

Dr. D. Jesús Ángel Velázquez Iturbide, Catedrático de Universidad del departamento de Lenguajes y Sistemas Informáticos I de la Universidad Rey Juan Carlos y director de la Tesis Doctoral "*Diseño de software educativo para la enseñanza de la programación orientada a objetos basado en la taxonomía de Bloom*" realizada por el doctorando D. Isidoro Hernán Losada

HACE CONSTAR

que esta Tesis Doctoral reúne los requisitos necesarios para su defensa y aprobación.

En Móstoles a 18 de mayo de 2012

Dr. D. Jesús Ángel Velázquez Iturbide

Agradecimientos

A mis hijos Saúl, Eloy y Zoe. Me aportan una felicidad y unas sensaciones especiales difíciles de describir. Sin duda alguna "mi mejor obra".

A mi mujer Mayte González de Lena Alonso. Sin su amistad, su amor, su cariño y la seguridad que me aporta esto no habría sido posible.

A mis padres, Isidoro y María del Pilar porque soy parte de ellos y todo lo mío es gracias a ellos. A mis hermanos María del Pilar, Luis Patricio, Fátima y Oscar porque el tiempo compartido con vosotros me ha servido para estar donde estoy.

A mi director de tesis, J. Ángel Velázquez, por su paciencia, sus ánimos y sus comentarios, los cuales han enriquecido esta tesis.

A mis sobrinos Carlos, Lucía, Luis, Alicia e Irene por todos los buenos ratos y los mejores que vendrán. A Antonio, Hortensia, Carlos, Goyo, Pili, Toni, Ana, Vero por aceptarme en vuestras familias. A mis abuelos, tíos, primos y el resto de familia, que sería imposible nombrar sin dejarme a alguno… Os quiero.

A mi 'otra' familia: Miguel y Marga, Alberto y Natalia, Luis A., Sergio si quieres algún día quedamos ;-) y Mayte Acicolla gracias por vuestra amistad.

A Almudena, Maxi, Paloma, Tuni, José Velez por estar aquí. Os debo unos cuantos cafés, por ser personas especiales dentro de la URJC. A Estefanía, sin tu inestimable ayuda y generosidad no habría sido posible.

En general a todos que deberían haber tenido un sitio en esta página que, por suerte o por desgracia, no los he puesto. A todos los que han influido en mi vida, para bien o para mal, porque gracias a vosotros estoy defendiendo la tesis. Sobre todo a la gente que guardo un recuerdo especialmente bueno de ellos.

A todos mis alumnos, especialmente a los que han desarrollado sus proyectos fin de carrera conmigo, pues han sido una fuente de inspiración y una gran ayuda en la implementación de las herramientas mostradas en esta tesis. Sin ellos, parte de esta tesis estaría coja.

A mis compañeros del Centro Menéndez Pelayo (CSIC), en el que trabajé unos cuatro años y en especial a Javi, quien tanto me ayudo a superar mi timidez. A los compañeros del Instituto de Automática Industrial (CSIC), por todos los buenos ratos que hemos pasado juntos.

A mis compañeros de la Universidad Rey Juan Carlos (URJC) a Carlos Lázaro, por ser compañero de fatigas en los inicios de este trabajo. Paco Nava, Jaime, Paco D., Alberto F., Ascen, Raquel, Manolo, Belén, José Mari y resto de colegas de departamento y de profesión. Incluyo aquí a todo el equipo de futbol, ese deporte que me sirve para relajarme y para desconectar.

Y finalmente, de nuevo, a mis hijos Saúl, Eloy y Zoe porque son todo, el principio y el fin, las mañanas y las noches, la claridad, la ingenuidad y la alegría que aportan a mi vida.

Índice general

Índice de figuras

Índice de tablas

RESUMEN

En esta tesis se presenta una nueva metodología para el diseño de herramientas educativas para la enseñanza de la programación orientada a objetos (POO) basadas en la taxonomía de Bloom, de modo que se logre un desarrollo sistemático del software educativo.

Aunque existen muchas aplicaciones de ayuda a la enseñanza-aprendizaje de la POO, no hemos hallado métodos de diseño de dichas herramientas que contengan un objetivo pedagógico claro. La taxonomía de Bloom es una jerarquía de seis niveles con grado creciente de aprendizaje del alumno. Cada nivel presupone la capacitación del alumno en los niveles precedentes. Según ascendemos por la jerarquía nos encontramos un mayor grado de aprendizaje. Al ser una taxonomía ampliamente aceptada, la usamos como un marco adecuado para fijar la meta educativa que debe poseer cualquier software educativo. La obtención de este objetivo hace que el diseño nos lleve a plantear una serie de características para cada uno de los tres primeros niveles de Bloom. La limitación a los niveles iniciales es debida a dos factores. El primero es la forma de aprendizaje de los discentes. Los alumnos primero captan la información (primer nivel o de conocimiento), la interiorizan (segundo nivel o de comprensión) y después la aplican (tercer nivel o de aplicación) ante nuevas situaciones. Si alguna de estas fases no es convenientemente adquirida, el aprendizaje no es duradero. El segundo factor es el software existente. Pensamos que para los niveles superiores ya hay herramientas educativas (por ejemplo, los entornos de programación) que les ofrecen soporte.

Cualquier propuesta debe probar su validez, así, en el marco de este trabajo de investigación, también hemos desarrollado, utilizando esta metodología, varias herramientas educativas para la enseñanza de distintos conceptos fundamentales y distintivos de la POO con el resto de paradigmas. Este software ha sido puesto a disposición de los alumnos, que han sido los encargados de valorar su eficacia docente. Algunos han participado en distintos experimentos cuidadosamente diseñados, siguiendo una metodología de investigación general. Los resultados han demostrado que las aplicaciones educativas realizadas cumplen con su principal objetivo: ayudar al alumno en un determinado nivel de la taxonomía de Bloom. También se ha recabado la opinión de los alumnos, juzgando estas aplicaciones como útiles para su proceso de aprendizaje.

ABSTRACT

This doctoral thesis presents a new methodology for the design of educational tools aimed at teaching object-oriented programming (OOP). The tools are based on Bloom's taxonomy in order to be able to develop educational software more systematically.

Although many applications exist to assist in the teaching-learning of OOP, we have not found design methods for these tools based on unequivocal, pedagogical objectives. Bloom's taxonomy is a hierarchy of six levels corresponding to an increasing learning performance of the student. Each level forejudges the capacity of the pupil in the previous levels. As we climb up in the hierarchy we find higher learning performance. Bloom's taxonomy is widely known and accepted among the Computer Science education community. For this reason, we used it to specify the educational level of any didactic tool to construct. In this thesis, we focus on the three first levels of Bloom's taxonomy. This limitation is due to two factors, the student's learning method and the software tools currently available. With respect to the first factor, students first perceive information (first level or knowledge level), interiorize it (second level or comprehension level) and later they apply it (third level or application level) in new situations. If any one of these phases is not suitably acquired, it does not lead to long-time learning. With respect to the second factor, we thought that the educational tools (for example, the programming environments) offer support to the superior levels.

We have developed several educational tools for the teaching of fundamental and distinctive concepts of the OOP, based on the assumptions and methodology claimed above. These tools have been used by students, and they were given an estimate on his educational efficacy. Some of them participated in experiments carefully designed according to the principles of general investigation.

Results have shown that the educational tools we have developed comply with their main objective: assisting the student at a given level of Bloom's taxonomy. We have also collected the students' opinions, who judge these tools as effective for their learning.

Capítulo 1 Introducción

En este capítulo se expone el planteamiento y la justificación de esta tesis doctoral; seguidamente se introduce la hipótesis planteada, así como los objetivos derivados directamente de la misma; se describe después el método de investigación seguido para la realización de la evaluación de las herramientas desarrolladas; a continuación se muestran las principales aportaciones de este trabajo; para finalizar, se presenta la organización y estructura de esta memoria.

1.1. Motivación

En la actualidad del panorama universitario, los nuevos planes de estudio definidos en Bolonia establecen de forma implícita que el mejor modelo de enseñanza-aprendizaje es el constructivista-cognitivo [Piaget 52, 54] [Ausubel 63] [Bruner 66]. Este paradigma pone como centro del proceso al alumno. Esta teoría conlleva un cambio en el método instruccional tradicional. Las lecciones magistrales deben ir acompañadas de nuevas técnicas que involucren al alumno para que aprenda de forma activa. Esto produce un aprendizaje duradero, que ha sido aceptado como un elemento básico para el Área de Educación Superior Europea [Bologna 03]. Los nuevos planes se fundamentan en que en una Europa construida sobre la sociedad y la economía basada en el conocimiento, las estrategias para fomentar el aprendizaje a largo plazo son necesarias para afrontar los cambios en competitividad y en el uso de las nuevas tecnologías, y mejorar la cohesión social, la igualdad de oportunidades y la calidad de vida. Desde entonces ha ido creciendo la conciencia de la necesidad de integrar este tipo de aprendizaje en la educación superior.

Este paradigma de la educación necesita otras metodologías que mejoren el papel activo del estudiante, su iniciativa y su pensamiento crítico. El aprendizaje consiste en participar activamente en ese proceso, frente al adquirido por la recepción pasiva de información dada por un experto. Los estudiantes son constructores activos de su propio conocimiento y aprenden a

comprender y analizar diferentes alternativas adquiridas de diversas maneras (a través del profesor, de herramientas pedagógicas, del diálogo con otros compañeros, etc.). El conocimiento no un objeto estático sino que es un ente en continua evolución. Es construido por el alumno a través de su propia experiencia. El aprendizaje significativo se logra cuando el sujeto percibe el tema como importante para sus propios intereses o si satisface alguna necesidad personal o social. Los discentes tienen que asumir su responsabilidad en este proceso y desarrollar sus habilidades y capacidades para guiar su propio conocimiento y su desempeño. Existen muchas formas distintas de dar apoyo a los elementos principales del aprendizaje activo como, por ejemplo, el aprendizaje basado en problemas –*Problem Based Learning* o PBL- o en proyectos, el aprendizaje colaborativo, el aprendizaje cooperativo, etc.. Prince [Prince 04] mostró que no existe una única técnica para conseguir un aprendizaje más eficaz, sino que es mejor combinar las distintas metodologías para garantizar una enseñanza de calidad.

Además de implicar al alumno en su aprendizaje, otra de las tareas del docente es medir el grado de conocimiento adquirido por el discente. La obtención de este indicador puede dar una pista de la bondad de una metodología que implique al alumno. La taxonomía de Bloom [Bloom 56] nos da un marco excepcional para este trabajo. Bloom y su equipo se dieron cuenta que "pueden observarse unos tipos de conducta esencialmente idénticos entre personas de carácter normal y en un determinado nivel de enseñanza (elemental, media y superior)". Esto permite describir una clasificación, denominada taxonomía de los objetivos de la educación o más comúnmente taxonomía de Bloom, que agrupe esos tipos de conducta. Esta jerarquía establece seis niveles con grado creciente de aprendizaje del alumno. Cada nivel presupone la capacitación del alumno en los niveles precedentes.

En este contexto, las TIC (Tecnologías de la Información y de la Comunicación) toman un papel muy importante ya que permiten desarrollar entornos de enseñanza-aprendizaje para dar soporte a las diferentes alternativas que existen para implicar al alumno en su propio proceso de aprendizaje. Se denominan herramientas pedagógicas informáticas o software educativo a toda aplicación creada con la finalidad específica de ser usada como facilitadores del aprendizaje, o dicho de otro modo, todo programa desarrollado para ayudar en los procesos de enseñanza y aprendizaje. Al inicio de esta tesis nos propusimos estudiar las herramientas pedagógicas informáticas desarrolladas para el aprendizaje de la programación de computadoras, lo que nos llevo a observar una serie de carencias que este trabajo trata de paliar. La mayor ausencia encontrada fue una falta de

sistematización en el diseño del software educativo, siendo esta fase del desarrollo del software la más crítica [Ruiz 96], según la ingeniería del software tradicional.

El software educativo se diseña, en general, siguiendo la intuición del docente o del desarrollador (a veces acertadamente, a veces no), detectándose problemas a la hora de su diseño, bien por falta de capacitación docente o de supervisión del técnico, o bien por falta de conocimientos de las metodologías del software por parte del docente. Esto provoca que el programa final no posea la facilidad de uso, la eficacia y calidad necesarias para conseguir el efecto deseado (ayudar en el proceso de enseñanza-aprendizaje). Otro problema añadido es que su eficacia pedagógica no puede ser medida hasta que la aplicación está terminada, lo que puede conllevar un rediseño de la misma y por lo tanto, un nuevo desarrollo.

Por lo tanto, la presente tesis trata de solucionar esos problemas, mezclando dos áreas de distinta naturaleza, por un lado, la pedagogía, con sus teorías del aprendizaje y la clasificación del nivel de aprendizaje de Bloom, con la ingeniería del software, con sus métodos para la producción efectiva y de calidad de aplicaciones informáticas. El interés mayor se centra en aportar una metodología para el diseño de herramientas informáticas de ayuda a la enseñanza en un determinado nivel de Bloom que sirva de guía a los futuros diseñadores de este tipo de software y que minimice al máximo el riesgo de obtener un producto poco eficaz, pedagógicamente hablando, y de poca calidad educativa y tecnológica. En el marco de este trabajo, se aplicará las metodologías al software educativo para la enseñanza de la Programación Orientada a Objetos (POO en adelante).

1.2. Hipótesis y objetivos

Al comienzo del trabajo de investigación que se ha llevado a cabo en esta tesis doctoral, y teniendo en cuenta el planteamiento de la anterior sección, se concibió la siguiente **hipótesis** de trabajo:

"Es factible la especificación de una metodología para el diseño sistemático de aplicaciones de ayuda a enseñanza de la programación orientada a objetos en un nivel especifico de la taxonomía de Bloom, que permita obtener un producto eficaz pedagógicamente hablando"

El **objetivo principal** de este trabajo de investigación, derivado directamente de la hipótesis, es:

"La especificación de una metodología para el diseño sistemático de aplicaciones de ayuda a enseñanza de la programación orientada a objetos en un nivel específico de la taxonomía de Bloom y la demostración de su aplicación y de su eficacia pedagógica".

Para la consecución de este objetivo se han planteado los siguientes subobjetivos u objetivos parciales:

Primer objetivo parcial:

"Estudio de las distintas herramientas de ayuda a la enseñanza de la programación, centrándonos en la programación orientada a objetos y relacionándolas con el nivel de Bloom que soportan, mostrando sus principales características y carencias."

Segundo objetivo parcial:

"Especificación de la metodología para el diseño sistemático de aplicaciones de ayuda a enseñanza de la programación orientada a objetos en un nivel específico de la taxonomía de Bloom, identificando los diferentes aspectos a considerar en el desarrollo de la aplicación en función del nivel de Bloom al que ofrezca soporte."

Tercer objetivo parcial:

"Desarrollar herramientas pedagógicas para la ayuda a la enseñanza y aprendizaje de la programación orientada a objetos basadas en esta metodología"

Cuarto objetivo parcial:

"Evaluación cuantitativa y cualitativa, estudio de resultados y posterior validación de la eficacia pedagógica de las herramientas desarrolladas usando esta metodología mediante evaluaciones de su uso por parte de los estudiantes de diversos cursos de la asignatura de programación orientada a objetos."

1.3. Metodología de investigación

Para la evaluación de la bondad de la metodología hemos decidido hacer dos tipos de evaluaciones: la evaluación cualitativa y la cuantitativa.

1.3.1. Evaluación cuantitativa

Para evaluar de forma cuantitativa alguna de las herramientas desarrolladas se ha seguido un proceso general de investigación (véase Figura 1) consistente en los siguientes pasos [Aliaga 00]:

1. Planteamiento del problema.
2. Formulación de hipótesis.
3. Identificación de variables.
4. Población y muestra.
5. Obtención de datos.
6. Obtención de resultados.
7. Análisis de datos.
8. Conclusiones.

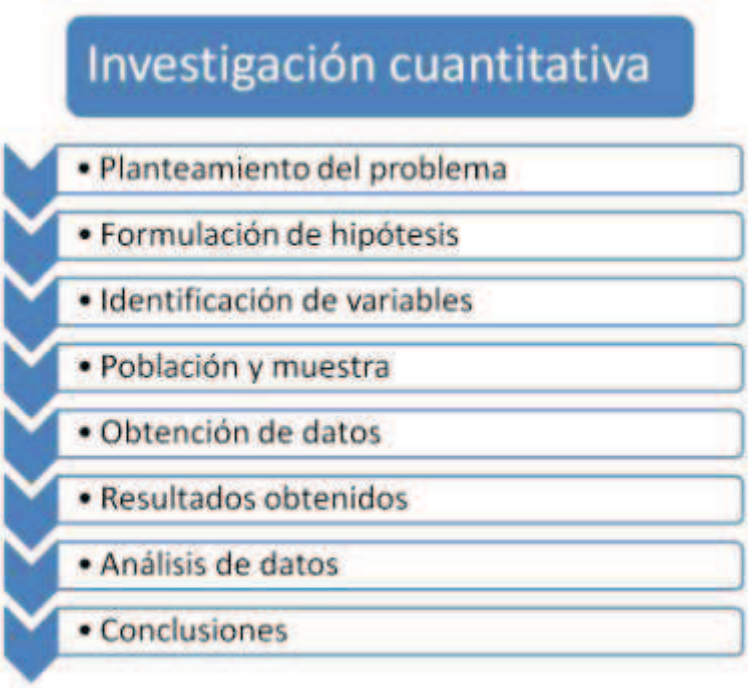


Figura 1. Metodología de investigación cuantitativa

Planteamiento del problema: El problema ha de formularse en términos directos, claros y sin ambigüedad, y el mejor método es plantearlo como un interrogante, en forma de pregunta. Esto nos permitirá distinguir el propósito de la investigación (algo de carácter más global y genérico) del problema de la investigación, que ha de ser más específico y ayudarnos a centrar la investigación en términos más concretos. La formulación del problema ha de hacerse de tal modo que su verificación empírica sea factible.

En nuestro caso concreto el problema planteado en todas las investigaciones realizadas es si el uso de la herramienta es eficaz pedagógicamente hablando. Dicho de otra forma, se pretende observar si existe una diferencia significativa entre los alumnos que aprenden los conceptos en estudio utilizando la herramienta y los discentes que no la usan.

Formulación de hipótesis: Una hipótesis es una afirmación demostrable sobre una relación potencial entre dos o más variables. En general ha de establecerse como una afirmación, de tal modo que podamos someter a prueba si se ajusta a los datos o no.

Identificación de variables: En este apartado se detallan las variables descriptivas de lo que se pretende medir. En nuestro caso se trata de medir el impacto del uso de una herramienta informática. Por lo que una de las variables será la utilización de dicho software (variable dependiente) y la otra será la nota

obtenida por los alumnos (variable independiente). En el capítulo de evaluación de las herramientas se describen estas variables con detalle.

Población y muestra: Población se define como la totalidad de individuos susceptibles de realizar el experimento. Como en la práctica es complicado realizarlo sobre la totalidad se suele restringir el experimento a un grupo reducido de individuos denominado muestra. Para que el estudio en una muestra sea generalizable hemos de procurar que dicha muestra sea representativa de la población a la que pretende representar. El tamaño de la muestra ha de ser suficiente para poder recoger la diversidad de los casos que se den en la población. Una muestra es representativa de una población cuando sus estadísticas son semejantes. Esto solo puede ser comprobado a posteriori.

Obtención de los datos: El proceso de medición es la conversión del mundo real al mundo matemático. En él, se cuantifica de forma apropiada los fenómenos. Si se realiza de forma incorrecta, el proceso total carece de validez.

Obtención de resultados: Una vez recopilada la información, ésta ha de agruparse en un conjunto elementos manejables y comprensibles, básicamente a una agrupación en categorías globales o a través de representaciones gráficas o mediante aplicación de los procedimientos de estadística descriptiva.

Análisis de datos: El objetivo de la mayoría de investigaciones consiste en someter a prueba nuestras hipótesis y el análisis de datos nos proporciona una poderosa herramienta matemática que nos permite llevar a cabo esta labor. Basándonos en la estadística, debemos mostrar si los datos analizados corroboran las hipótesis o, por el contrario, la contradicen.

Conclusiones: En este paso se debe poner en relieve la relación entre el trabajo empírico con las hipótesis previas y resumir los principales hallazgos, haciendo referencia a sus implicaciones teóricas y prácticas.

1.3.2. Evaluación cualitativa

En cuanto a la evaluación cualitativa, se pretende conocer la satisfacción del uso de la herramienta. Existen diversas técnicas de evaluación y pueden ser clasificadas en dos categorías [Gediga 99]: las técnicas de evaluación descriptiva y las técnicas de evaluación predictiva:

Técnicas de evaluación descriptiva. Son usadas para representar el estado de una manera objetiva, confiable y válida. Estas técnicas están basadas en el usuario y pueden ser subdivididas en:

- Técnicas de evaluación basada en la conducta. Consiste en registrar de alguna forma la conducta del usuario mientras está trabajando con un sistema. Estos procedimientos pueden incluir técnicas de observación y el protocolo "pensando en voz alta" (*thinking-aloud*).
- Técnicas de evaluación basada en la opinión. Se fundamenta en la extracción de la opinión del usuario. En este caso se suelen usar (de forma no excluyente) entrevistas, encuestas y cuestionarios.
- Pruebas de usabilidad. Son una combinación de medidas basadas en la opinión y la conducta de los usuarios, normalmente seleccionadas por un experto.

Las técnicas descriptivas necesitan que el sistema este desarrollado (o al menos tener un prototipo) y un usuario. Los datos recogidos por estas técnicas requieren ser interpretados por uno o más expertos.

Técnicas de evaluación predictiva. Son métodos que permiten obtener información que sirve para hacer recomendaciones para un futuro desarrollo de software y para la prevención de errores. Estas técnicas están basadas en el conocimiento de expertos en el tema a diseñar. En ocasiones a los usuarios se les permite participar para obtener más información y distintos puntos de vista. En general los datos son obtenidos por las simulaciones que realizan los especialistas. El beneficio principal de las técnicas predictivas es que permiten la evaluación en la etapa de diseño, antes de que se produzca el desarrollo del sistema. Es importante hacer notar que las predicciones hechas por modelos teóricos están basadas en hipótesis, no en datos reales, por lo que puede ser que se obtengan falsas alarmas o hipotéticos éxitos.

En nuestro caso, se ha utilizado una técnica de evaluación predictiva, en concreto, la basada en la opinión del usuario. Para la recogida de datos se ha usado un cuestionario. Se ha solicitado a los alumnos usuarios que lo rellenen de forma anónima. Este cuestionario consta de varias afirmaciones o preguntas, a las que deben responder con el grado de acuerdo/desacuerdo. Para cuantificar estos grados se midieron utilizando una escala de Likert.

1.4. Aportaciones principales

Las mayores contribuciones de esta tesis son el resultado de todos estos años de investigación sobre la sistematización del diseño de las aplicaciones pedagógicas. En este camino hemos conseguido las siguientes aportaciones:

- Un completo estudio de las herramientas pedagógicas de ayuda a la enseñanza de la programación orientada a objetos

- Se ha desarrollado unas metodologías de diseño de herramientas útiles y pedagógicamente eficaces basadas en la taxonomía de Bloom para ayudar a los discentes en su tarea de ir incrementado su nivel de experiencia o maestría en la programación orientada a objetos.

- Se han desarrollado varias aplicaciones siguiendo esta metodología, obteniéndose muy buena acogida por parte de los principales usuarios: los alumnos.

- Se ha evaluado la eficacia pedagógica, realizando diversos experimentos diseñados cuidadosamente, de varias de las herramientas implementadas, mostrando sus bondades en este terreno.

La difusión de las aportaciones y resultados obtenidos se han materializado en las siguientes publicaciones, ordenadas por fecha de publicación:

1. Isidoro Hernán Losada, Carlos A. Lázaro Carrascosa, J. Ángel Velázquez Iturbide: "Hacia el diseño de herramientas educativas de programación basadas en la taxonomía de Bloom". 5º Simposio Internacional en Informatica Educativa (SIIE), Braga ,Portugal 2003. Relacionada con los capítulos 3 y 4.

2. Isidoro Hernán Losada, Carlos A. Lázaro Carrascosa, J. Ángel Velázquez Iturbide: "On the use of Bloom's taxonomy as a basis to design educational software on programming", World Conference on Engineering and Technology Education WCETE2004 Guarujá / Santos, Brasil, pp: 351-355. 2004. Relacionada con los capítulos 2 y 3.

3. Isidoro Hernán Losada, Carlos A. Lázaro Carrascosa: "Un prototipo de herramienta educativa basada en la taxonomía de Bloom", 6º Simposio Internacional de Informática Educativa (SIIE), Cáceres, España 2004. Relacionada con el capítulo 4.

4. Isidoro Hernán Losada, Carlos Alfredo Lázaro Carrascosa, J. Ángel Velázquez Iturbide: "Una Aplicación Educativa Basada en la Jerarquía de

Bloom para el Aprendizaje de la Herencia de POO", 7º Simposio Internacional de Informática Educativa (SIIE), Leiria, Portugal, 2005. Relacionada con los capítulos 3, 4 y 5.

5. Isidoro Hernán Losada, J. Ángel Velázquez Iturbide, Carlos Alfredo Lázaro Carrascosa: "Programming Learning Tools Based on Bloom's Taxonomy: Proposal and Accomplishments", 8º Simposio Internacional de Informática Educativa (SIIE) León, España 2006. Relacionada con los capítulos 2 y 3.

6. Isidoro Hernán Losada, J. Ángel Velázquez Iturbide, Carlos A. Lázaro Carrascosa: "Dos Herramientas Educativas para el Aprendizaje de Programación: Generación de Comentarios y Creación de Objetos", 7º Congreso Internacional de Interacción Persona-Ordenador. Puertollano, España 2006. Relacionada con los capítulos 3 y 4.

7. Isidoro Hernán Losada, J. Ángel Velázquez Iturbide: "Hacia el nivel de aplicación en la taxonomía de Bloom: Generación de ejercicios semiabiertos sobre POO" VIII Simposio Nacional De Tecnologías De La Información Y Las Comunicaciones En La Educación (Sintice 2007) Zaragoza, España 2007. Relacionada con los capítulos 3 y 4.

8. Isidoro Hernán Losada, Carlos Lázaro Carrascosa, Ángel Velázquez Iturbide: "An educative application based on Bloom's taxonomy for the learning of inheritance in oriented-object programming". En Computers and Education: Towards Educational Change and Innovation, António Mendes, Isabel Pereira y Rogério Costa (eds.), págs. . **Springer**. Berlín, 2007. Relacionada con los capítulos 4 y 5.

9. Isidoro Hernán Losada, J. Ángel Velázquez Iturbide y Carlos A. Lázaro Carrascosa: "Diseño y evaluación de una herramienta para el aprendizaje de la creación de objetos en Java". En Actas del IX Simpósio Internacional de Informática Educativa SIIE, Escola Superior de Educação do Instituto Politécnico do Porto, 2007, págs. 91-96. Relacionada con los caps. 4 y 5.

10. Isidoro Hernán Losada, Cristóbal Pareja Flores, J. Ángel Velázquez Iturbide: "Testing-Based Automatic Grading: A Proposal from Bloom's Taxonomy", 8th IEEE International Conference on Advanced Learning

Technologies (**ICALT 2008**), Santander, España, 2008. Relacionada con el capítulo 2.

11. Isidoro Hernán Losada, Carlos A. Lázaro Carrascosa, Maximiliano Paredes Velasco y J. Ángel Velázquez Iturbide, "Tutores interactivos basados en objetivos pedagógicos para el aprendizaje de la programación". En I Encuentro de Intercambio de Experiencias en Innovación Docente, Félix Labrador Arroyo y Rosa Santero Sánchez (eds.), 2009. Relacionada con el capítulo 4.

12. Isidoro Hernán-Losada: "Conclusiones sobre la aplicación de la Taxonomía de Bloom al diseño de herramientas pedagógicas". Actas del I Seminario de Investigación en Tecnologías de la Información Aplicadas a la Educación (SITIAE 2007); Ed Dykinson S.L. 2009. Relacionada con el capítulo 3.

13. Estefanía Martín, Carlos Lázaro, Isidoro Hernán Losada: "Active learning in Telecommunication Engineering: A case study", IEEE Engineering Education (**Educon 2010**), Madrid, España, 2010. Relacionada con el capítulo 2.

14. Isidoro Hernán Losada: "GenPR: generación aleatoria de problemas", Actas de las I Jornadas en Innovación y TIC Educativas – JITICE 2010, Estefanía Martín Barroso, Manuel Rubio Sánchez y Jaime Urquiza Fuentes (eds.), Serie de Informes Técnicos DLSI1-URJC, número 2010-05, Departamento de Lenguajes y Sistemas Informáticos I, Universidad Rey Juan Carlos, 2010. Relacionada con el capítulo 4.

15. Isidoro Hernán-Losada Cristóbal Pareja-Flores, J. Ángel Velázquez-Iturbide: "Pedagogical use of automatic graders". En Advances in learning processes, 265-274. Editorial: **In-tech**. Vukovar, Croatia, January 2010. Relacionada con el capítulo 2.

16. Isidoro Hernán Losada: "Comparación de la taxonomía de Bloom con otras teorías del aprendizaje". Actas del III Seminario de Investigación en Tecnologías de la Información Aplicadas a la Educación (SITIAE 2009), páginas 203-213. Editorial Dykinson, S.L. Madrid 2011. Relacionada con el capítulo 2.

17. Isidoro Hernán Losada y J. Ángel Velázquez Iturbide, "Aplicación de la investigación social a la evaluación y su relación con la taxonomía de Bloom". En Old Meets New: Media in Education – Proceedings of the 61st International Council for Educational Media and the XIII International Symposium on Computers in Education (ICEM&SIIE'2011) Joint Conference, António Moreira, Maria José Loureiro, Ana Balula, Fernanda Nogueira, Lúcia Pombo, Luís Pedro y Pedro Almeida (eds.), Universidade de Aveiro, 2011. Relacionada con los capítulos 2 y 5.

18. Isidoro Hernán Losada y Estefanía Martín: "Experiencia Docente Innovadora en la Enseñanza. Aplicación a la Programación Orientada a Objetos". En el I Congreso Internacional sobre aprendizaje, innovación y competitividad (CINAIC 2011). Editorial UPM, pp. 254-259, Madrid 2011. Relacionada con el capítulo 2.

19. Isidoro Hernán Losada y J. Ángel Velázquez Iturbide, "Aplicación de la investigación social a la evaluación y su relación con la taxonomía de Bloom". En Indagatio Didactica, 3(3):141-158, diciembre 2011. Relacionada con los capítulos 2 y 5.

Otras contribuciones relacionadas con la tesis fueron:

Fuller U., Johnson C.G., Ahoniemi T., Cukierman D., Hernán Losada I., Jackova J. , Lahtien E., Lewis T.L., McGee Thompson D., Riedesel C., Thompson E. "Developing a Computer Science-specific Learning Taxonomy", Sigcse Bulletin ISSN- 0097-8410 Vol 39, Number 4, pp.: 152-170, December 2007, New York USA. Relacionada con el capítulo 2.

Carlos A. Lázaro Carrascosa, J. Ángel Velázquez Iturbide, Isidoro Hernán Losada, Francisco Gortázar Bellas y Micael Gallego Carrillo: "TextOO: Learning object oriented modeling using the enunciate", Koli Calling 2005. En Proceedings of the Fifth Koli Calling Conference on Computer Science Education, Tapio Slakoski, Tomi Mäntylä y Mikko Laakso (eds.), Turku Centre for Computer Science, TUCS General Publications Series, nº. 41, 2005, págs. 181-182. Relacionada con el capítulo 2.

J. Ángel Velázquez Iturbide, Carlos A. Lázaro Carrascosa e Isidoro Hernán Losada: "Asistentes interactivos para el aprendizaje de algoritmos voraces", IEEE Revista Iberoamericana de Tecnologías del Aprendizaje, IEEE-RITA, 4(3):213-220, agosto 2009 (ISSN 1932-8540). Relacionada con el capítulo 2.

1.5. Organización de la memoria

Este trabajo está estructurado en seis capítulos. A continuación describimos brevemente el contenido de cada uno de ellos, que van cubriendo cada uno de los objetivos propuestos.

El presente capítulo ha mostrado las carencias en las metodologías de las herramientas software para la ayuda al estudio dirigidas a un nivel concreto de la taxonomía de Bloom y por lo tanto la necesidad de desarrollar esas metodologías, lo que nos conduce al trabajo de esta tesis. También se ha descrito el método general de investigación que se va a utilizar en la presente tesis. Además se ha fijado una hipótesis de partida que conlleva unos objetivos a lograr, así como se ha mostrado los logros más importantes producidos por el presente trabajo.

En el segundo capítulo se realiza un estudio del arte sobre los objetivos. Se contextualiza y se analiza la taxonomía de Bloom como herramienta pedagógica, y se realiza un estudio bibliográfico sobre los usos más comunes de esta jerarquía en el campo de la Informática. Se identifican los diferentes problemas que se generan al intentar utilizarla y se intentan dar soluciones para su mejor aplicación. Se realiza un estudio sobre el software educativo existente y las características que debe tener. De esta forma, se le da al lector una visión global del marco de trabajo, uniendo la taxonomía de Bloom y el software educativo.

En el tercer capítulo se hace un estudio detallado de los tipos y características de las herramientas pedagógicas apropiadas a cada nivel de la taxonomía. Posteriormente se introduce el software educativo a diseñar en el presente trabajo. Nos centramos sobre todo en el que da soporte al aprendizaje basado en problemas, poniendo las bases para el correcto desarrollo de los enunciados orientados a un nivel de Bloom. Por último, se da la propuesta principal de esta tesis, estableciendo las diferentes metodologías de diseño de herramientas software pedagógicas para la ayuda a la enseñanza de la

programación orientada a objetos para cada uno de los tres primeros niveles de la taxonomía de Bloom.

En el cuarto capítulo se aplica las metodologías para diseñar y desarrollar distintas aplicaciones, en concreto cuatro aplicaciones. La primera, una herramienta de ayuda a la enseñanza del concepto de herencia simple en POO, tiene tres componentes distintas, orientada cada una a un nivel de la taxonomía distinto. La segunda, denominada GeCom, orientada al segundo nivel de la taxonomía. La tercera, denominada CreOO, orientada al tercer nivel, que además permite la generación y corrección de ejercicios de forma automática (mediante plantillas). La cuarta, llamada AplicOO, también está orientada al nivel tres. Esta herramienta genera y corrige también problemas de manera automática, pero en este caso, la respuesta del alumno es libre. Al final de este capítulo se muestras las dos últimas versiones finalizadas de CreOO y AplicOO que permiten al profesor introducir las plantillas en ejecución.

En el capítulo quinto se muestra la evaluación tanto cuantitativa como cualitativa de dos herramientas. Es esencial evaluar el producto obtenido para comprobar la bondad de la metodología. En concreto se ha evaluado la herramienta de ayuda al estudio del concepto de herencia simple en POO y CreOO. En las evaluaciones cuantitativas se hace una descripción exhaustiva del experimento realizado así como un completo estudio estadístico. Posteriormente se muestran los resultados obtenidos y se analizan los datos. De este análisis se llega a la conclusión de que las herramientas cumplen los objetivos con que fueron diseñadas.

En el último capítulo se describen las conclusiones de este trabajo, así como las líneas abiertas de investigación, fuente de trabajos futuros.

Capítulo 2 Estado del Arte

En este capítulo se sitúa la taxonomía de Bloom en un contexto histórico, explicando los diferentes paradigmas de aprendizaje, se da un repaso a los conceptos fundamentales de esta taxonomía y así como otras teorías o explicaciones alternativas a esta taxonomía. Después se hace un estudio sobre los distintos usos que se han dado esta jerarquía, centrándonos sobre todo en el campo de la Informática y dentro de ella, en la enseñanza de la programación. A continuación se hace una reflexión sobre esta taxonomía, identificando problemas sobre su uso.

También se realiza una descripción del software educativo, poniendo de relieve las características principales que debe tener. Después se hace un estudio de las herramientas educativas para la enseñanza de la programación, distinguiendo la programación orientada a objetos del resto de paradigmas, publicadas en los congresos más relevantes sobre la educación y enseñanza en Informática.

2.1. Contexto histórico de la taxonomía de Bloom

2.1.1. Introducción

La RAE define el aprendizaje en su acepción psicológica como: "adquisición por la práctica de una conducta duradera", es decir, como un cambio permanente del comportamiento del discente como resultado de la aplicación de los conocimientos.

El proceso de aprendizaje es una actividad muy compleja, pues involucra comunicación, memorización, interpretación, comprensión, reflexión e interiorización. ¿Por qué estudiar teorías psicológicas del aprendizaje humano, como uno de los fundamentos para un proceso de diseño de una aplicación de ayuda a la enseñanza? La respuesta es obvia: el desarrollo de entornos de enseñanza-aprendizaje basados en la computadora sin tener una buena base teórica en relación al aprendizaje humano y a las característica de interacción

persona-ordenador como medio de enseñanza pueden conducir a un producto ineficaz. El conocimiento de estas teorías no garantiza el éxito de una aplicación, pero minimiza el riesgo de cometer algún error desde el punto de vista pedagógico. Existen multitud de teorías pedagógicas y psicológicas del aprendizaje, las cuales estudian y dan explicaciones y descripciones, algunas diferentes y a veces contrapuestas del qué y del cómo ocurren estos procesos de construcción del conocimiento. Muchas de estas son teorías parciales que ponen énfasis en determinados aspectos o factores del desarrollo y del aprendizaje, en detrimento de otros. Las teorías de aprendizaje han estado asociadas a la implantación de una metodología pedagógica y su aplicación en la educación. Aquí se muestran algunos de los paradigmas más extendidos junto con una breve descripción de las principales características aportadas y una visión del papel del docente y del alumno, sin pretender ser exhaustivos en dichos contenidos.

2.1.2. Paradigmas del aprendizaje

En este apartado se realiza una clasificación de las teorías del aprendizaje dependiendo de cómo se interpreta al aprendiz y su entorno. En ningún caso se trata de ser taxonómicos, pues existen multitud de paradigmas. La clasificación presentada coincide con la realizada por J. Piaget [Piaget 77], agrupando los paradigmas en dos grandes conjuntos: los que entienden el aprendizaje como acumulación de información (entre los que destaca el conductismo) y los que conciben el aprendizaje como construcción (resto de los paradigmas de la clasificación). Esta clasificación, aun a riesgo de obviar alguna, es la siguiente:

- Paradigma conductista
- Paradigma constructivista
- Paradigma humanista
- Paradigma cognitivo
- Paradigma sociocultural

2.1.3. Paradigma conductista

El conductismo surge como una teoría psicológica y posteriormente se aplica su uso a la educación. Esta fue la primera teoría del aprendizaje ampliamente aceptada. El conductismo surge como una propuesta que aporta un enfoque externo al aprendizaje, pues propone medir este proceso realizando controles a través de fenómenos observables. Antes de este paradigma el aprendizaje era explicado como un proceso interno y era analizado mediante un método llamado introspección, que consistía en preguntar al discente que describiera qué era lo que estaba pensando. A partir de las respuestas, se deducía la forma de aprender. Fue J.B. Watson [Watson 13] el que matizo dicha explicación,

argumentando que la introspección, al ser un método subjetivo, carece de validez científica.

Su axioma principal es que todas las acciones son fruto de las respuestas a un estímulo. La persona es un organismo que se adapta al ambiente y puede ser considerada como una máquina. Debido a esto, cualquier palabra relacionada con la conciencia de las personas es evitada de forma intencionada. Según esta teoría una conducta está motivada por dos elementos. Por un lado, los estímulos ambientales que recibe el estudiante, y, por otro, la respuesta que éste emite a continuación. Tanto estímulos como respuestas son perceptibles, cuantificables y susceptibles de ser utilizados en experimentos científicos. El aprendizaje se realiza por condicionamiento a través del modelo de estímulo-respuesta.

Dentro de esta teoría existen dos corrientes: el condicionamiento básico y el condicionamiento operante o instrumental. Sus inicios se remontan a las primeras décadas del siglo XX, su fundador fue J.B. Watson [Watson 13]. En 1913 publicó "*Psychology as the behaviorist views it*" que marco el punto de partida de esta teoría. De acuerdo con Watson "para que la psicología lograra un estatus verdaderamente científico, tenía que olvidarse del estudio de la conciencia y los procesos mentales (procesos inobservables) y, en consecuencia, nombrar a la conducta (los procesos observables) su objeto de estudio".

Los estudios sobre condicionamiento animal de I. Pavlov [Pavlov 27] y E. Thorndike influyeron en gran medida para fijar las bases del conductismo. En los años 20 el conductismo fue aceptado entre la comunidad de investigadores de la psicología y de la pedagogía y rápidamente se asocio a otras escuelas con principios análogos, tal fue el caso de B.F. Skinner [Skinner 38, 74] con el conductismo operante, cuyas ideas llegaron a convertirse en la principal corriente del conductismo. Gran parte de las teorías del aprendizaje elaboradas entre la década de los veinte y sesenta se puede relacionar con el conductismo.

La teoría del aprendizaje propuesta por Watson, se basa en dos pilares. Primero: El principio de la asiduidad; Las respuestas aumentan de fortaleza de acuerdo con su frecuencia de ocurrencia. La respuesta más frecuente se convierte en la respuesta más fuerte. Segundo: El principio de la cercanía temporal; Las respuestas más recientes se refuerzan más por su frecuencia de uso que las respuestas más antiguas.

El aprendiz, según esta propuesta, es una máquina de estados, por lo que es predecible. Dicho de otra forma, es posible prever su comportamiento si es posible conocer el estado del discente y las fuerzas o conductas que interactúan con él en cada momento. Este tipo de aprendizaje se asocia con el aprendizaje

de tipo repetitivo, memorístico y mecánico. El conductismo propuso el uso de métodos destinados a manipular las conductas. De esta forma, la información y los datos organizados de manera adecuada son los estímulos básicos frente a los que los estudiantes, como simples receptores, deben hacer elecciones y asociaciones dentro de un estrecho margen de posibles respuestas correctas que, de ser ejecutadas, recibían el correspondiente refuerzo. Como consecuencia de esto se obtiene dependencia del estudiante, en la mayor parte, de los estímulos externos, predominio del método ensayo–error, enseñanza y evaluación sometidas al premio-castigo, motivación ajena al estudiante, repetición y memorización. La aplicación de esta teoría conllevó la obtención de recursos técnicos y operativos para que el clásico papel del proceso educativo se fortaleciera gracias a una planificación educativa sumamente sofisticada.

De esta forma el proceso de enseñanza aprendizaje se entiende como la transmisión de contenidos desde alguien que es experto en el tema tratado hacia alguien que no sabe. El aprendizaje se produce cuando el docente le transmite la información al alumno. El profesor para realizar bien el proceso debe de estar dotado de competencias aprendidas, que pone en práctica según las necesidades. Un buen método de enseñanza garantiza un buen aprendizaje. El alumno debe ser un buen receptor de contenidos y su tarea principal es asimilar lo que se enseña.

La enseñanza se centra en los contenidos como directivas a aprender y memorizar para aprobar. La motivación deja de ser propia del estudiante, pasando a ser externa o extrínseca y se apoya en premios o castigos como catalizadores del aprendizaje. Para los conductistas los objetivos pedagógicos se clasifican y secuencian en generales, específicos y operativos, donde lo importante es llegar a identificar conductas observables, medibles y cuantificables. La evaluación ha de centrarse en un producto que sea evaluable. El criterio de evaluación radica en los objetivos operativos. Si no hay cambio observable no hay aprendizaje. La relación entre el educador y el discente es escasa, limitándose a la mera transferencia de contenidos entre un emisor y un receptor pues no se concibe que los estudiantes tengan iniciativas intelectuales.

2.1.4. Paradigma constructivista

El constructivismo es una teoría que intenta explicar cuál es la naturaleza del conocimiento humano. La base de esta teoría la comparten numerosos investigadores entre los más importantes están Piaget [Piaget 52], Vygotsky [Vygostky 78], Ausubel [Ausubel 63], y Bruner [Bruner 66]. Su principal idea es que el conocimiento no se encuentra aislado en el individuo ni en el medio que le rodea, sino que se construye en la interrelación de ambos (sujeto y entorno).

El constructivismo no rompe con las teoría anteriores, así asume que el conocimiento previo sirve de soporte a conocimiento nuevo. También argumenta que el aprendizaje debe ser esencialmente activo. Una persona que aprende algo nuevo, lo incorpora a sus experiencias previas y a sus propias estructuras mentales, transformándolo para usarlo para nuevos fines. Cada información captada es interiorizada y puesta dentro de su red de conocimientos y experiencias previas. Esto conlleva a que el aprendizaje es un proceso subjetivo que cada persona va modificando constantemente dependiendo de sus experiencias, o dicho de otra forma, dos personas no tienen porqué aprender de la misma forma ante una misma vivencia.

Existen dos enfoques constructivistas: el constructivismo psicológico y el constructivismo social. En el psicológico el aprendizaje se efectúa por descubrimiento, manipulación de información, experimentación, pensamiento crítico y diálogo. En el constructivismo social añade al origen de todo conocimiento la sociedad en la que vive el individuo, la cual trae consigo una cultura dentro de una época histórica. Para este enfoque el lenguaje es la herramienta cultural de aprendizaje por excelencia. El individuo construye su conocimiento por que es capaz de leer, escribir y preguntar a otros y preguntarse a si mismo sobre aquellos asuntos que le interesan. La persona construye su conocimiento porque se le ha educado de esa manera y no porque su cerebro lo haga de forma innata.

En ambos contextos el discente dirige su aprendizaje y este se ve como una actividad personal enmarcada en contextos funcionales, significativos y auténticos. El profesor es un guía del aprendizaje y no asume el rol del elemento más importante en el proceso de enseñanza-aprendizaje. Es el alumno quien se convierte en el responsable de su propio aprendizaje, mediante su participación y la colaboración con sus compañeros. Para esto habrá de automatizar nuevas y útiles estructuras intelectuales que le llevarán a desempeñarse con suficiencia no sólo en su entorno social inmediato, sino en su futuro profesional. Y es él quien habrá de lograr la transferencia de lo teórico hacia ámbitos prácticos, situado en contextos reales.

El nuevo plan de estudios de Educación Europea de Enseñanza Superior (EEES) se basa en este paradigma. El aprendizaje de contenidos o dominios de conocimiento por parte del alumno no es suficiente. El estudiante debe además desarrollar una serie de habilidades intelectuales, estrategias, capacidades y competencias para ser capaz de afrontar de forma eficaz cualquier tipo de situaciones de aprendizaje, así como para aplicar los conocimientos adquiridos frente a situaciones nuevas. En este nuevo contexto el énfasis esta puesto en el alumno, independientemente de cualquier situación instruccional, para que

desarrolle su potencialidad cognitiva y se convierta en un aprendiz estratégico (que sepa cómo aprender y solucionar problemas).

2.1.5. Paradigma humanista

El humanismo es una rama del constructivismo. Se basa en que el ser humano es el eje sobre el que gira todo. Esta teoría ofrece una nueva perspectiva con una visión compleja en la que cada hombre, además de la naturaleza genérica, común a la de otros hombres, posee una naturaleza individual, que es única e irrepetible. Existen varias corrientes (cristiano, socialista, existencialista, científica, etc.) que ofrecen su particular interpretación de dicha perspectiva. El hecho de que cada persona sea diferente hace de deba ser estudiado y tratado de forma especial. Esto obliga a evitar, por tanto, en lo posible, el uso de esquemas o conceptos genéricos y prediseñados. Según este paradigma los esquemas preestablecidos solo permiten explicar la conducta, más o menos parecida, de un colectivo pero no la del individuo como ser único que recibe e interioriza cada experiencia de una manera característica y personal. En resumen, para el paradigma humanista la metáfora básica es que el individuo es un organismo único y diferente a los demás que tiende hacia su desarrollo, en relación continua con su entorno.

El movimiento se difundió sobre los años cincuenta y sesenta e influyó tanto en la psicología y pedagogía como en otras áreas del conocimiento. En esos años, algunos de los principales promotores y divulgadores fueron A. Maslow [Maslow 54] (padre del movimiento), L. Bingswanger, G. W. Allport, R. May, C. Roger, entre otros.

Se basa en unos axiomas básicos:

- Un individuo es más que la suma de sus partes. Por dicho motivo, el ser humano debe estudiarse en su totalidad y no separadamente.
- Una persona es la esencia en un contexto humano y vive en relación con otras.
- El ser humano es consciente de sí mismo y de su existencia. El hombre posee un núcleo central estructurado, es decir, su "yo", que es el origen y estructura de todos sus procesos psicológicos.
- El hombre es un ser encaminado hacia un objetivo. Tiende de forma natural a su autorrealización, ya sea en su vertiente cognitiva, afectiva-social o actitudinal.
- Una persona está capacitada para elegir. El hombre tiene facultades de decisión, autonomía y conciencia para optar y tomar sus propias decisiones, lo que le hace ser activo y constructor de su propia vida.

El ser humano es guiado por sus objetivos, sus experiencias, su creatividad y su comprensión de los hechos. La persona progresa al ir superando una serie de necesidades. Estas son, ordenadas jerárquicamente de mayor a menor importancia, necesidades biológicas o fisiológicas, de seguridad, de amor y de pertenecer (sociales), de estima y de autorrealización. El hombre se siente feliz cuando desarrolla todas sus cualidades y, en última instancia, cuando sus necesidades prioritarias llegan a ser las de autorrealización como pueden ser la contemplación de la belleza, la búsqueda de la verdad y el encuentro religioso.

En esta teoría, los objetivos pedagógicos contribuyen al desarrollo humano, de potencialidades, individualidad y autoreconocimiento personal. El proceso de enseñanza-aprendizaje se debe dirigir hacía la ayuda a los discentes para que decidan lo que son y lo que quieren llegar a ser. La educación humanista tiene la idea de que los alumnos son únicos y trata de ayudarles a ser más como ellos mismos y menos como los demás. En este contexto la evaluación se centra en la autoevaluación, ya que no tiene sentido otro tipo de valoración dados los objetivos arriba descritos.

El profesor debe entender a sus alumnos poniéndose en su lugar (empatía) y ser receptivo a sus percepciones y sentimientos. Por lo tanto debe alejarse del autoritarismo y de las clases magistrales. El docente permite que los discentes aprendan favoreciendo todas las investigaciones, experiencias y proyectos, que estos preferentemente inicien o decidan emprender y logren aprendizajes con sentido.

Cada alumno es diferente de los demás, posee intereses, afectos y valora cada cosa de forma particular y se le debe considerar en su totalidad. El discente realizará su aprendizaje cuando llegue a ser significativo. Este aprendizaje se produce cuando se involucra a la persona como totalidad, incluyendo sus procesos afectivos y cognitivos, y se desarrolla en forma experimental. Es importante que el alumno considere el tema a tratar como algo importante para sus objetivos personales. El aprendizaje es mejor si se involucra al discente, dejando que el alumno decida, use sus propios recursos y se responsabilice de lo que va a aprender. Esta teoría promulga que la instrucción debe realizarse en un ambiente de respeto, comprensión y apoyo para los alumnos, y que el profesor sea él mismo y no utilice los mismos métodos de forma continua, es decir, que proceda de manera innovadora.

2.1.6. Paradigma cognitivo

Los estudios de enfoque cognitivo surgen a comienzos de los años cincuenta y se presentan como la alternativa al conductivismo, que era el paradigma más

usado en la psicología. Piaget y la psicología genética [Piaget 52, 54, 72], Ausubel y el aprendizaje significativo [Ausubel 63], la teoría de la Gestalt, Bruner y el aprendizaje por descubrimiento [Bruner 66] fueron dando forma y matizando este nuevo paradigma. Esta teoría estudia la representación mental del conocimiento y las categorías o dimensiones de lo cognitivo: la atención, la percepción, la memoria, el lenguaje, el pensamiento, la inteligencia, la creatividad. El cognitivismo, desde el punto de vista del procesamiento de la información, parte de la hipótesis de que el individuo es un sistema capaz de buscar, organizar, reorganizar, transformar y emplear creativamente la información con diferentes fines.

Para el paradigma cognitivo las personas son receptores, procesadores y asimiladores de información activos y exploratorios y, por lo tanto, son constructores de su propio conocimiento. Los seres humanos no recogen el conocimiento como una respuesta a la experiencia o a la instrucción. El aprendizaje es un proceso activo, que sucede en la mente de los estudiantes. En consecuencia, son los discentes lo que deben elegir entre la información que poseen para enfrentarse a una determinada situación o problema, además de establecer la conexión con la información recibida, lo que los conduce a nuevas formas de conocimiento. Así se establecen nuevas herramientas para tomar decisiones y resolver las situaciones pedagógicas a las que se enfrentan. Según Piaget, el conocimiento se inicia con la fase de asimilación, en la cual el alumno toma información del entorno (profesor, compañeros, material didáctico…) que es relevante para él. Le sigue la fase de acomodación, en la que se compara con los conocimientos previos y se producen asociaciones, creando un esquema en el que incorpora la nueva información. La fase de equilibrio es transversal a las otras dos. En ella el discente va comparando las experiencias con su propia actividad y reajustando sus esquemas con los resultados obtenidos.

Otro de los aspectos a tener en cuenta en este paradigma es que considera al ser humano como una entidad entendida como una totalidad cognitiva y afectiva. Por lo tanto, todo lo cognitivo posee siempre connotaciones afectivas. Los objetivos pedagógicos se programan por contenidos, que sirven para desarrollar las capacidades del alumno, y por métodos que favorecen la ampliación de los valores. La evaluación en el ámbito cognitivo se plantea desde una perspectiva cualitativa para el proceso (formativa, para ver si los objetivos de la enseñanza están siendo alcanzados o no) y cuantitativa para el producto (sumativa, para calificar el grado de conocimiento). Para ello es indispensable realizar una evaluación inicial de conceptos previos y destrezas básicas. Respecto a la formalización de los objetivos, debe considerarse los esfuerzos de B. Bloom y sus colaboradores sobre la clasificación cognitiva de las

metas educativas, en su ya tan conocida taxonomía [Bloom 56] que se describe en el apartado siguiente.

El docente debe primar el aprendizaje sobre la enseñanza. Para ello tiene que reflexionar sobre los métodos de instrucción utilizados para mejorar el aprendizaje a los discentes. En las clases, el profesor ejerce de guía en el proceso de enseñanza-aprendizaje, pero no fuerza este proceso, con el fin de no imponer al alumno a llevar un rol pasivo. El alumno no es un mero receptor y debe ser el impulsor activo de su propio aprendizaje. La enseñanza se centra en el desarrollo de estrategias de aprendizaje orientadas a los objetivos cognitivos y afectivos. La motivación debe ser intrínseca al discente y el aprendizaje debe ser significativo para incorporar su esencia a su esquema mental. La finalidad del proceso de enseñanza-aprendizaje está en enseñar a pensar o aprender a aprender, de forma que los alumnos sean procesadores activos y críticos del conocimiento.

2.1.7. Paradigma sociocultural

En esta teoría, llamada también constructivismo situado, el aprendizaje significativo solo se logra en un contexto social. El paradigma histórico cultural o sociocultural fue impulsado por Vigostky [Vygostky 78] a partir de la década de 1970. Contrario a la teoría de Piaget [Piaget 72] del paradigma cognitivo, donde se afirma que es el sistema cognitivo lo que estructura significados, este paradigma cambia esta idea y afirma que el aprendizaje se consigue gracias a la interacción social. Cada sujeto percibe la información a partir de su propia realidad sociohistórica en la que vive. De esta forma el discente recibe la información y ésta es modificada por el propio sujeto de acuerdo a sus condicionamientos socioculturales. Se aprende solo, o se aprende de los otros o con la ayuda de los otros desarrollando capacidades hasta ser capaz de resolver independientemente un problema. Según este paradigma, los niños adquieren primero el aprendizaje cultural a nivel social y luego lo interiorizan y esto lo hacen primero interactuando con personas y luego de forma individual. Para Vigotsky [Vygostky 78] las funciones superiores se originan a través de relaciones entre seres humanos. El aprendizaje está primero en el nivel social (intersicológico) y después en un nivel interno (intrasicológico). La idea primordial de este paradigma se puede sintetizar diciendo que el individuo aunque es importante no es la única variable en el aprendizaje. Su clase social, sus vivencias, y consecuentemente sus oportunidades sociales, su momento histórico, las herramientas que ha tenido a su disposición, son variables que, además de ayudar al aprendizaje son parte integral de él.

La idea básica de este paradigma es que el individuo es un sujeto inmerso en un espacio donde se realizan las interrelaciones entre personas y el medio ambiente. Los objetivos pedagógicos se pueden plantear en grupo (aprendizaje colaborativo) donde el lenguaje es la principal, pero no la única, herramienta cultural de aprendizaje. El discente construye su conocimiento por que es capaz de leer, escribir, hablar con otros y razonar sobre aquellos asuntos que le interesan. Según esta corriente el individuo construye su conocimiento no porque sea una función natural de su cerebro sino por que literalmente se le ha enseñado a construir a través de la comunicación con otras personas. Los objetivos pedagógicos deben estar diseñados de forma que incluyan la interacción social, no sólo entre alumnos y profesor, sino entre los propios discentes y, a veces, con la sociedad en la que viven. En el proceso de aprendizaje o en la construcción de los conocimientos las herramientas más importantes son la búsqueda, la investigación, la exploración y la solución de problemas. La evaluación se ha de realizar a través de un contexto interactivo entre el profesor, el alumno y la tarea.

El profesor se presenta como un intermediario con la cultura social. Su misión es articular el aula de forma que ayude a fomentar las interacciones, la creación de expectativas y a generar un clima de confianza. Así, a través de actividades conjuntas e interactivas, el docente procede creando espacios de construcción para que el alumno se apropie de los conocimientos, en parte gracias a sus aportes y en parte gracias a las ayudas estructuradas en las actividades propuestas, siguiendo cierta dirección intencionalmente determinada que acompañe al discente a alcanzar el objetivo marcado. El profesor debe reflexionar y actuar sobre la base de esta teoría, de que la mente es un conjunto de capacidades –capacidad de observación, atención, memoria, razonamiento, etc.– y que cada mejora de cualquiera de esas capacidades significa la mejora de todas las capacidades en general. El alumno debe ser entendido como un ser social, fruto y protagonista de las múltiples interacciones sociales en que se ve envuelto a lo largo de su vida.

2.2. Taxonomía de Bloom

Esta jerarquía surgió en pleno proceso de desarrollo del paradigma constructivista, y dentro de este, de la vertiente cognitivista. Independientemente de la explicación dada por las distintas teorías del aprendizaje, existe un problema clave que es: ¿Cómo medir el grado de conocimiento adquirido por un alumno? o dicho de una forma más ligada a la pedagogía: ¿Cómo clasificar las reacciones del discente como resultado consciente del proceso de la educación? De esa pregunta surgen otras complementarias en el proceso de enseñanza-aprendizaje: ¿Cómo pueden los

profesores describir o fijar los objetivos pedagógicos? ¿Los objetivos dados son trasladables a otros profesores de la misma materia? Un grupo de psicólogos y pedagogos norteamericanos se hicieron estas preguntas y trataron (y consiguieron) de establecer un sistema de clasificación universal que facilitará fijar objetivos educativos y como consecuencia poder evaluar si los discentes conseguían alcanzarlos. Se dieron cuenta que "pueden observarse unos tipos de conducta esencialmente idénticos entre personas de carácter normal y en un determinado nivel de enseñanza (elemental, media y superior)". Esto permite describir una clasificación que agrupe esos tipos de conducta. Además se buscaba que este marco teórico pudiera usarse para facilitar la comunicación entre examinadores, promoviendo el intercambio de materiales de evaluación e ideas de cómo llevar ésta a cabo.

El proceso de desarrollo de esta taxonomía estuvo liderado por Benjamín Bloom, doctor en educación de la Universidad de Chicago (USA). Empezó a fraguarse en la Convención de la Asociación Norteamericana de Psicología, que se celebró en Boston en 1948. Terminó con la publicación de un libro titulado "*Taxonomy of educational objetives*" [Bloom 56] donde se establecen los resultados de esos años de investigación sobre el tema. En él se formuló una taxonomía de los objetivos de la educación, desde entonces conocida como Taxonomía de Bloom, que puede entenderse como una clasificación de las metas educativas.

El libro explica detalladamente cada uno de los niveles, estableciendo una definición. Luego especifica algunos objetivos característicos de cada nivel y subnivel. A continuación realiza una breve discusión de los problemas, haciendo ciertas observaciones sobre ellos, para finalizar con bastantes ejemplos de las cuestiones destinadas a comprobar los objetivos de cada tipo. Para cada ejemplo se hace un breve comentario para poner de relieve lo que se exige al alumno y los resultados logrados. Todos los ejemplos son de materias ajenas a la Informática, ya que por esa época todavía estaba en ciernes.

La taxonomía establece una jerarquía de seis niveles con grado creciente de aprendizaje del alumno (véase Figura 2). Cada nivel presupone la capacitación del alumno en los niveles precedentes. Según ascendemos por la jerarquía nos encontramos un mayor grado de aprendizaje:

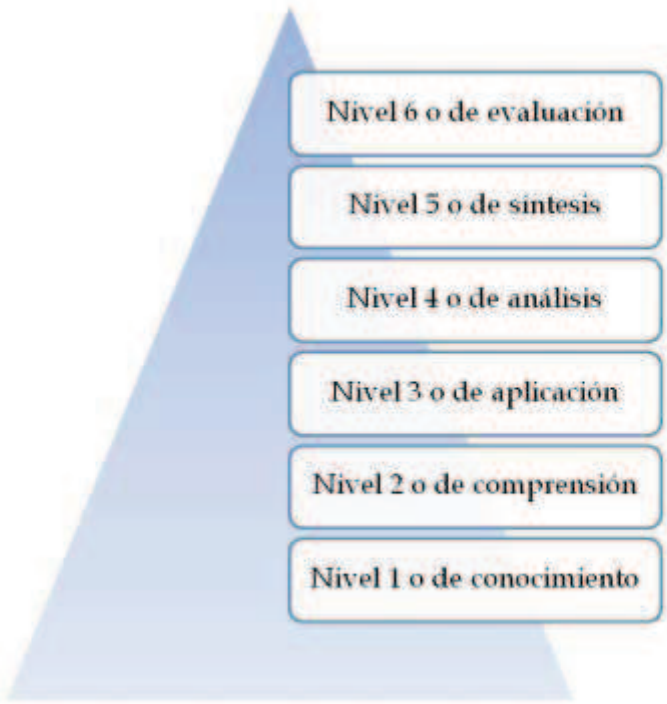


Figura 2. Taxonomía de Bloom

– **Nivel 1 o nivel de conocimiento.** El conocimiento comprende el recuerdo de los conceptos específicos y universales, el de métodos y procesos o el de unas normas, estructuras o situaciones. La memorización constituye el principal proceso psicológico. Por este motivo, el estudiante puede reconocer o recordar la información sin ser necesario cualquier clase de entendimiento o razonamiento sobre su contenido. Este nivel tiene varias subcategorías:
  - Conocimiento de lo especifico (terminología -1.11- y hechos -1.12-).
  - Conocimiento de los modos y maneras de tratar hechos específicos (convencionalismos -1.21-, tendencias y secuencias -1.22-, clasificaciones y categorías -1.23-, criterios -1.24- y metodologías -1.25-).
  - Conocimiento de conceptos o leyes universales y abstracciones de un campo concreto (principios y generalizaciones -1.31- y de estructuras y teorías -1.32-).

Bloom en su libro explica textualmente que la evaluación de este nivel hay que hacerla de manera cuidadosa: "aunque para el conocimiento se requiere algo más que unos actos exclusivamente memorísticos, tanto la forma en que se expone la pregunta como el grado de precisión y exactitud de la respuesta no ha de diferenciarse mucho del modo con que adquirió dichos conocimientos".

- **Nivel 2 o nivel de comprensión**. Para los autores de la taxonomía, el término comprensión engloba objetivos, actitudes y reacciones que expresan cierto entendimiento del mensaje literal contenido en una comunicación. El individuo logra enterarse de las ideas fundamentales de un mensaje, pudiendo aplicarlas sin que le sea preciso relacionarlas con otras materias ni llegar a sus últimas consecuencias. Por lo tanto, el estudiante puede entender y explicar el significado de la información recibida. Dentro de este se encuentran los subniveles:
  - Transferencia -2.10-. El discente es capaz de "captar frases no literales" (metáforas, ironías, exageraciones…). También puede cambiar expresiones verbales matemáticas a expresiones simbólicas y viceversa.
  - Interpretación -2.20-. B. Bloom entiende este nivel como: "La capacidad para captar el mensaje global de un trabajo dentro de un nivel determinado de generalización".
  - Extrapolación -2.30-. Esta subcategoría es explicada como: "La capacidad para manipular las conclusiones de un trabajo de acuerdo con las deducciones inmediatas derivadas de toda aserción explicita".

  Para realizar la comprobación de los objetivos de este nivel, proponen ejercicios donde el alumno deba definir términos o establecer correspondencias entre definiciones y términos. Insisten en que la respuesta del alumno debe llevar integrada alguna característica distinta a la aprendida, bien porque el enunciado lleva alguna situación inédita o bien porque este lleva algunas claves contextuales que permiten al alumno extrapolar o interpretar de forma análoga a la aprendida.

- **Nivel 3 o nivel de aplicación.** El estudiante puede seleccionar y utilizar datos y métodos para solucionar una tarea o un problema dado. Entre otras cosas, en el libro se describe como: "La aplicación a los fenómenos estudiados en un trabajo de los términos o conceptos científicos utilizados en otros". Para entender la diferencia con el nivel anterior (comprensión) B. Bloom explica que: "para resolver un problema en el nivel 2, es absolutamente necesario que el alumno domine una abstracción en un grado suficiente para que pueda demostrar su uso correcto cuando se le invite a ello. Sin embargo, el nivel de aplicación exige algo más. El alumno deberá aplicar la abstracción adecuada sin que se le haya advertido de antemano si es o no correcta y sin haberle preparado para utilizarla en esa circunstancia específica". Este nivel es el que se exige al alumno en las situaciones de la vida real, de ahí que sea muy importante incluirlo como objetivo de la enseñanza general.

– **Nivel 4 o nivel de análisis.** El discente puede distinguir, clasificar y relacionar hipótesis y las evidencias de la información dada, así como descomponer un problema en sus partes. Bloom lo explica: "Los procesos de análisis tienen por objeto dar una mayor claridad al mensaje, poner de manifiesto cómo se halla estructurado y explicar el modo con que se desarrolla el proceso, de sus consecuencias, de sus orígenes y de su estructura". Se pueden distinguir tres clases de análisis:
  - Análisis de los elementos -4.10-. Consiste en "el descubrimiento de los factores que integran un mensaje".
  - Análisis de las relaciones -4.20-. Son "las conexiones e influencias recíprocas entre los elementos y partes de un mensaje".
  - Análisis de los principios de la organización o de las normas de estructuración -4.30-. Es "la organización sistemática que hace posible la permanencia del mensaje como un solo conjunto, incluyendo la estructura implícita y la explicita".

  El fin último de este nivel es que los alumnos vayan capacitándose para distinguir lo hipotético de lo cierto en un mensaje determinado, para discernir las conclusiones y probar una afirmación, para diferenciar los aspectos principales de los secundarios, para observar cómo se encadenan unas ideas con otras, para descubrir los verdaderos propósitos de una comunicación, etc.

– **Nivel 5 o nivel de síntesis.** El estudiante puede generalizar ideas y aplicarlas para solucionar un nuevo problema. Se puede crear una nueva solución, manejando diferentes aspectos de conocimiento y métodos, de tal manera que el resultado sea algo más que la suma de los componentes. En el libro se expone como "el proceso de la manipulación de piezas, factores, partes, etc. ordenación y combinación de modo que lleguen a constituir una estructura que hasta ese momento se hallaba en estado más o menos confuso". Se subdivide en:
  - Elaboración de un mensaje único -5.10-. Consiste en la capacidad de producir una redacción capaz de transmitir las ideas o las experiencias de forma estructurada.
  - Realización de un plan o un conjunto de operaciones programadas -5.20-. Esto incluye el estudio de alternativas y la toma de decisiones dentro de un contexto.
  - Deducción de un conjunto de relaciones abstractas 5.30-. Es "la capacidad para elaborar hipótesis correctas fundamentadas en el análisis de los

factores sometidos a estudio, y para modificar dichas hipótesis de acuerdo con los datos aportados por otros nuevos factores."

– **Nivel 6 o nivel de evaluación.** El estudiante puede comparar, criticar y evaluar métodos o soluciones para solucionar un problema o para elegir el mejor. Incluye "juicios cuantitativos y cualitativos sobre el grado en que dichos métodos y materias cumplen con los criterios que los condicionan". A su vez se distinguen:
  - Evaluación en términos de evidencias internas -6.10-. Consiste en "enjuiciar, por medio de normas internas, la aptitud para distinguir entre un cálculo aproximado de la exactitud en la presentación de hechos y la labor minuciosa de comprobación de la precisión de una prueba, de unos datos o de una afirmación.
  - Juicio en términos de criterios externos -6.20-. Esta subcategoría trata de "enjuiciar, por medio de normas externas, la aptitud para cotejar un trabajo concreto con los que se consideran modelos de la materia, es decir, con otros trabajos de calidad contrastada.

Para su aplicación existen listas de verbos (véase Tabla 1) que pueden ser usados como guías para el diseño de preguntas de evaluación o como palabras clave a la hora de definir los objetivos pedagógicos.

Tabla 1. Ejemplos de verbos de la taxonomía de Bloom

Nivel	Ejemplos de verbos
Conocimiento	Nombrar, listar, reconocer, identificar, etiquetar.
Comprensión	Explicar, traducir, resumir, predecir
Aplicación	Aplicar, usar, construir, hacer, desarrollar, implementar
Análisis	Analizar, comparar, clasificar, dividir, simplificar, depurar
Síntesis	Construir, estructurar, hacer, diseñar, planear, solucionar
Evaluación	Comparar, juzgar, justificar, evaluar, medir

Se han hecho muchos estudios sobre esta taxonomía. En el apartado siguiente se presentan estos estudios así como alternativas a ella.

2.3. Estudios y alternativas a la taxonomía de Bloom.

Este apartado recoge y resume algunos estudios relevantes sobre la taxonomía de Bloom y algunas propuestas alternativas [Hernán 11b]. Está basado en parte en el capítulo "*Empirical Studies of the Structure of the Taxonomy*" del libro de Anderson *et al.* [Anderson 01] y parte en la bibliografía consultada por el autor de la tesis.

2.3.1. Taxonomía de Gerlach y Sullivan (1967)

Para los autores [Gerlach 67], a la taxonomía de Bloom le falta precisión al definir los comportamientos asociados a sus seis niveles. Para suplir dicha ausencia, usan verbos para etiquetar sus niveles. Así al primer nivel le denominan "nombrar", al segundo "ordenar", al tercero "identificar", al cuarto "describir", al quinto "demostrar" y al sexto y último "construir". Si analizamos los verbos y los comparamos con diversas listas de verbos para aplicar la taxonomía de Bloom podemos asociar nombrar y ordenar al primer nivel de la taxonomía, identificar y describir al segundo, demostrar al tercero y construir al quinto, quedando sin correspondencia directa el nivel cuarto y el sexto de la jerarquía de Bloom (análisis y evaluación) [Anderson 01].

2.3.2. Categorías de Ausubel y Robinson (1969)

Definen sus seis niveles ordenados por categorías [Ausubel 69]. En este caso los autores intentan dar una explicación a la naturaleza del conocimiento, distanciándose del trabajo de Bloom en el que establece una jerarquía del conocimiento. Aún así es interesante remarcar las similitudes y diferencias entre ambas [Anderson 01].

D. Ausubel y F. Robinson clasifican la naturaleza del aprendizaje como: aprendizaje representacional, aprendizaje conceptual, aprendizaje proposicional o metodológico, aplicación del aprendizaje, aprendizaje de resolución de problemas y aprendizaje de la creatividad. Existe una relación casi uno a uno entre ambas clasificaciones. La única diferencia podemos observarla si hilamos muy fino. El primer tipo de aprendizaje se corresponde con el nivel de conocimiento. Los dos siguientes tipos de aprendizaje, el representacional y el conceptual, pueden estar asociados al nivel de comprensión. Y los tres siguientes a los tres siguientes niveles de Bloom, quedando sin correspondencia el último nivel (evaluación) [Anderson 01].

2.3.3. Sinónimos de Metfessel, Michael y Kirsner (1969)

En este caso, los autores [Metfessel 69] piensan que la taxonomía de Bloom es difícil aplicar por su falta de ejemplos y su generalidad. Para paliar esas carencias tratan de proporcionar unos descriptores alternativos a los niveles de Bloom. Para cada categoría y subcategoría proporcionan verbos. Por ejemplo, para el nivel de Aplicación: generalizar, relacionar, elegir, desarrollar, organizar, usar… Para el nivel de Análisis: distinguir, detectar, identificar, clasificar, deducir… Por lo tanto, en este caso no se trata de una teoría nueva, sino simplemente de dar un enfoque práctico a la taxonomía de Bloom [Anderson 01].

2.3.4. Jerarquía del Aprendizaje de Gagné (1970)

R. Gagné trata de definir y clasificar los distintos tipos de aprendizaje, tomando los principios de los distintos paradigmas del aprendizaje [Gagné 70]. Según Robert Gagné el aprendizaje puede clasificarse en: aprendizaje por señales, aprendizaje por respuesta ante estímulos, aprendizaje por encadenamiento, aprendizaje por asociación verbal, aprendizaje por discriminación, aprendizaje de conceptos, aprendizaje de principios generacionales, aprendizaje de resolución de problemas. Si comparamos y asociamos con la taxonomía de Bloom, podemos corresponder al primer nivel los aprendizajes por señales, por respuesta ante estímulo, por asociación verbal y por discriminación. Al segundo nivel le pueden corresponder los aprendizajes por discriminación y de conceptos. Al tercer nivel, los aprendizajes de principios generacionales. Y a los tres últimos niveles el aprendizaje de resolución de problemas.

2.3.5. Dominio Cognoscitivo de Stahl y Murphy (1981)

Los autores [Stahl 81] definen una nueva taxonomía concentrándose en cómo es procesada la información. Las etapas para adquirir nuevos conocimientos son las siguientes. La primera consta de preparación y observación. Se trata de recibir la información a través de los sentidos. La segunda es la recepción, donde se almacena los datos relevantes. La tercera se compone de la trasformación y la adquisición o interiorización. En ella se le da sentido a la información recibida. La cuarta es la retención, en la cual se clasifica la información. La quinta es la transferencia, en la cual se usa la información existente para usarla ante la nueva experiencia. La sexta se denomina incorporación. En esta etapa se usa la información interiorizada. La séptima es la organización, donde se interrelaciona y se da prioridad a la información. Y la última es la generación, en la cual se sintetiza la información para formar nuevas ideas y conocimientos. Haciendo una correspondencia de cada nivel con las etapas en las que nos los podemos encontrar, se puede asociar el primer nivel de Bloom a las etapas de recepción y retención. El nivel de comprensión y de aplicación con la etapa de transferencia. Y la etapa de generación se puede asociar al nivel de síntesis. El resto de niveles estarían o bien difuminados en las etapas o bien sin establecer ninguna correspondencia con las etapas. [Anderson 01]

2.3.6. Integración del Conocimiento de Bruce (1981)

R. L. Bruce reorganiza los niveles de la taxonomía de Bloom [Bruce 81]. Elimina el nivel de conocimiento y lo inserta en otras categorías. Así la integración del conocimiento queda con los siguientes niveles y subniveles: comprensión, aplicación (aplicando conceptos y aplicando reglas), análisis (identificar

elementos esenciales, reconocer relaciones, reconocer principios organizacionales), síntesis (crear un única comunicación, combinar en un plan, formular principios básicos) y evaluación (establecer criterios, hacer juicios) [Anderson 01].

2.3.7. Análisis del Conocimiento y Habilidades de Romizowski (1981)

A. J. Romizowski hace una clasificación atendiendo a los conocimientos en sí y a la puesta en práctica de dichos conocimientos [Romizowski 81]. Por lo tanto tiene dos categorías: conocimientos y habilidades. Dentro de la categoría de los conocimientos, distingue entre conocimientos de hechos, de métodos, de conceptos y de principios. Y en la categoría de habilidades diferencia entre aprendizaje reproductivo y aprendizaje productivo. Haciendo una correspondencia con la taxonomía de Bloom, la primera categoría entraría dentro del nivel de conocimiento. Respecto a la segunda categoría, el aprendizaje reproductivo entraría en los tres primeros niveles de Bloom y el aprendizaje productivo en los tres últimos [Anderson 01].

2.3.8. S.O.L.O. (Structure of the Observed Learning Outcome). Biggs y Collis. (1982)

J. B. Biggs y K. F. Collis observaron que los alumnos, en su evolución desde la etapa inicial del conocimiento hasta la etapa final o maestría, muestran una secuencia consistente, o ciclo de aprendizaje, que es generalizable a una gran variedad de tareas y en particular a las tareas escolares. Esta secuencia se refiere a un progreso jerárquico en la complejidad estructural de sus respuestas. Así, esta teoría trata de observar los resultados del aprendizaje del alumno y clasificarlos dentro de las cinco siguientes categorías, clasificadas en orden creciente de complejidad [Biggs 82]:

- *Pre-estructural*. El alumno responde de forma vaga o da una respuesta que no tiene que ver con el problema planteado, bien porque no ha entendido la cuestión o porque desconoce la respuesta.
- *Mono-estructural*. La respuesta del discente corresponde solo a una parte del problema complejo. Esto indica que el alumno no conoce las relaciones ni reconoce alguno de los distintos aspectos de la cuestión.
- *Multi-estructural*. El estudiante demuestra con su respuesta que posee conocimiento y quizá la comprensión de muchos hechos, pero no muestra evidencias suficientes de la comprensión del conjunto.
- *Relacional*. Respuestas en las que se pone de manifiesto una comprensión conjunta de las interrelaciones entre los diferentes

aspectos usados para dar la solución. Todas las partes relevantes e importantes de la tarea se encuentran bien ubicadas y se unen en un conjunto coherente.

- *Abstracción extendida*. Respuestas que hacen uso de principios, hechos, procesos, etc. más abstractos que aquéllos que describen el problema actual. La respuesta del estudiante va más allá de lo que se ha aprendido. Demuestra que el alumno ha alcanzado un nivel de generalización mayor que en el nivel anterior.

Según los autores, esta clasificación puede usarse tanto para evaluar la calidad del aprendizaje como para establecer los objetivos del currículo.

Las categorías de pre, mono y multi-estructural pueden corresponder a los dos niveles inferiores de la taxonomía de Bloom. La categoría relacional puede estar emparentada con el nivel de aplicación, análisis y diseño. La última categoría se puede relacionar con el nivel mayor de la taxonomía.

2.3.9. Taxonomía del Proceso Cognitivo de Quellmalz (1987)

La taxonomía del proceso cognitivo establece cinco tipos de procesos, con el objeto de establecer una jerarquía fácil de usar [Quellmalz 87]. Estos procesos son recuerdo, análisis, comparación, inferencia y evaluación. Se puede establecer una correspondencia con los niveles de Bloom de la forma siguiente: recuerdo con el primer nivel, análisis con el nivel de análisis, comparación con el nivel de comprensión, inferencia con el de síntesis y evaluación con su homónimo en la taxonomía de Bloom. Queda sin asignar el nivel de aplicación [Anderson 01].

2.3.10. Marco Conceptual de Hauenstein (1998)

A. D. Hauenstein intenta aunar los tres dominios del aprendizaje (el cognitivo, el afectivo y el psicomotor) para dar consistencia a sus teorías [Hauenstein 98]. Dentro del dominio cognitivo identifica cinco categorías, con sus respectivas subcategorías: conceptualización (identificación, definición y generalización), comprensión (traducción, interpretación y extrapolación), aplicación (simplificación y solución), síntesis (hipótesis y resolución), evaluación (análisis y calificación). La correspondencia con la taxonomía de Bloom es evidente y en este caso, por lo tanto, se omite [Anderson 01].

2.3.11. Marco Comparativo de Reigeluth y Moore (1999)

En este caso, los autores realizan una compresión de las seis categorías o niveles de Bloom en cuatro, para dar un marco para comparar distintas teorías de diseño instruccional [Reigeluth 99]. Las categorías son: memorización de

información, comprensión de relaciones, aplicación de habilidades y aplicación de habilidades genéricas. Las tres primeras categorías corresponden a los tres primeros niveles y la última categoría a los tres niveles superiores [Anderson 01].

2.3.12. Taxonomía de Bloom revisada (2001)

David R. Krathwohl (colaborador de B. Bloom) junto a Lorin W. Anderson revisaron la taxonomía y el mayor inconveniente que encontraron es que está definida en términos estáticos, sin incluir los procesos de aprendizaje. Aunque Bloom pone ejemplos de todos los niveles, de los cuales se pueden deducir las acciones que el alumno debe realizar, D. Krathwohl y L. Anderson redefinieron la taxonomía dividiéndola en dos dimensiones [Anderson 01]: la dimensión del conocimiento, basada en la materia que se pretende enseñar y la dimensión del proceso cognitivo, fundamentada en el verbo o acción que se quiere conseguir enseñar. Así en la primera dimensión quedan cuatro niveles principales y no jerarquizados, denominados: conocimiento de hechos, conocimiento de conceptos, conocimiento de procedimientos y conocimiento metacognitivo. La segunda dimensión, que mide el proceso cognitivo, se compone de seis niveles, en este caso jerarquizados, y son: recordar, comprender, aplicar, analizar, evaluar y crear. Como se puede observar, esta última dimensión es muy parecida a la taxonomía original de Bloom en la que el primer nivel se pasa a denominar recordar, y los dos últimos niveles se cambian de orden, pasando a ser el quinto evaluar y el sexto y último crear.

2.3.13. Taxonomía del aprendizaje para Informática (2007)

Un grupo de investigadores internacionales, entre los que se encontraba el autor de esta tesis, se reunieron en Dundee para intentar resolver el problema de la aplicación de la taxonomía de Bloom a la enseñanza de la Informática [Fuller 07]. De dicha reunión surgió un nuevo planteamiento de la taxonomía denominada "taxonomía matricial" (véase Figura 3), donde se plantea que la comprensión e interpretación de un código o programa es una capacidad casi independiente de la capacidad para diseñar y construir programas. De ahí surgen las dos dimensiones de la matriz: la interpretación (*interpreting*) y la producción (*producing*). Cada dimensión tiene sus niveles. La interpretación tiene los subniveles de recuerdo (*remember*), comprensión (*understand*), análisis (*analyse*) y evaluación (*evaluate*). La producción tiene los subniveles nulo (*none*), aplicar (*apply*) y crear (*create*). Así los discentes pueden situarse en cualquier celda de la matriz, teniendo en cuenta que el recorrido siempre es secuencial, empezando por la primera celda de los subniveles elementales (recuerdo y nulo respectivamente). Como se observa, los nombres de los subniveles han sido

extraídos de la taxonomía de Bloom y la idea es parecida a la taxonomía revisada, donde se distingue la parte de conocimientos para interpretar y los procedimientos para generar programas.

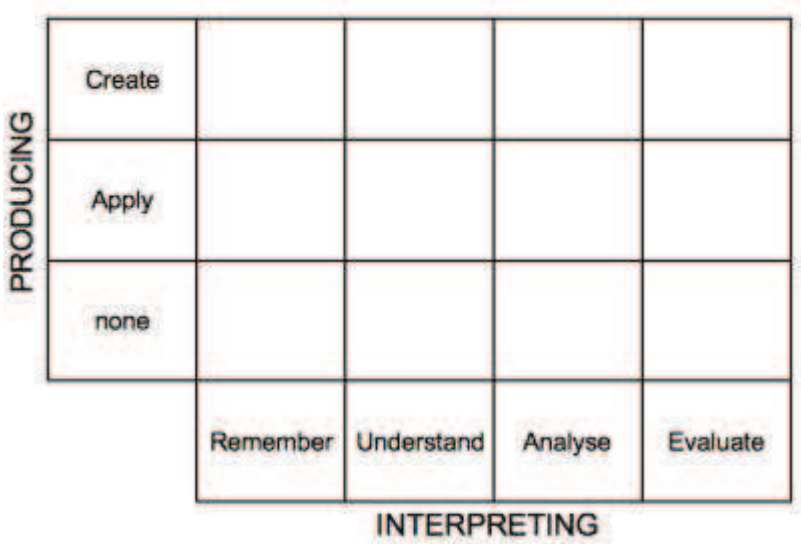


Figura 3. Representación gráfica de la taxonomía matricial

2.4. Discusión sobre la taxonomía de Bloom y sus alternativas.

Como el propio Benjamin Bloom comenta en su libro, realizar una taxonomía de los objetivos de la educación es un objetivo muy complicado, pues trata de jerarquizar unos fenómenos que no pueden observarse ni manipularse con la objetividad con que se dan otros fenómenos en las ciencias exactas. Por el mero hecho de tratar de medir las reacciones humanas y al ser cada persona única, no se deben observar conductas individuales. Pero sí que existen reacciones comunes entre seres normales. Así la taxonomía establece niveles de forma general, pues se trata de agrupar estas manifestaciones. Esta generalidad, que en principio puede parecer un problema, se eligió de forma premeditada, para evitar una fragmentación de los niveles. De hecho, se concretaron unos subniveles para particularizar algunos aspectos de cada nivel.

Aun así, entre la mayoría de los investigadores que han dedicado su tiempo al estudio de la taxonomía, se producen discrepancias y se muestran dificultades al tratar de aplicarlas a distintas materias, como por ejemplo, la ingeniería eléctrica [Apple 02] o la física [Hestenes 92, Rhoads 99] o al *marketing* [Healy 2011]. En informática, Colin Johnson y Ursula Fuller encontraron discrepancias en su aplicación entre profesores de este área [Johnson 07]. El Apéndice A muestra un estudio del autor donde también se demuestra los

distintos puntos de vista de los profesores de distintas asignaturas con los expertos en la taxonomía de Bloom. E. Lathinen [Lathinen 07] hace un estudio sobre los alumnos y distingue entre alumnos competentes, prácticos, teóricos, memorísticos, sin preparación e indiferentes. Compara las respuestas dadas a unos ejercicios clasificados en los seis niveles de Bloom de estos grupos llegando a la conclusión de que los niveles no son incrementales, ya que hay alumnos del grupo práctico que ofrecen respuestas correctas en niveles superiores e incorrectas en niveles inferiores. En este articulo se muestran las preguntas realizadas según el nivel de Bloom, y bajo nuestro punto de vista, no se encuentran bien situadas.

2.5. Aplicaciones de la taxonomía de Bloom

La taxonomía de Bloom ha sido usada para conseguir distintos fines. En el campo de la Informática, después de una revisión de la literatura existente sobre el tema, hemos agrupado los usos en: diseñar cursos con distintos niveles de granularidad o establecer metodologías para impartirlos, evaluar y/o explicar el conocimiento del discente y para desarrollar materiales docentes. En esta sección, haremos una visita a cada uno de las aplicaciones de la taxonomía.

2.5.1. Diseño de cursos

Algunos autores han usado esta taxonomía para el diseño de asignaturas, cursos o de planes de estudio. La lista de artículos es muy larga, por lo que procedemos a mostrar algunos de los más representativos, sabiendo que dejamos muchos sin exponer. A. Howard *et al.* [Howard 96] proponen diseñar un curso de programación estructurada (que trata de estructuras de datos básicas y algoritmos recursivos) identificando los objetivos de la mayoría de las cuarenta lecciones del curso de forma clara usando los niveles de Bloom para establecerlos. Las lecciones tienen unas metas y concluyen con un gráfico mostrando la evolución del curso dibujando el nivel de cada lección. M. Doran y D. Langan [Doran 95] publican los resultados de un proyecto que usa la taxonomía de Bloom para definir los objetivos de los dos primeros años de la carrera de Informática. Este proyecto trata de implementar una estrategia espiral de presentación de contenidos y realización de prácticas para afianzar el aprendizaje, afirmando que los estudiantes principiantes no están capacitados para llegar a los niveles más altos de la taxonomía, de ahí el enfoque en espiral. De esta manera se consigue que los discentes vuelvan a revisar conceptos para conseguir un mayor grado de experiencia y por lo tanto, subir a nivel mayor de la jerarquía. C. Reynolds y C. Fox matizan las unidades didácticas del curriculum del área de la tecnología de la información [Reynolds 96] añadiéndoles objetivos pedagógicos con la taxonomía.

Otros ejemplos lo tenemos en el artículo de I. Sanders y C. Mueller [Sanders 00] donde los autores proponen el rediseño del curriculum de su universidad, haciendo que los primeros años del grado de programación contengan materiales relacionados con los niveles iniciales de la taxonomía e ir incrementado el nivel a medida que pasan los cursos. P. Machanick [Machanick 00] describe su experiencia en aplicar la taxonomía en el diseño de tres cursos totalmente diferentes de Informática (estructuras de datos, algoritmos e inteligencia artificial y arquitectura de computadores) concluyendo que esta taxonomía es adecuada para este fin, ya que los alumnos presentan mejores resultados que cuando los cursos eran diseñados de forma tradicional. L. Schatzberg [Schatzberg 02] también la aplica con éxito a la asignatura de diseño y análisis de sistemas. P. Bourque *et al.* utilizan la jerarquía para el cuerpo de conocimiento de la guía de ingeniería del software [Bourque 03], en concreto para cuatro áreas del conocimiento de esta disciplina: ingeniería de gestión del software, mantenimiento del software, ingeniería del proceso del software, y calidad del software. B. Manaris y R. McCauley [Manaris 04] usan la taxonomía para realizar el diseño curricular de la asignatura de interacción persona-computadora, dotándola de esta forma de mayor objetividad en su presentación. R. Lister [Lister 01] también propone fijar los objetivos del primer curso de programación en los dos primeros niveles a la vez que propone los métodos de evaluación.

También existen estudios que se han basado en Bloom para establecer nuevas metodologías. D. Buck y D. Stucki [Buck 00] afirman que en los primeros cursos de la enseñanza de la programación no están en consonancia con las teorías pedagógicas, por lo que proponen usar la jerarquía para realizar un nuevo enfoque denominado *inside/out*. Básicamente trata de estructurar la presentación de contenidos, empezando por los de los niveles inferiores para ir avanzando a lo largo de la taxonomía. C. Piombo *et al.* [Piombo 03] proponen un marco de referencia para la enseñanza adaptativa, donde se modeliza al alumno según su estilo de aprendizaje y los materiales a presentar según los objetivos docentes medidos por la taxonomía de Bloom. De esta forma se logra presentar a los alumnos los contenidos de acuerdo a sus preferencias de aprendizaje y a los objetivos pedagógicos.

Como se observa por la cantidad de estudios efectuados la taxonomía de Bloom ofrece un marco excepcional para fijar objetivos y metodologías de cursos de distintas materias en Informática.

2.5.2. Evaluación del conocimiento

La evaluación del conocimiento es una tarea complicada, pues conlleva considerar multitud de factores, unos relacionados con el evaluador (objetividad, criterios, diseño de las pruebas, etc.) y otros relacionados con el alumno (preparación, conocimientos previos, actitud, etc.). La evaluación y las pruebas usadas para ello necesitan estar acorde con el curso educativamente hablando. El profesor debería preparar la tarea y las estrategias de evaluación con el fin de que los estudiantes estén animados para aprender y demostrar su conocimiento. Para ayudar a esta tarea (diseño y desarrollo correcto de las prueba) existen multitud de herramientas automáticas que permiten realizar evaluaciones de distintos aspectos de un programa [Ala-Mutka 05] y así evitar al profesor el largo proceso de corrección. La evaluación automática ofrece una serie de ventajas como: son un soporte más rápido y versátil, más coherente, más objetivo y da una retroalimentación instantánea a los discentes y a los estudiantes. Desafortunadamente, existen algunos aspectos de la Informática que no pueden ser evaluados de forma automática.

Para realizar el diseño y las estrategias de las pruebas, que es la tarea más importante de la evaluación, la taxonomía de Bloom ofrece un marco excepcional. De hecho, aquí se encuentran la mayoría de los usos de esta jerarquía. A continuación se ofrecen algunos ejemplos de esta utilización.

R. Rademacher [Rademacher 99] une la taxonomía de Bloom con la teoría de Greenwood para dar un nuevo enfoque a la evaluación del nivel cognitivo necesario que deben tener los distintos roles para realizar las actividades implicadas en los sistemas de gestión del conocimiento. J. Buckley y C. Exton realizan un estudio sobre la tarea de mantenimiento del software. Usan la taxonomía para dar a cada una de las tareas de mantenimiento del software un nivel de Bloom [Buckley 03] y realizan un experimento piloto para comprobar el grado de conocimiento (usando la jerarquía) sobre el código de cada alumno. R. Lister y J. Leaney [Lister 03] proponen una mezcla de formas de evaluar para medir el nivel de Bloom que el alumno ha conseguido alcanzar. T. Scott aplica la taxonomía para clasificar los ejercicios propuestos y da ejemplos aplicándolos a la enseñanza de la programación [Scott 03]. T. Naps *et al*. [Naps 03b] realizan un estudio sobre la eficacia educativa de las visualizaciones en el campo de la programación de computadoras. Detectan un conjunto de buenas prácticas eficaces educacionalmente hablando. Proponen usar la taxonomía como marco estándar para que los educadores e investigadores puedan medir la eficacia educativa.

D. Oliver *et al.* van más allá de la evaluación de los alumnos. Ellos tratan de evaluar un curso, introduciendo el concepto de "*Bloom rating*" o puntuación de Bloom [Oliver 04]. Está nueva métrica consiste en fijar cada prueba de evaluación en un nivel de Bloom, puntuado de 1 al 6 (de acuerdo con el nivel) y dar un peso a cada prueba de acuerdo con su importancia en la calificación final. Se realiza la suma ponderada y se normaliza, obteniéndose así un valor. Este resultado es la puntuación de Bloom de ese curso. Los resultados mostrados se han medido usando cuatro investigadores para puntuar y pesar cada prueba y haciendo la media.

E. Shneider y O. Gladkikh [Shneider 06] proponen un enfoque genérico para diseñar cuestiones de evaluación basados en Bloom que consiste en realizar esquemas o plantillas válidos para desarrollar preguntas de los niveles requeridos. Así establecen dos tipos de evaluaciones, las sumativas y las formativas. Para ambas realizan una tabla con una fila para cada nivel y tres columnas. La primera contiene las clases de problema (que contiene enunciados genéricos con palabras o verbos clave que encajan en el nivel correspondiente a la fila). La segunda columna contiene los conocimientos previos del alumno. Y la última columna ofrece la solución esperada que el alumno debe dar. Esta tabla puede contener una cuarta columna con alguna anotación de los autores para clarificar o modificar los enunciados.

E. Thompson *et al*. [Thompson 08] hacen un estudio detallado de la taxonomía, interpretándola y dando ejemplos de preguntas de evaluación que pueden ser usadas como guías por los docentes para determinar el nivel de conocimiento del discente. En la misma línea investigadora se encuentra el artículo de N. Khairuddin y K. Hashim [Khairuddin 08], pero con menor detalle en el estudio realizado. La taxonomía de Bloom sigue siendo marco de referencia para clasificar problemas, como muestran J. Rutkowski *et al.* [Rutkowski 10].

2.5.3. Desarrollo de materiales docentes

En este apartado nos referimos a material docente, entendiendo como tal a las herramientas software de ayuda al aprendizaje, de nuevo poniendo énfasis en el área Informática. La preparación del temario de una materia usando la taxonomía de Bloom se ha explicado en el apartado de diseño de cursos. La utilización de la jerarquía para desarrollar aplicaciones de ayuda a la enseñanza, es la menos documentada en profundidad.

El software educativo se utiliza actualmente en muchas materias de distintas ciencias e ingenierías. La mayoría de aplicaciones educativas son diseñadas e implementadas de manera intuitiva, incorporando características

que el autor o los autores creen que son convenientes para ayudar al alumno en su proceso de aprendizaje. Esta intuición a veces es acertada, pues, tras una evaluación de la herramienta, ésta demuestra su bondad desde el punto de vista de la eficacia pedagógica. Otras veces, o bien no se comprueba o bien se realiza una evaluación cualitativa que depende de las sensaciones de los usuarios más que de los progresos que se consiguen.

De los artículos sobre herramientas relacionadas con la taxonomía de Bloom, analizados y publicados en congresos y revistas relevantes podemos destacar el trabajo de Duane Buck y David Stucki sobre la aplicación JKarelRobot [Buck 01]. Esta herramienta software de ayuda a la enseñanza a la programación estructurada pretende dar soporte a todos los niveles de la jerarquía. JKarelRobot es una ampliación de *Karel the robot* [Pattis 95]. Para el primer nivel permite escribir instrucciones (en varios lenguajes) para comprobar su sintaxis. El segundo nivel se consigue mediante ejercicios donde el alumno tiene que predecir el estado del robot después de una instrucción. En el nivel de aplicación el discente tiene que poner la instrucción que conduce al robot a un estado determinado. En el nivel de análisis tienen varios ejercicios. Uno de ellos consiste en traducir un programa en un diagrama de flujo. Otro consiste en dar un fragmento de código que contiene un error, encontrarlo, explicarlo razonadamente y corregirlo. En otro se ofrece un programa que sigue unas especificaciones y se pide cambiarlo para que cumpla otras especificaciones ligeramente diferentes. Para dar soporte al nivel de diseño, JKarelRobot permite realizar programas y ejecutarlos. Y en el nivel superior, el de evaluación, se pide a los alumnos que comparen dos implementaciones que siguen una especificación común.

Otro autor que ha usado las aplicaciones pedagógicas para dar soporte a la taxonomía de Bloom es Amruth N. Kumar. Este autor, junto a sus colaboradores, ha desarrollado *applets*, que ha denominado *problets*, que ayudan a conseguir el nivel de aplicación en conceptos muy concretos de la programación estructurada [Kumar 00, Krishna 01, Kumar 02, Dancik 03] y a su vez, ha evaluado su impacto educativo, consiguiendo buenos resultados.

E. Lahtinen y T. Ahoniemi [Lahtinen 05] la usan para el diseño de visualizaciones que ayudan a los alumnos a comprender la programación. Para cada nivel de la taxonomía presentan distintos tipos de material visual y de interacciones con esas visualizaciones, relevantes para conseguir dicho nivel. En la misma línea encontramos dos herramientas de visualización de programas, denominadas SRec [Velázquez 08] y VAST [Almeida 09]. Ambas están orientadas al nivel de análisis. La primera sirve para la enseñanza de la recursividad mediante diversas técnicas de visualización. La segunda se usa

para la enseñanza de procesadores de lenguajes mediante la visualización del análisis sintáctico.

I. Hernán *et al.* [Hernán 08, Hernán 10b] usamos esta taxonomía para incluir meta-información en las colecciones de problemas que son susceptibles de ser corregidos de forma dinámica por jueces automáticos. De esta forma, los alumnos pueden utilizarlos para aprender o auto-evaluarse en un determinado nivel. También se ha usado para el desarrollo de asistentes interactivos para el aprendizaje de algoritmos voraces [Velázquez 09]. En el mismo se identifican los objetivos, que consisten en dar soporte a diferentes niveles de la taxonomía de Bloom. Se relaciona cada parte de las aplicaciones con el correspondiente nivel. En el artículo se habla de dos herramientas, AMO y SEDA, que en la actualidad están agrupadas en un solo sistema, llamado GreedEx.

Aunque existe multitud de material docente desarrollado, no hemos encontrado en la literatura ningún artículo que detalle las técnicas para diseñar aplicaciones de ayuda al estudio siguiendo criterios pedagógicos fundamentados en Bloom.

2.6. Problemas de la aplicación de la taxonomía a la enseñanza de la programación

Algunos miembros del grupo de investigación [LITE], después de realizar un estudio de la problemática de la aplicación de la taxonomía al campo de la programación, procedimos a clasificar esos problemas [Hernán 04a] en: contextuales, terminológicos e inherentes a la complejidad del dominio de la informática dentro del campo de la programación. A continuación detallamos estos problemas.

2.6.1. Contextuales

B. Bloom y sus colaboradores realizaron una prueba que consistía en ver si entre varios investigadores se llegaba a un acuerdo para clasificar unos objetivos educativos y unos test, llegándose a numerosas ambigüedades e incongruencias. Uno de los máximos problemas que se pusieron en evidencia en este estudio es que siempre es necesario conocer o presuponer las experiencias precedentes de los discentes. Se llegó a la conclusión de que sólo se podrán clasificar de forma satisfactoria los test cuando sean conocidos los antecedentes en que se expusieron los ejercicios de la prueba. Por lo tanto podemos concluir que la aplicación de la taxonomía depende del contexto del alumno, de su experiencia previa.

2.6.2. Terminológicos

Algunas actividades de la programación comparten la misma palabra que algunos niveles de la taxonomía de Bloom. Esto lleva a considerar a estas actividades en el nivel que se denomina de idéntica forma. Sin embargo, dichas tareas pueden clasificarse en otros niveles diferentes. Por ejemplo, el análisis de la complejidad de los algoritmos puede ser situado en el nivel de conocimiento si consideramos los conceptos básicos (ej. la notación O o complejidad en el peor caso), en el nivel de comprensión por la capacidad de comprender y repetir el análisis de complejidad de un algoritmo dado o en el nivel de aplicación si el problema es totalmente novedoso para el discente. Estos problemas suelen ser debidos a que muchos investigadores se limitan a leer el resumen de la taxonomía ofrecida en el propio libro. Nosotros recomendamos la lectura completa, pues en ella se encuentran guías y discusiones que clarifican la clasificación.

2.6.3. Complejidad del dominio de la programación

La programación es un dominio muy complejo, ya que posee varios niveles de razonamiento, varias áreas de conocimiento, varias claves para resolver un ejercicio y varias especificaciones de un mismo problema. La existencia de muchos niveles de discusión es probablemente el mayor hándicap y contribuye a aumentar la confusión. En el estudio realizado pudimos distinguir varios elementos de diferente naturaleza a tener en cuenta a la hora de resolver un problema informático. Estos universos son: la materia sobre la que versa el problema, el dominio del problema, el lenguaje de programación, el método de resolución o paradigma de programación y el entorno de programación. Una asignatura tiene diferentes materias a estudio: el lenguaje de programación, la metodología de programación, las estructuras de datos y algoritmos, etc. Por ejemplo, una asignatura de algoritmia y estructuras de datos típicamente utiliza un lenguaje de programación como herramienta, sin embargo en una asignatura de introducción a la programación el lenguaje de programación es uno de los objetos de estudio.

Como consecuencia, la implementación de un algoritmo en un lenguaje de programación se sitúa en diferentes niveles de la jerarquía de Bloom dependiendo del curso. En un curso de algoritmia se coloca como una actividad de comprensión, pues la implementación implica una traducción o reescritura del algoritmo a un lenguaje y habitualmente el lenguaje no es nuevo para el estudiante. Sin embargo, en un curso de programación, esta actividad se coloca en el nivel de aplicación si requiere que use construcciones del lenguaje que ha aprendido pero no domina. Consideraciones similares se pueden hacer con el

entorno de programación (o cualquier otra herramienta de programación) o con el dominio del problema.

2.7. Software educativo

Como se ha comentado en el capítulo de introducción se puede definir el software educativo como toda aplicación creada con la finalidad específica de ser usada como facilitadores del aprendizaje, o dicho de otro modo, todo programa desarrollado para ayudar en los procesos de enseñanza y aprendizaje. En este trabajo utilizaremos indistintamente los términos software educativo, herramientas informáticas de ayuda a la enseñanza o aplicaciones pedagógicas o educativas como sinónimos.

2.7.1. Tipos de software educativo

Existen muchas clasificaciones del software educativo dependiendo del criterio elegido. P. Marqués [Marquès 99] describe varias clasificaciones que reproducimos a continuación: según los contenidos (temas, áreas curriculares...); según los destinatarios (criterios basados en niveles educativos, edad, conocimientos previos...); según su estructura: tutorial (lineal, ramificado o abierto), base de datos, simulador, constructor, herramienta; según sus bases de datos: cerrado, abierto o modificable; según los medios que integra (convencional, hipertexto, multimedia, hipermedia, realidad virtual); según su "inteligencia" (convencional, experto o con inteligencia artificial); según los objetivos educativos que pretende facilitar (conceptuales, procedimentales, actitudinales); según las actividades cognitivas que activa (control psicomotriz, observación, memorización, evocación, comprensión, interpretación, comparación, relación -clasificación, ordenación-, análisis, síntesis, cálculo, razonamiento -deductivo, inductivo, crítico-, pensamiento divergente, imaginación, resolución de problemas, expresión -verbal, escrita, gráfica…-, creación, exploración, experimentación, reflexión metacognitiva, valoración...); según el tipo de interacción que propicia (recognitiva, reconstructiva, intuitiva/global, constructiva); según su función en el aprendizaje: instructivo, revelador, conjetural, emancipador); según su comportamiento (tutor, herramienta, aprendiz); según el tratamiento de errores (tutorial -controla el trabajo del estudiante y le corrige-, no tutorial); según sus bases psicopedagógicas sobre el aprendizaje (conductista, cognitivista, constructivista,…); según su función en la estrategia didáctica (entrenar, instruir, informar, motivar, explorar, experimentar, expresarse, comunicarse, entretener, evaluar, proveer recursos -calculadora, comunicación telemática-...); según su diseño (centrado en el aprendizaje, centrado en la enseñanza, proveedor de recursos).

Otra clasificación más general se puede hacer escogiendo como criterio la forma en la que el alumno interacciona y aprende con la herramienta. Según este criterio podemos dividir las herramientas y materiales para asistir el aprendizaje en algorítmicos y heurísticos.

En los materiales o **herramientas algorítmicas** predomina (aunque no es único) el aprendizaje vía transmisión de conocimiento desde un experto en el tema hacia quien lo desea aprender. Esta herramienta tiene que diseñarse de forma que presente los contenidos o las actividades de forma que el estudiante sea guiado por un determinado camino. Por lo tanto el rol del discente es adquirir los conocimientos a base de realizar las actividades o visionar el contenido del material docente preparado para él.

Las herramientas algorítmicas, se pueden clasificar dependiendo de la forma de interaccionar y ayudar al aprendizaje al alumno. Aquí hay que comentar que esta clasificación no es cerrada, los límites entre algunas herramientas son difusos y podrían catalogarse incluso en herramientas heurísticas. Esta clasificación es la siguiente:

Libros Electrónicos

Se pueden definir como sistemas de información que ponen a disposición del usuario páginas conceptualmente organizadas del mismo modo que las de un libro de papel, con las que además poder interaccionar. Su objetivo es presentar información al estudiante a partir del uso de texto, gráficos, animaciones, videos, hipertexto, etc., pero con un nivel de interactividad y motivación que le facilite las acciones que realiza.

Sistemas Tutoriales

Son aplicaciones informáticas basadas en el diálogo con el estudiante, que van presentando información de forma normalmente guiada. Estos sistemas suelen tener en cuenta las características del alumno, siguiendo una estrategia pedagógica para la transmisión de conocimientos.

Sistemas Entrenadores

Son sistemas que ayudan a afianzar conceptos y destrezas previamente adquiridas, por lo que su propósito es contribuir al desarrollo de una determinada habilidad, intelectual, manual o motora.

En los materiales o **sistemas heurísticos**, en cambio, el aprendizaje se suele producir por experimentación y descubrimiento. En estos casos, el diseñador crea entornos con escenarios o actividades que el alumno debe

explorar. El discente adquiere el conocimiento a partir de su propia experiencia, por lo tanto crea sus propios modelos de pensamiento y sus propias interpretaciones del concepto en estudio, las cuales puede someter a prueba con la herramienta.

Los sistemas heurísticos se pueden clasificar teniendo en cuenta su función educativa. Aquí de nuevo hacemos notar que los límites entre los distintos sistemas son borrosos. Esta clasificación es:

Simuladores

Su objetivo es apoyar el proceso de enseñanza – aprendizaje, intentando reflejar la realidad de forma entretenida.

Juegos Educativos

Son aplicaciones que pretenden crear situaciones excitantes y entretenidas para que el alumno adquiera conocimientos o destrezas mediante la exploración o el entretenimiento. Se suelen utilizar metáforas de la realidad.

Sistemas Expertos

Un sistema experto o sistema basado en el conocimiento es un sistema informático capaz simular el comportamiento de un experto humano en un área concreta de conocimiento especializado. Pueden ser usados en educación para generar contenidos o actividades dependiendo del usuario, y evolucionar de forma dinámica a medida que se usa.

2.7.2. Software educativo para la enseñanza de la programación

Existe un acuerdo entre la mayoría de investigadores en este campo que la enseñanza de la programación es esencial en las ciencias de la computación y que el aprendizaje es una tarea compleja, independientemente del paradigma usado para su enseñanza. Se han realizado muchas herramientas de ayuda al aprendizaje para complementar los típicos cursos basados en lecciones y prácticas en laboratorios. La programación tiene una problemática propia de esta materia ya que es un dominio muy complejo, pues posee varios niveles de razonamiento, varios elementos de conocimiento, varios problemas clave y varias especificaciones de un problema. La existencia de muchos niveles de discusión es probablemente el problema que contribuye a aumentar la confusión. Estos niveles son: la materia sobre la que versa el problema a resolver, la complejidad intrínseca de los conceptos y estructuras de los

programas, el lenguaje de programación (con su propia sintaxis y semántica) y la complejidad del entorno de programación usado.

Existe multitud de software educativo para la enseñanza de la programación. La mayoría de herramientas han sido desarrolladas para satisfacer la demanda de los programadores profesionales. A menudo, estas aplicaciones tratan muchos conceptos y poseen muchas características que las convierten en herramientas difíciles de usar por parte de alumnos que se inician en la programación. Además, en general, los mensajes de error y de advertencias no son claros.

En este apartado mostramos algunas aplicaciones de ayuda a la enseñanza-aprendizaje de la programación, dividiéndolas en cuatro grandes grupos [Gómez 05] que no son excluyentes entre sí: herramientas basadas en visualizaciones, entornos basados en ejemplos, herramientas de entornos de programación simplificados y entornos de simulación. Existen otras clasificaciones como la producida por [Pears 07]. En ella clasifica las aplicaciones educativas también en cuatro grupos: herramientas de visualización, herramientas de corrección automática, entornos de programación y otras herramientas. Nosotros hemos optado por unir ambas clasificaciones, ya que de esta forma se tiene una mayor división o especialización de las herramientas y dejar las siguientes categorías: herramientas basadas en visualizaciones, entornos basados en ejemplos, entornos de programación, entornos de simulación, herramientas de corrección automática y otras herramientas. Debido a la ingente cantidad de software educativo existente, aquí se muestran únicamente los más conocidos por sus publicaciones en los diferentes congresos y revistas especializadas o los que están más relacionados con la programación orientada a objetos, por ser este paradigma parte de este trabajo. En ningún caso tratamos de ser exhaustivo, únicamente queremos dar una visión general de los sistemas existentes.

Herramientas de visualización o animación: las persona son muy buenas procesando información visual. Por otra parte, la mayoría de conceptos de programación, la mayoría de algoritmos y la mayoría de estructuras de datos son conceptos abstractos sin forma gráfica obvia. Además, los programas y los algoritmos son elementos dinámicos y reconocer y comprender las partes importantes es una tarea complicada para los alumnos. No es por tanto sorprendente, que muchos investigadores [Stasko 98] han dedicado su trabajo a la visualización de la estructura y al funcionamiento de programas y a la animación de algoritmos.

Se puede distinguir entre herramientas que visualizan estructuras estáticas o la ejecución de código, como Jeliot y sus descendentes [Jeliot3], JGrasp [JGrasp] o los depuradores integrados en los entornos de programación y las herramientas de animación de algoritmos, que muestran el comportamiento dinámico del código, desde distintos puntos de vista. Como ejemplos de estas herramientas podríamos citar JHAVE [Naps 05], MatrixPro [Karavirta 04] y SRec [Velázquez 08].

Entornos basados en ejemplos: también se han desarrollado herramientas basadas en ejemplos para la ayuda al aprendizaje de la programación. Los autores justifican este tipo de herramientas en que los alumnos usan las soluciones de problemas que ya han visto para intentar resolver problemas nuevos parecidos a los anteriores. De hecho, existen metodologías orientadas a esta forma de resolver los problemas [Ben-Bassat 11]. Entre este software educativo podemos destacar ELM-PE, ELM-ART dos tutores inteligentes para la enseñanza del lenguaje funcional Lisp [Brusilovsky 96, Weber 01] y Web Examples (WebEx) para el lenguaje C [Brusilovsky 01]. También se basan en la misma técnica los libros electrónicos. Por ejemplo, C. Gregorio *et al.* desarrollaron un sistema de generación automática de libros electrónicos dedicados a presentar ejercicios de programación a medida de las necesidades del usuario [Gregorio 02].

Si nos referimos a la enseñanza de la POO, podemos destacar Javy [Gomez-Martin 03] que es una herramienta diseñada para la enseñanza del proceso de compilación en Java, que usa un entorno virtual en 3D que simula la maquina virtual de Java. Otra herramienta parecida a la anterior es VisualJVM [Garrido 08], que permite visualizar como trabaja la maquina virtual mientras se ejecuta un programa real. Aunque si bien es cierto que ambas herramientas no muestran conceptos de POO directamente, estudiando cómo trabaja la maquina virtual se pueden comprender dichos conceptos.

Entornos de programación: se han desarrollado multitud de entornos dependiendo del lenguaje y del paradigma de programación al que ofrecen soporte, que agrupan herramientas como editores de texto inteligentes, depuradores, herramientas de prueba, generadores de documentación, etc. que las convierten en software educativo con mucha funcionalidad. Como se ha comentado en la introducción a este apartado, una fuente de problemas en el aprendizaje de la programación es la complejidad de uso del entorno de programación, ya que tantas características abruman al alumno novel. Se han

desarrollado algunos entornos de programación simplificados, que tratan de minimizar este problema. Podemos destacar THETIS, que se desarrollo para la enseñanza del lenguaje Ansi C [Freund 96] y AnimPascal [Satratzemi 01] para la enseñanza de la programación imperativa en Pascal. Algunos de estos sistemas además incluyen facilidades de visualización y animación, incrementando así sus características pedagógicas.

Si nos centramos en los entornos para la enseñanza de la programación orientada a objetos tenemos los entornos de programación profesionales (Eclipse [Eclipse], Netbeans [Netbeans], JCreator [JCreator], etc…). A continuación hacemos un repaso de los entornos de programación simplificados y diseñados con alguna característica especial para la enseñanza de la programación orientada a objetos en Java.

DrJava [DrJava] es un entorno de programación ligero, con una interfaz simple (véase Figura 4) y que ofrece las características típicas de estas aplicaciones [Allen 02]. Se basa en una interfaz sencilla apoyada en un bucle de lectura, evaluación y escritura que posibilita al programador desarrollar, probar y depurar programas Java de una forma interactiva e incremental.

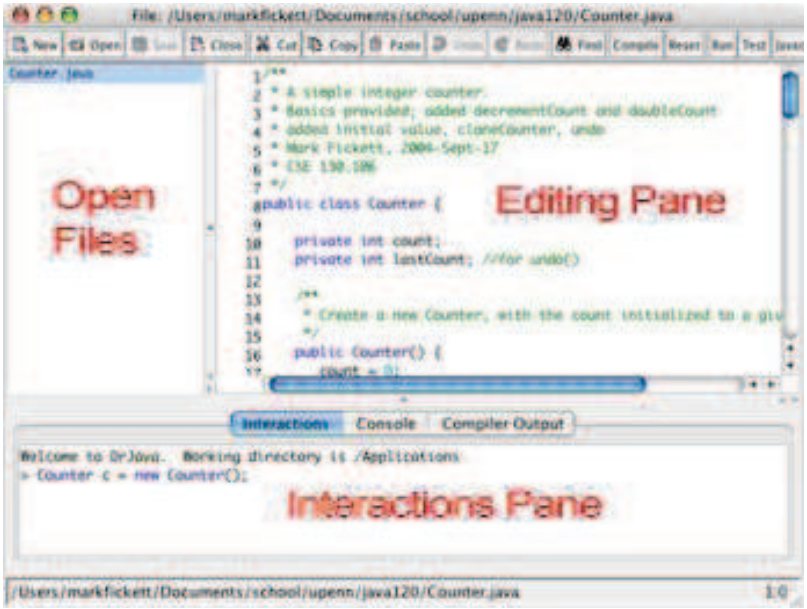


Figura 4. Ventana principal de DrJava

BlueJ [BlueJ] fue diseñado e implementado por Michael Kölling y John Rosenberg para mejorar la enseñanza y el aprendizaje de programación orientado a objetos usando Java como el lenguaje de implementación [Kolling 03]. BlueJ permite a los estudiantes tener una vista gráfica de las clases y los objetos en un sistema (véase Figura 5), permite a los alumnos interactuar directamente con ellos, simplifica la prueba de métodos y las clases, y elimina la necesidad de codificar código Java difícil y confuso como por ejemplo, el método principal (*main*) de una clase. Ofrece un entorno de programación con

una buena base pedagógica pero no usa la taxonomía explícitamente. Esta herramienta ha sido ampliamente usada y evaluada [Van Haaster 04], mostrando su bondad para la enseñanza.

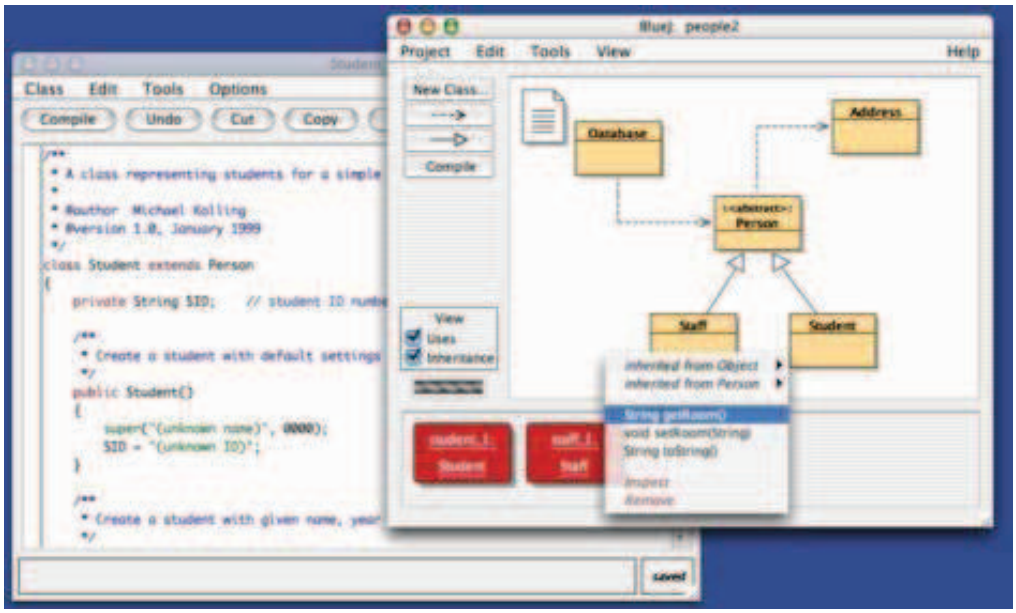


Figura 5. Ventanas del BlueJ

Del mismo autor, M. Kölling, tenemos otro entorno: Greenfoot [Greenfoot]. Es un entorno de desarrollo integrado educativo (véase Figura 6 y Figura 7) que facilita escribir aplicaciones gráficas interactivas [Kolling 08]. Está dedicado a la enseñanza y aprendizaje de programación orientada a objetos. Este sistema permite escribir programas fácilmente y ver mediante gráficos bidimensionales animados el efecto de cada instrucción. Para ello contiene varias herramientas educativas que ayudan a la comprensión de diversos conceptos fundamentales del paradigma orientado a objetos. Es altamente motivante ya que da una retroalimentación gráfica instantánea de lo que el alumno programa. También ha sido usada con éxito [Gallant 08]. Como precursor de estas ideas podemos citar a JKarel [Bergin 02], que es la versión mejorada para la enseñanza de la programación orientada a objetos de *Karel the robot* [Pattis 95]. De nuevo se muestra un entorno gráfico donde podemos aprender los distintos conceptos de la programación dando instrucciones a Karel el robot.

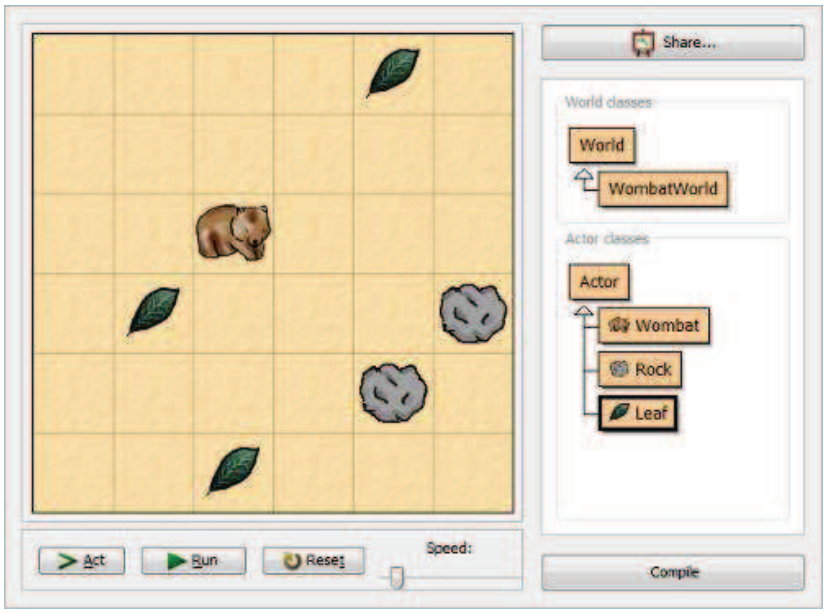


Figura 6. Ventana principal de Greenfoot

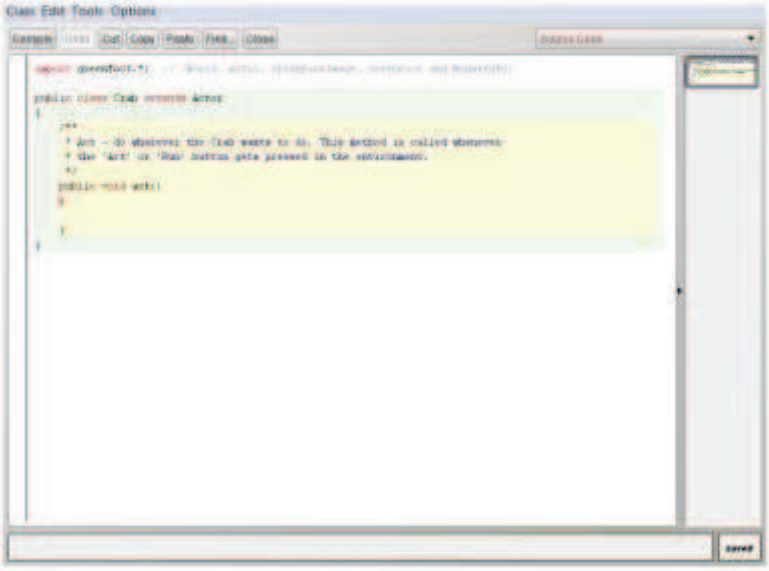


Figura 7. Ventana de edición de código de Greenfoot

Jeroo [Jeroo], es una herramienta que permite aprender los conceptos básicos de la programación orientada a objetos. Está inspirada en Karel el robot y sus versiones posteriores. Jeroo es un animal parecido a un canguro raro que vive en una isla (*Santong Island*). El usuario tiene una ventana en la cual todo los elementos son visibles (véase Figura 8). El código fuente se resalta a medida que se va ejecutando y va produciendo una animación del comportamiento del canguro. Ha demostrado cualitativamente ser una herramienta efectiva cuando se usa al inicio del curso en estudiantes noveles [Dorn 03].

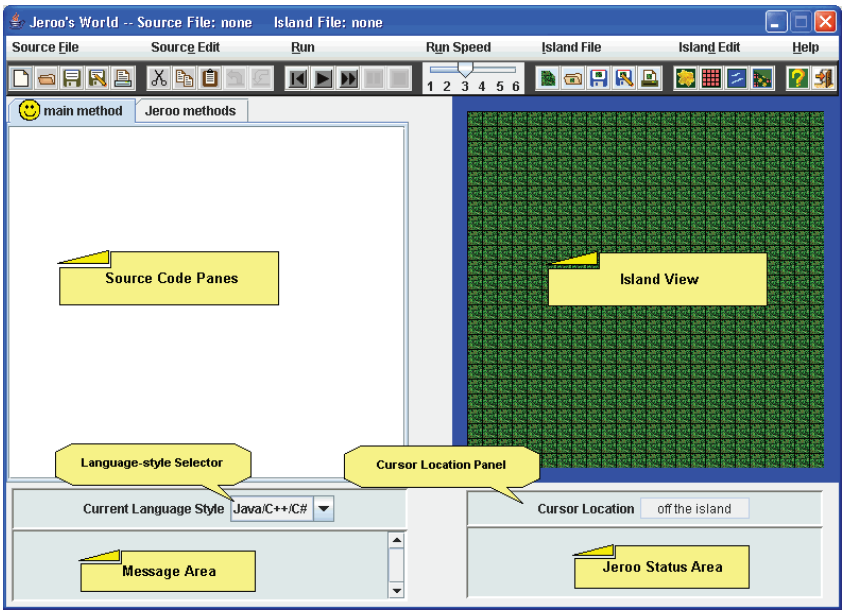


Figura 8. Ventana principal de Jeroo

Jeliot3 [Jeliot3] también es una buena herramienta para la enseñanza de la programación orientada a objetos. Básicamente es un visualizador de programas (véase Figura 9) que ofrece una animación de la ejecución del programa. Contiene muchas de las características que detectamos en nuestro estudio: ofrece visualizaciones y demostraciones del funcionamiento de un programa, permite crear y manipular un programa así como depurarlo Esta aplicación ha demostrado su eficacia educativa cuando es usada en la introducción a la programación orientada objetos por programadores noveles [Moreno 04]. Pero de nuevo, no se basa en la taxonomía.

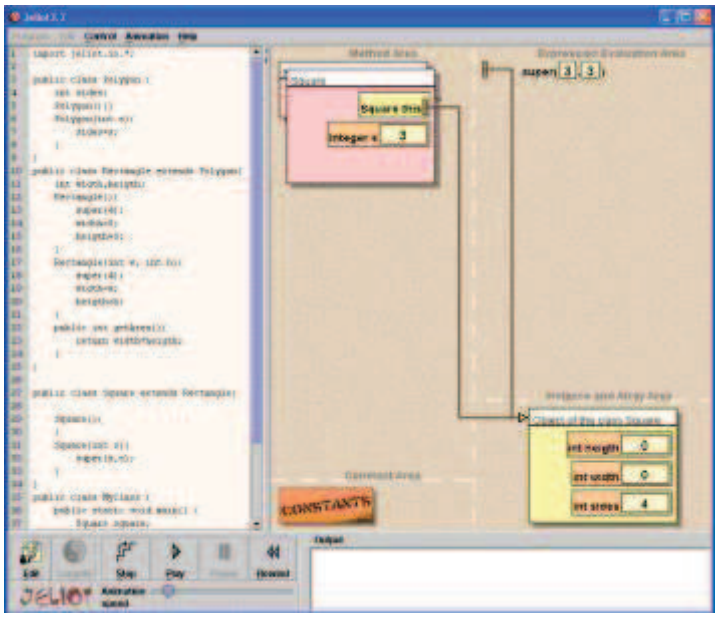


Figura 9. Ventana principal de Jeliot3

Alice [Alice] es un entorno de programación con una interfaz en tres dimensiones (3D), interactiva y animada para crear mundos virtuales, diseñada para programadores principiantes. Alice provee un entorno donde los estudiantes pueden usar y modificar objetos 3D y escriben programas para generar animaciones (véase Figura 10 y Figura 11). Ha sido usada como primer entorno de programación en un curso de programación orientada a objetos, mostrando sus bondades educativas para alumnos que dan ese curso como primer curso de programación [Cooper 03].

Figura 10. Ventana de presentación de Alice

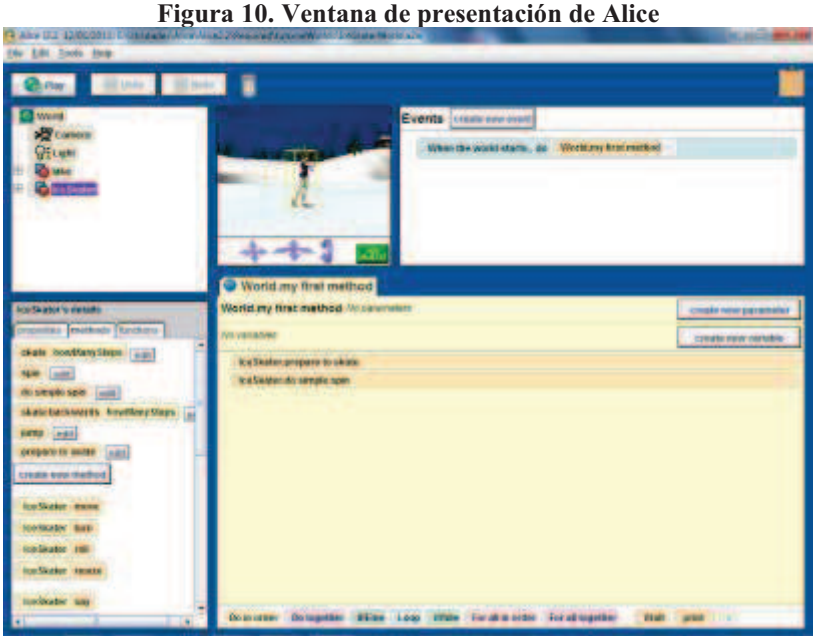


Figura 11. Ventana principal de Alice

Entornos de simulación: Estas herramientas tratan de reflejar la ejecución del código en acciones de habitantes de mundos imaginarios. En esta categoría entrarían Karel the robot [Pattis 95] y sus descendientes, entre los que

destacamos por su relación con la POO, JKarelRobot [Buck 01]. D. Buck y D. Stucki [Buck 01] ampliaron el entorno JKarelRobot para dar soporte a los seis niveles de la taxonomía. Hemos de decir que estos autores son de los pocos que han relacionado la taxonomía de Bloom con su herramienta. También podrían considerarse entornos de simulación los sistemas Alice [Alice], Greenfoot [Greenfoot] y Jeroo [Jeroo] comentados anteriormente.

Herramientas de corrección automática: los correctores o jueces automáticos [Ala-Mutka 05, Douce 05] también pueden ser tipificados como software educativo ya que, aunque no es su uso habitual, pueden permitir a los alumnos que los utilizan aprender ciertas destrezas, si lo usan como tutor de sus prácticas. Podemos citar algunos de los correctores más usados en la actualidad como pueden ser PC^2, Mooshak, RockTest, PC-CAT y Online Judge.

Típicamente, los ejercicios de programación requieren que el estudiante desarrolle un programa que funcione. El material (información) que se entrega al alumno para solucionar el problema puede variar, la simplicidad del problema, el tamaño de la solución, etc. En la literatura podemos encontrar distintas clasificaciones de ejercicios según varios criterios, por ejemplo según la dificultad [Pareja-Flores 00] o según como esté compuesto o según su uso [Edwards 08, Gregorio-Rodríguez 01]. Sin embargo, todas estas clasificaciones tienen el inconveniente que no consideran la taxonomía de Bloom.

De un punto de vista pedagógico, requerir a los estudiantes entregar un programa que funcione tiene consecuencias negativas en su actitud y en el éxito de su aprendizaje. La mayoría de los problemas corresponden al nivel de síntesis de la taxonomía de Bloom, y esa tarea (diseñar el programa) no es trivial. Como consecuencia, el programador novel acomete problemas de gran dificultad desde el principio, una situación que a menudo le frustra y le conduce a abandonar el estudio de ese tema.

Afortunadamente, se pueden eliminar algunas de las deficiencias de este enfoque considerando a la vez la evaluación automática y los niveles educativos. El estudiante puede aprovecharse de la retroalimentación del juez automático para obtener información formativa, y no como se hace normalmente, que se usa esa información para evaluar. Para ello los problemas que se presentan a los alumnos (para ser evaluados por un juez automático) deben ser diseñados de forma apropiada para lograr los niveles de comprensión, aplicación, análisis y síntesis de taxonomía de Bloom.

En [Hernán 08, Hernán 10b], describimos un conjunto de criterios para diseñar problemas de programación adecuados para ser evaluados de forma automática y apropiados para cada uno de los cuatro niveles centrales de taxonomía de Bloom. Un sistema informático con base pedagógica puede extraer y puede mostrar problemas apropiados, de acuerdo a las metas de curso o las etapas en el proceso de aprendizaje. Estos problemas tienen ventajas adicionales: Aumentan la motivación y favorecen la práctica de leer programas como forma de aprendizaje.

Otras herramientas: aquí incluimos software pedagógico que no tiene cabida en ninguna de las categorías anteriores. S. Bridgeman *et al.* realizaron dos sistemas SAIL y PILOT. SAIL [Bridgeman 00b] permite hacer varias versiones del mismo problema, para evitar plagios, y admite soluciones a los ejercicios mediante un fichero auxiliar, pero carece de realimentación con el alumno. PILOT [Bridgeman 00a] es una herramienta que genera ejercicios para un curso de algoritmia e incluye generación de grafos. Es independiente de la plataforma, pues esta escrita en Java. Tiene dos tipos de realimentación: (a) al final del problema, verificando si es correcto o incorrecto, (b) de manera simultánea a la realización del ejercicio, indicando si el alumno se está alejando de la solución a medida que va solucionando el ejercicio propuesto. Este sistema no clasifica los problemas por grado de dificultad, aunque tiene parámetros para ajustar la densidad del grafo.

C. Gregorio *et al.* construyeron un sistema diseñado para archivar y publicar ejercicios de programación. Consta de una colección de documentos estructurados, cada documento describe un ejercicio y varias herramientas para tratarlo. Los documentos están marcados con una extensión de LaTeX diseñada por los autores que permite publicar de forma automática los ejercicios como archivos PostScript o páginas de Web [Gregorio-Rodríguez 01].

Amruth Kumar y su equipo han desarrollado [Kumar 00, Krishna 01, Kumar 02, Dancik 03] varios *applets* (que llama "*problets*") orientados al nivel 3 (aplicación) para conceptos de programación estructurada (ámbito, bucles, memoria dinámica…). Cada *problet* permite generar instancias de un problema, proporciona alguna forma de interacción y visualización para resolver el problema y plantea una pregunta tipo test que el alumno debe responder. Permiten corregir la respuesta inmediatamente y por último, en cada sesión guardan un histórico sobre las preguntas acertadas, parcialmente acertadas y falladas. Existe la opción de generar ejercicios con distintos grados de

dificultad, pero ésta se mide como número de opciones a escoger o como complejidad del problema propuesto.

2.8. Resumen

El diseño y desarrollo de aplicaciones educativas para la enseñanza de la programación orientada a objetos necesita el conocimiento de dos disciplinas aparentemente disjuntas, la pedagogía y la informática. Hemos realizado un estudio sobre la taxonomía de Bloom, de esta forma hemos fijado la base pedagógica, dándole un contexto histórico, en la que nos apoyaremos para que el software a diseñar posea unas buenas características en la vertiente educativa. También hemos repasado el concepto de software educativo, dando un vistazo a las principales aportaciones en el campo de la enseñanza de la programación, centrándonos sobre todo en el paradigma de la programación orientada a objetos. Los tres niveles superiores de la taxonomía de Bloom pueden ser cubiertos de forma apropiada (excepto quizás el nivel de evaluación) por las herramientas existentes, sobre todo por los entornos de programación. Para los niveles inferiores, existen herramientas que pueden ser útiles, sobre todo las que usan la resolución de problemas. El mayor hándicap de estas herramientas es la falta de definición de los objetivos pedagógicos que se pretenden alcanzar. Haciendo este estudio hemos detectado un hueco que este trabajo trata de cubrir: el diseño de herramientas pedagógicas, basadas en la taxonomía de Bloom, para la enseñanza de la programación orientada a objetos, orientada a los niveles inferiores, no se realiza de forma sistemática.

Capítulo 3 Metodología de diseño

En este capítulo se hace un estudio de los tipos de herramientas pedagógicas apropiadas a cada nivel de la taxonomía. Posteriormente se introduce el software educativo a diseñar en el presente trabajo. Nos centramos sobre todo en el que da soporte al aprendizaje basado en problemas, poniendo las bases para el correcto desarrollo de los enunciados orientados a un nivel de Bloom. A continuación se expone la principal aportación de esta tesis: una metodología de diseño del software educativo para la ayuda a la enseñanza de la programación en cualquiera de los tres primeros niveles de la taxonomía de Bloom. Se basa en metodologías de la ingeniería del software pero teniendo en cuenta los aspectos de naturaleza pedagógica que raramente son considerados de forma sistemática a la hora de diseñar aplicaciones educativas. Para finalizar, particularizamos esta metodología para cada uno de los tres primeros niveles de la taxonomía de Bloom. El hecho de limitar esta metodología a los niveles iniciales (justificado en los siguientes apartados) es debido a que creemos que las herramientas (entornos de programación) existentes ya dan soporte a los niveles superiores a excepción, quizás, del nivel de evaluación.

3.1. Relación entre herramientas existentes y la taxonomía de Bloom

Nuestro interés se centra en el diseño y desarrollo sistemático de herramientas educativas basadas en la taxonomía de Bloom. Existen estudios al respecto, por ejemplo L. Fernández *et al.* [Fernández 02] proponen una serie de herramientas existentes para la programación orientada a objetos que sitúan en cada nivel de la taxonomía, bajo nuestro punto de vista, su propuesta no está lo suficientemente justificada. Para dar apoyo a cada uno de los niveles de la jerarquía, identificamos [Hernán 04a] una serie de herramientas informáticas genéricas que describimos y ampliamos aquí:

- El nivel de **conocimiento** puede ser logrado utilizando libros electrónicos, gestores de contenidos, plataformas de enseñanza y resolución de problemas (sobre todo de tipo test). El uso de estas

herramientas permite al alumno la memorización del tema en estudio.

- El nivel de **comprensión** puede ser conseguido con el uso adecuado de las herramientas anteriores y con demostraciones, visualizaciones o simulaciones de programas. El término simulación de programas se refiere a la ejecución de un programa (o fragmento) sustentada por medio de visualizaciones y controles adecuados. Es conocido que la facilidad de interactuar con la demostración determinará el nivel de implicación y por consiguiente de conocimiento del discente [Naps 03a]. El uso de cualquiera de estas herramientas por parte del alumno le conducen a entender y explicar el significado de la información recibida sobre el tema presentado.
- Para el nivel de **aplicación**, es necesaria una herramienta que permita crear o manipular programas (o al menos procese un subconjunto del lenguaje usado para los ejercicios) o un software que permita la resolución de problemas novedosos para el alumno, bien de tipo test o de respuesta libre. Los programas y los ejercicios deben estar relacionados con el tema a tratar. De esta forma, el alumno debe seleccionar y utilizar datos y métodos para solucionar una nueva tarea o un nuevo problema.
- En el nivel de **análisis**, la herramienta debería permitir seleccionar y/o modificar partes de un programa y mostrar los efectos de cada una de ellas. También pueden permitir analizar cada parte de forma flexible, usando algunas funciones de interacción (filtrar de datos, mostrar distintas vistas, cambiar el nivel de abstracción, etc.). Algunas de estas características pueden ser ofrecidas por depuradores, editores de estructuras y visualizadores de programas. Normalmente estas herramientas vienen integradas en los entornos de programación. Su uso adecuado permite al alumno poner de manifiesto cómo se halla estructurado un programa y explicar el modo con que se desarrolla el proceso, de sus consecuencias, de su estado inicial y de su estructura.
- Para el penúltimo nivel, el de **diseño**, las aplicaciones deberían permitir todas las operaciones anteriores además de ayudar en el diseño del software. Los entornos de programación actuales ofrecen ya dichas características. Estas herramientas permiten que el discente pueda crear una nueva solución, manejando diferentes aspectos de conocimiento y métodos, de tal manera que el resultado sea algo más que la suma de los componentes.

- Para el nivel de **evaluación**, la herramienta adecuada sería una que permita comparar de forma simultánea varias soluciones a un mismo problema, pudiendo elegir el usuario el criterio de comparación.

3.2. Tipos de aplicaciones a diseñar

Ya que el alumno es el factor más importante del proceso de enseñanza-aprendizaje, las herramientas deben ayudarle a mejorar, en la medida de lo posible, en este apasionante camino. Esta ayuda puede hacerse desde varias vertientes: mediante la exposición del concepto usando diferentes métodos (textual, gráfico, interactivo), su ejemplificación y la realización de test o problemas prácticos que ayuden a reforzar e interiorizar los conceptos de estudio. Estos aspectos por si mismos tienen la suficiente entidad como para ser el objetivo de una herramienta software de ayuda al estudio. De hecho, así existen en el panorama pedagógico.

Analizando el apartado anterior, se deduce que existen muchas herramientas actuales que dan soporte a los distintos niveles de la taxonomía de Bloom. Por ejemplo, la presentación de contenidos mediante libros electrónicos, plataformas de enseñanza on-line y gestores de contenidos cubren el primer nivel de la taxonomía de Bloom. Los entornos de programación existentes en la actualidad, integran la mayoría de las herramientas citadas para dar soporte a los niveles de análisis y diseño. Existen también simuladores y visualizadores de programas, de hecho es una de las líneas de investigación del grupo LITE [LITE], que pueden ofrecer soporte a los niveles de conocimiento, comprensión y análisis. Para la resolución de problemas también hay soluciones software (gestores de problemas tipo test, *problets* [Kumar 00]) que pueden cubrir los tres primeros niveles de Bloom. En ningún artículo analizado se muestra de forma clara como se pueden diseñar estas herramientas a un nivel concreto de la taxonomía de Bloom.

En el marco de este trabajo, por lo tanto, nos centraremos en las herramientas algorítmicas que presenten materiales o problemas que den apoyo a un determinado nivel de los tres primeros de la taxonomía de Bloom. De acuerdo a los tipos de software educativo presentados en el capítulo anterior, diseñaremos herramientas o sistemas tutoriales y fijaremos las reglas para el diseño de aplicaciones de uso individual y sin necesidad de realimentación o ayuda por parte del profesor o tutor. Este software educativo debe poder usarse en cualquier lugar (de forma general con acceso a internet), para que el alumno pueda adecuar de forma personal su ritmo de estudio y de práctica.

3.3. Características de las herramientas educativas

En esta sección proponemos una serie de características que deberían tener las herramientas educativas basadas en la taxonomía de Bloom. El primer grupo de propiedades ayudan a fijar el nivel de Bloom. El segundo conjunto son características que deberían tener el software educativo para incrementar su potencial en el ámbito de la enseñanza-aprendizaje.

3.3.1. Relacionadas con la taxonomía de Bloom

Característica 1. *La herramienta debe centrarse en uno o pocos conceptos*.

La mayor dificultad es la complejidad intrínseca del dominio de la programación. Nosotros podemos intentar pensar en una formidable herramienta que de soporte a todos los conceptos involucrados en una materia. Esta herramienta podría ser un entorno de programación, que es una aplicación de propósito general. Sin embargo, ¿en qué nivel de Bloom podríamos clasificar esta herramienta? La respuesta es fácil: Depende del uso que hagamos de ella. Si la usamos para implementar un algoritmo dado en un curso de algoritmia, podríamos fijarla en el nivel de comprensión. Sin embargo si la usamos para desarrollar un programa que resuelve un problema nuevo, podría estar en el nivel de aplicación o incluso en un nivel superior. Por lo tanto, sin restringimos el alcance de la herramienta facilitaremos la identificación del nivel que consigue alcanzar, así como su diseño e implementación.

Característica 2. *Una herramienta puede dar soporte de igual manera a la teoría y a la práctica.*

Los niveles de la taxonomía de Bloom no diferencian entre teoría y práctica. De hecho, en la revisión de la teoría o teoría revisada [Anderson 01] se divide en la dimensión del conocimiento, basada en la materia que se pretende enseñar, y la dimensión del proceso cognitivo, fundamentada en el verbo o acción que se quiere conseguir enseñar. Por lo tanto, la teoría es necesaria para todos los niveles de la taxonomía. Por otra parte, es difícil alcanzar los niveles altos sin resolver problemas.

Como consecuencia, una herramienta educativa debería poder dar soporte a ambos aspectos, el teórico y el práctico. La importancia relativa de cada uno varía, dando lugar a diferentes aplicaciones. Por ejemplo, para dar soporte a la teoría se puede diseñar un libro electrónico [Martínez 02].

La decisión de ofrecer teoría o práctica o ambas debe ser pragmática. Por ejemplo, los recursos disponibles para desarrollar una herramienta (incluidos el tiempo necesario para escribir la teoría) pueden ser escasos. O pueden existir otros recursos externos que den soporte de manera satisfactoria a alguno de los dos aspectos.

Característica 3. *Una herramienta puede ofrecer ayuda para alcanzar distintos niveles de la Taxonomía de Bloom*.

Si bien es cierto que teniendo una herramienta centrada en pocos conceptos (característica 1) favorece la clasificación dentro de la jerarquía, la aplicación educativa debería soportar los niveles inferiores, para que el alumno fuera capaz de conseguir alcanzar un nivel determinado de manera incremental. Esta propiedad no es esencial, pues los niveles básicos de la taxonomía se pueden alcanzar por medios externos a la herramienta. En concreto, si nos centramos en un curso de programación orientada a objetos y a un determinado concepto, por ejemplo, la herencia, los niveles de conocimiento y de comprensión se pueden alcanzar atendiendo a las clases impartidas por el profesor o leyendo un libro o consultando alguna herramienta pedagógica existente.

3.3.2. Relacionadas con la eficacia pedagógica

Estas propiedades no están relacionadas con la taxonomía, pero no podemos olvidarnos de ellas cuando diseñamos herramientas de ayuda a la enseñanza interactivas. Estas características han sido extraídas sobre todo de los estudios sobre el uso educativo de programas y animaciones de algoritmos [Naps 03a, Naps 03b].

Característica 4. *La herramienta debe involucrar al alumno.*

El software educativo debería aumentar la motivación del alumno sobre el concepto en estudio y además debería ser eficaz pedagógicamente hablando. Para conseguir estos objetivos, la herramienta debe involucrar al discente. En algunas ocasiones el programa puede forzar al alumno a realizar alguna tarea (por ejemplo, a resolver correctamente un ejercicio) antes de presentarle la próxima actividad. En otras, debe permitir al alumno llevar su propio ritmo de aprendizaje y por lo tanto, decidir sobre las actividades que desea realizar.

Otros métodos para involucrar al usuario son menos directos. Por ejemplo, dar al alumno información clara y concreta sobre los ejercicios a realizar puede incrementar su seguridad y provocar una mejora ostensible en el proceso de aprendizaje (ya que los conceptos son aprendidos mejor y los errores son detectados más eficientemente). Al aumentar la seguridad sobre lo que está haciendo, el alumno seguirá trabajando con la herramienta.

Característica 5. *La herramienta debe ser fácil de usar para el profesor*.

Uno de los problemas con algún software educativo existente es la dificultad de su uso y su adaptación a una materia dada. Para realizar un buen diseño se debe tener en cuenta una serie de consideraciones [Naps 03b], como por ejemplo, la dificultad para desarrollar nuevos materiales, la adaptación del material existente a un nuevo curso, etc. Por lo tanto, la herramienta debería dar la posibilidad de cambiar la lección o el ejercicio sin un esfuerzo considerable o sin conocimientos técnicos avanzados.

Característica 6. *La herramienta debe incluir elementos hipermedia que ayuden al éxito en la comprensión de la teoría y la realización de problemas.*

Es un hecho conocido que el uso de representaciones externas diferentes de las textuales incrementan la motivación del alumno. De hecho, las explicaciones son complementadas frecuentemente con gráficos, ecuaciones, tablas, figuras, etc. En programación, los elementos hipermedia más usados son las visualizaciones de programas y algoritmos. Recordemos que el uso de cualquier representación externa diferente de la textual requiere una explicación explícita [Ainsworth 99] para su correcta comprensión.

Una vez fijado el tipo de software educativo basado en la taxonomía de Bloom a diseñar (herramientas algorítmicas que presenten materiales o problemas que den apoyo a un determinado nivel de los tres primeros de la taxonomía de Bloom) y conocidas las características que deben tener, pasamos a introducir las bases para producir de forma correcta los problemas apropiados a cada concepto a estudiar.

3.4. Resolución de problemas

La resolución de problemas es una forma de aprendizaje que tiene su base pedagógica que pasamos a explicar. El aprendizaje basado en problemas (*Problem-Based Learning*, PBL) se fundamenta en la resolución de problemas por el alumno [Barrows 80]. Esta técnica mejora la retención de los conceptos a largo tiempo [Farnsworth 94] al obligar al alumno, no sólo a comprender, sino a aplicar o razonar sobre el concepto aprendido. Además, hace que el alumno practique el aprendizaje activo, lo que conlleva indudables beneficios [McConnell 96] para el discente: el alumno consigue un mayor grado de comprensión, se incrementa la sensación de confort con la asignatura y de confianza en sí mismo.

Una forma en la que la informática puede dar soporte al PBL es mediante la presentación y corrección automática de problemas a resolver por el alumno. El aprendizaje basado en problemas mejora la retención a largo plazo al obligar al alumno, no sólo a comprender sino a aplicar o razonar sobre el concepto aprendido. Para que este método tenga éxito el alumno debe obtener una respuesta rápida sobre la corrección de la solución hallada [Farnsworth 94]. Entre la información que al menos debe recibir el alumno en el proceso de corrección debe estar su respuesta, la respuesta correcta y una explicación de la pregunta y de la respuesta correcta. Si el alumno realiza estos problemas fuera del aula la realimentación sobre la solución no la tendrá disponible hasta que el tutor la publique. Las herramientas pedagógicas pueden jugar un papel fundamental en este marco ya que el estudiante puede disponer de ellas en cualquier momento para consultarlas o usarlas.

Un tipo de problemas que se pueden corregir de forma automática y fácilmente son los de tipo test. Estos problemas son adecuados para los primeros niveles de la taxonomía. Sus ventajas son conocidas: fácil corrección y objetividad en la misma. Además está comprobado experimentalmente que pueden ser utilizados como método de aprendizaje. Se conoce como efecto test

al fenómeno que supone que hacer una prueba de memoria no solo evalúa lo que ya se sabe sino que también incrementa la capacidad de recuerdo a largo plazo [Mcdaniel 07]. De hecho existen experimentos que dicen que la realización de test provoca un mayor aprendizaje que la relectura del material y los test con realimentación mejores resultados que los que no dan esa realimentación [Macdaniel 07]. Desde nuestro punto de vista, la resolución de problemas mediante test posee un inconveniente: la solución es una de las opciones. Si las posibles respuestas no están bien diseñadas el alumno puede contestar por aproximación o por eliminación de posibilidades.

Un remedio consiste en diseñar las preguntas y las opciones de manera que se minimice el inconveniente [Lister 01], pero no es una tarea trivial ni es siempre posible. Otra alternativa consiste en que el alumno escriba su solución como una respuesta abierta o, al menos, semiabierta (con opciones limitadas pero no prefijadas). Existen diversos métodos [Pérez 06] para evaluar automáticamente preguntas abiertas. Desde un punto de vista técnico se basan en técnicas de procesamiento de lenguaje natural combinadas con otras técnicas, como son extracción de información, análisis sintáctico, reconocimiento de patrones, redes bayesianas y árboles de decisión, estadísticas, etc. Tienen la ventaja de su mayor flexibilidad y naturalidad para el usuario, pero su fiabilidad no es absoluta.

El uso de preguntas semiabiertas es una opción menos flexible pero, si pueden corregirse automáticamente, tiene la ventaja de ser totalmente fiable y subjetiva. La corrección puede realizarse mediante una cadena de texto fija o mediante expresiones regulares. La primera técnica es poco flexible (compara carácter a carácter y por lo tanto, no permite ninguna variación) aunque rápida de implementar. La segunda, sin embargo, permite comparar la respuesta del usuario con un patrón, con lo que da un mayor repertorio a la hora de corregir si una respuesta semiabierta es correcta o no.

Pueden adoptarse varias enfoques, pero la que nos parece natural es la consistente en pedir al usuario trozos de código correctos, ya que es una aproximación al aprendizaje basada en los tres primeros niveles de la taxonomía de Bloom.

3.5. Bases para la producción de preguntas adecuadas a un nivel de Bloom

Tener un método con ejemplos [Hernán 11d] de cómo diseñar correctamente preguntas para ayudar al alumno a alcanzar un nivel puede ser útil al profesor o investigador a la hora de preparar una evaluación o un experimento. Existen

multitud de clasificaciones de evaluaciones basadas en la taxonomía de Bloom [Scott 03, Lister 03], pero todas ellas están basadas en las apreciaciones de los autores. De hecho, es bien sabido que adjudicar una pregunta a un nivel determinado no depende de la pregunta en sí, sino del alumno al que se le presenta. Es decir, la misma pregunta puede situarse en distintos niveles dependiendo del grado de conocimiento previo del discente. Es muy importante, a la hora de evaluar una pregunta, preguntarnos qué conocimientos y qué habilidades han usado los estudiantes para contestar correctamente [Deimel 90]. En los siguientes párrafos se trata de proporcionar un método para conseguir un mayor grado de objetividad a la realización de preguntas para evaluar. Para ello se explican los pasos y se ejemplifica con cuestiones centradas en el aprendizaje del concepto de herencia simple en programación orientada objetos usando el lenguaje Java para implementar dicho concepto.

Por otro lado, la evaluación del conocimiento de un alumno también es complicada. De forma habitual, se realizan pruebas que constan de varios ejercicios. A estos problemas se les puntúa de forma homogénea, si son de dificultad parecida, o de forma heterogénea a criterio del profesor/evaluador. La suma (ponderada) de los puntos da una nota numérica que sirve como indicativo del grado de conocimiento del alumno en esa materia a la vez que puede servir para establecer una clasificación entre los discentes: los de mayor nota son mejores estudiantes que los que obtienen una nota inferior. La clave para obtener una buena nota indicativa, está en evaluar la dificultad. Para realizar esta tarea de forma lo más subjetiva posible se puede usar la taxonomía de Bloom [Bloom 56].

Una metodología adecuada de preparar las preguntas para evaluaciones y/o experimentos nos la da la investigación social. En los siguientes apartados se explica esta metodología y se aplica a un concepto clave en la programación orientada a objetos: la herencia.

3.5.1. Dimensiones, indicadores e índices

Las dimensiones de un concepto son los distintos aspectos en que puede ser considerado el mismo. Las dimensiones que consideremos deben ser mensurables de forma que por medio de medidas (indicadores) se haga operativo el concepto. Generalmente hay una pérdida de información al descomponer un todo en sus partes y analizar esas partes por separado. Por ello hay que operar de forma que las medidas (indicadores) de las dimensiones reflejen lo más ajustadamente posible el concepto del que partimos [Carmona 77].

Un indicador es algo que da señal o cuenta de algo concretándolo. Se puede refinar esta definición diciendo que un indicador es la medida estadística de un concepto o de una dimensión de un concepto o de una parte de aquélla, basado en un análisis teórico previo e integrado en un sistema coherente de medidas semejantes, que sirve para describir el concepto es estudio [Carmona 77]. Los indicadores deben tener dos propiedades fundamentales: estar relacionados con el concepto o dimensión del que tratan de ser indicación y ser expresión numérica, cuantitativa, de la dimensión que reflejan. Existen distintos tipos de indicadores [De Miguel 67], de los que tomaremos únicamente los descriptivos, que son aquellos que tratan de poner de manifiesto o explicar la posible regularidad existente en un conjunto de datos.

Un índice es una medida obtenida por la agrupación adecuada de varios indicadores. Si los indicadores que se usan pertenecen todos a una misma dimensión, ese índice representará numéricamente la dimensión medida (véase Figura 12 , i_1). Pueden existir índices de indicadores de varias dimensiones [García 03], y este índice representará al conjunto de dimensiones (i_4). Un índice también puede representar al concepto en estudio (i_G). Aquí aplicaremos estos conceptos al estudio de la herencia simple en programación orientada a objetos.

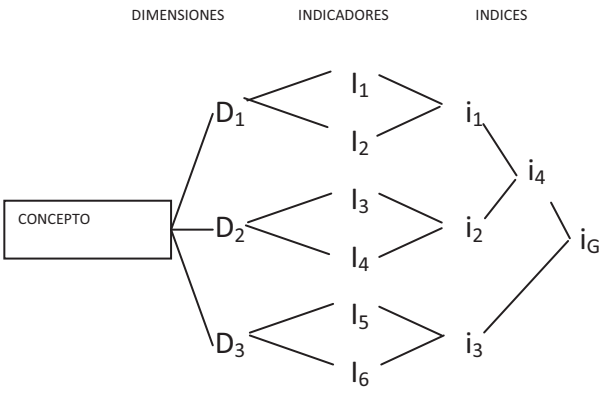


Figura 12. Dimensiones, indicadores e índices

3.5.2. Aplicación del método al concepto de herencia simple

Nuestro interés se centra en poder preparar preguntas para ayudar a alcanzar un determinado nivel y/o evaluar si el alumno ha conseguido esos conocimientos. Si echamos un vistazo a los exámenes de una asignatura de POO, esta evaluación, en la actualidad, se suele hacer de manera vaga. Las cuestiones planteadas al alumno suelen englobar diversos conceptos y diversos

métodos que se aplican sobre esos conceptos. El alumno debe usar todo su conocimiento y su pericia para resolver problemas y por tanto para superar el examen con éxito. Para intentar evitar que el discente falle la resolución de un problema por el desconocimiento de una parte de él, proponemos la división de los conceptos en sus dimensiones. De esta forma, un alumno podrá conocer qué es en lo que realmente anda flojo y el profesor podrá valorar, de una manera más objetiva, el conocimiento global del alumno. Si se trata de diseñar preguntas, son aplicables las mismas consideraciones. En concreto, vamos a ejemplificar este método con el concepto de herencia simple.

En general, una asignatura o materia experimental, podemos dividirla en dos partes. Primero, la parte teórica esencial, donde se enseñan los conocimientos (terminologías, conceptos, hechos, clasificaciones, etc.) y los métodos (leyes, modos, formas, etc.) propios de la materia en cuestión. Segundo, la parte práctica, donde se aplican los conocimientos teóricos para resolver problemas.

3.5.3. Dimensiones del concepto de Herencia

La herencia en programación orientada objetos, como se comenta en el párrafo anterior, se puede dividir en dos dimensiones: (D_1) la parte teórica relacionada con el concepto y (D_2) la parte práctica de dicho concepto. Podemos refinar a su vez D_2 en dos facetas o subdimensiones: (D_{21}) el diseño de clases con herencia y (D_{22}) su implementación en un lenguaje concreto.

1) El conocimiento del concepto (D_1)
2) El uso de la herencia en POO: (D_2)
    a. En el diseño de clases con herencia (D_{21})
    b. En la implementación de la herencia (D_{22})

3.5.4. Indicadores del concepto de herencia

Construir indicadores no es sencillo. Hay que tener en cuenta que precisa un trabajo previo de diseñar lo que vamos a investigar, de extraer sus características y sus problemas. Esto nos dará un boceto inicial que se refina con la experiencia. Es decir, normalmente realizar indicadores será un proceso iterativo, que se basa en los conocimientos del investigador, en la observación de experimentos previos y de experiencias actuales, que guiarán esta construcción.

Los indicadores propuestos, tras un estudio del concepto de herencia y de la experiencia docente como profesor de la asignatura Programación Orientada a Objetos, para cada una de las dimensiones explicadas son:

Conocimiento del concepto D_1

Como indicadores del conocimiento del concepto propondremos:

Grado de recuerdo del concepto de herencia. (I_1)

Grado de comprensión del concepto y sus implicaciones. (I_2)

Uso del concepto D_2

Diseño de clases con herencia D_{21}

Un indicador de cómo el alumno es capaz de aplicar el conocimiento de la herencia, para realizar un diseño puede ser:

Medida del uso correcto de la herencia en el diseño (I_3)

Implementación de clases con herencia (con un lenguaje de POO) D_{22}

El indicador de cómo el discente es capaz de usar el conocimiento de la herencia para resolver un problema mediante un programa puede ser:

Medida de la utilización de instrucciones y mecanismos de la herencia en un lenguaje de programación concreto. (I_4)

3.5.5. Ejemplos

Los ejemplos sirven de guía práctica para los investigadores que quieran usar esta técnica. En este caso, se muestra una única pregunta por indicador, pero hay que tener en cuenta que para que la obtención de datos sea correcta, debemos dar un número de preguntas suficiente. Esto permitirá conseguir unas medidas más precisas y minimizar el efecto no deseable que pueden producir otras variables ocultas (como puede ser el diseño poco adecuado o poco cuidadoso de las preguntas).

INDICADOR I_1

Las preguntas para el indicador (I_1) del grado de recuerdo del concepto de herencia deben ser tipo test, formuladas de manera similar a como se presenta el concepto con la metodología impartida o en la herramienta utilizada. Estas preguntas han de ser puntuadas para obtener un valor numérico. Un ejemplo de pregunta con varias respuestas posibles para este indicador es:

Ejemplo (I_1):

Si B hereda de A entonces es cierto que B puede:

a) añadir nuevos atributos
b) redefinir métodos
c) añadir nuevos métodos
d) eliminar atributos.

INDICADOR I_2

Para el indicador (I_2) del grado de comprensión de la herencia en POO, se deben plantear preguntas que el alumno pueda responder razonando sobre la información recibida. En este caso las preguntas pueden ser tipo test o preguntas semiabiertas. El número y el contenido de las preguntas deben ser tal que permitan al investigador o al profesor tener la suficiente confianza en que el alumno que obtenga una puntuación media-alta comprende lo que es y cómo funciona la herencia. Bajo nuestro punto de vista, los contenidos a cubrir en estas preguntas deberían ser:

- Necesidad de la herencia. Hacer ver que las clases no son suficientes para conseguir reutilización y extensibilidad.
- Relaciones entre clases: extensión, especialización y combinación.
- Herencia como solución a: representar esas relaciones y posibilitar la creación de una clase a partir de otra.
- Consecuencias de la herencia: incorporación de atributos y métodos de la clase padre, sustitución de objetos de subclases.
- Posibilidades de la herencia: adicionar nuevos atributos y métodos, redefinición de métodos (refinando o reemplazando).
- Tipos de herencia simple:
    - Especialización (la clase hija es un subtipo de la clase padre)
    - Especificación (la clase padre define comportamientos que son implementados por la clase hija)
    - Construcción (la clase hija usa el comportamiento proporcionado por la clase padre pero no es un subtipo de ella)
    - Extensión (la clase hija añade nuevas funcionalidades a la clase padre, pero sin cambiar el comportamiento heredado)
    - Limitación (la clase hija restringe el uso de algún comportamiento de la clase padre)

Un ejemplo de pregunta para este indicador es:

Ejemplo (I_2):

Si un Rectángulo es un tipo especial de Polígono, el tipo de herencia que usaríamos para diseñar la clase Rectángulo a partir de la clase Polígono será:

a) Herencia por construcción
b) Herencia por especialización
c) Herencia por especificación
d) Herencia por limitación

INDICADOR I_3

El indicador (I_3) de la medida del uso correcto de la herencia en el diseño podemos obtenerlo generando preguntas donde el alumno se tenga que enfrentar al diseño de una aplicación nueva, sin referencias anteriores. Para detectar la herencia durante el diseño, el discente debe ser capaz de extraer información del enunciado y ver si existen clases con comportamientos comunes (generalización) o si una clase es un caso especial de otra (especialización). Un ejemplo de pregunta en este indicador es:

Ejemplo (I3):

Sean los tres siguientes conceptos geométricos:

Línea recta (infinita en ambas direcciones);

Rayo (sale de un punto fijo y es infinito en una dirección); Segmento (es una porción de una línea con dos puntos finales fijos)

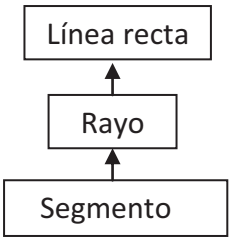
¿Cómo podría representar esos tres conceptos en una jerarquía de herencia?

INDICADOR I_4

El indicador (I_4) de medida de la utilización de instrucciones y mecanismos de la herencia en un lenguaje concreto, debe dar la destreza del alumno en el uso del lenguaje para implementar los distintos tipos de herencia. Para ello, las preguntas de este indicador deben estar orientadas a la escritura de un programa (o fragmento) con un lenguaje de programación particular.

Ejemplo (I_4):

Dada la siguiente jerarquía de herencia:



¿Cómo podría implementar en JAVA estas tres clases, sabiendo que una línea recta se puede identificar por dos puntos que atraviesa, un rayo es una recta con un punto inicial (o final) y un segmento es una recta limitada por dos puntos (inicial y final)?

3.5.6. Índices

En primer lugar vamos a fijar un índice para cada indicador. El índice (i_1) para el indicador (I_1) puede ser calculado de la siguiente forma:

- Si las preguntas son tipo test con cuatro opciones, de las cuales solo una es correcta, se calcula sumando un valor *a* (normalmente el 100% del valor de la pregunta) para la respuesta acertada, restando b (normalmente 1/número_de_opciones) para las falladas (para eliminar el azar) y cero para las no contestadas. En general, el índice (i_1) se calculará aplicando la siguiente fórmula:

$$i_1 = \frac{a*n - m*b}{t} \quad (i_1)$$

  Siendo *n* el número de respuestas acertadas y *m* el número de respuestas falladas y *t* el número total de preguntas.

- Si las preguntas son semiabiertas, la puntuación dependerá del criterio del profesor, pudiendo obtenerse un índice similar al anterior.

Este índice (i_1) caracterizará el grado de recuerdo del concepto de herencia que el alumno posee.

Para el índice (i_2) se puede hacer un análisis similar al anterior, caracterizando de nuevo la dimensión que representa el indicador (I_2).

$$i_2 = \frac{a*n - m*b}{t} \quad (i_2)$$

Uniendo ambos índices podemos formar el índice (i_{1_2}) para sacar conclusiones sobre la dimensión D_1. Este índice se puede definir como la media de los índices i_1 e i_2. Por lo tanto, tendremos un valor indicativo de esta dimensión.

$$i_{1\_2} = \frac{i_1 + i_2}{2} \quad (i_{1\_2})$$

Análogos razonamientos se pueden hacer para los índices (i_3, i_4). Teniendo en cuenta que en ambos se plantean preguntas abiertas, se puede puntuar cada respuesta de forma que se admitan cinco posibles valores (por ejemplo: perfecto, bien, regular, bastantes fallos, mal), asignando un peso a cada categoría (por ejemplo: perfecto = 4, bien = 3, regular = 2, bastantes fallos = 1, mal = 0). En este caso, los índices se podrían calcular con la fórmula:

$$i_3 = \frac{v_1 * p_1 + v_2 * p_2 + v_3 * p_3 + v_4 * p_4 + v_5 * p_5}{t}$$

$$i_4 = \frac{v_1 * p_1 + v_2 * p_2 + v_3 * p_3 + v_4 * p_4 + v_5 * p_5}{t}$$

Siendo v_1 el número de respuestas perfectas, v_2 el número de respuestas bien, v_3 el número de respuestas regulares, v_4 el número de respuestas con bastantes fallos, v_5 el número de respuestas mal contestadas, p_i el peso de cada categoría y t el número de preguntas.

El índice (i_{3_4}) construido como la media aritmética de i_3 e i_4 caracterizará la dimensión del uso de la herencia es sus dos vertientes (diseño e implementación).

$$i_{3\_4} = \frac{i_3 + i_4}{2} \quad (i_{3\_4})$$

Observando las dimensiones obtenidas usando el método de investigación social, se puede ver cierto paralelismo con los niveles de la jerarquía de Bloom. La dimensión D_1 (conocimiento del concepto) junto con sus indicadores se puede asociar a los dos primeros niveles de Bloom (recuerdo y comprensión). D_2 (uso del concepto) se puede asociar a los dos niveles

siguientes (aplicación y análisis). Debido a esta relación se puede intentar sacar un índice general (i_G) que sea la media de i_{1_2} e i_{3_4}.

$$i_G = \frac{i_{1\_2} + i_{3\_4}}{2} \quad (i_G \text{ Ver.1})$$

Con este índice obtendríamos teóricamente el grado de sabiduría del alumno respecto del concepto de herencia (en cuanto a su conocimiento y a su uso). Hipotéticamente un valor de i_G menor correspondería a un alumno con poco grado de conocimiento y un valor correspondería a un alumno con un alto grado de conocimiento del concepto de herencia.

Pero aquí surge nuestra duda: ¿se puede usar ese índice para medir el grado de conocimiento según la taxonomía de Bloom? O formulada desde otra perspectiva ¿el alumno que obtenga un mayor índice general (i_G) se encuentra en un nivel más alto de la taxonomía de Bloom? Nuestra experiencia dice que no. En varios experimentos con la enseñanza de la programación se ha encontrado a alumnos que son capaces de (intuitivamente) resolver problemas del nivel de aplicación en la taxonomía de Bloom teniendo bajo porcentaje de aciertos en las preguntas de los niveles inferiores [Hernán 07b] y al contrario [Lathinen 07]. No es difícil imaginar en un alumno que haya estudiado y aprendido los conceptos de memoria, que comprenda las distintas relaciones que pueden dar lugar a la herencia simple, y que obtenga una puntuación del índice general mayor (al obtener un índice i_{1_2} muy alto –por ejemplo un 9- y un índice i_{3_4} bajo –por ejemplo un 3- obtendría un índice general de 6) que otro alumno que sepa medianamente aplicar e implementar esas relaciones pero no recuerde el nombre de dichas relaciones y por tanto, obtenga un valor menor de dicho índice (por ejemplo con índice i_{1_2} de 3 y un índice i_{3_4} de 7 obtendría un índice general de 5).

La contradicción encontrada puede dar lugar a reformular el cálculo del índice general, añadiendo pesos p_{1_2} e p_{3_4} a cada índice i_{1_2} e i_{3_4} de forma que el índice i_{1_2} tenga menos peso que el índice i_{3_4}. De esta manera podemos compensar los casos presentados en el párrafo anterior.

$$i_G = \frac{i_{1\_2} * p_{1\_2} + i_{3\_4} * p_{3\_4}}{2} \quad (i_G \text{ ver. 2})$$

3.6. Metodología general de diseño

3.6.1. Introducción

Se debe pensar en las aplicaciones educativas como complemento a otros métodos instruccionales basados en el uso de otros medios (clases presenciales, información impresa, información audio-visual, etc.), teniendo claro el rol de cada uno de los medios educativos seleccionados y la viabilidad de usarlos. Hemos de hacer notar que el diseño no muestra con exactitud lo que hemos de hacer, sino que nos señala las direcciones que hemos de seguir para conseguir un producto adecuado. La metodología que proponemos a continuación se debe aplicar, como ya hemos comentado, al diseño de herramientas algorítmicas que presenten materiales o problemas que den apoyo a un determinado nivel de los tres primeros de la taxonomía de Bloom

3.6.2. Metodología general de diseño propuesta

En el diseño de una aplicación de ayuda a la enseñanza basada en la taxonomía de Bloom se distinguen los siguientes pasos (véase Figura 13).

Primera fase: Análisis de requisitos o necesidades docentes

En esta etapa se trata, fundamentalmente, de definir el tema a enseñar y la profundidad con la que se quiere tratar (nivel de la taxonomía de Bloom al que se va a dar soporte). Es decir, se fija la intencionalidad de la herramienta pedagógica a diseñar. Al definir un nivel de la taxonomía de Bloom estamos declarando el estado final que queremos que los alumnos alcancen. También es importante fijar los posibles escenarios de interacción que tendrá el usuario.

Una de las mayores dificultades para diseñar una herramienta en un determinado nivel de Bloom es la complejidad intrínseca de programación [Hernán 04a]. La programación es una tarea multidisciplinaria, donde se mezclan varias áreas de conocimiento. Como primer intento podemos pensar en desarrollar herramientas ambiciosas que pueden soportar todos los conceptos y métodos de un asunto dado de programación, como pueden ser, por ejemplo, los entornos de programación, los cuales son herramientas de uso general. Sin embargo, restringir el alcance de una herramienta [Hernán 06a] y por lo tanto su dominio, hace más fácil identificar claramente el nivel de

taxonomía de Bloom a la que está orientada, así como diseñar e implementar sus prestaciones. La interacción con la aplicación se hará con un ordenador o con un dispositivo móvil, con las características adecuadas para mostrar información multimedia o interactuar con el usuario de forma simple e individual.

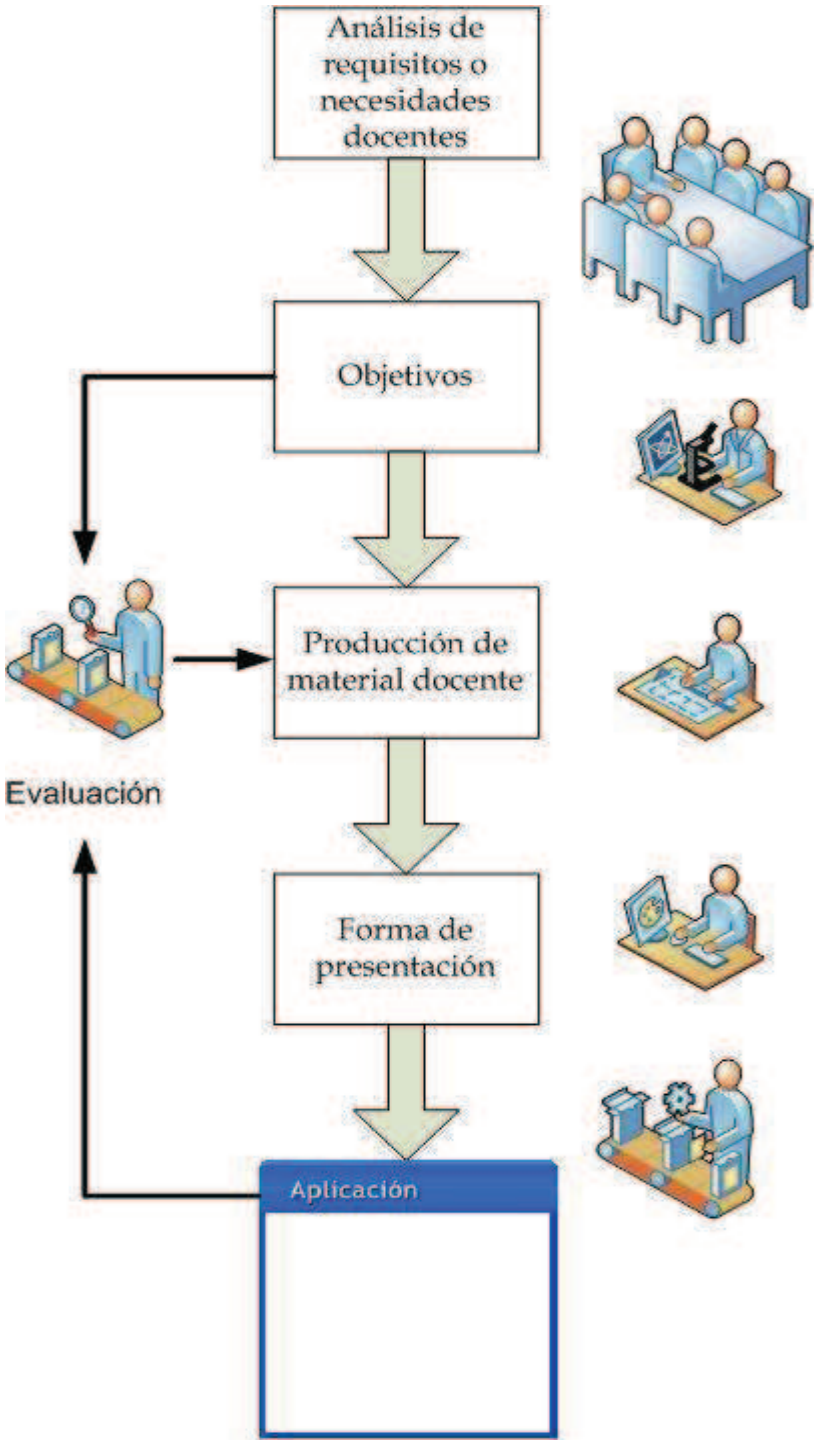


Figura 13. Metodología de diseño de software educativo

Segunda fase: Objetivos

En los objetivos se definen con claridad los requisitos del docente. Aquí deben quedar muy claros cuáles son los conocimientos previos que pueden ser asumidos, qué tema debe incluirse y qué tipo de ejercicios son adecuados para el nivel de Bloom definido en la primera fase.

Esta fase es la más importante. Primero se definen los conocimientos previos del alumnado. Esta tarea debe realizarla el profesor, ya que es la persona que ha seguido la evolución de sus pupilos. De esta forma se sitúa aproximadamente el estado de partida de los discentes. En segundo lugar se definen las estrategias a emplear. Conocidos los puntos de partida y de llegada, se deben configurar la estructura de la aplicación para conseguir guiar al alumno a conseguir la meta. En el marco de esta tesis, los objetivos se fijaran en términos del nivel de la taxonomía de Bloom al que se quiere dar soporte. La meta final es que el alumno logre alcanzar o afianzar los conocimientos propios del nivel correspondiente.

Tercera fase: Producción de material docente

Se entiende como material docente la información a mostrar o las preguntas a realizar al alumno. Como la aplicación debe tener un enfoque práctico de la materia o aspecto a estudiar o resolver, debe realizarse una generación de materiales que apoyen este nivel. También debe incluirse información de retroalimentación al alumno, de tal forma que sea inmediata, para una mejor asimilación de los contenidos aprendidos. Esta realimentación deberá mostrar, al menos, la acción o respuesta del alumno (en caso de que no sea visible), independientemente de que sea correcta o no, la corrección de la respuesta dada así como información sobre la respuesta correcta.

Esta fase tiene que ser acorde con las dos anteriores, de tal forma que los contenidos o ejercicios producidos sean adecuados a los niveles de la taxonomía

fijada en la primera fase y al conocimiento previo del alumno fijado en la segunda. En el caso de que el material docente sean ejercicios prácticos, para que esta fase tenga éxito, el alumno debe obtener una respuesta rápida sobre la corrección de la solución dada [Farnsworth 94].

Richard Felder hizo un estudio [Felder 96] sobre los cuatro modelos de estilos de aprendizaje que existían (activo/reflexivo, sensorial/intuitivo, visual/verbal, secuencial/global) llegando a la conclusión que los cuatro son útiles y reúnen todas las necesidades que tienen los diferentes tipos de alumnos. Según Felder: "Desempeñar de manera efectiva cualquier profesión, sin embargo, requiere que se trabaje bien con todos los estilos de aprendizaje. Si el profesor enseña exclusivamente de una manera que favorece a un tipo de estudiantes, el nivel de disconformidad puede crecer e interferir en el aprendizaje. Por otro lado, si el profesor explica en el modo preferido por los estudiantes, estos pueden no desarrollar la destreza mental que necesitan para sus estudios o profesiones." Un objetivo de la educación es ayudar a que los alumnos construyan sus habilidades con todos los estilos de aprendizaje. Felder hace notar que los estudios de Ingeniería en la pasada década habían estado dirigidos a los estilos de aprendizaje intuitivos, verbales, reflexivos y secuenciales y que pocos estudiantes se incluyen dentro de esas categorías. Existe una estrecha relación entre el estilo de aprendizaje y el de enseñanza. Esto puede desembocar en el desinterés y apatía de los estudiantes en grado menor y en el abandono de los estudios en el caso extremo, haciendo que alguien que puede contribuir de manera importante a una profesión se quede fuera, por lo que es necesario que se incorporen todos los estilos de aprendizaje a la hora de enseñar.

En [Thomas 02] se realiza un experimento en un curso de introducción a la programación con Java. Primero clasificaron a los alumnos según el estilo de aprendizaje de Felder-Silverman preferido y luego estudiaron las notas de clase y del examen. Se observa que los aprendices reflexivos obtienen mejor nota que los activos; ocurre lo mismo entre los verbales (mejor nota) y los visuales. En base a estos resultados, los autores de extendieron sus estudios a grupos de alumnos, tratando de establecer relaciones entre los distintos grupos (combinaciones óptimas, etc.). Se destaca que los aprendices activos, juiciosos, visuales y globales están en desventaja debido fundamentalmente a la forma de explicar en Ingeniería.

Nuestro objetivo es ayudar al mayor rango de alumnos posibles, por lo que nuestra metodología debe tener en cuenta los resultados de los estudios previos y generar material docente (contenidos o problemas) de forma que se presente la información a aprender de manera tanto textual como gráfica, lo

cual motiva a los alumnos que según Thomas et al. [Thomas 02] se encuentran desfavorecidos por la manera de impartir las clases.

Cuarta fase: Forma de presentación

Este paso es importante en este proceso, ya que aquí se conforma todas las fases anteriores, produciendo el deseado efecto didáctico. Se responde a las pregunta ¿Qué aspectos deben aparecer, en qué orden y cómo se plantearán? Para el diseño de la interfaz se deben seguir los principios de diseño de interfaces gráficas: que sea funcional y que posea facilidad de uso (usabilidad). La facilidad de uso se entiende como una interfaz que necesite poco tiempo y esfuerzo a aprender la herramienta, que facilite la retención de lo aprendido, que minimice los errores, que se eficiente (referida a la velocidad para realizar una tarea), que sea agradable y cómoda de usar.

La forma de presentación debe incrementar la motivación del alumno [Hernán 06a], sobre todo la motivación intrínseca, que es la más fuerte, ya que proviene del propio sujeto y de su compromiso con su propio proceso de aprendizaje. Si además se consigue incrementar la motivación extrínseca (relacionada con recompensas externas, por ejemplo, con premios o mejores notas) la herramienta tendrá más éxito. Para fomentar la motivación intrínseca se deben proponer situaciones que sean interesantes, que despierten curiosidad o que inviten al usuario a indagar a través de la experimentación.

También es importante que el alumno pueda conocer sus resultados. Por ello creemos básico que se presente un informe estadístico del progreso del discente. Este informe puede ser textual (con número de preguntas respondidas correctamente, o erróneamente, o sin contestar) y/o gráfico (con diagramas de barras o gráficas de los resultados).

Desarrollo de la aplicación

Una vez definido de manera clara todas las etapas anteriores, el siguiente paso es la implementación de la herramienta. Fijándonos en el material docente a producir deberemos deducir el tipo de herramienta a desarrollar. Así, si lo que se pretende es únicamente mostrar información, la herramienta debe ser algo que nos permita visualizar esos datos de forma clara y amena, como por ejemplo, una página web. Si se pretende realizar ejercicios de tipo test, la herramienta será un gestor, visualizador y corrector de test. Si se quiere presentar problemas abiertos o semiabiertos, la aplicación será un gestor y visualizador de enunciados y corrector de texto libre.

En cuanto a la tecnología a emplear para cada tipo de aplicación, queda fuera del ámbito de esta tesis, por lo que se deja a criterio del programador buscar y encontrar la mejor forma de implementarla, siguiendo las pautas marcadas en la metodología.

Evaluación de la aplicación

La etapa final es comprobar que el software desarrollado cumple con los requisitos dados en la metodología. Para ello se toma el producto y se evalúa su calidad docente o pedagógica. Si cumple los objetivos para los que fue diseñado, la metodología habrá tenido éxito. Si por el contrario, los objetivos no se alcanzan, habrá que revisar el material producido y la forma de presentación, ya que en uno de esas dos etapas no se habrá considerado algún aspecto importante para alcanzar la meta prefijada.

3.7. Metodología específica para el diseño de una herramienta educativa según el nivel deseado de la taxonomía de Bloom

A continuación exponemos las fases que requieren un tratamiento especial, dependiendo del nivel de la taxonomía al que se ofrece apoyo. La primera fase es común a todas, por lo que no es necesario presentarla de nuevo.

3.7.1. Nivel 1 o de conocimiento

Si el nivel que se quiere apoyar es el primer nivel de Bloom debemos seguir la metodología general y tener en cuenta las recomendaciones dadas en este apartado.

Segunda fase: Objetivos

Para el nivel de conocimiento, el objetivo es claro: introducir al alumno en el concepto definido en la primera fase. En este caso, los conocimientos de partida son nulos o escasos. La estrategia a seguir es presentar el concepto con explicaciones tanto textuales como gráficas y multimedia (si puede ser [Hernán 06a]), ya que hay que tener en cuenta el estilo de aprendizaje [Felder 96] de cada alumno. Realizar la presentación con una única técnica corre el riesgo de que la herramienta deje de ser atractiva a algún grupo de alumnos. También se debe ejemplificar los conceptos con fragmentos de programas. Si se desea plantear problemas estos deben ser sencillos. Los tipos de ejercicios adecuados para el nivel de Bloom son los test formulados de forma que no difieran demasiado de la manera en que fue explicado el concepto en estudio.

Tercera fase: Producción de material docente

El material puede incluir tanto elementos textuales como gráficos. Recordemos que el uso de cualquier representación externa diferente de la textual requiere una explicación explícita [Ainsworth 99]. En el caso de elegir producir ejercicios de tipo test, estos deben ser formulados de tal forma que la manera en que se expone la pregunta como el grado de precisión y exactitud de las respuestas no ha de diferenciarse mucho del modo con que adquirieron dichos conocimientos [Bloom 56]. Por lo tanto, los ejercicios deben ser redactados por el profesor, que es la persona que conoce el material producido o la clase impartida. También debe incluirse información de retroalimentación al alumno, de tal forma que sea inmediata, para una mejor asimilación de los contenidos aprendidos.

Cuarta fase: Forma de presentación

El material, en caso de ser informativo, debe presentarse con hipermedia [Hernán 06a], para permitir al discente explorar todas las posibilidades. La hipermedia permite al usuario navegar por el documento de la forma que él considere apropiada, accediendo a los contenidos dependiendo de sus necesidades.

En caso de ser problemas tipo test, se deberá presentar toda la información referente al problema (enunciado, posibles soluciones, así como la realimentación) en la misma ventana, de forma que el alumno pueda consultar cualquier parte del problema sin tener que cambiar de escenario.

3.7.2. Nivel 2 o de comprensión

Para el nivel de comprensión de la taxonomía de Bloom debemos tener en cuenta las siguientes recomendaciones.

Segunda fase: Objetivos

Para el nivel de comprensión, el objetivo es ayudar al alumno para que sea capaz de entender y explicar la información recibida. Para ello se pueden proponer ejercicios de tipo test o semiabiertos. Se entiende por ejercicios semiabiertos a los ejercicios que se pueden contestar con un número limitado de respuestas, siendo estás respuestas abiertas (el alumno debe escribirla). En este nivel también se pueden incluir demostraciones del funcionamiento de un programa, con ejemplos y contraejemplos. Podemos imaginarnos una demostración de una ejecución con ayuda de visualizaciones y controles adecuados (como, por ejemplo, algunos que permitan introducir datos de entrada, cambiar el sentido de la visualización, preguntar sobre la instrucción en curso, etc.). Las facilidades de interacción que proporcione determinan el nivel de implicación del alumno [Naps 03a] y por tanto su nivel cognitivo. La herramienta diseñada debe hacer reflexionar al alumno.

Tercera fase: Producción de material docente

El material pueden ser ejercicios o demostraciones. Si son ejercicios, se deben proponer de forma que el alumno deba definir términos o establecer correspondencias entre definiciones y términos. La respuesta del alumno debe llevar integrada alguna característica distinta a la aprendida, bien porque el enunciado lleva alguna situación inédita o bien porque este lleva algunas claves contextuales que permiten al alumno extrapolar o interpretar de forma análoga a la aprendida. Recordamos en este punto que los ejercicios de tipo test pueden ser usados como método de aprendizaje, ya que hacer una prueba de este tipo no solo evalúa lo que ya se sabe sino que también incrementa la capacidad de recuerdo a largo plazo [Mcdaniel 07].

Cuarta fase: Forma de presentación

Si se decide incluir una demostración, se debe permitir que el discente pueda interactuar con cada animación, avanzando o retrocediendo al ritmo que considere oportuno. Esto convierte al discente en un usuario de tipo visualizador activo [Lauer 08]. Con las demostraciones se ayuda a los alumnos activos, que necesitan experimentar para comprender el concepto a enseñar.

Si son ejercicios, de nuevo se debe pedir que toda la información referente al mismo se encuentre en la misma ventana, con cada parte visible y fácilmente distinguible del resto. La corrección de los ejercicios debe mostrarse claramente en la misma ventana o en una ventana modal, que impida continuar si el usuario no realiza la acción pertinente.

3.7.3. Nivel 3 o de aplicación

Para el este nivel debemos de nuevo seguir la metodología general y tener en cuenta las recomendaciones dadas en este apartado.

Segunda fase: Objetivos

Para el nivel de aplicación, el objetivo es ayudar al alumno para pueda seleccionar y utilizar datos y métodos para solucionar una tarea o un problema dado. Para ello se pueden proponer ejercicios semiabiertos o de tipo test. Se entiende por ejercicios semiabiertos a los ejercicios que se pueden contestar con un número limitado de respuestas, siendo estás respuestas abiertas (el alumno debe escribirla.

Tercera fase: Producción de material docente

La resolución de problemas es la forma adecuada de dar soporte a este nivel. Por lo tanto el material debe ser ejercicios de tipo test (cuidadosamente diseñados) o bien ejercicios semiabiertos. Si son ejercicios de tipo test o semiabiertos se deben proponer de forma que el alumno deba enfrentarse a situaciones nuevas donde deban aplicar el conocimiento y los métodos adquiridos, pero que no los hayan utilizado previamente en esa situación. La respuesta del alumno debe demostrar que el discente domina una abstracción en un grado suficiente y que hace un uso correcto de ella.

Cuarta fase: Forma de presentación

Esta fase coincide con la fase del nivel anterior aplicada a la presentación de problemas.

3.8. Resumen

En este capítulo hemos mostrado varias aportaciones de esta tesis. La primera es una propuesta del tipo de herramientas informáticas existentes apropiadas a cada uno de los cinco primeros niveles de Bloom. Luego se ha razonado el porqué diseñar software educativo para los niveles de conocimiento, comprensión y aplicación. A continuación se ha descrito otro de los aportes de este trabajo, enumerando una serie de características generales que todo tipo de aplicación de ayuda al estudio basada en la taxonomía deberían tener. Luego se ha propuesto una forma de diseñar de manera correcta preguntas y problemas apropiados a cada nivel. Por último se ha dado otra contribución de esta tesis, que es una metodología de diseño de herramientas pedagógicas para los tres primeros niveles de la taxonomía de Bloom.

Capítulo 4 Herramientas desarrolladas

Aquí se detallan las herramientas desarrolladas [Hernán 05, Hernán 06b, Hernán 07a, Hernán 07c, Hernán 09, Hernán 10a] en base a la metodología de diseño descrita y que ofrecen ayuda a la enseñanza-aprendizaje de la programación orientada a objetos. En concreto, nos hemos centrado en varios aspectos básicos de las misma, como son el concepto de herencia simple y sus implicaciones así como en la construcción de clases y la creación de objetos. La elección de estos conceptos de la POO es debida a que su comprensión es fundamental para entender el paradigma. Las clases y objetos, así como la herencia entre clases son elementos novedosos para los alumnos que ya conocen la programación imperativa, por lo que su aprendizaje suele ser costoso, ya que implica un cambio en la forma de enfrentarse a la resolución de problemas mediante el ordenador.

El software educativo diseñado y desarrollado lo hemos clasificado según el nivel de la taxonomía de Bloom al que ofrecen soporte. En los siguientes apartados describimos cada una de ellas con sus características principales, acordes con el diseño centrado en dicho nivel. A modo de resumen presentamos la siguiente Tabla 2 donde se muestran las herramientas desarrolladas, el concepto en estudio al que ofrecen ayuda, el nivel superior de Bloom al que dan soporte, así como el tipo de herramienta planteada, el método de corrección de la solución y la realimentación dada al alumno.

Tabla 2. Resumen de las herramientas desarrolladas

Herramienta	Nivel de Bloom	Concepto	Tipo de herramienta	Corrección de soluciones	Realimentación
Ayuda a la herencia					
Teoría	Primero	Herencia	Web	No	No
Demostración	Segundo	Herencia	Flash	No	No
Test	Tercero	Herencia	Test	Fija	A demanda
GeCom	Segundo	Declaración de clases	Texto	No	Añade comentarios al código
CreOO	Tercero	Objetos	Test mediante plantillas	Fija, mediante plantillas	Siempre, mediante plantillas
AplicOO	Tercero	Declaración de clases	Semiabiertos mediante plantillas	Fija, mediante plantillas	Siempre, mediante plantillas
AplicOO v2	Tercero	Declaración de clases	Semiabiertos mediante plantillas	Expresiones regulares	Siempre, mediante plantillas

4.1. Herramienta de ayuda al estudio del concepto de Herencia en POO

Esta aplicación [Hernan 05] está desarrollada en Java y consta de varios componentes, implementados como applets. Cada componente está orientada a un nivel de la taxonomía de Bloom, por lo que procedemos a distinguir cada parte como una única herramienta informática. Es una aplicación de uso individual y se puede utilizar en cualquier lugar con acceso a internet, ya que está implementada como una aplicación que se ejecuta en un navegador.

La aplicación sigue las directrices dadas en los fundamentos pedagógicos. El usuario percibe tres módulos en la aplicación:

- Teoría.
- Demostración.
- Test.

La aplicación está estructurada internamente en directorios, cada uno correspondiente a un módulo. En la carpeta correspondiente a la teoría almacena la página web de contenido que va a mostrar. El directorio de demostraciones alberga subcarpetas denominadas $demo_i$ (donde *i* indica el número de demostración), que contienen los archivos necesarios para visualizar la demostración (archivo *.swf* producido en FLASH, archivo *.html* para la carga del mismo, el archivo *flash.jar* para su correcta visualización, y un fichero de texto con el título de la demostración). El número actual de demostraciones es de dos. Por último, el directorio de los test, que consta de tres subcarpetas, denominadas nivel1, nivel2 y nivel3, que corresponden con los tres primeros niveles de la taxonomía de Bloom, cuyo contenido es el siguiente: enunciado de las preguntas (en archivo denominados $test_n.html$ donde *n* corresponde al número de la pregunta), respuesta correcta y explicación (en un archivo denominado $explicacion_n.txt$).

Se eligió esta organización por su sencillez de instalación y de gestión (modificación, inserción, eliminación). Por ejemplo, para añadir preguntas de tipo test, basta crear un archivo (*html*) con el enunciado, denominarlo $test_m.html$ donde m será el número siguiente al último añadido, modificar el archivo de respuestas correctas y crear un archivo nombrado $explicacion_m.txt$ que contendrá información de realimentación al alumno. La aplicación reconoce automáticamente la creación de nuevas carpetas (en el caso de las demostraciones) o de nuevos archivos. La interfaz gráfica es sencilla e intuitiva para que el alumno no tenga que dedicar tiempo extra a aprender su manejo.

El tema escogido para el estudio es el concepto de herencia simple en el paradigma de la programación orientada a objetos. Este concepto es complejo por lo que se descompuso en todas sus dimensiones [Budd 00]. La herencia en programación orientada objetos se puede dividir en dos dimensiones (veasé Capítulo 3): la parte teórica relacionada con el concepto y la parte práctica de dicho concepto. Podemos refinar a su vez la dimensión práctica en dos facetas o subdimensiones: el diseño de clases con herencia y su implementación en un lenguaje concreto [Hernán 11a]. Para ejemplificar este tema se eligió Java como lenguaje de programación base.

Esta herramienta está dirigida a estudiantes que ven este paradigma de programación por primera vez. Sus conocimientos sobre el concepto son nulos

o básicos (lo enseñado en una clase previa tradicional por el profesor de la asignatura, sin profundizar en el concepto). El contexto en el que se va a usar la herramienta es un segundo curso de la carrera de Ingeniería Informática, donde los alumnos conocen previamente la programación imperativa usando el lenguaje Pascal como el vehículo de sus prácticas y conocen la programación orientada a objetos básica usando el lenguaje Java para implementar los conceptos aprendidos, pero no lo han usado para implementar herencia simple.

4.1.1. Componente de Teoría

Este componente está orientada al primer nivel o nivel de conocimiento. Por lo tanto, siguiendo la metodología descrita se procedió a diseñarla:

Primera fase: Análisis de requisitos o necesidades docentes

El nivel de Bloom al que se pretende dar soporte es el primer nivel, como ya se ha comentado anteriormente, por lo que el software a diseñar se centrará en la primera dimensión del concepto en estudio.

Segunda fase: Objetivos

Se deben incluir conceptos básicos teóricos, así como ejemplos, para que el alumno vaya familiarizándose con el tema. Como el nivel es el de conocimiento, el objetivo de esta herramienta es presentar de forma atractiva y potencialmente extensible (a través de hiperenlaces) el concepto de herencia. Para conseguirlo se puede incluir hipermedia, para dar soporte a los distintos estilos de aprendizaje del alumnado y potenciar la propia iniciativa del alumno en su proceso de aprendizaje.

Tercera fase: Producción de material docente

El material docente producido es una explicación tanto textual como gráfica del concepto de herencia. La teoría presentada ayuda a los alumnos reflexivos, que estudian solos. También ayuda a los alumnos visuales pues la teoría no sólo se explica de forma verbal (mediante código) sino también de manera gráfica.

***Cuarta fase*:** Forma de presentación

La explicación del concepto así como los ejemplos se presenta como una página web (Figura 14), por reunir todas las características sugeridas en la segunda fase, así como por su sencillez de implementación y las ventajas educativas del hipermedia [Dillon 98].

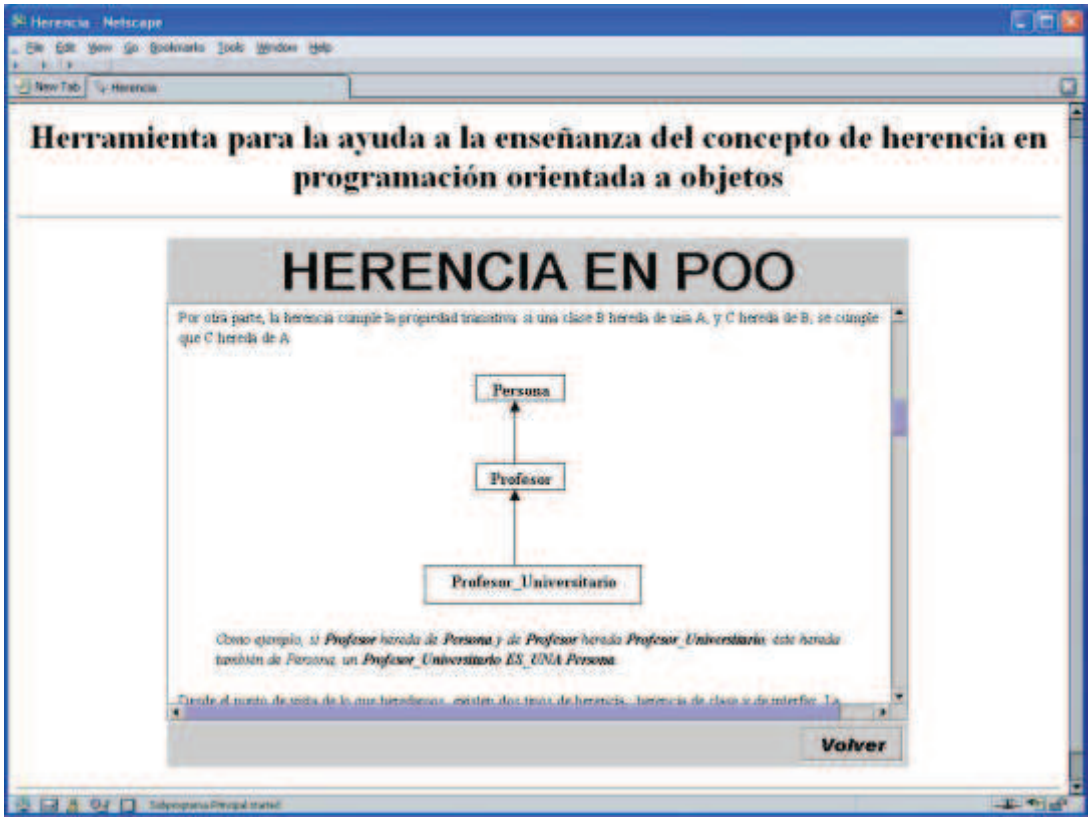


Figura 14. Componente de teoría de la herramienta de ayuda al estudio del concepto de herencia

4.1.2. Componente de demostración

Este componente está orientada al segundo nivel o nivel de comprensión. Por lo tanto, siguiendo la metodología descrita se procedió a diseñarla:

***Primera fase*:** Análisis de requisitos o necesidades docentes

En este caso, se pretende permitir visualizar los conceptos explicados en el componente de teoría o en clase. El nivel de Bloom al que se pretende dar soporte es el segundo nivel o de comprensión, por lo que el software a diseñar se centrará en las dos dimensiones del concepto en estudio.

Segunda fase: Objetivos

En este nivel se pueden incluir demostraciones del funcionamiento, con ejemplos y contraejemplos. El objetivo de esta herramienta es hacer reflexionar al alumno sobre el funcionamiento y características de la herencia. Podemos imaginarnos una demostración de una ejecución con ayuda de visualizaciones y controles adecuados. Las facilidades de interacción que proporcione determinan el nivel de implicación del alumno [Naps 03a] y por tanto su nivel cognitivo. En nuestro caso la implicación del alumno se centra sobre todo en el visual activo [Lauer 08] ya permite controlar el ritmo de demostración (avanzar hacia adelante y atrás).

***Tercera fase*:** Producción de material docente

Las demostraciones incluidas en esta herramienta son animaciones. Estas animaciones han sido realizadas en Flash ya que pueden visualizarse en cualquier navegador con el *plug-in* correspondiente instalado. Se ha decidido ilustrar el proceso que tiene lugar cuando una clase hija hereda de una clase padre. La animación debe presentar de forma textual como visual el proceso. Para ello se van generando las instrucciones en Java y el diagrama UML de clases correspondiente a cada instrucción.

Se pensaron dos demostraciones. La primera, muestra dos clases heredando de una clase padre y como se produce la herencia de atributos y métodos públicos y protegidos. También modela la extensibilidad, añadiendo atributos y la redefinición y la extensión de métodos. La segunda debe enseñar el uso adecuado e inadecuado de los métodos heredados.

***Cuarta fase*:** Forma de presentación

El alumno puede interactuar con cada animación, avanzando o retrocediendo al ritmo que considere oportuno. Con las demostraciones se ayuda a los alumnos activos, que necesitan experimentar para comprender el concepto a enseñar. Gráficamente se ve como las clases incorporan los atributos y métodos a medida que se van definiendo. La primera (Figura 15) va mostrando el proceso de construcción de una clase padre, y dos clases hijas, que heredan de la primera. Se muestra como recogen sus atributos y sus métodos y como la extienden y la especializan. En rojo se marca cada uno de los cambios producidos por las correspondientes instrucciones.

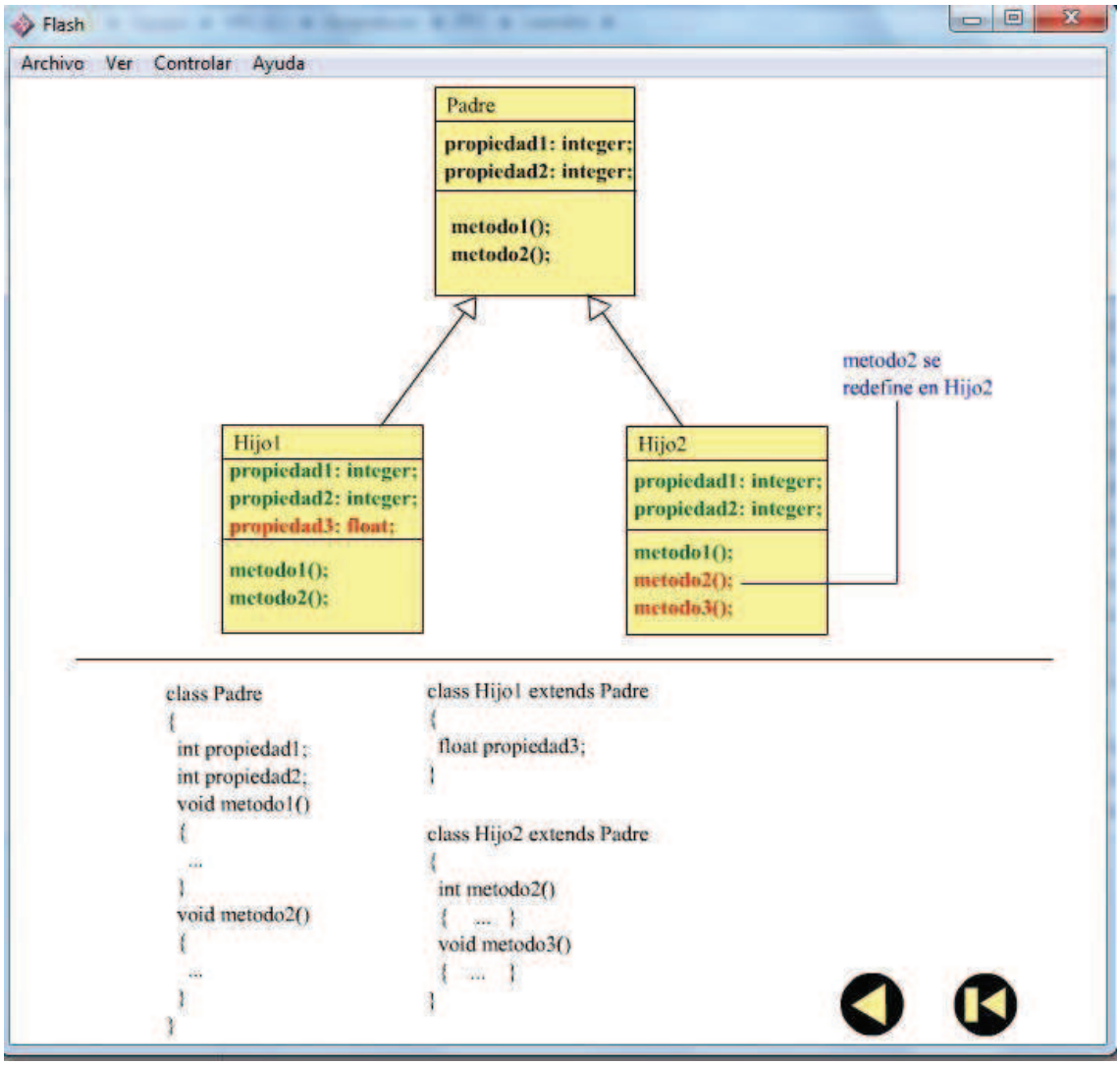


Figura 15. Componente de demostración, primera animación.

La segunda demostración ilustra el uso correcto e incorrecto de los métodos heredados o supuestamente heredados. La siguiente figura (Figura 16) muestra esta última demostración. Las visualizaciones se basan en la notación UML. Puede observarse en la parte inferior derecha los botones de avance y retroceso para controlar la animación. Las clases se representan con rectángulos que encierran el nombre en la parte superior, debajo los atributos y por último los métodos (notación UML). Las elipses representan objetos instanciados. En la parte inferior van apareciendo las instrucciones, resaltándose la que está en ejecución. En la parte superior derecha, el objeto que recibe el método se marca con un trazo perimetral grueso y simultáneamente aparece una explicación del mensaje que recibe el objeto en cuestión. Si se intenta ejecutar una instrucción errónea, el sistema explica el motivo del error y la tacha para resaltar el hecho.

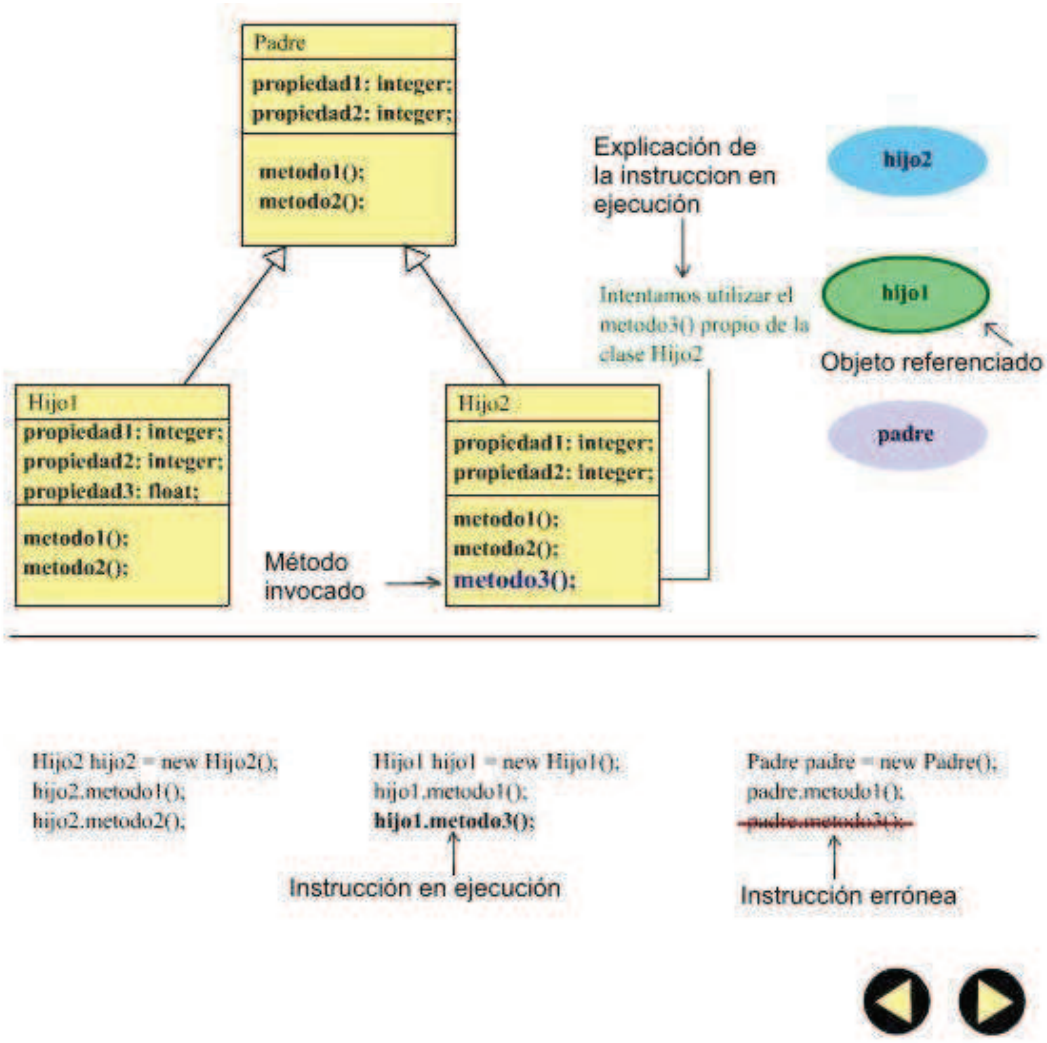


Figura 16. Componente de demostración, segunda animación.

4.1.3. Componente de test

Esta componente está orientada a los tres primeros niveles, conocimiento, comprensión y aplicación. El usuario puede elegir el grado de dificultad de las preguntas de tipo test.

Primera fase: Análisis de requisitos o necesidades docentes

En este caso, se pretende permitir practicar y autoevaluar los conceptos explicados en la componente de teoría o en clase. Dependiendo del nivel de Bloom al que se pretende ayudar el software a diseñar se centrará en la correspondiente dimensión del concepto en estudio.

Segunda fase: Objetivos

Los conocimientos previos del alumno dependen del nivel de la taxonomía que el usuario elija. Si escogen el primer nivel, se suponen conocimientos mínimos por lo que las preguntas serán básicas. Si deciden optar por el segundo nivel, se supone que el discente ya tiene asentados los conocimientos básicos sobre la herencia, por lo que las preguntas serán más elaboradas, haciendo que el estudiante tenga que reflexionar sobre la respuesta correcta. Si elijen el tercer nivel, el alumno tendrá conocimientos y comprenderá el mecanismo de la herencia simple, por lo que estará capacitado para contestar aplicando los conceptos aprendidos.

Tercera fase: Producción de material docente

En este caso, se diseñaron tres baterías de preguntas apropiadas a distintos niveles de Bloom. Cada pregunta de tipo test consta de enunciado, cuatro posibles respuestas y una explicación para cada pregunta, para así poder realimentar al alumno, siempre y cuando éste lo solicite. Dada la estructura de la aplicación es sencillo modificar, añadir y eliminar preguntas.

Para el primer nivel, los enunciados deben ser formulados de tal forma que la forma en que se expone la pregunta como el grado de precisión y exactitud de las respuestas no ha de diferenciarse mucho del modo con que adquirieron dichos conocimientos [Bloom 56].

Para el segundo nivel, los enunciados pueden introducir alguna novedad con respecto a la forma de enseñanza. Al ser de tipo test, las respuestas elegibles deben contener elementos novedosos, buscando que la respuesta correcta difiera de la forma en la que fue introducido el concepto.

Para el tercer nivel, los enunciados deben ser totalmente novedosos. Deben plantear al alumno tareas nuevas relacionadas con el concepto en estudio, que exijan una aplicación de los métodos ante la nueva situación.

Cuarta fase: Forma de presentación

La primera vez que se ejecuta pregunta al alumno empieza una sesión y debe escoger el nivel donde desea comenzar. Una vez elegido, las preguntas se presentan de manera aleatoria y asegurando que no se repiten en la misma sesión. La herramienta va almacenando las estadísticas de las preguntas (aciertos y fallos). De esta forma y para reforzar la motivación extrínseca del alumno la aplicación aumenta automáticamente de nivel (pasa al siguiente nivel

cognitivo) cuando éste logra acertar el 80% de las preguntas efectuadas. Para que sea efectivo, el porcentaje se debe calcular después de un número fijo de preguntas. En nuestro caso hemos elegido que deba contestar al menos cinco para que se pueda subir de nivel.

Un ejemplo de la presentación de contenidos se muestra en la siguiente figura (véase Figura 17). En la parte superior muestra el concepto en estudio, el nivel de la taxonomía de Bloom. En la parte central aparece el enunciado con las posibles respuestas. En la parte inferior se encuentra la parte de interacción, donde el discente puede elegir la respuesta correcta. Cuando el usuario pulsa el botón de Validar, la aplicación muestra en una ventana emergente si la respuesta es correcta o no. En ambos casos, esta ventana contiene un botón de Explicación, donde el alumno puede ver la respuesta correcta y una explicación del porqué. Esta realimentación inmediata hace más efectivo el aprendizaje [Farnsworth 94].

Figura 17. Componente de test

En la Figura 17 4 se muestra una pregunta del nivel de conocimiento o nivel uno. Un ejemplo de las preguntas formuladas por la aplicación para el nivel de comprensión es:

Dado el siguiente código ¿Qué sentencias son válidas?

```
class Automovil {
  int puertas;
}
class Coche extends Automovil {
  private boolean gasolina;
  ...
}
```

a) Coche coche = new Coche();
   coche.puertas = 4;

b) Automovil auto = new Automovil();
   auto.gasolina = false;

c) Para que la opción b) sea correcta la propiedad gasolina debería ser pública.

d) a) y c) son correctas.

Esta forma de presentar los ejercicios tiene las ventajas descritas en la metodología general (véase Capítulo 3).

4.1.4. Descripción detallada de la herramienta y conclusiones

Todas las componentes se unieron en una única herramienta, para una mejor distribución. Al ser aplicaciones de tipo *applets* se pueden visualizar con cualquier navegador. Se puede acceder a cada una de ellas de forma simple, como se observa en la Figura 18. Dependiendo de la necesidad, el discente elige la componente adecuada.

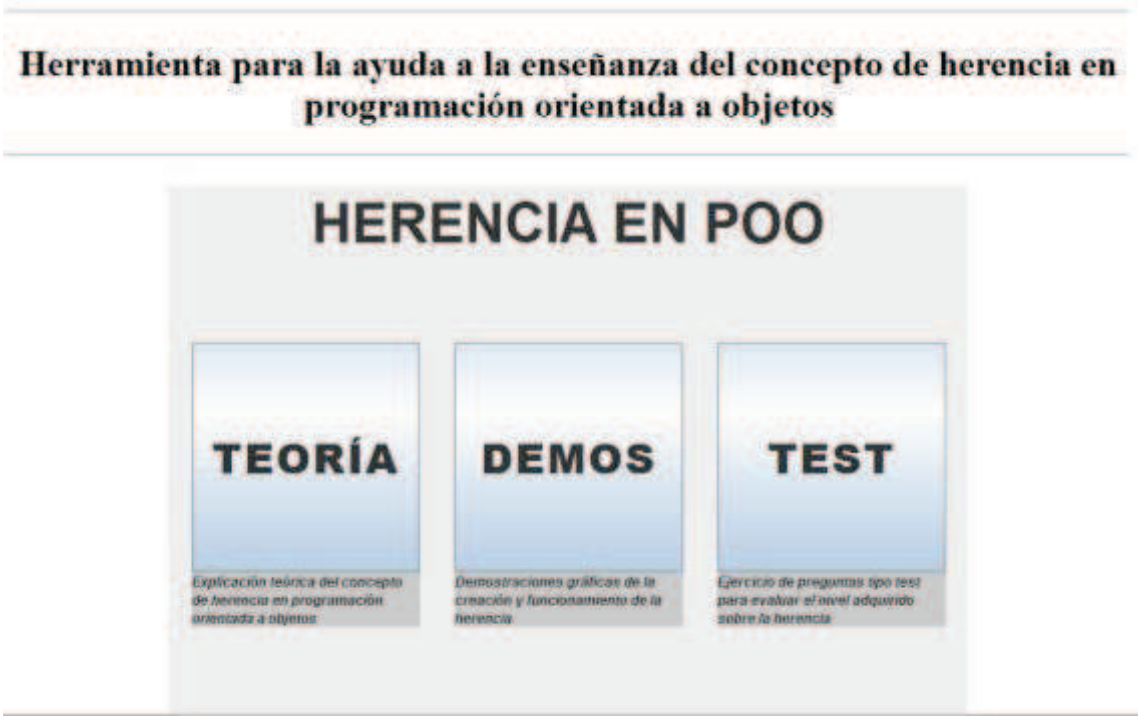


Figura 18. Herramienta completa de ayuda a la enseñanza de la herencia

Se ha conseguido desarrollar todas las componentes siguiendo el diseño aportado en esta tesis. La implementación por parte de un técnico informático ha sido sencilla, puesto que la metodología de diseño dejaba todos los aspectos a tener en cuenta de forma clara. Esta herramienta ha sido probada y evaluada por los alumnos, consiguiendo una valoración muy buena y demostrando su eficacia educativa mediante una evaluación descrita en el capítulo siguiente.

4.2. Generación de comentarios GeCom.

Esta aplicación [Hernán 06b] está desarrollada en Java. Es una aplicación de escritorio, de uso individual y se puede instalar en cualquier ordenador que tenga la máquina virtual de Java (JRE). Para realizar el diseño, se siguió la metodología específica para el segundo nivel.

***Primera fase*:** Análisis de requisitos o necesidades docentes

El tema escogido es la declaración de una clase, con sus atributos y sus métodos. El nivel de Bloom al que se pretende dar soporte es el segundo o de comprensión, por lo que el software a diseñar se centrará en ayudar al entendimiento del concepto de clase. Para implementar los ejemplos se eligió Java como lenguaje de programación base.

Segunda fase: Objetivos

Esta herramienta está dirigida a estudiantes que ven este paradigma de programación por primera vez y sus conocimientos sobre la creación de clases son básicos, ya han visto o asistido a una clase de explicación del concepto y conocen la sintaxis de Java para implementar dicha implementación. El contexto en el que se va a usar la herramienta es un segundo curso de la carrera de Ingeniería Informática, donde los alumnos conocen previamente la programación imperativa usando el lenguaje Pascal como el vehículo de sus prácticas y además han visto el concepto de clase de la programación orientada a objetos y han usado el lenguaje Java.

En este nivel se van a incluir demostraciones del funcionamiento, con ejemplos. Estas demostraciones no son visuales, sino textuales. El objetivo de esta herramienta es explicar de forma detallada la semántica de cada una de las instrucciones que existen en la implementación de una clase. Estas explicaciones se hacen de forma automática y se insertan en el código en forma de comentarios, de forma que el código original no se ve modificado (salvo con los comentarios). En las listas de verbos apropiados para cada nivel, en el nivel de comprensión se incluye traducir, que es justamente lo que realiza esta aplicación.

***Tercera fase*:** Producción de material docente

En este caso, la producción del material adecuado (los comentarios) los hace la propia herramienta, analizando instrucción a instrucción y añadiendo los comentarios correspondientes. Los comentarios se añadirán justo después de la instrucción, para relacionar de forma clara e inequívoca ambos. El código puede ser introducido directamente por el usuario o ser leído de un fichero. Además, se han introducido más de 90 ejemplos para que el alumno pueda usarlos y comprender mejor el proceso de la implementación de una clase.

***Cuarta fase*: Forma de presentación**

El alumno puede interactuar con la aplicación decidiendo cómo va a introducir el código (ver Figura 19) o si prefiere consultar ejemplos predefinidos. Además se ofrece las restricciones actuales que tiene la aplicación, con el fin de ver el alcance de esta herramienta.

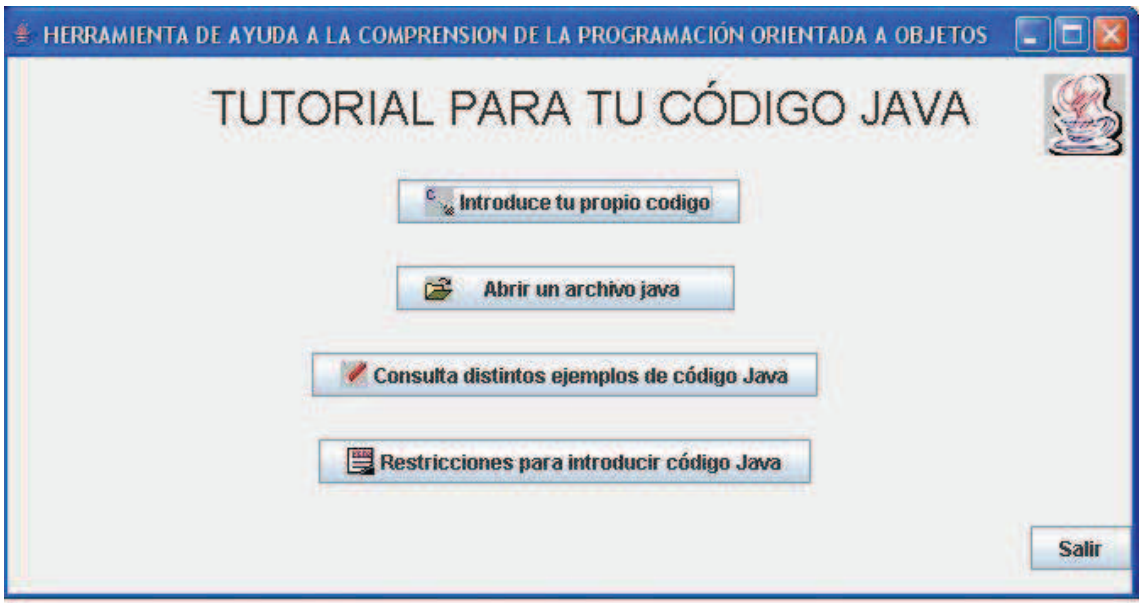


Figura 19. Ventana principal de GeCom

Una vez elegida la forma de visualizar el código, se abrirá una nueva ventana con un editor textual, donde se verá el código escrito o leído de un archivo (Figura 20). Este fragmento de programa podrá ser modificado por el usuario. Cuando este decida añadir los comentarios, pulsará el botón de Ver explicación, con lo que GeCom añadirá automáticamente los comentarios. En la Figura 21 se observan estos comentarios, resaltados en color azul. Este nuevo código podrá ser guardado en un archivo para su posterior tratamiento (compilación o ejecución).

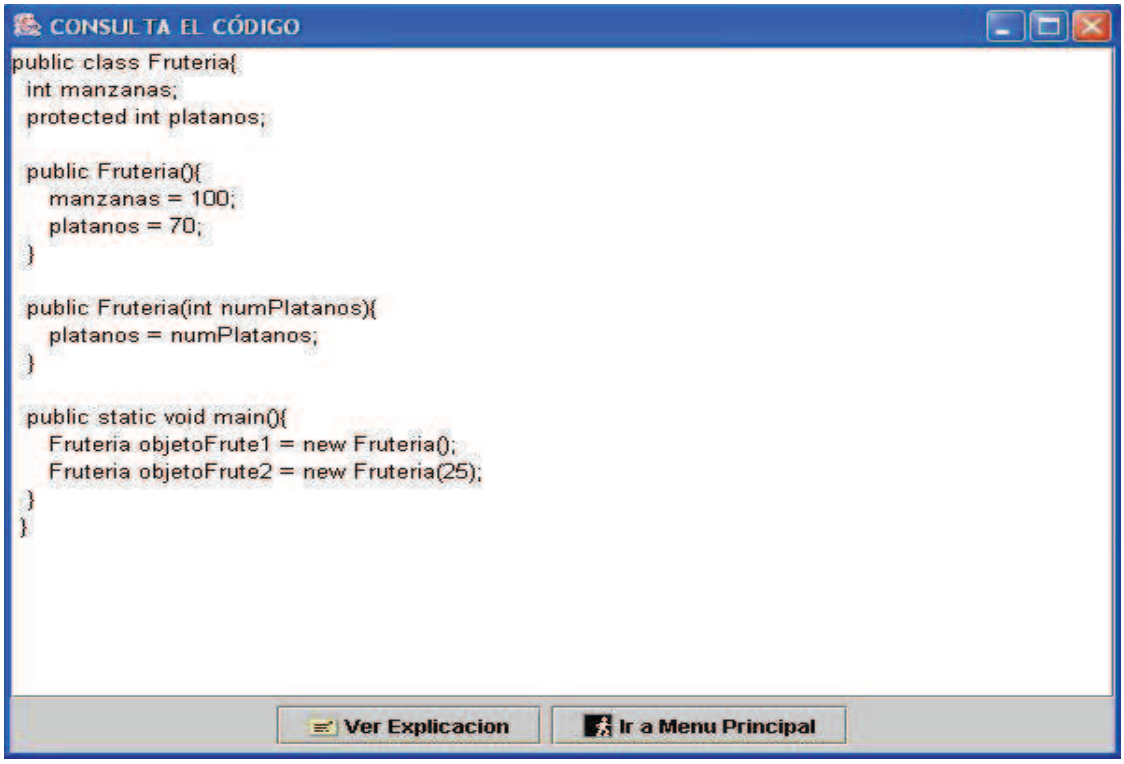


Figura 20. Editor de GeCom

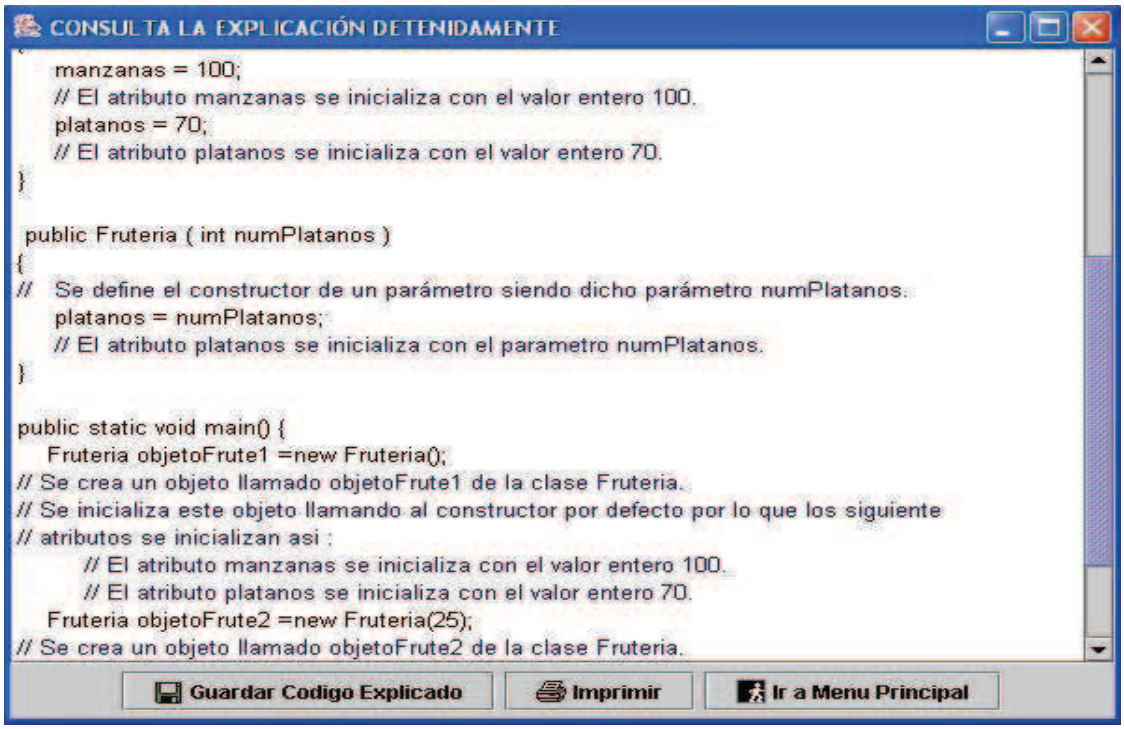


Figura 21. Comentarios generados por GeCom

4.2.1. Descripción detallada de GeCom y conclusiones

Esta aplicación genera automáticamente comentarios a un programa sencillo implementado en Java. Por sencillez, GeCom restringe el lenguaje Java permitido a un subconjunto sencillo del mismo, que se puede consultar en la herramienta. En concreto, en la versión actual solamente se admite la declaración de una clase, con sus atributos de tipo int, con distintos constructores y el método main que contiene la creación de un objeto de la clase declarada. Se considera un "error" no seguir las restricciones, aunque el código escrito sea correcto según la sintaxis de Java. Estas limitaciones pueden ser eliminadas haciendo un analizador de instrucciones más completo.

El alumno puede realizar tres acciones básicas (véase Figura 19) más otra de consulta de las restricciones que GeCom impone:

a) Escribir directamente el programa y que la aplicación genere los comentarios, siempre y cuando este programa satisfaga las restricciones impuestas por GeCom.

b) Consulta de ejemplos de código previamente diseñados. Se pretende que el alumno observe y se familiarice con el esquema de un programa básico, así como con la sintaxis para crear una clase. En la actualidad hay 90 ejemplos distintos.

c) Abrir un archivo con un programa Java previamente codificado.

En las tres acciones, el usuario siempre tiene la opción de modificar el código (propio, del ejemplo o del archivo), así como de generar comentarios. GeCom informa al alumno de cualquier incumplimiento de las restricciones. Para ello, GeCom incorpora un analizador sintáctico que comprueba el código introducido por el estudiante. Una vez generados los comentarios de un programa, GeCom permite imprimir el código explicado o almacenarlo en un archivo para su estudio posterior.

GeCom es una herramienta de ayuda a la enseñanza que únicamente da apoyo a las explicaciones teóricas de la programación en Java. Tiene un campo de aplicación limitado dentro de la POO, la declaración de una clase, con sus atributos y constructores, así como la creación de un objeto. Únicamente admite Java como lenguaje de programación para la ejemplificación de este paradigma. Asimismo, su interfaz es muy simple e intuitiva. Ambas características permiten al alumno centrarse en la comprensión del concepto explicado. También facilitan la clasificación de la herramienta en el nivel de comprensión de la taxonomía de Bloom.

Obsérvese que los comentarios generados se visualizan en distinto color que el código para distinguirlo y poder leerlo con facilidad. Esta característica es común a las facilidades de edición de los entornos de programación contemporáneos. Desde el punto de vista docente, interesa incluir esta característica porque está demostrada la bondad del uso de técnicas de visualización para mejorar el aprendizaje [Naps 03b].

4.3. CreOO

Esta herramienta [Hernán 06b] sigue la línea de la componente de test de la herramienta de ayuda al estudio de la herencia que permite resolver ejercicios agrupados por nivel de dificultad. Aparte de cambiar el concepto en estudio, que pasa a ser el mecanismo de la creación de objetos, la mayor aportación de esta nueva aplicación es la forma de generar los enunciados, las respuestas posibles, la respuesta correcta y las explicaciones, ya que se hacen de forma automática (a través de plantillas). Para resolver los problemas, el estudiante debe haber entendido y comprendido el mecanismo de la creación de objetos. Está dirigida a alumnos con conocimientos básicos de POO.

***Primera fase*:** Análisis de requisitos o necesidades docentes

El tema escogido es la creación de objetos. El nivel de Bloom al que se pretende dar soporte es el tercero o de aplicación, por lo que el software a diseñar se centrará en ayudar a practicar el concepto, que ha tenido que ser previamente introducido y comprendido, bien mediante explicaciones y ejercicios realizados en clase o mediante algún software orientado a los dos primeros niveles. De nuevo para implementar los ejemplos se eligió Java como lenguaje de programación base.

Segunda fase: Objetivos

Esta herramienta está dirigida a estudiantes que se introducen en el paradigma de programación orientada a objetos por primera vez. Como conocimientos previos, los alumnos deben saber la forma de crear objetos y sus implicaciones. El contexto en el que se va a usar la herramienta es el segundo curso de la carrera de Ingeniería Informática, donde los alumnos conocen previamente la programación imperativa y además conocen los conceptos de clase y objeto de la programación orientada a objetos y han usado el lenguaje Java. El objetivo es que los discentes puedan poner en práctica sus conocimientos mediante la realización de ejercicios de tipo test con preguntas novedosas. Para conseguir esta característica (la novedad), es necesario que las preguntas no se obtengan de una colección, pues después de varios usos es casi inevitable que los enunciados se repitan. Si los problemas se repiten, los alumnos pueden memorizar las respuestas correctas sin necesidad de comprender el concepto en estudio ni de aplicarlo.

Tercera fase: Producción de material docente

La producción de enunciados (código), posibles respuestas, respuesta correcta y explicación (realimentación), se hace en forma de plantilla. Estas plantillas las genera el profesor, especificando las partes fijas y las partes variables. Las partes fijas se presentarán igual en todos los problemas y las partes variables se instanciaran en cada ejemplar de forma distinta, de manera que exista una probabilidad baja de tener dos problemas iguales.

Por su carácter pedagógico, es muy importante que el enunciado de cada pregunta, así como el código presentado, corresponda al nivel de aplicación de la taxonomía de Bloom. Deben plantear al alumno tareas nuevas relacionadas con el concepto en estudio, que exijan una aplicación de los métodos ante la nueva situación.

Cuarta fase: Forma de presentación

Las funciones básicas de CreOO son: generar un nuevo problema, variar el nivel de dificultad y visualizar la estadística de respuestas acertadas/falladas. La aplicación presenta una ventana (véase Figura 22) donde se distinguen tres zonas. La zona izquierda presenta un código escrito en Java, con una clase y la creación de un objeto. En la parte derecha superior se presenta el enunciado y las posibles respuestas, para que el usuario escoja una de ellas. En la parte derecha inferior se presenta la zona de realimentación, donde se muestra la corrección de la respuesta dada, así como una explicación (véase Figura 23).

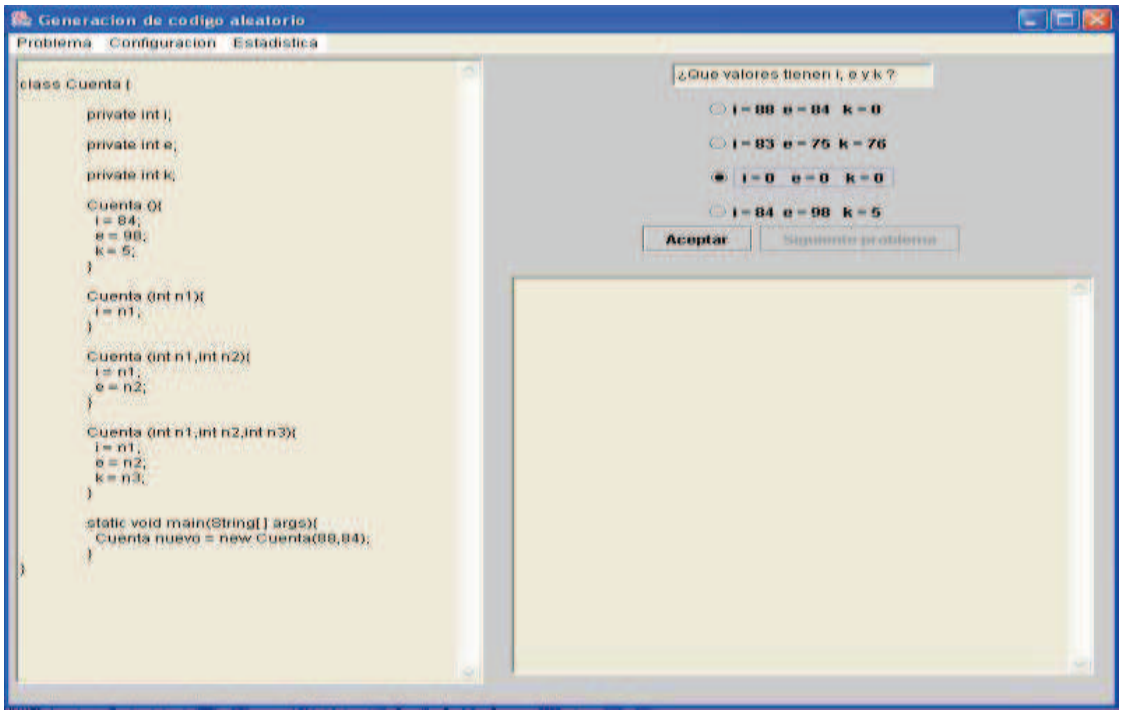


Figura 22. Ventana principal de CreOO

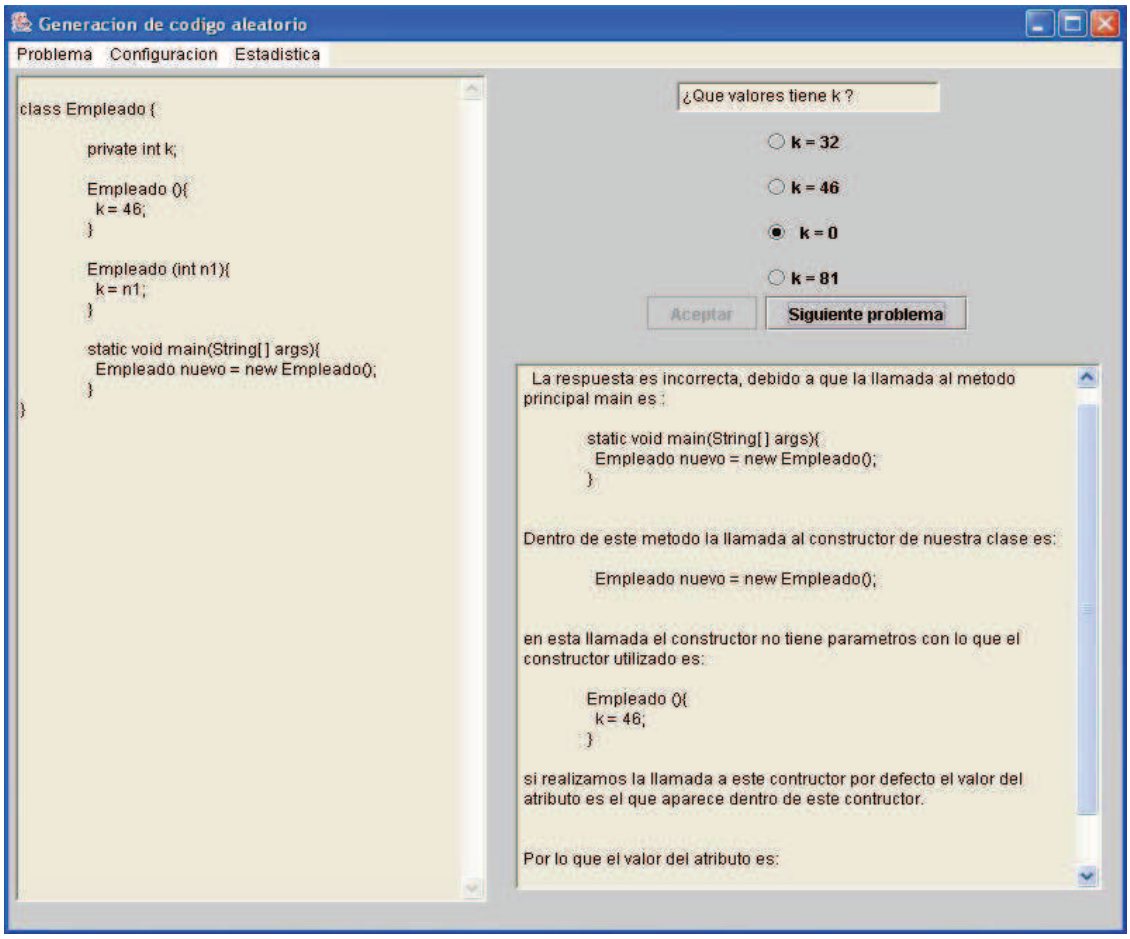


Figura 23. Realimentación de CreOO

4.3.1. Descripción detallada de CreOO y conclusiones

CreOO realiza automáticamente tanto la generación de problemas sobre la creación de objetos en Java como la corrección de la solución del alumno. Informa al alumno del resultado de la corrección y, en caso de haber dado una respuesta errónea, genera una explicación de porqué ha errado. Como ya se ha comentado, esta realimentación es esencial para que el autoaprendizaje tenga lugar de forma correcta y el concepto se fije en la memoria a largo plazo.

Las funciones que tiene son las siguientes:

Generar un problema. Permite al alumno generar un nuevo problema consistente en una pregunta sobre un fragmento de código que el estudiante tendrá que examinar para poder responder correctamente (Figura 22). Tanto el código como la pregunta y las opciones de respuesta son generados de forma aleatoria.

La generación de problemas se basa en una plantilla con partes variantes. En la actualidad existen seis plantillas distintas diseñadas por el profesor. La plantilla siempre consta de una clase que se denomina de forma cambiante, con nombres escogidos al azar de una lista de nombres. En la actualidad la lista consta de diez nombres y su ampliación es muy sencilla. El número de atributos lo fija el alumno con la opción Configuración y puede escoger entre que la clase tenga uno, dos o tres atributos. El tipo de los atributos siempre es *int* por sencillez, pero se podría modificar fácilmente para elegir de una lista de tipos. Al ser los atributos de tipo entero, los valores se escogen también al azar entre 0 y 100. Los nombres de los atributos son escogidos aleatoriamente de las letras del alfabeto.

El número de constructores también es elegido al azar teniendo en cuenta que existe un número máximo, que depende del número de atributos, ya que todos los atributos son del mismo tipo. Por ejemplo, si el usuario escoge generar un problema con un atributo, el número máximo de constructores que puede tener la clase es dos (el constructor sin argumentos y el constructor con un parámetro). Si el usuario elige un problema con dos atributos, el número de constructores máximo es tres (esto es debido a que todos los atributos son del mismo tipo). En general, para n atributos del mismo tipo tendremos n+1 posibles constructores. En el caso de que la herramienta soportara tener atributos de distinto tipo el número máximo de constructores se ampliaría considerablemente. Así con dos atributos de distinto tipo se podrían combinar para tener como máximo cinco constructores, y con tres atributos se podrían tener dieciséis constructores.

Veamos la forma básica de uno de estos esquemas o plantillas. Las palabras encerradas entre < > son variables, que toman valores al azar de una lista de nombres, salvo <valor> que se genera aleatoriamente entre un rango de números. Así, una plantilla para el código podría ser:

```
class <nombre_clase>;

private int <nombre_atributo>;

<nombre_clase> () {

    <nombre_atributo> = <valor>;

}

static void main (String[] args) {

    <nombre_clase> Nuevo = new <nombre_clase> ( );

}
```

Una vez producido el código se genera una pregunta relacionada. Las posibles opciones de respuesta también son generadas de forma aleatoria, excepto la correcta, poniendo restricciones para que no se repitan. Los rangos de posibles valores vienen limitados entre 0 y 100 y para evitar casos simples se prohíbe la repetición.

Como consecuencia de todo lo expuesto en los párrafos anteriores, es muy poco probable que se repitan dos problemas iguales en una misma sesión. Esto permite que el alumno no aprenda por memorización, pues la aleatoriedad le obliga a razonar sobre la forma de resolverlo, garantizando de este modo que el discente este en el nivel de aplicación. Esta característica hace que esta herramienta pueda ser usada en evaluaciones, ya que previene frente al plagio entre alumnos.

Variar el grado de dificultad de las preguntas. CreOO permite tres niveles de dificultad. Un aumento de la dificultad se concreta en un incremento del número de atributos. Al tener más atributos, el número máximo de constructores crece y con ello se amplían las posibilidades al crear un nuevo objeto. En consecuencia, al aumentar las posibilidades de respuesta, la dificultad del problema aumenta.

Mostrar el número de preguntas realizadas y el número de aciertos y de fallos. En esta opción, se muestra en una nueva ventana una tabla con las estadísticas del alumno en la sesión actual.

La interfaz que presenta es intuitiva y muy sencilla de usar. Toda la información se encuentra en la misma ventana, con lo que el acceso a cualquier función es rápido.

El número de problemas distintos que puede generarse es muy grande. Para una plantilla con un solo constructor sin argumentos, una vez fijado la cantidad de atributos, dicho número está únicamente limitado por la variedad de identificadores *a* introducidos para la clase (en nuestro caso diez), el número de nombres *b* de los atributos y el rango de valores *c* que son asignados a los atributos:

$$n = a^* b * c \quad \text{(Numero de problemas distintos)}$$

Por ejemplo, para la plantilla mostrada anteriormente tenemos 10 nombres de clases, 26 identificadores de atributos y 100 posibles valores para los atributos el número de problemas distintos es 26000. Este número aumenta si añadimos constructores con argumentos.

El sistema es capaz de resolver cada problema que genera y dar una explicación sobre cuál sería la respuesta correcta. Esta explicación es generada mediante plantillas relacionadas con el código generado por la aplicación (véase Figura 23). A cada problema le corresponde un esquema de explicación. Estas plantillas tienen el mismo formato que las plantillas para generar código, por lo que no es necesario introducirla de forma explícita.

Resumiendo, CreOO es una herramienta de ayuda a la enseñanza que únicamente da apoyo a la parte práctica del concepto, por lo que el alumno ha debido recibir la parte teórica de forma externa a la aplicación. Al igual que GeCom, se centra en un solo concepto. Esto facilita que la herramienta se centre en uno o pocos niveles de la taxonomía de Bloom; en este caso, los niveles de comprensión y de aplicación. El nivel de dificultad mínima permite comprobar que el alumno ha comprendido el mecanismo de creación de objetos. Con los otros dos grados de dificultad, el alumno debe saber aplicar su conocimiento para responder correctamente. Además, la explicación dada al alumno es un valor añadido para éste cuando comete un error, ya que le ayuda a aprender de sus propios errores.

La aplicación tiene cierta similitud con los tutores para la enseñanza de la programación imperativa desarrolladas por Amruth Kumar. Estas aplicaciones, denominadas genéricamente problets o problettes, están dirigidas al nivel de aplicación y se ha demostrado su eficacia pedagógica [Dancik 03, Kumar 02]

4.4. AplicOO

Describimos una herramienta educativa, denominada AplicOO [Hernán 07a], diseñada para el aprendizaje de la POO mediante resolución de problemas de respuesta semiabierta. Consideramos problemas de respuesta semiabierta a los que se pueden solucionar de una forma o de formas limitadas, en las que el alumno debe introducir la solución de forma textual. AplicOO permite resolver problemas generados automáticamente sobre varios aspectos clave de Java, como son la declaración de atributos, la declaración de métodos, la creación de objetos, y la declaración de clases con herencia. La respuesta debe ser un fragmento de código Java, por lo que puede corregirse automáticamente. En concreto, la aplicación tiene por objetivo que el alumno alcance el tercer nivel de la taxonomía de Bloom (nivel de aplicación).

***Primera fase*:** Análisis de requisitos o necesidades docentes

En el caso de esta herramienta se pretende dar soporte a las declaraciones de los distintos elementos de una clase. En concreto a la declaración de atributos, con los distintos modificadores que le pueden afectar, declaración de métodos de una clase, con sus modificadores, declaración de constructores de una clase, creación de objetos de una clase y declaración de clases que heredan de otra clase. El nivel de Bloom al que se pretende dar soporte es el tercero o de aplicación, por lo que el software a diseñar se centrará en ayudar a practicar el concepto, que ha tenido que ser previamente introducido y comprendido, bien mediante explicaciones y ejercicios realizados en clase o mediante algún software orientado a los dos primeros niveles. De nuevo para implementar los ejemplos se eligió Java como lenguaje de programación base.

Segunda fase: Objetivos

Esta herramienta está dirigida a estudiantes que se introducen en el paradigma de programación orientada a objetos por primera vez. Como conocimientos previos, los alumnos deben saber el concepto y los métodos en estudio. El contexto en el que se va a usar la herramienta es el segundo curso de la carrera de Ingeniería Informática, donde los alumnos conocen previamente la programación imperativa y además conocen los conceptos de clase y objeto de la programación orientada a objetos y han usado el lenguaje Java. El objetivo es que los discentes puedan poner en práctica sus conocimientos mediante la

realización de ejercicios de tipo semiabierto con preguntas novedosas. Para conseguir esta característica (la novedad), es necesario que las preguntas no se obtengan de una colección, pues después de varios usos es casi inevitable que los enunciados se repitan. Si los problemas se repiten, los alumnos pueden memorizar las respuestas correctas sin necesidad de comprender el concepto en estudio ni de aplicarlo.

Tercera fase: Producción de material docente

La producción de los códigos, enunciados, respuesta correcta y explicación (realimentación) de cada pregunta, se hace en forma de plantilla. Estas plantillas las genera el profesor, especificando las partes fijas y las partes variables. Las partes fijas se presentarán igual en todos los problemas y las partes variables se instanciaran en cada ejemplar de forma distinta, de manera que exista una probabilidad baja de tener dos problemas iguales.

Por su carácter pedagógico, es muy importante que el enunciado de cada pregunta, así como el código presentado, correspondan al nivel de aplicación de la taxonomía de Bloom. Deben plantear al alumno tareas nuevas relacionadas con el concepto en estudio, que exijan una aplicación de los métodos ante la nueva situación.

En la actualidad tenemos implementadas seis grupos de plantillas, correspondientes a distintos aspectos de la programación orientada a objetos. A modo de ejemplo, se presentan dos de estos grupos de plantillas que hasta ahora tiene la aplicación:

Declaración de atributos

Los atributos son cualidades comunes a los objetos de una misma clase. Estas propiedades poseen diferentes modificadores, que el alumno debería conocer para diseñar de manera correcta una clase. Por lo tanto, las preguntas van relacionadas con los distintos modificadores que los atributos pueden poseer.

En la Plantilla 1 puede verse el formato de la plantilla correspondiente al código que se genera en el tema de declaración de atributos. La nomenclatura para expresar estos esquemas es la siguiente: Las palabras encerradas entre < > son meta-variables, que toman valores al azar de distintas listas de nombres, salvo <valor> que se genera aleatoriamente entre un intervalo de números.

```
public class <clase1> {
   int <atrib1> = <valor>;
   int <atrib2> = <valor>;
public <clase1> (int <param1>) {
      <atrib1> = <param1>;
   }
}
```

Plantilla 1. Código de la declaración de atributos de AplicOO

A partir de la plantilla, el sistema genera automáticamente el código. El enunciado se produce a partir de otra plantilla relacionada con la plantilla de código. Su organización e instanciación se realiza de forma similar. La plantilla para el enunciado relacionado con el tema en cuestión es (véase Plantilla 2):

```
¿Cómo debe ser la declaración de un atributo <atrib3>
de la clase <clase1> sin inicializar y que sea privado,
estático y booleano?
```

Plantilla 2. Enunciado de la declaración de atributos de AplicOO

La respuesta correcta también se genera mediante una plantilla, que en el caso que estamos tratando será (véase Plantilla 3):

```
private static boolean <atrib3>
```

Plantilla 3. Solución de la declaración de atributos de AplicOO

Y la explicación que recibe el alumno también es generada mediante plantillas (véase Plantilla 4):

```
Para que un atributo llamado <atrib3> sea privado hay que
anteponer la palabra clave "private".

Para que un atributo llamado <atrib3> sea estático hay que
anteponer la palabra clave "static".

Para que un atributo llamado <atrib3> sea de tipo booleano
hay que anteponer la palabra clave "boolean".
```

Plantilla 4. Explicación de la declaración de atributos de AplicOO

Declaración de métodos

Los métodos marcan el comportamiento de los objetos de la misma clase. Los objetos se comunican entre sí por el paso de mensajes, que no es otra cosa que la invocación de los métodos públicos que poseen.

En este caso, la plantilla usada para que el alumno ejercite este aspecto fundamental de la programación orientada a objetos es la Plantilla 5:

```
public class <clase1> {
   int <atrib1> = <valor>;
   int <atrib2> = <valor>;
public <clase1> (int <param1>) {
      <atrib1> = <param1>;
   }
}
class <clase2> extends <clase1> {
   int <atrib2> = <valor>;
   int <atrib3> = <valor>;
   public <clase2> (int <param2>,
                    int <param3>) {
      <atrib2> = <param2>;
      <atrib3> = <param3>;
   }
```

Plantilla 5. Código de la declaración de métodos de AplicOO

Y la plantilla que genera el enunciado es (véase Plantilla 6):

```
¿Cómo debe ser la declaración de un método <metodo1> de la
clase <clase1> que sea protegido y no devuelva nada?
```

Plantilla 6. Enunciado de la declaración de métodos de AplicOO

Cuarta fase: Forma de presentación

AplicOO muestra en la misma ventana el código, el enunciado, la corrección de la pregunta y si el usuario lo solicita, la explicación o realimentación. De esta forma, el alumno puede acceder a toda la información de manera rápida y fácil.

Este software ofrece varias funciones básicas como son: gestionar los usuarios, generar un problema y visualizar la estadística de respuestas acertadas/falladas. La aplicación presenta una ventana (véase Figura 24) donde se distinguen tres zonas. La zona izquierda presenta un código escrito en Java. En la parte derecha superior se presenta el enunciado y una línea de texto vacía para que el alumno introduzca su respuesta. El alumno puede elegir entre contestar (botón de "aceptar") o pasar al siguiente problema (botón de "pasar") sin contestar el actual. En la parte derecha inferior se presenta un cuadro de texto donde se muestra el resultado y dos botones. El botón de "respuesta" genera una explicación sobre el problema. El usuario puede pulsarlo haya o no introducido correctamente la respuesta. El botón de "siguiente" sirve para que la aplicación presente un nuevo problema.

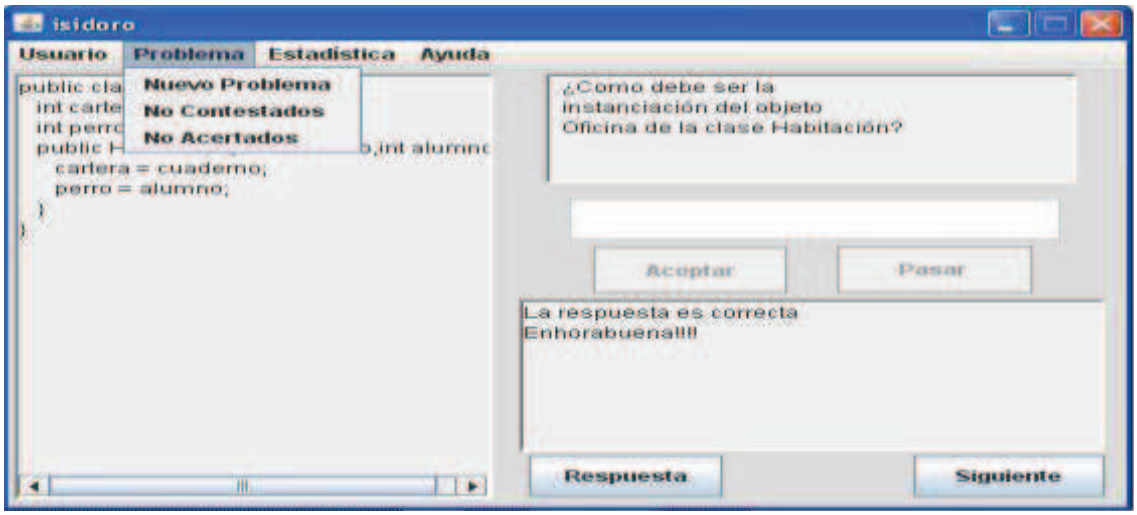


Figura 24. Ventana principal de AplicOO

4.4.1. Descripción detallada de AplicOO y conclusiones

La herramienta guarda estadísticas de la actividad de todos los usuarios. Por tanto, comienza cada sesión pidiendo al alumno que se identifique. Entonces se presentan las opciones mostradas en el menú de la herramienta que son: gestionar los usuarios, generar un problema y visualizar la estadística de respuestas acertadas/falladas:

Gestionar usuarios.

Esta opción permite guardar los datos de la sesión actual así como crear un nuevo usuario o trabajar con un usuario anteriormente creado. Por lo tanto, la herramienta posee información de cada alumno, lo que podría usarse para realizar estudios estadísticos, para ver la evolución del alumno o para adaptar los problemas al progreso del discente.

Generar un problema.

Permite al alumno generar un nuevo problema, abrir un problema no contestado, o abrir un problema que ha fallado en esa misma sesión. La Figura 24 muestra esta posibilidad de elección de una pregunta tras haber respondido a otra previa. Un nuevo problema consiste en una pregunta sobre un fragmento de código inédito relacionado con los aspectos de POO antes citados. El funcionamiento de la aplicación cuando se genera un problema se muestra en la Figura 25. El estudiante tendrá que examinar el código para poder responder correctamente. Tanto el código como la pregunta y las opciones de respuesta son generados de forma aleatoria. Si el usuario no sabe la respuesta, puede pasar a un nuevo problema. El problema anterior se almacena por si el usuario decidiera en esa sesión intentarlo de nuevo. También se guardan temporalmente los problemas resueltos erróneamente por el mismo motivo. Estos problemas son eliminados al finalizar la sesión.

La generación de problemas se basa en plantillas. En la actualidad hay cinco temas disponibles, cada uno con una plantilla asociada. La elección de un determinado tema, y por tanto de su correspondiente plantilla lo realiza de forma automática la herramienta. Esta aleatoridad se puede suprimir de manera sencilla, con el objetivo de que el alumno elija el tema.

Cada plantilla consta de al menos una clase que se denomina de forma cambiante, con nombres escogidos al azar de una lista de nombres. El tipo de los atributos siempre es *int* por sencillez, pero se podría modificar fácilmente para elegir de una lista de tipos. Los nombres de los atributos son escogidos también al azar, haciendo casi imposible la aparición de dos códigos idénticos.

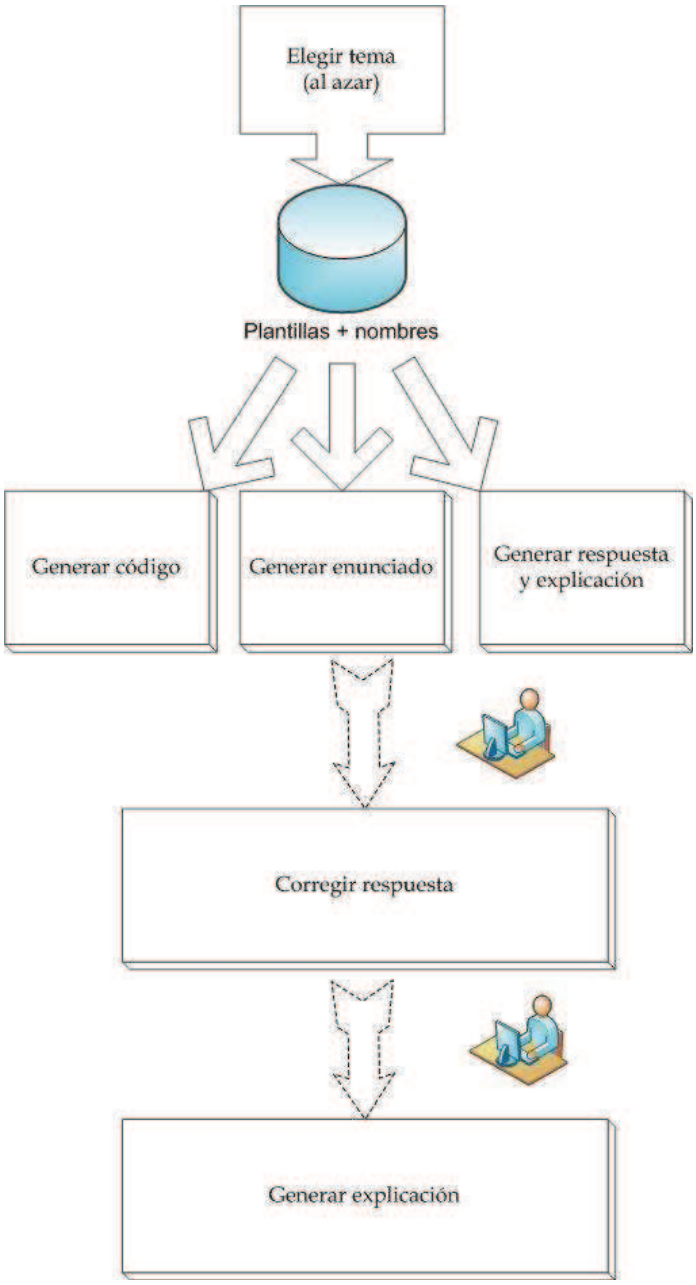


Figura 25. Funcionamiento de AplicOO

Mostrar la estadística.

Se muestra el número de preguntas que el usuario ha acertado, ha dejado sin contestar y las que ha fallado, en todas las sesiones que la ha usado.

AplicOO posee algunas características relacionadas con la taxonomía de Bloom [Fuller 07, Hernan 06a] que la hacen fácilmente clasificable. Por una parte, se centra en pocos conceptos de la POO, que ya se han enumerado anteriormente. Por otra parte, incluye explicaciones teóricas (al utilizar la opción de "respuesta") y un número muy grande de problemas distintos.

La herramienta debería ser eficaz educativamente hablando, pues implica al alumno gracias a la autoevaluación y a la respuesta inmediata y detallada de sus soluciones. Esta realimentación es esencial para que el autoaprendizaje tenga lugar de forma correcta y el concepto se fije en la memoria a largo plazo.

También es fácil de utilizar con un fin docente, pues su instalación es sencilla (es una aplicación Java, con lo que sólo es necesario tener la máquina virtual instalada en el ordenador) y su interfaz es sencilla e intuitiva como se muestra en la figuras adjuntadas.

Existen aplicaciones educativas que tratan de ayudar al alumno de programación a alcanzar el nivel de aplicación [Dancik 03, Hernán 05, Kumar 02], pero todas poseen problemas tipo test, con lo que siempre muestran entre sus opciones la respuesta correcta, con lo que el alumno puede recordar (nivel de conocimiento) y contestar correctamente.

Es casi imposible que se repitan dos problemas iguales en una misma sesión, debido a la gran cantidad de códigos y de enunciados distintos que se pueden generar. El número de códigos y enunciados distintos que se pueden generar para una determinada plantilla dependen del número de combinaciones que se pueden formar con nombres diferentes que existan (en la actualidad unos cuarenta en AplicOO) y del rango de valores para los atributos (del -10 al 10). Por ejemplo, la plantilla presentada en Plantilla 1 contiene 7 meta-variables y 2 valores. Suponiendo que solamente existieran 2 valores posibles para instanciar cada meta-variable o valor (en realidad son más), tendríamos un total de 512 posibilidades.

Esta aleatoriedad y la facilidad para ampliar el número de plantillas, permite que el alumno no aprenda por memorización, pues le obliga a razonar sobre su resolución, a la vez que previene frente al plagio entre alumnos. La ausencia de opciones de respuesta también obliga al alumno a conocer, comprender y aplicar los conocimientos previamente adquiridos. Por tanto, la herramienta posee las características adecuadas [Hernán 06a] para ayudar al alumno a alcanzar el nivel de aplicación de la taxonomía de Bloom.

4.5. Otras herramientas desarrolladas

Uno de los problemas principales que tienen algunas de las herramientas que hemos diseñado y desarrollado es que el profesor siempre debe estar involucrado en el momento del diseño. Es decir, sus contenidos son cerrados, ya que una vez desarrollada solo puede ser modificados mediante programación. También nos hemos encontrado con una dificultad en AplicOO y es la forma de corregir las respuestas dadas por los usuarios. Para solucionar ambos problemas, hemos diseñado y desarrollado dos aplicaciones para la ayuda a la enseñanza de la programación orientada a objetos (CreOO v2 y AplicOO v2).

4.5.1. CreOO v2

CreOO es una herramienta diseñada para ser usada por un único actor: el alumno. El profesor solo interviene en el diseño, mostrando las necesidades docentes, los objetivos y desarrollando el material docente. La versión actual, CreOO v2, además de las características de CreOO, añade otro tipo de usuario, el profesor. Ofrece al mismo nuevas funcionalidades entre las que destacan la gestión de usuarios y la gestión de las preguntas, pudiéndose agrupar para realizar exámenes. Todos los datos de esta aplicación (preguntas, exámenes, usuarios etc.) están almacenados en una base de datos guardada en un servidor. Pasamos a detallar las nuevas opciones que se ofrecen al usuario profesor:

Realizar la gestión usuarios.

Con esta opción el profesor puede limitar el uso de la herramienta a determinados usuarios (alumnos u otros profesores). Con ella dará de alta, borrará y modificará los usuarios que usarán la aplicación y nos mostrará un listado de usuarios, independientemente del tipo que sean, con sus datos personales.

Gestión preguntas/exámenes.

Esta funcionalidad es la mayor aportación a la antigua herramienta. Con ella se gestiona todas las preguntas con sus plantillas de códigos, enunciados, posibles soluciones, la solución correcta y una breve explicación. En CreOO las plantillas había que realizarlas en la etapa de diseño. Ahora se permite la inclusión, modificación, eliminación y agrupación (exámenes) de las preguntas en tiempo de ejecución. La única restricción que tiene es que para modificar o eliminar una pregunta, está no debe pertenecer a un examen. También permite

crear exámenes (agrupar preguntas para su posterior realización por parte de los alumnos), modificarlos o eliminarlos.

Para que una aplicación tenga éxito es imprescindible que sea usable y que no suponga un esfuerzo considerable su aprendizaje y su uso (capítulo 3). La introducción de las plantillas de los problemas se puede realizar básicamente de dos formas. La primera es especificarla en algún medio (papel o formato electrónico) y que el desarrollador la incorpore directamente al código del programa. Esta forma ha sido la utilizada en las herramientas CreOO y AplicOO. La segunda es introducirla directamente en la aplicación. La versión 2 de CreOO permite esta opción. Para ello, la aplicación ofrece una interfaz sencilla (véase Figura 26) que facilita esta tarea. No es necesario aprender ningún lenguaje de marcado para especificar las meta-variables (partes variantes) ni los meta-valores (valores cambiantes). El software ofrece botones que permiten realizar esta tarea de forma automática.

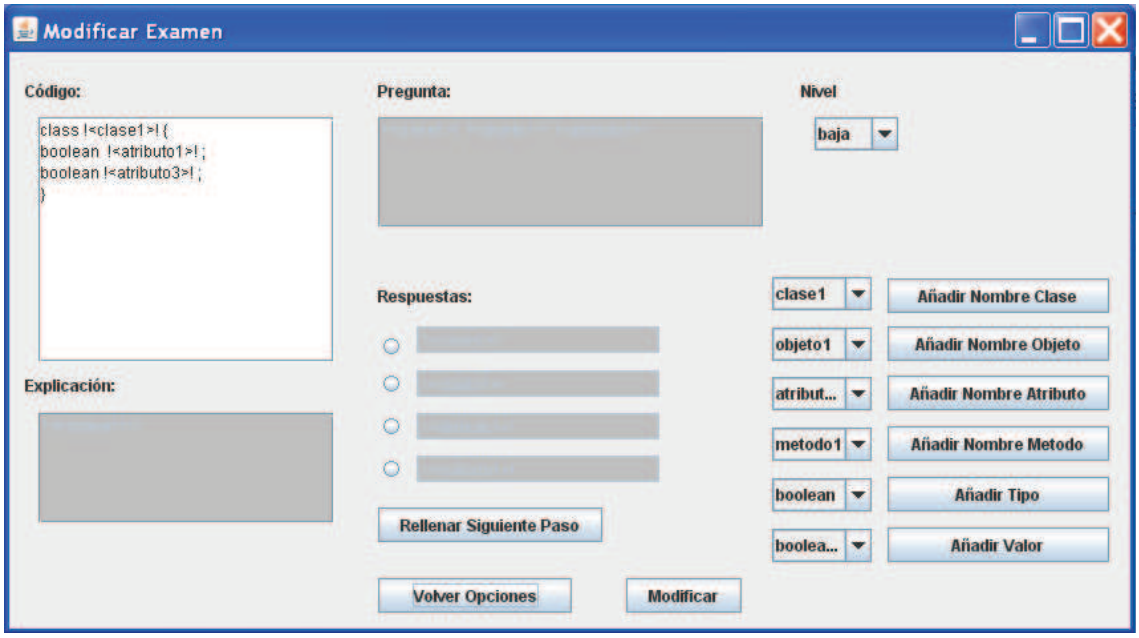


Figura 26. Pantalla de modificar/introducir pregunta en CreOO v2.

La forma de introducir o modificar una pregunta es la siguiente: Primero se elige la parte a modificar/introducir, por ejemplo, el código. Se escribe la parte fija, que no lleva ninguna etiqueta. La parte variable (meta-variables o valores) van encerrados entre "!<" y ">!". Esta parte se puede introducir por teclado o mediante los botones y cuadros desplegables situados en la parte derecha. En el ejemplo mostrado el nombre de la clase "!<clase1>!" ha sido introducido eligiendo del cuadro desplegable la primera opción y pulsando en el botón de "Añadir Nombre Clase". De esta forma se puede generar las plantillas de todos los campos de una forma sencilla.

La aplicación entonces se encargará de generar de forma adecuada a las plantillas las preguntas, posibles respuestas y explicación a los alumnos, así como la corrección automática.

Ver estadísticas.

Aquí el profesor podrá ver las estadísticas de las pruebas realizadas por sus alumnos, tanto de forma individual, seleccionando un alumno en concreto, como de forma colectiva, con un listado.

4.5.2. AplicOO v2

En AplicOO igual que en CreOO todo lo concerniente los problemas se crea a través de plantillas. En esta segunda versión, se añade al usuario profesor. Las novedades de esta aplicación con respecto a AplicOO son: a) permite introducir las plantillas de forma sencilla, de manera similar a CreOO v2; b) la corrección de la respuesta introducida por el usuario se hace a través de expresiones regulares, con lo que permite ampliar la precisión de esta corrección.

La funcionalidad general en cuanto a la gestión de usuarios y la introducción de las plantillas de este software es similar a CreOO versión 2 por lo que la omitimos, centrándonos en la parte novedosa de esta herramienta, que es la introducción de la expresiones regulares para la corrección de la solución producida por el alumno. La escritura de una expresión regular suele ser complicada. La primera dificultad es debida a que hay que aprender el significado de los símbolos implicados en el patrón. Estos símbolos suelen depender del lenguaje de programación usado. En nuestro caso, este lenguaje es Java, por lo que habría que aprender dicho significado en este lenguaje. La segunda dificultad es la definición en sí del patrón, es decir, poner todos los símbolos apropiados para la descripción de la solución correcta. Para facilitar esta tarea, la aplicación ofrece una interfaz (véase Figura 27) con unas expresiones regulares predefinidas, con lo que el profesor solo tiene que elegir entre las ofertadas. Las expresiones que actualmente tiene la aplicación son la declaración de atributos de tipos primitivos, la declaración de métodos, la declaración de la cabecera de una clase, con y sin herencia y la instrucción de creación de objetos. Si la expresión que el docente busca no se encuentra predefinida, puede optar por introducirla por teclado. También se ofrece a través del botón de "ayuda" un pequeño manual introductorio a las

expresiones regulares en Java y ejemplos prácticos del uso de estas expresiones. En la Figura 28 se enseña la ventana de ayuda con un ejemplo concreto para crear el patrón de declaración de atributos.

Figura 27. Introducción de una expresión regular en AplicOO v2.

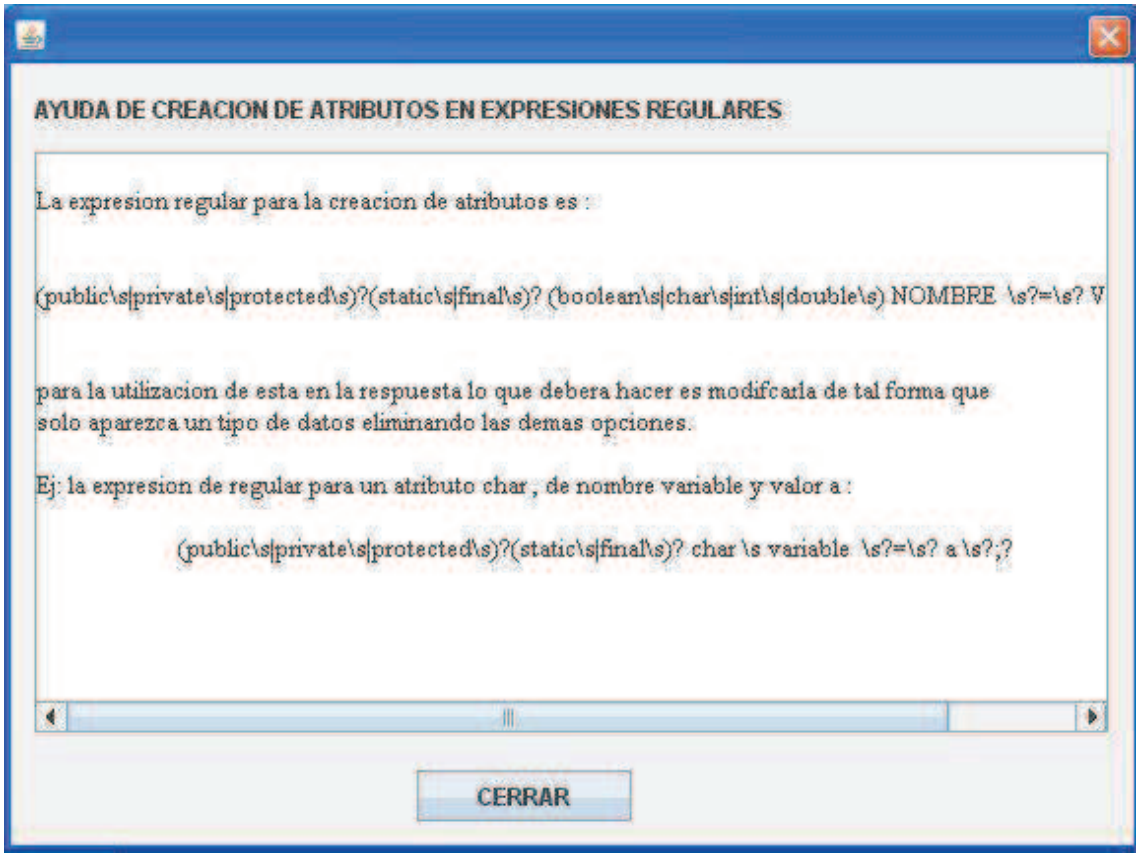


Figura 28. Ventana de ayuda de las expresiones regulares en AplicOO v2.

.

Capítulo 5 Evaluación

En el capítulo 3 hemos descrito unas metodologías de diseño de aplicaciones educativas para los tres primeros niveles de la taxonomía de Bloom. En el capitulo anterior hemos mostrado como se aplica esa técnica para el diseño y desarrollo de varias herramientas de ayuda a la enseñanza de la programación. Queda por demostrar su validez, entendiéndose como tal su eficacia educativa y su facilidad de uso. Para ello hemos desarrollado diferentes experimentos que nos han permitido observar el uso de dichas herramientas. En el presente capítulo describimos las evaluaciones realizadas, los resultados obtenidos y su interpretación estadística.

5.1. Evaluación de la herramienta de ayuda al estudio del concepto de Herencia en POO

Esta aplicación [Hernan 05] está desarrollada en Java y consta de varios componentes, implementados como *applets*. El usuario puede decidir a qué parte de la herramienta quiere acceder y por lo tanto, que nivel de Bloom del concepto de herencia simple pretende alcanzar. Se pretende ver la utilidad de esta herramienta. Se han realizado dos evaluaciones, una cuantitativa y otra cualitativa. En la cuantitativa para demostrar o refutar la eficacia educativa se ha utilizado la metodología de investigación explicada en el tema 1.

5.1.1. Evaluación cuantitativa

Planteamiento del problema

El problema planteado es el siguiente: Se dispone de un software educativo de ayuda a la enseñanza del concepto de la herencia simple y se quiere observar si el aprendizaje de ese concepto de los alumnos que lo usan es mayor que de los alumnos que no lo utilizan. O dicho de otro modo, se pretende realizar un experimento que demuestre la eficacia pedagógica de esta herramienta.

Formulación de hipótesis

Se plantearon dos cuestiones relacionadas:

a. determinar si el uso de la aplicación influye significativamente en el aprendizaje global de los alumnos y,
b. ver si refuerza el aprendizaje en algún nivel concreto de la taxonomía de Bloom.

Para ello se diseña un experimento cuantitativo, en el que se hará un estudio estadístico de los datos y se establece un intervalo de confianza para rechazar o aceptar la hipótesis del 95%. Este valor es el valor típico que se suele tomar en estudios de ciencias sociales y en estudios de medicina, para validar las hipótesis de trabajo.

Identificación de variables

Para realizar el estudio se consideran dos variables, una independiente y otra dependiente de la primera. Se pretende estudiar la relación entre ambas. La variable independiente es el uso de la herramienta diseñada y desarrollada siguiendo la metodología propuesta en este trabajo y la variable dependiente es la diferencia entre las notas obtenidas por cada grupo en la realización del pretest y del postest.

Como en este experimento se establecen dos hipótesis, la variable dependiente (la diferencia entre las notas del pretest y postest) se calcula de la siguiente manera: Para la primera hipótesis, la diferencia de notas se obtiene de la puntuación final de los test, entendiéndose como tal la suma de todas las respuestas correctas; para la segunda hipótesis las diferencias de notas se obtienen de la suma de los puntos obtenidos en cada respuesta correcta de un nivel concreto. El objetivo de la investigación es determinar la influencia de la variable independiente (el uso de la aplicación) sobre la variable dependiente (diferencia de notas).

Población y muestra

La población objeto del estudio fueron los alumnos de la asignatura Programación Orientada a Objetos perteneciente al segundo curso de Ingeniería Informática de la Universidad Rey Juan Carlos en el curso 2004/2005.

Los 36 alumnos que asisten regularmente a las clases prácticas fueron divididos en dos grupos: uno experimental y otro de control. La división se realizó al azar por apellidos y teniendo en cuenta el sexo (está demostrado que hombres y mujeres se enfrentan de forma distinta a los ordenadores [Carter 99], de manera que quedaron dos grupos de 18 miembros (13 alumnos y 5 alumnas). Todos los alumnos están matriculados en la asignatura por primera vez y han recibido las mismas clases teóricas de la misma profesora (que no pertenece al mismo grupo de investigación que los autores del experimento, con lo que se

evita cierto sesgo en el estudio). Con esta división se intentó que los grupos fueran lo más homogéneos posible. También se comenta que los investigadores no habían tenido contacto previo con los alumnos, para evitar el sesgo al generar los test.

Obtención de los datos

Los tests (véase Apéndice B) fueron diseñados con 3 preguntas por nivel de Bloom, por lo que contenía 9 preguntas con 4 posibles respuestas. Para la evaluación, las respuestas acertadas sumaban 1 punto y las falladas 0 (máximo 9 puntos), es decir, no se penalizó el uso del azar. El diseño de los test se hizo cuidadosamente. Para garantizar los niveles, la profesora colaboró en las preguntas del nivel de conocimiento, para que tuvieran una redacción lo más parecida posible a lo explicado [Bloom 56]. El resto de las preguntas fueron enunciadas por los autores.

Las clases prácticas de la asignatura de POO, son clases que se imparten en los laboratorios. Duran dos horas y generalmente se ponen en práctica los conceptos adquiridos en las clases teóricas. La obtención de los datos tuvo lugar en la clase práctica, correspondiente al tema de herencia, sin realizar la división de los grupos, se pasó un test previo (pretest) al uso de la aplicación. Finalizado el pretest, se dividió a los alumnos en los dos grupos anteriormente mencionados. Los alumnos del grupo de control realizaron con el ordenador la práctica correspondiente al tema de herencia simple. En contraposición, el grupo experimental instaló la aplicación y la usó durante aproximadamente media hora. Antes de finalizar la sesión práctica, se les entregó un cuestionario para evaluar la herramienta. Al día siguiente, en la clase teórica, a ambos grupos se les pasó un test (postest) con el mismo diseño que el pretest y distintas preguntas.

Resultados

Los test fueron respondidos por los alumnos de los dos grupos. Se pueden observar los resultados en la Tabla 3, Tabla 4 y Tabla 5. Es de notar que se produjeron tres bajas en el grupo experimental y en el grupo de control, por lo que los datos de esos alumnos no se han tenido en cuenta en los resultados. Para el análisis de datos se usó la aplicación estadística R [Venables 05].

Tabla 3. Resultados generales de los tests

	Pretest		Postest	
GRUPO	Control	Experimental	Control	Experimental
Media	5,00	6,28	4,60	5,27
Mediana	5	7	4	6
Varianza	3,14	2,35	2,69	2,64
Desviación típica	1,77	1,53	1,64	1,62

Rango	1..8	3..9	2..7	2..8

Tabla 4. Diferencias entre el pretest y el postest

GRUPO	Control	Experimental
Media	0,4	1,0
Mediana	1,0	1,0
Varianza	2,69	3,14
Desviación típica	1,64	1,77
Rango	-2..3	-2..4

Tabla 5. Diferencias agrupadas por niveles de Bloom

	Control			Experimental		
	N1	N2	N3	N1	N2	N3
Media	0,93	-0,80	0,27	0,47	-0,13	0,67
Mediana	1	-1	0	0	0	1
Varianza	1,35	0,59	0,77	1,56	0,85	0.51
Desviación típica	1,16	0,77	0,88	1,25	0,92	0,72
Rango	-2..3	-2..0	-1..2	-1..3	-1..2	0..2

Análisis de los datos

Una vez recogidos los datos y mostrados los resultados generales, se procede a analizarlos y hacer el estudio estadístico que permita corroborar o desechar las hipótesis de partida.

Para estudiar la normalidad de las muestras (Tabla 6) hemos utilizado las pruebas de Shaphiro-Wilk y la de Kolmogorov-Smirnov que dan un valor del estadístico P. Si el valor de P obtenido en cada una de las pruebas no supera 0.05 (debido al criterio del intervalo de confianza de 95% anteriormente escogido), se concluye que la muestra no sigue una distribución normal. En este caso, todos los valores P calculados son mayores que ese valor, por lo tanto, siguen la distribución normal. Esto indica que se pueden usar los métodos paramétricos para su estudio.

Tabla 6. Prueba de normalidad de las muestras (P)

	Pretest		Postest	
GRUPO	Control	Experimental	Control	Experimental
Shaphiro-Wilk	0,27	0,49	0,56	0,61
Kolmogorov-Smirnov	0,67	0,56	0,95	0,54

A continuación comprobamos si la división realizada en grupos (por apellidos y sexo) es correcta o, en cambio, presenta algún sesgo (por ejemplo, que los alumnos más brillantes se encuentren en el mismo grupo). Como las muestras tienen distribución normal, se hizo el test F de igualdad de varianzas para los pretest dando P=0,59. Como las muestras cumplen los dos requisitos (normalidad e igualdad de varianza) usamos la prueba T de Student para comprobar si ambos grupos provienen de la misma población (en nuestro caso, del mismo aula y con conocimientos parecidos). Este test se basa en probar que las medias de los pretest no difieren significativamente. Aplicada la prueba se obtuvieron los siguientes resultados (véase Tabla 7):

Tabla 7. Prueba de T-student para el pretest

T	Grados de libertad	P
-1,9828	33	0,056

Al ser tan justo el valor de P en la Tabla 7 (recordamos que si es menor que 0,05 se rechaza la hipótesis y por tanto, se concluye que ambos grupos no son de la misma población), se realizó el cálculo del intervalo de confianza del 95% de la media (IC 95% de la media). Los resultados obtenidos se muestran en la Tabla 8:

Tabla 8. Intervalo de confianza del pretest

	Pretest	
GRUPO	Control	Experimental
IC 95%	4,38-6,07	5,54-7,02

Como ambos intervalos se solapan parcialmente, junto con la prueba de T-Student, nos permite afirmar que las diferencias entre ambos grupos son achacables al azar y por lo tanto pertenecen a la misma población.

El siguiente estudio consiste en analizar el efecto de la aplicación en su aprendizaje. En la Tabla 4 se muestran los resultados generales de la variable dependiente, que es la diferencia entre ambos test (nota del pretest menos nota del postest). Antes de hacer la prueba de T-Student, hay que probar la normalidad de las diferencias y la igualdad de varianzas. Se aplican los mismos test explicados anteriormente. Los resultados sobre normalidad son expuestos en la Tabla 9.

Tabla 9. Prueba de normalidad (P) de las diferencias

GRUPO	Control	Experimental
Shaphiro-Wilk	0,12	0,57
Kolmogorov-Smirnov	0,56	0,80

La igualdad de varianza, aplicando el test F da un valor de P = 0,77, por lo que las varianzas son semejantes.

Una vez comprobado la normalidad y la igualdad de varianzas de las diferencias de las notas obtenidas, se aplica la prueba de T-Student da los valores de la Tabla 10.

Tabla 10. Prueba de T-student para las diferencias

T	Grados de libertad	P
-0,9625	28	0,34

Para una mejor interpretación del resultado estadístico procedimos a realizar una gráfica con la distribución de ambos grupos (véase Figura 29)

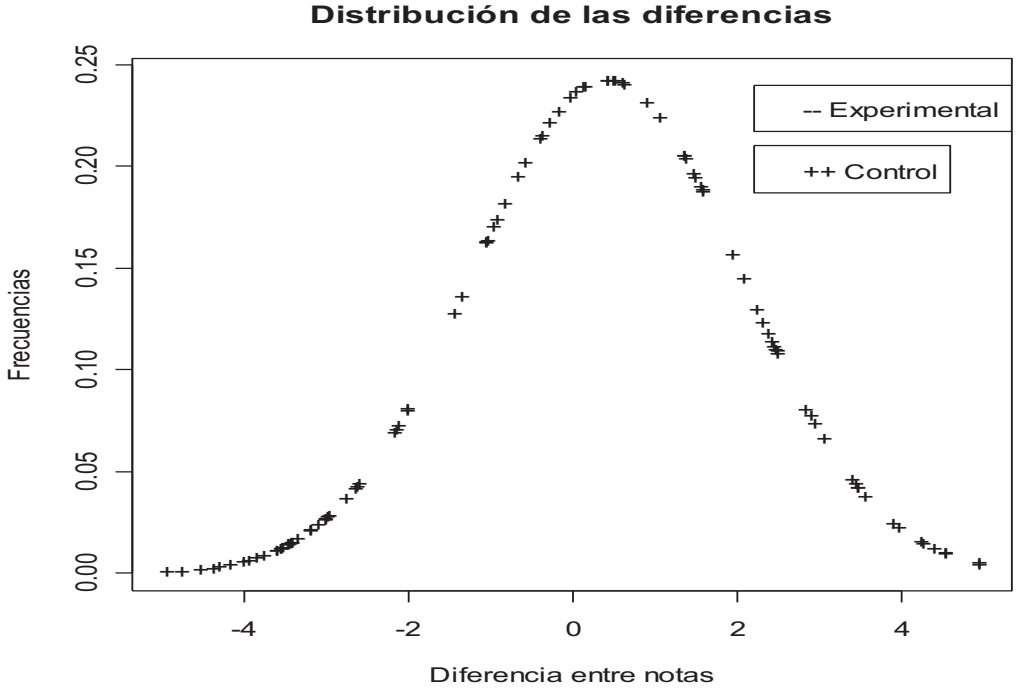


Figura 29. Distribución de las diferencias de notas entre ambos grupos

Para estudiar la segunda hipótesis (la influencia de la aplicación sobre los niveles de Bloom) se han realizado las mismas pruebas. Los resultados generales se encuentran agrupados en la Tabla 5 y las resultados obtenidos de la pruebas sobre normalidad en la Tabla 11 e igualdad de varianza y la T-Student en la Tabla 12.

Tabla 11. Prueba de normalidad (P) de las diferencias por nivel de Bloom

	Control			Experimental		
	N1	N2	N3	N1	N2	N3
Shaphiro-Wilk	0,07	0,00	0.06	0,13	0,01	0,00
Kolmogorov-Smirnov	0,27	0.31	0.47	0,72	0,34	0,17

Tabla 12. Pruebas de igualdad de varianzas y T-student

	N1	N2	N3
Test F (P)	0,8	0.54	0,46
T-STUDENT			
T	1,07	-2,15	-1,36
Grados de libertad	28	27	27
P	0,29	0.04	0,19

Hemos dibujado las gráficas en forma de diagrama de barras de las frecuencias de diferencias de notas agrupadas por niveles de la taxonomía. En la primera (véase Figura 30) se muestra esa frecuencia para el nivel de conocimiento. En la segunda (véase Figura 31) para el nivel de comprensión y la tercera (véase Figura 32) para el de aplicación.

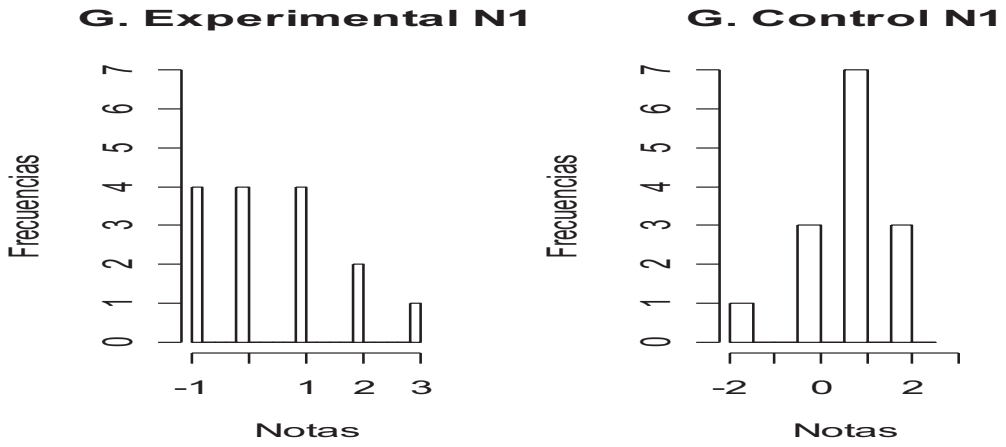


Figura 30. Frecuencia de notas para el nivel 1

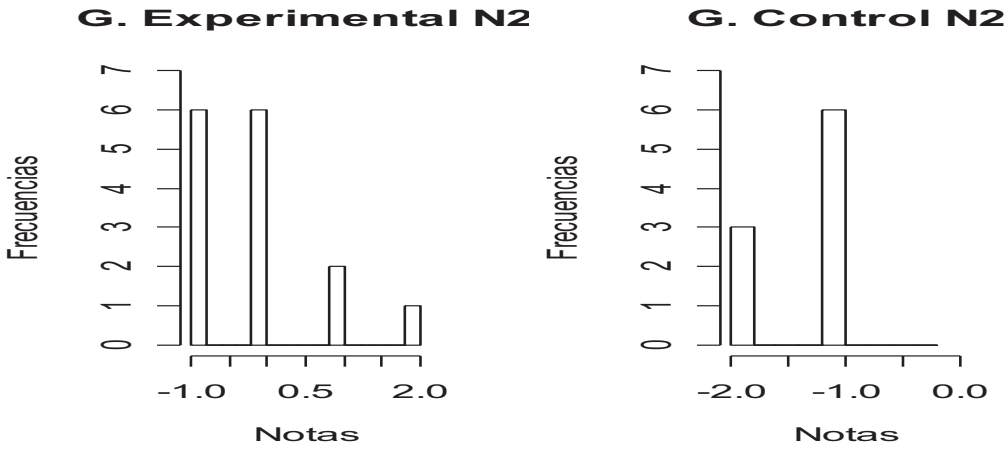


Figura 31. Frecuencia de notas para el nivel 2

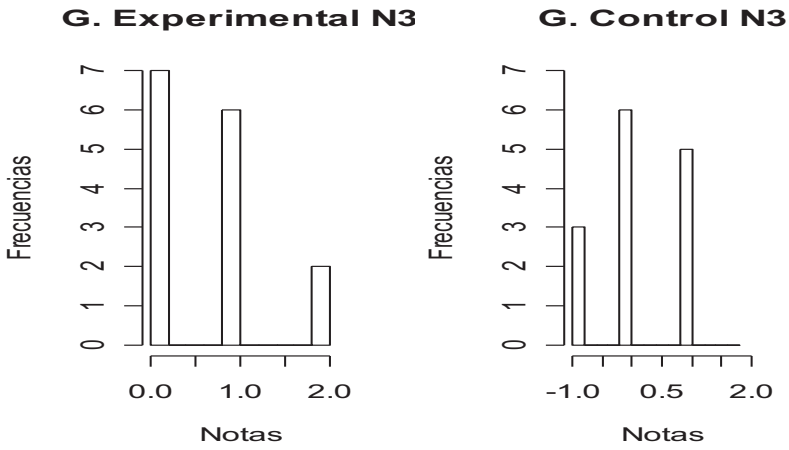


Figura 32. Frecuencia de notas para el nivel 3

Discusión de los resultados

Primero se intentó demostrar que la división realizada por grupos era correcta. Para ello, con las notas del pretest, se obtuvo que ambas muestras seguían distribuciones normales (Tabla 6) y presentaban varianzas homogéneas. Aplicada la prueba de T-Student (Tabla 7) y la prueba del IC 95% para contrastar las notas medias en el pretest, el resultado fue que no existían diferencias significativas entre ambos grupos. De esta forma nos aseguramos de que las diferencias entre notas del pretest y del postest sólo pueden deberse al uso de la aplicación.

El uso de la diferencia de notas entre los dos test como variable dependiente es una práctica habitual en este tipo de estudios, pues se eliminan factores externos al experimento, como la distinta dificultad de las preguntas entre ambos test (que de hecho existe según muestran las medias en la Tabla 3). Al restar las notas, cada alumno se mide a sí mismo.

Para comprobar la primera hipótesis (la aplicación influye significativamente en el aprendizaje del concepto de herencia), se han analizado las diferencias entre las notas totales del pretest y del postest. Como se muestra en la Tabla 4, la media del grupo experimental es mayor que la del grupo de control, lo que podría conducir erróneamente a pensar que el uso de la aplicación mejora considerablemente el aprendizaje. Sin embargo, el estudio estadístico muestra que no existen diferencias significativas entre ambos grupos (Tabla 10 y Figura 29). Esto no quiere decir que el uso de la herramienta no sea eficaz: habría que experimentar con un grupo de alumnos mayor para demostrar o refutar definitivamente la hipótesis.

La segunda hipótesis es que la herramienta refuerza el aprendizaje en algún nivel de Bloom. Los datos obtenidos en la Tabla 5 y los estudios estadísticos de la Tabla 11 y la Tabla 12 muestran que salvo para el nivel 2 (de

comprensión), las diferencias no son significativas. La prueba de T-Student para el nivel 2 arroja un P-valor de 0.04, que es menor que el valor fijado a priori de 0.05. Esto implica que existe una diferencia significativa (mejoría en el grupo experimental), no achacable al azar. Justamente nuestra aplicación tiene más componentes que apoyan al nivel de comprensión, que al resto de niveles.

Conclusiones

La herramienta para el aprendizaje del concepto de herencia ha mostrado su bondad para el nivel de comprensión, ya que dos de sus tres módulos dan soporte a este nivel. Según los resultados del experimento, el producto de las fases de la metodología propuesta en esta tesis (el objetivo marcado, la producción del material docente y la forma de presentación) es el adecuado para conseguir ayudar en el proceso de aprendizaje al alumno en el nivel 2.

También ha mostrado que la elección del tipo de software (problemas de tipo test así como las demostraciones) es apropiada para ese nivel. Dado que esta herramienta tiene preguntas de tipo test adecuadas para el nivel de aplicación y los resultados no muestran una diferencia significativa entre las notas de ambos grupos en ese nivel, se puede deducir que, o los test no han sido diseñados cuidadosamente para dar apoyo al nivel de aplicación, o no hay el número suficiente de preguntas de test del nivel 3 o los alumnos no usaron esas preguntas al considerarlas de una mayor dificultad.

5.1.2. Evaluación cualitativa

Para evaluar su usabilidad se utilizo un método de evaluación cualitativa, tratando de recoger las impresiones y sensaciones de los usuarios de la herramienta. Para ello se diseño un cuestionario con nueve preguntas concernientes a la instalación y uso de la misma. La recogida de datos tuvo lugar después de la utilización de la misma por los alumnos del grupo experimental.

Para medir el grado de satisfacción se usó la escala de Likert, desde totalmente de acuerdo (5) hasta totalmente en desacuerdo (1). Los resultados, junto con las preguntas, se ven en la Tabla 13.

Tabla 13. Evaluación de la herramienta (grupo experimental)

PREGUNTA	OPINION	Media
Es fácil de instalar	Totalmente de acuerdo	5
El uso es intuitivo	Totalmente de acuerdo	4,7
La explicación teórica es completa	Bastante de acuerdo	3,8
Las demostraciones me ayudan a comprender la materia	Bastante de acuerdo	4,2
La realización de tests me parece acertada para	Bastante de acuerdo	4,4

motivarme		
He notado que la dificultad aumenta con los niveles	Totalmente de acuerdo	4,5
La explicación de las respuestas de los tests la encuentro útil	Bastante de acuerdo	3,9
La usaría en casa para practicar	Bastante de acuerdo	4,4
A grandes rasgos, la herramienta me parece útil	Totalmente de acuerdo	4,5

Analizando los datos obtenidos, podemos obtener algunas conclusiones. Observando las notas medias menores, cabe reseñar como aspectos a mejorar de la aplicación, los siguientes: la explicación teórica y la explicación de las respuestas de los test. Es decir, el apoyo de la herramienta al nivel de conocimiento. El motivo de ello es que los autores de la aplicación dedicamos poco tiempo a recopilar información útil sobre el concepto teórico, ya que es una parte que, creíamos, los alumnos poseían suficientes recursos para ella.

Como partes a destacar, según los alumnos, están la facilidad de instalación y el uso intuitivo. El segundo factor es crucial para el éxito (o al menos, para no rechazar el uso) de cualquier aplicación interactiva.

Podemos concluir que según la opinión de los alumnos que usaron la aplicación, la impresión fue muy buena. Como se muestra en los resultados, es intuitiva y obtuvo una buena valoración global.

5.2. Evaluación de CreOO

CreOO es una herramienta de ayuda al estudio del mecanismo de la creación de objetos. Esta aplicación genera los enunciados, las respuestas posibles, la respuesta correcta y las explicaciones de forma automática (a través de plantillas). Para resolver los problemas, el estudiante debe haber entendido y comprendido el mecanismo de la creación de objetos. Está dirigida a alumnos con conocimientos básicos de POO.

En el caso de esta herramienta se hizo únicamente una evaluación cuantitativa, para determinar la eficacia educativa. La evaluación cualitativa no se realizó ya que la interfaz, la instalación y el funcionamiento son similares a los de la aplicación anterior.

5.2.1. Planteamiento del problema

El problema planteado es: se dispone de un software educativo de ayuda a la enseñanza del mecanismo de la creación de objetos en POO con Java y se quiere observar si el aprendizaje de ese mecanismo por parte de los alumnos que usan la herramienta es mayor que el de los alumnos que no lo utilizan. O dicho de otro modo, se pretende realizar un experimento que demuestre la eficacia pedagógica de esta herramienta.

Para poder resolver el problema se realizó un experimento consistente en realizar un test previo (pretest) a todos los alumnos para evaluar sus conocimientos anteriores al uso de la herramienta. Al siguiente día, se dividieron los alumnos que habitualmente asistían a clases prácticas en dos grupos potencialmente equivalentes, en términos de sexo y años de curso de la asignatura. Un grupo, que denominaremos experimental, utilizó la herramienta y otro grupo, que denominaremos de control, siguió el ritmo habitual de prácticas, que consiste en solucionar, implementado un programa, un supuesto práctico. Ambos grupos se alojaron en aulas separadas, por lo que no tuvieron contacto. Al finalizar las dos horas de clase, se les pasó un cuestionario, denominado postest, para medir el conocimiento adquirido.

5.2.2. Formulación de hipótesis

Queremos contrastar si las diferencias encontradas al comparar los dos grupos son lo suficientemente grandes como para que su única causa sea atribuible al azar. Por lo tanto, se plantearon dos hipótesis relacionadas:

H_{0T}: "El uso de la aplicación **no** influye significativamente en el aprendizaje global de los alumnos",

H_{0i}: "El uso de la aplicación **no** refuerza el aprendizaje en algún nivel *i* concreto de la taxonomía de Bloom", donde *i* va de 1 a 3 (nivel de conocimiento o recuerdo al nivel de aplicación).

Para ello se diseña un experimento cuantitativo, en el que se hará un estudio estadístico de los datos y se establece un intervalo de confianza para rechazar o aceptar la hipótesis del 95%.

5.2.3. Identificación de variables

Para realizar el estudio se tomaron dos variables: la aplicación es la variable independiente y la diferencia entre las notas obtenidas por cada grupo en la realización del pretest y del postest, la variable dependiente. Para comprobar la primera hipótesis H_{0T}, la diferencia de notas se obtiene de la puntuación final del test entero, mientras que para la segunda hipótesis H_{0i} las diferencias de notas se obtienen de los puntos obtenidos en cada nivel. El objetivo de la investigación es determinar la influencia de la primera sobre la segunda.

5.2.4. Población y muestra

La población objeto del estudio fueron los alumnos de la asignatura "Programación Orientada a Objetos" perteneciente al segundo curso de Ingeniería Informática de la Universidad Rey Juan Carlos en el curso 2006/2007.

Los 30 alumnos que asisten regularmente a las clases prácticas se dividieron en dos grupos: uno experimental y otro de control. Una primera

división se realizó alfabéticamente por apellidos. Luego se intentó reducir las posibles diferencias entre ambos, teniendo en cuenta el sexo (está demostrado que hombres y mujeres se enfrentan de forma distinta a los ordenadores [Carter 99]) y el número de años que los alumnos llevan cursando esta asignatura. Se intercambiaron individuos entre los grupos y de esta manera quedaron dos grupos de 15 miembros (14 alumnos y 1 alumna, con dos repetidores en cada grupo). Todos los alumnos han recibido las mismas clases teóricas del mismo profesor. Con esta división se intentó que los grupos fueran lo más homogéneos posible.

5.2.5. Obtención de los datos

El pretest (véase Apéndice B) se diseñó con 28 preguntas, de las cuales 14 estaban relacionadas con la creación de objetos. Estas 14 preguntas se encontraban divididas (no físicamente) en 3 niveles de dificultad, según la taxonomía de Bloom. En el nivel de conocimiento se encontraban 6, en el de comprensión otras 6 y en el de aplicación 4. La preparación de los ítems de los test se hizo cuidadosamente. Para garantizar los niveles, el profesor colaboró en las preguntas del nivel de conocimiento, para que tuvieran una redacción lo más parecida posible a lo explicado [Bloom 56]. El resto de las preguntas fueron enunciadas por los autores.

El postest (véase Apéndice B) fue diseñado únicamente con 14 preguntas que eran las mismas 14 preguntas relacionadas con la creación de objetos del pretest. Para la evaluación, las respuestas acertadas contaban 1 punto y las falladas 0 (máximo 14 puntos), es decir, no se penalizó el uso del azar. Las preguntas del nivel de conocimiento fueron propuestas por el profesor de la asignatura y el resto de las preguntas fueron enunciadas por los autores.

En la clase teórica, correspondiente al tema de clases y objetos, se pasó a ambos grupos de manera conjunta el pretest. Al día siguiente, en distintas aulas, los alumnos del grupo de control realizaron con el ordenador la práctica del tema. En contraposición, el grupo experimental instaló la aplicación y la usó durante aproximadamente una hora. Antes de finalizar la sesión práctica, se les entregó el postest para que lo contestaran.

5.2.6. Resultados

Los test fueron respondidos por los dos grupos. Se pueden observar los resultados en la Tabla 14 y en la Tabla 15 . Es de notar que se produjeron cuatro bajas en el grupo experimental y una en el de control, por lo que los datos de esos alumnos no se han tenido en cuenta en los resultados. Para el análisis de datos se usó la aplicación estadística SPSS.

Tabla 14. Resumen de los resultados del PRETEST

Grupo		Nivel1	Nivel2	Nivel3	Total
Control	N	14	14	14	14
	Media	5,214	4,714	3,357	13,285
	Mediana	5,000	5,000	3,000	13,000
	Varianza	0,489	0,835	0,247	2,066
	Desv. típ.	0,699	0,914	0,497	1,4373
Experimental	N	11	11	11	11
	Media	4,818	3,727	3,000	11,545
	Mediana	5,000	4,000	3,000	12,000
	Varianza	1,364	0,818	0,400	4,473
	Desv. típ.	1,167	0,905	0,632	2,1148
Total	N	25	25	25	25
	Media	5,040	4,280	3,200	12,520
	Mediana	5,000	4,000	3,000	13,000
	Varianza	0,873	1,043	0,333	3,760
	Desv. típ.	0,935	1,0214	0,577	1,939

Tabla 15. Resumen de los resultados del POSTEST

Grupo		Nivel1	Nivel2	Nivel3	Total
Control	N	14	14	14	14
	Media	5,071	4,857	3,357	13,285
	Mediana	5,000	5,000	3,000	13,500
	Varianza	0,379	0,901	0,247	1,451
	Desv. típ.	0,616	0,949	0,497	1,204
Experimental	N	11	11	11	11
	Media	5,181	4,545	3,363	13,090
	Mediana	5,000	5,000	3,000	13,000
	Varianza	0,564	0,473	0,255	2,091
	Desv. típ.	0,751	0,688	0,505	1,446
Total	N	25	25	25	25
	Media	5,120	4,720	3,360	13,200
	Mediana	5,000	5,000	3,000	13,000
	Varianza	0,443	0,710	0,240	1,667

	Desv. típ.	0,666	0,843	0,489	1,291

Para comprobar la eficacia de la herramienta, medimos el rendimiento de los estudiantes comparándolos con ellos mismos. Es decir, obtenemos la diferencia entre la nota obtenida en el postest y el pretest. La Tabla 16 contiene los estadísticos más representativos.

Tabla 16. Diferencias (POSTEST-PRETEST)

Grupo		Nivel1	Nivel2	Nivel3	Total
Control	N	14	14	14	14
	Media	-0,143	0,143	0,000	0,000
	Mediana	0,000	0,000	0,000	0,000
	Varianza	0,593	0,440	0,154	1,231
	Desv. típ.	0,770	0,663	0,392	1,109
Experimental	N	11	11	11	11
	Media	0,364	0,818	0,364	1,546
	Mediana	0,000	1,000	0,000	2,000
	Varianza	1,255	1,564	0,255	3,273
	Desv. típ.	1,120	1,250	0,505	1,809
Total	N	25	25	25	25
	Media	0,080	0,440	0,160	0,680
	Mediana	0,000	0,000	0,000	0,000
	Varianza	0,910	1,007	0,223	2,643
	Desv. típ.	0,954	1,003	0,473	1,626

5.2.7. Análisis de los resultados

En primer lugar, vamos a investigar si la división hecha es correcta, en el sentido de haber conseguido dos grupos homogéneos (con parecidos

conocimientos previos). Para ello realizamos el test de Levene (Tabla 17) para los resultados totales.

Tabla 17. Prueba de igualdad de varianzas

	Prueba de Levene para la igualdad de varianzas	
	F	Sig.
Total	2,171	,154

El nivel de significación de la prueba de Levene es mayor de 0.05 (valor mínimo). Por tanto, los grupos son homogéneos.

Para contrastar hipótesis correctamente, hay que demostrar la normalidad de las muestras. Aplicamos el método de Kolmogorov -Smirnov a las diferencias entre el postest y el pretest (Tabla 18).

Tabla 18. Prueba de normalidad

	Grupo	Kolmogorov-Smirnov(a)			Shapiro-Wilk		
		Estad	gl	Sig.	Estad	gl	Sig.
Nivel 1	Control	0,359	14	0,000	0,800	14	0,005
	Experimental	0,355	11	0,000	0,779	11	0,005
Nivel 2	Control	0,300	14	0,001	0,801	14	0,005
	Experimental	0,194	11	0,200	0,938	11	0,498
Nivel 3	Control	0,429	14	0,000	0,551	14	0,000
	Experimental	0,401	11	0,000	0,625	11	0,000
Total	Control	0,286	14	0,003	0,792	14	0,004
	Experimental	0,236	11	0,089	0,928	11	0,388

Debido al bajo valor de significación (sig.) de la mayoría de las muestras y al tener un tamaño muestral pequeño (menor que 30) no podemos usar métodos paramétricos (por ejemplo el test *t* de *student*) para realizar el contraste. En contraposición, usaremos métodos no paramétricos para verificar

o desechar las hipótesis principales H_{0T} y H_{0i} . En nuestro caso, usaremos la prueba U de Mann-Whitney. Los resultados se muestran en la Tabla 19.

Tabla 19. Contraste de hipótesis

	Nivel1	Nivel2	Nivel3	Total
U de Mann-Whitney	63,000	50,000	51,000	33,500
W de Wilcoxon	168,000	155,000	156,000	138,500
Z	-0,868	-1,558	-1,913	-2,443
Sig. asintót. (bilateral)	0,385	0,119	0,056	0,015

Los resultados obtenidos nos permiten aceptar las hipótesis planteadas para cada nivel (H_{01}, H_{02}, H_{03}). Es decir, con los datos obtenidos para la significación asintótica de los niveles 1, 2 y 3, todos mayores de 0.05, no podemos deducir que CreOO favorezca el aprendizaje en un nivel específico. Se observa que este estadístico disminuye a medida que se sube en la taxonomía de Bloom, hasta llegar a su valor fronterizo en el nivel de aplicación. Esta disminución puede indicar que CreOO tiene un efecto mayor en los niveles de comprensión y de aplicación que en el nivel de conocimiento. Sin embargo no hay que olvidar que estas diferencias pueden deberse a otros factores o al azar.
La significación de la diferencia de notas totales entre el pretest y el postest es menor de 0.05, así que tenemos que rechazar la hipótesis H_{0T}. En este caso podemos, por tanto, concluir que el uso de CreOO hace que las notas del grupo experimental son significativamente mejores (globalmente) que las del grupo de control. Si observamos la tabla 3, podíamos intuir este resultado, ya que la media de la diferencia (postest-pretest) del grupo experimental son todas positivas, mientras que las del grupo de control son negativas o cero (salvo en el nivel de comprensión). Pese a esa intuición, es necesario el análisis estadístico riguroso realizado para llegar a esta conclusión.

Discusión

La aplicación tiene cierta similitud con los tutores de programación desarrollados por Amruth Kumar. Sin embargo, éstos están dirigidos al nivel de aplicación [Dancik 03, Kumar 02].

Una limitación de CreOO es que sus plantillas y su mecanismo de generación son limitados. Con respecto a las primeras, se generan problemas distintos, todos tienen exactamente el mismo formato. Por tanto, el alumno que se dé cuenta de esta situación tras generar varios problemas puede terminar memorizando la forma de resolverlos. Con respecto a las opciones, una

generación puramente aleatoria tiene el riesgo de que sea fácil determinar la respuesta correcta mediante el descarte de opciones imposibles.

En cuanto al diseño del experimento, los resultados de la evaluación muestran que la clasificación de las preguntas de los tests por niveles se ha hecho de forma correcta. Observando las tablas 1 y 2 se ve que los alumnos han obtenido mejor nota media en el primer nivel que en el segundo nivel y este a su vez que el tercer nivel. Estos datos corroboran el hecho de que en la taxonomía de Bloom, cada nivel presupone la capacitación del alumno en los niveles precedentes.

5.2.8. Conclusiones

Hemos presentado las características principales de CreOO, una herramienta orientada al aprendizaje del mecanismo de creación de objetos en Java. También hemos descrito la forma de generar, corregir y comentar problemas. Hemos realizado una evaluación controlada de CreOO en el aula, con resultados satisfactorios a nivel general, puesto que se ha demostrado su eficacia educativa.

Podemos mejorar CreOO rediseñando las plantillas para que haya más variabilidad. En primer lugar, podrían generalizarse algunas partes de la plantilla. Por ejemplo, si permitiera tener atributos de tipos distintos, el número máximo de constructores aumentaría considerablemente. Así, con dos atributos de distintos tipos se pueden tener cinco constructores y con tres atributos, hasta 16.

También podríamos aumentar la complejidad de los problemas para que incluyan otros mecanismos de creación de objetos en Java, como inicializadores de atributos, bloques de inicialización, herencia, las palabras clave *this* y *super*, o polimorfismo.

Una última mejora consistiría en que las opciones de respuesta no se generen de forma aleatoria pura, sino que tengan en cuenta errores frecuentes de los alumnos. Otras experiencias han seguido esta línea, aunque de forma manual [Lister 01].

Capítulo 6 Conclusiones

Presentamos en este capítulo las conclusiones finales del trabajo realizado, relacionándolas con los objetivos que nos marcamos al inicio de este largo camino. Señalamos las ventajas y limitaciones de la metodología propuesta, que nos van a marcar los trabajos futuros, nuevas vías de investigación, fruto de esta tesis.

Las aplicaciones informáticas han sido usadas con profusión en la enseñanza debido a su alto potencial pedagógico. El hecho de poder contar con una herramienta educativa en cualquier momento y las restricciones de tiempo que tienen los profesores o tutores hace que los alumnos las usen cada vez más. Este creciente interés puede ser usado por:

- los diseñadores de software para realizar más programas con fines educativos. El problema general que surge es la falta de conocimientos pedagógicos
- los profesores de las distintas asignaturas. Los docentes son conscientes de este potencial, pero, a veces, no tienen los conocimientos técnicos adecuados de ingeniería del software y carecen de tiempo para su aprendizaje.

La taxonomía de Bloom es una jerarquía del conocimiento por la que los discentes van ascendiendo en su proceso de aprendizaje, empezando en el nivel más básico, nivel 1 o de conocimiento y llegando, en su extremo superior, al nivel de experto o nivel de evaluación. De esta forma se obtiene un marco general de trabajo para ir fijando objetivos pedagógicos claros, que ayuden realmente al alumno en su aprendizaje.

En el estudio del arte realizado, analizando las diversas herramientas de ayuda a la enseñanza de la programación existentes y poniendo mayor énfasis en el paradigma de la programación orientada a objetos, nos dimos cuenta que ese software se suele diseñar *ad hoc* para apoyar al estudio de un determinado concepto o de la programación genérica, pero falto de rigor pedagógico. Pocas

aplicaciones educativas fijan sus objetivos utilizando algún tipo de clasificación pedagógica y menos usan la taxonomía de Bloom como referencia.

Otro problema detectado en el estudio fue la falta de metodologías de diseño pedagógico de esas herramientas. En los artículos analizados no encontramos un método claro para diseñar las aplicaciones.

Por lo tanto, los trabajos realizados durante esta tesis han ido encaminados a arrojar un poco de luz sobre ambos aspectos, proponiendo una metodología de diseño de aplicaciones pedagógicas orientada a los tres primeros niveles de la taxonomía de Bloom (las primeras etapas del aprendizaje) y comprobar científicamente su validez.

6.1. Relación entre hipótesis, objetivos y logros.

Con el ánimo de contribuir al diseño y desarrollo sistemático de aplicaciones educativas surgió este trabajo de investigación, planteando la **hipótesis**:

"Es factible la especificación de una metodología para el diseño sistemático de aplicaciones de ayuda a enseñanza de la programación orientada a objetos en un nivel especifico de la taxonomía de Bloom, que permita obtener un producto eficaz pedagógicamente hablando"

De ella se dedujeron el **objetivo principal** y los **subobjetivos** descritos en el primer capítulo y que aquí repetimos:

"La especificación de una metodología para el diseño sistemático de aplicaciones de ayuda a enseñanza de la programación orientada a objetos en un nivel específico de la taxonomía de Bloom y la demostración de su aplicación y de su eficacia pedagógica".

Primer objetivo parcial:

"Estudio de las distintas herramientas de ayuda a la enseñanza de la programación, centrándonos en la programación orientada a objetos y relacionándolas con el nivel de Bloom que soportan, mostrando sus principales características y carencias."

Segundo objetivo parcial:

"Especificación de la metodología para el diseño sistemático de aplicaciones de ayuda a enseñanza de la programación orientada a objetos en un nivel específico de la taxonomía de Bloom, identificando los diferentes aspectos a considerar en el desarrollo de la aplicación en función del nivel de Bloom al que ofrezca soporte."

Tercer objetivo parcial:

"Desarrollar herramientas pedagógicas para la ayuda a la enseñanza y aprendizaje de la programación orientada a objetos basadas en esta metodología"

Cuarto objetivo parcial:

"Evaluación cuantitativa y cualitativa, estudio de resultados y posterior validación de la eficacia pedagógica de las herramientas desarrolladas usando esta metodología mediante evaluaciones de su uso por parte de los estudiantes de diversos cursos de la asignatura de programación orientada a objetos."

El primer subobjetivo se ha conseguido y ha sido reflejado en el segundo capítulo de la tesis. De ese estudio, se han reflejado las carencias existentes en la actualidad del software educativo y se han deducido características principales que tienen que tener las aplicaciones educativas para que tengan ciertas garantías de éxito.

En la consecución del segundo objetivo parcial, se han hallado algunos elementos esenciales para realizar una correcta metodología de diseño del software educativo para la ayuda a la enseñanza de la programación en uno de los tres primeros niveles de la taxonomía de Bloom. Uno de ellos es fijar un objetivo pedagógico claro, usando la taxonomía. De esta forma, el diseñador puede pensar en la herramienta como una aplicación encaminada a ayudar en ese nivel. Otro elemento es la preparación del material didáctico. Normalmente el concepto a enseñar es complejo. En el capítulo tercero hemos dado las bases

para descomponerlo en partes más simples y poder enfocar la herramienta a cada una de las partes. Un elemento esencial más es la forma en la que se deben preparar las cuestiones encaminadas a practicar o evaluar un concepto (o parte de él). En el mismo capítulo hemos ofrecido un método para la correcta construcción de preguntas. Por último, hemos desarrollado una metodología de diseño orientada a los tres primeros niveles de Bloom, aunando los elementos anteriores.

Una vez descrita la metodología, se han diseñado y desarrollado varias aplicaciones siguiendo la línea trazada. De esta manera el tercer objetivo ha sido alcanzado. En concreto se han realizado cinco programas educativos (y dos versiones mejoradas) para la ayuda a la enseñanza-aprendizaje de la programación orientada a objetos, en concreto para conceptos esenciales de la misma como son la herencia, la construcción de clases y la creación de objetos. En el capítulo cuarto han sido descritos en detalle, comentando cada fase de diseño y el resultado final obtenido.

El último subobjetivo también ha sido alcanzado. Hemos realizado evaluaciones de dos de las herramientas desarrolladas que han sido descritas en el quinto capítulo. Cada evaluación ha constado de dos partes, una cuantitativa y una cualitativa para cada uno del software educativo. En la primer tipo de evaluación se ha medido con éxito la eficacia pedagógica de las herramientas evaluadas. Para ello se ha usado una metodología de investigación científica descrita en el primer capítulo. El segundo tipo de evaluación (cualitativa) ha reflejado la opinión de los usuarios en aspectos como la facilidad de uso, la motivación y la utilidad obteniendo ambas aplicaciones buenas valoraciones.

La consecución de todos los objetivos parciales nos han conducido a obtener el objetivo principal de la tesis y ha poder demostrar la hipótesis de esta tesis. Se puede afirmar que es posible diseñar y desarrollar de forma sistemática software educativo eficaz pedagógicamente hablando.

6.2. Trabajos futuros

La pedagogía aplicada a la Informática abre muchas líneas de investigación futuras. La metodología de mostrada en este trabajo se circunscribe a unas aplicaciones concretas de ayuda a la enseñanza. Estas aplicaciones son de uso individual y permiten al alumno, en general, elegir la presentación de contenido o preguntas a su ritmo de estudio.

Una de las líneas futuras se abre a las herramientas de uso compartido y, por lo tanto, al trabajo colaborativo. Es bien conocido las ventajas que tiene el trabajo colaborativo [Slavin 90] como promover la cooperación, la interacción y la familiaridad entre estudiantes y profesores. Además, desde el punto de vista del investigador, los entornos colaborativos facilitan el desarrollo de habilidades de razonamiento tales como hacer explicitas las ideas, discutir e interactuar con otros estudiantes para construir una solución común, etc. También se incrementa la motivación, participación y autoestima del discente cuando obtienen buenos resultados en sus actividades colaborativas. El aprendizaje colaborativo crea un entorno seguro en el cual los estudiantes pueden expresar sus propias ideas sin temor a los fallos o a las críticas, ayudándole a desarrollar sus habilidades comunicativas. Sus compañeros pueden hacer críticas constructivas a las diferentes ideas aportadas por el grupo, mientras que el profesor puede evaluar el proceso de aprendizaje como un todo (incluyendo el proceso de razonamiento) y no teniendo que ceñirse al resultado final de la actividad. De esta forma, nos parece interesante crear una metodología que permita diseñar aplicaciones pedagógicas que den soporte al trabajo colaborativo y fije sus objetivos educativos usando la taxonomía de Bloom.

Otro trabajo futuro surge con la adaptación de la herramienta al alumno. Como ya hemos comentado, es el propio discente quien elige el material didáctico que se presenta. Se puede pensar en una aplicación que recoja las características del alumno y pueda ir adaptando de forma automática y dinámica su contenido al perfil del alumno. De esta forma, la metodología debería describir una fase donde se hagan explicitas las características más importantes del perfil del usuario que deberían ser recogidas por la aplicación para adaptarse de forma dinámica, de acuerdo al discente y a su actividad con la herramienta. Existen muchos trabajos sobre software adaptativo entre los que podemos destacar el trabajo de Rosa Carro y su equipo [Carro 99, Carro 00]. La línea futura que proponemos es unir las características del alumno con el nivel de Bloom al que pretende dar soporte la herramienta y de esta manera, mostrar el contenido según ciertas reglas prefijadas en el momento del diseño.

En la misma línea se abre la posibilidad de incluir otra fase donde tuviera cabida la sugerencia o recomendación de actividades. Aunque en este campo ya hay trabajos realizados [Martín 08], la recomendación solo se produce de acuerdo al perfil del alumno, del grupo y del contexto, pero no se tiene de nuevo en cuenta el objetivo pedagógico según la taxonomía de Bloom. Sería un trabajo interesante poder unir los dos elementos para poder tener una

herramienta completamente adaptativa, de acuerdo al discente, al uso de la aplicación y al nivel de Bloom al que se pretende dar soporte.

Un trabajo que se presenta a corto plazo es extender esta metodología para los niveles superiores de la taxonomía de Bloom. Hemos comentado que ya existen herramientas que cubren los niveles de análisis y diseño, pero un diseño sistemático de las mismas podría abrir nuevas vías a nuevas herramientas y facilitar su desarrollo para la enseñanza de conceptos concretos.

Otro trabajo futuro que queda por realizar es medir el esfuerzo que supone al diseñador o profesor la aplicación de la metodología propuesta en esta tesis. Recordamos que para que una aplicación realizada *ad-hoc* tenga éxito, es muy importante que no suponga un esfuerzo excesivo su desarrollo o que la relación esfuerzo-resultado obtenido merezca la pena.

El aprendizaje de la programación conlleva la correcta realización de varias tareas, como lectura y comprensión de código, efectuar trazas de programas, predecir resultados, analizar datos, calcular complejidades, realizar cambios en el código para producir otras salidas, programar… Sería interesante hacer un estudio similar al expuesto en este trabajo de investigación, pero enfocado a potenciar cada una de estas tareas. De esta forma, poder proponer herramientas de ayuda al aprendizaje adecuadas a cada quehacer en programación.

Apéndice A

Estudio comparativo sobre la aplicación de la taxonomía de Bloom

Presentamos en este apéndice el estudio comparativo sobre la aplicación de la taxonomía de Bloom para valorar distintos tipos de preguntas realizadas en las pruebas de evaluación de cinco asignaturas del primer curso de Ingeniería Informática. Este trabajo consistió en:

Primero: Introducción de la taxonomía de Bloom. Se explicó a cada profesor de las distintas asignaturas en qué consistía la taxonomía de Bloom y los distintos niveles jerárquicos que poseen.

Segundo: Recopilación de pruebas de evaluación. Cada profesor de esas cinco asignaturas aporto varios exámenes, compuestos de varias preguntas. En concreto se obtuvieron 144 preguntas de 9 pruebas de evaluación.

Tercero: Valoración de cada pregunta. El profesor y tres expertos en la taxonomía de Bloom valoraron de forma independiente cada problema de las pruebas de evaluación aportadas. Cada profesor únicamente valoró las preguntas de sus exámenes, mientras los expertos puntuaron las 144 preguntas.

Cuarto: Estudio de los resultados.

A.1. Valoración de cada pregunta.

Aquí se muestra en una tabla (véase Tabla 20) la valoración dada por los profesores y expertos a cada una de las preguntas de los exámenes aportados.

Tabla 20. Valoración de cada pregunta

	Exp1	Exp2	Exp3	Profesor

ASIGNATURA	FECHA				
BM	sep-06	3	3	4	3
BM	sep-06	3	2	4	2
BM	sep-06	3	3	4	2
BM	sep-06	3	3	4	2
BM	sep-06	2	2	4	2
BM	sep-06	3	3	4	2
BM	sep-06	3	3	4	2
BM	sep-06	3	2	4	2
BM	sep-06	2	2	4	2
BM	sep-06	3	4	4	3
BM	sep-06	3	3	4	3
BM	sep-06	2	2	4	2
BM	sep-06	2	2	4	2
BM	sep-06	2	2	4	2
BM	sep-06	3	2	4	2
BM	feb-07	2	4	4	2
BM	feb-07	3	3	4	2
BM	feb-07	3	3	3	2
BM	feb-07	3	3	3	1
BM	feb-07	3	4	4	3
BM	feb-07	2	3	4	2
BM	feb-07	2	3	4	2
BM	feb-07	2	3	4	2
BM	feb-07	3	3	4	2

BM	feb-07	3	3	4	2
MD	feb-07	2	3	3	2
MD	feb-07	2	3	3	2
MD	feb-07	3	3	3	3
MD	feb-07	2	3	2	1
MD	feb-07	2	3	2	1
MD	feb-07	2	3	2	2
MD	feb-07	2	2	2	1
MD	feb-07	2	2	4	3
MD	feb-07	2	2	4	3
MD	feb-07	3	5	3	3
MD	feb-07	2	3	4	2
MD	feb-07	2	2	3	2
MD	feb-07	2	2	2	1
MD	feb-07	2	2	3	2
MD	sep-06	3	3	3	2
MD	sep-06	3	3	3	3
MD	sep-06	3	3	4	3
MD	sep-06	3	2	4	3
MD	sep-06	3	2	4	3
MD	sep-06	3	4	3	3
MD	sep-06	3	3	3	3
MD	sep-06	3	2	3	2
MD	sep-06	2	3	2	2
MD	sep-06	2	2	2	2

MD	sep-06	2	2	3	2
AL	may-06	4	3	4	4
AL	may-06	3	3	4	3
AL	may-06	3	3	4	3
AL	may-06	3	3	4	4
AL	may-06	2	3	4	4
AL	may-06	3	3	3	4
AL	may-06	3	3	3	4
AL	sep-06	3	3	4	4
AL	sep-06	3	3	4	4
AL	sep-06	2	3	3	4
AL	sep-06	2	3	3	4
AL	sep-06	3	3	3	4
AL	sep-06	3	2	4	4
AL	sep-06	3	3	4	4
AL	sep-06	3	3	3	4
AL	sep-06	3	3	3	5
AL	jun-06	3	3	4	3
AL	jun-06	3	2	4	4
AL	jun-06	3	3	3	4
AL	jun-06	3	3	3	5
AL	jun-06	2	3	3	5
AL	jun-06	3	3	3	5
AL	jun-06	3	3	3	5
AL	jun-06	3	3	2	4

BLP	jun-06	1	2	1
BLP	jun-06	1	2	1
BLP	jun-06	1	1	1
BLP	jun-06	1	1	2
BLP	jun-06	2	2	1
BLP	jun-06	2	2	1
BLP	jun-06	2	2	2
BLP	jun-06	2	2	2
BLP	jun-06	3	2	3
BLP	jun-06	1	2	1
BLP	jun-06	3	4	4
BLP	jun-06	3	2	4
BLP	jun-06	3	3	3
BLP	jun-06	2	2	2
BLP	jun-06	2	2	3
BLP	jun-06	3	3	4
BLP	sep-06	1	1	1
BLP	sep-06	1	1	1
BLP	sep-06	1	1	1
BLP	sep-06	1	1	2
BLP	sep-06	2	2	2
BLP	sep-06	2	2	1
BLP	sep-06	2	2	2
BLP	sep-06	2	2	2
BLP	sep-06	3	2	3

BLP	sep-06	1		2	1
BLP	sep-06	3		3	3
BLP	sep-06	2		2	4
BLP	sep-06	3		2	4
BLP	sep-06	3		3	4
BLP	sep-06	2		2	2
BLP	sep-06	2		2	1
BLP	sep-06	3		3	4
MTP	jun-06		1	2	2
MTP	jun-06		2	2	2
MTP	jun-06		2	2	1
MTP	jun-06		2	2	2
MTP	jun-06		1	2	1
MTP	jun-06		1	2	2
MTP	jun-06		3	2	3
MTP	jun-06		3	2	3
MTP	jun-06		1	2	2
MTP	jun-06		3	2	2
MTP	jun-06		4	2	2
MTP	jun-06		4	3	3
MTP	jun-06		4	3	3
MTP	jun-06		4	2	2
MTP	jun-06		4	3	3
MTP	jun-06		4	3	3
MTP	jun-06		4	3	3

MTP	sep-06	1	2	1
MTP	sep-06	1	2	1
MTP	sep-06	3	2	2
MTP	sep-06	3	2	2
MTP	sep-06	2	2	1
MTP	sep-06	3	2	2
MTP	sep-06	3	2	3
MTP	sep-06	2	2	1
MTP	sep-06	3	2	2
MTP	sep-06	3	2	1
MTP	sep-06	4	2	2
MTP	sep-06	4	3	3
MTP	sep-06	4	2	2
MTP	sep-06	4	3	3
MTP	sep-06	4	2	2
MTP	sep-06	4	3	3
MTP	sep-06	4	3	3
MTP	sep-06	4	3	3

A continuación se calculó la diferencia (en valor absoluto) de los niveles otorgados entre todas las posibles parejas formadas entre expertos y el profesor para ver la disparidad o igualdad entre las valoraciones (véase Tabla 21). En las últimas filas se muestra la media de las diferencias. Comentamos que la diferencia ideal es 0, ya que eso significa que ambas personas de la pareja han coincido en su valoración.

Tabla 21. Diferencias absolutas de las valoraciones

		Exp1-Exp2	Exp1-Exp3	Exp2-Exp3	Exp1-Prof	Exp2-Prof	Exp3-Prof
ASIGNAT	**FECHA**						
BM	sep-06	0	1	1	0	0	1
BM	sep-06	1	1	2	1	0	2
BM	sep-06	0	1	1	1	1	2
BM	sep-06	0	1	1	1	1	2
BM	sep-06	0	2	2	0	0	2
BM	sep-06	0	1	1	1	1	2
BM	sep-06	0	1	1	1	1	2
BM	sep-06	1	1	2	1	0	2
BM	sep-06	0	2	2	0	0	2
BM	sep-06	1	1	0	0	1	1
BM	sep-06	0	1	1	0	0	1
BM	sep-06	0	2	2	0	0	2
BM	sep-06	0	2	2	0	0	2
BM	sep-06	0	2	2	0	0	2
BM	sep-06	1	1	2	1	0	2
BM	feb-07	2	2	0	0	2	2
BM	feb-07	0	1	1	1	1	2
BM	feb-07	0	0	0	1	1	1
BM	feb-07	0	0	0	2	2	2
BM	feb-07	1	1	0	0	1	1
BM	feb-07	1	2	1	0	1	2
BM	feb-07	1	2	1	0	1	2

BM	feb-07	1	2	1	0	1	2
BM	feb-07	0	1	1	1	1	2
BM	feb-07	0	1	1	1	1	2
MD	feb-07	1	1	0	0	1	1
MD	feb-07	1	1	0	0	1	1
MD	feb-07	0	0	0	0	0	0
MD	feb-07	1	0	1	1	2	1
MD	feb-07	1	0	1	1	2	1
MD	feb-07	1	0	1	0	1	0
MD	feb-07	0	0	0	1	1	1
MD	feb-07	0	2	2	1	1	1
MD	feb-07	0	2	2	1	1	1
MD	feb-07	2	0	2	0	2	0
MD	feb-07	1	2	1	0	1	2
MD	feb-07	0	1	1	0	0	1
MD	feb-07	0	0	0	1	1	1
MD	feb-07	0	1	1	0	0	1
MD	sep-06	0	0	0	1	1	1
MD	sep-06	0	0	0	0	0	0
MD	sep-06	0	1	1	0	0	1
MD	sep-06	1	1	2	0	1	1
MD	sep-06	1	1	2	0	1	1
MD	sep-06	1	0	1	0	1	0
MD	sep-06	0	0	0	0	0	0
MD	sep-06	1	0	1	1	0	1

MD	sep-06	1	0	1	0	1	0
MD	sep-06	0	0	0	0	0	0
MD	sep-06	0	1	1	0	0	1
AL	may-06	1	0	1	0	1	0
AL	may-06	0	1	1	0	0	1
AL	may-06	0	1	1	0	0	1
AL	may-06	0	1	1	1	1	0
AL	may-06	1	2	1	2	1	0
AL	may-06	0	0	0	1	1	1
AL	may-06	0	0	0	1	1	1
AL	sep-06	0	1	1	1	1	0
AL	sep-06	0	1	1	1	1	0
AL	sep-06	1	1	0	2	1	1
AL	sep-06	1	1	0	2	1	1
AL	sep-06	0	0	0	1	1	1
AL	sep-06	1	1	2	1	2	0
AL	sep-06	0	1	1	1	1	0
AL	sep-06	0	0	0	1	1	1
AL	sep-06	0	0	0	2	2	2
AL	jun-06	0	1	1	0	0	1
AL	jun-06	1	1	2	1	2	0
AL	jun-06	0	0	0	1	1	1
AL	jun-06	0	0	0	2	2	2
AL	jun-06	1	1	0	3	2	2
AL	jun-06	0	0	0	2	2	2

AL	jun-06	0	0	0	2	2	2
AL	jun-06	0	1	1	1	1	2
BLP	jun-06	1	1	2	0	1	1
BLP	jun-06	1	1	2	0	1	1
BLP	jun-06	1	0	1	0	1	0
BLP	jun-06	1	0	1	1	2	1
BLP	jun-06	2	0	2	1	1	1
BLP	jun-06	2	0	2	1	1	1
BLP	jun-06	2	0	2	0	2	0
BLP	jun-06	2	0	2	0	2	0
BLP	jun-06	3	1	2	0	3	1
BLP	jun-06	1	1	2	0	1	1
BLP	jun-06	3	1	4	1	4	0
BLP	jun-06	3	1	2	1	4	2
BLP	jun-06	3	0	3	0	3	0
BLP	jun-06	2	0	2	0	2	0
BLP	jun-06	2	0	2	1	3	1
BLP	jun-06	3	0	3	1	4	1
BLP	sep-06	1	0	1	0	1	0
BLP	sep-06	1	0	1	0	1	0
BLP	sep-06	1	0	1	0	1	0
BLP	sep-06	1	0	1	1	2	1
BLP	sep-06	2	0	2	0	2	0
BLP	sep-06	2	0	2	1	1	1
BLP	sep-06	2	0	2	0	2	0

BLP	sep-06	2	0	2	0	2	0
BLP	sep-06	3	1	2	0	3	1
BLP	sep-06	1	1	2	0	1	1
BLP	sep-06	3	0	3	0	3	0
BLP	sep-06	2	0	2	2	4	2
BLP	sep-06	3	1	2	1	4	2
BLP	sep-06	3	0	3	1	4	1
BLP	sep-06	2	0	2	0	2	0
BLP	sep-06	2	0	2	1	1	1
BLP	sep-06	3	0	3	1	4	1
MTP	jun-06	1	2	1	2	1	0
MTP	jun-06	2	2	0	2	0	0
MTP	jun-06	2	2	0	1	1	1
MTP	jun-06	2	2	0	2	0	0
MTP	jun-06	1	2	1	1	0	1
MTP	jun-06	1	2	1	2	1	0
MTP	jun-06	3	2	1	3	0	1
MTP	jun-06	3	2	1	3	0	1
MTP	jun-06	1	2	1	2	1	0
MTP	jun-06	3	2	1	2	1	0
MTP	jun-06	4	2	2	2	2	0
MTP	jun-06	4	3	1	3	1	0
MTP	jun-06	4	3	1	3	1	0
MTP	jun-06	4	2	2	2	2	0
MTP	jun-06	4	3	1	3	1	0

MTP	jun-06	4	3	1	3	1	0
MTP	jun-06	4	3	1	3	1	0
MTP	sep-06	1	2	1	1	0	1
MTP	sep-06	1	2	1	1	0	1
MTP	sep-06	3	2	1	2	1	0
MTP	sep-06	3	2	1	2	1	0
MTP	sep-06	2	2	0	1	1	1
MTP	sep-06	3	2	1	2	1	0
MTP	sep-06	3	2	1	3	0	1
MTP	sep-06	2	2	0	1	1	1
MTP	sep-06	3	2	1	2	1	0
MTP	sep-06	3	2	1	1	2	1
MTP	sep-06	4	2	2	2	2	0
MTP	sep-06	4	3	1	3	1	0
MTP	sep-06	4	2	2	2	2	0
MTP	sep-06	4	3	1	3	1	0
MTP	sep-06	4	2	2	2	2	0
MTP	sep-06	4	3	1	3	1	0
MTP	sep-06	4	3	1	3	1	0
MTP	sep-06	4	3	1	3	1	0

MEDIA	1,401	1,056	1,161	0,992	1,197	0,823
DESVIACIÓN TÍPICA	1,337	0,947	0,827	0,967	0,973	0,753
VARIANZA	1,789	0,898	0,685	0,936	0,947	0,567

A.2. Estudio de los resultados.

Las medias de las parejas son muy parecidas, obteniendo la mayor diferencia (1,401) entre las valoraciones del experto 1 con el experto 2 y la menor diferencia (0,823) entre el experto 3 y el profesor. Esto corrobora los estudios del propio Benjamín Bloom [Bloom 56] cuando afirma que hay una disparidad de opiniones a la hora de aplicar la taxonomía y decidir en qué nivel de la jerarquía se encuentra cada pregunta. También Colin Johnson y Ursula Fuller [Johnson 07] llegaron a la misma conclusión al intentar aplicar la taxonomía al campo de la Informática. La discordancia hallada en el estudio realizado por el autor de esta tesis puede ser debida bien al desconocimiento de la materia o del contexto por parte de los expertos o bien a la falta de destreza en el uso de la taxonomía por parte del profesor.

Si suponemos que la valoración hecha por el discente es la correcta, una consecuencia que se debe sacar de este estudio es el hecho que para situar de forma correcta es imprescindible saber del contexto en el que se va a aplicar ese problema. El conocimiento previo que tienen los alumnos es fundamental para situar una pregunta en un determinado nivel y el que más conocimiento tiene sobre ese aspecto es sin duda alguna el profesor. Es, por este motivo, que el discente es una pieza clave en el diseño de una herramienta de ayuda a la enseñanza basada en la taxonomía de Bloom.

Si suponemos que la valoración hecha por uno de los expertos es correcta (aquí decimos uno de los expertos porque el estudio muestra disparidad entre los expertos en la aplicación de la taxonomía), podríamos concluir que el uso de la taxonomía al campo de la Informática por parte del profesor no es fácil. La discrepancia de las valoraciones muestra una evidente dificultad en su aplicación a este campo, bien por motivos terminológicos (algunos nombres de niveles coinciden con tareas de la programación) bien porque el dominio de la programación es amplio.

Apéndice B

Presentamos en este apéndice las preguntas de los pretest y postest planteados a los alumnos para realizar los experimentos para la evaluación cuantitativa de la herramienta pedagógica de ayuda al aprendizaje del concepto de herencia y de CreOO, por si son de utilidad en posteriores estudios.

B.1. Pretest de la herramienta de herencia.

1. El concepto de herencia implica:
    a. Transmisión de atributos y métodos privados
    b. Transmisión de métodos privados y públicos
    c. Transmisión de atributos y métodos
    d. Ninguna de las anteriores

2. Los atributos de una clase abstracta deben ser:
    a. Públicos
    b. Privados
    c. Con visibilidad por defecto
    d. Protegidos

3. La redefinición de métodos:
    a. Es una forma de especialización de una clase hija
    b. Se puede dar sin herencia
    c. Es posible para métodos privados
    d. Ninguna de las anteriores

4. Se tiene una clase denominada EquipoFutbol, que como atributos tiene todos los datos y los métodos referentes a un equipo de fútbol (presupuesto, jugadores, estadio, comprarJugador( ), venderJugador( ), etc.). Si queremos crear un equipo (por ejemplo, el SevillaFC) ¿Qué afirmación es verdadera?

    a. SevillaFC será una clase que hereda de EquipoFutbol.
    b. SevillaFC será un objeto de EquipoFutbol.
    c. SevillaFC imlpementará la interface EquipoFutbol.
    d. SevillaFC será una clase independiente de EquipoFutbol.

5. Dado el siguiente código se puede afirmar que:

```
class Plato {
```

```
    String material;

    …

 }

class PlatoCena extends Plato {

    String forma;

    …

   public static void main(String[] args){

      PlatoCena pc=new PlatoCena();

      Plato plato = new Plato();

    }

}
```

a. Se puede usar plato.forma para saber la forma del plato.
b. La clase Plato hereda la propiedad forma de PlatoCena
c. Se puede usar pc.material para saber de que esta hecho el plato de cena.
d. La clase PlatoCena implementa la interfaz Plato

6. Dado el siguiente código se puede afirmar que:

```
interface Musica {

    void tocar( );

    void componer( );

 }

class Rock implements Musica {

    void tocar( ){

        …

     };

  }
```

a. Se produce herencia múltiple, pues la clase Rock implementa la interface Musica.
b. Se produce herencia por implantación, pues Rock implementa el método tocar() de la interface Musica.
c. Se produce un error, pues Rock tiene que implementar todos los métodos de la interface Musica.
d. Se produce herencia simple, pues Rock solo hereda de Musica.

7. Dado el siguiente código, marque la opción correcta:

```
import gestor.GestorIO;
class ClaseBase {
   private int cb1;
   private int cb2;
   public ClaseBase(int cb1) {
      this.cb1 = cb1;
   }
   public ClaseBase(int cb1, int cb2) {
      this(cb1);
      this.cb2 = cb2;
   }
   public void mostrar() {
      GestorIO io = new GestorIO();
      io.out("valor 1 (cb): "+this.cb1);
      io.out("valor 2 (cb): "+this.cb2);
   }
}
class ClaseDerivada extends ClaseBase {
   private int cd1;
   private int cd2;
   private int cd3;
  public void mostrar() {
      super.mostrar();
      GestorIO io = new GestorIO();
      io.out("valor 1 (cd): "+this.cd1);
      io.out("valor 2 (cd): "+this.cd2);
      io.out("valor 3 (cd): "+this.cd3);
   }
}
```

a. Si se crea un objeto de la ClaseDerivada se ejecuta su constructor por defecto, y a la vez el primer constructor de ClaseBase
b. Si se crea un objeto de la ClaseDerivada se ejecuta únicamente el primer constructor de la ClaseBase.
c. El compilador dará un error, debido a que ClaseDerivada no tiene constructor por lo que no sabe que constructor de la ClaseBase ha de ejecutar.
d. El compilador no dará un error ya que aunque ClaseDerivada no tiene constructor con parámetros, la ClaseBase sí.

8. Si a la clase Derivada del ejercicio anterior se le añade el constructor siguiente, elija que opción es correcta:

```
public ClaseDerivada(int i) {
     super(i);
     this.cd1 = i;
   }
```

a. Con la instrucción ClaseDerivada clased=new ClaseDerivada(7) clased.cb1=7 y clased.cd1=7.
b. Con la instrucción ClaseDerivada clased=new ClaseDerivada(7) clased.cb1=0 y clased.cd1=7.
c. Con la instrucción ClaseDerivada clased=new ClaseDerivada(7) clased.cb1=7 y clased.cd1=0.
d. Con la instrucción ClaseDerivada clased=new ClaseDerivada(7) clased.cb1=0 y clased.cd1=0.

9. Sea el código del ejercicio 7 con el constructor del ejercicio 8. Cuándo se ejecuta ClaseDerivada clased=new ClaseDerivada(7) y clased.mostrar(), ¿Qué saldrá en pantalla?

a. Nada, pues hay un error en el código.

b. valor 1 (cb): 7
valor 2 (cb): 0

valor 1 (cd): 7

valor 2 (cd): 0

valor 3 (cd): 0

c. valor 1 (cd): 7
valor 2 (cd): 0

valor 3 (cd): 0

valor 1 (cb): 7

valor 2 (cb): 0

d. valor 1 (cb): 0
valor 2 (cb): 0

valor 1 (cd): 0

valor 2 (cd): 0

valor 3 (cd): 0

B.2. Postest de la herramienta de herencia.

1. Los atributos de un interface de Java por defecto son:
   a. Privados
   b. Con visibilidad de paquete
   c. Protegidos
   d. Públicos

2. Una clase abstracta:
   a. Debe tener al menos un método abstracto
   b. Puede tener todos sus métodos implementados
   c. No puede tener constructores
   d. Ninguna de las anteriores

3. Tenemos dos clases CuentaCorrienta y LibretaAhorro que tienen unas características comunes que se recogen en la clase Cuenta. ¿Pueden heredar ambas clases de Cuenta?
   a. sí porque este caso no es herencia múltiple.
   b. sólo si lo hacemos en dos paquetes distintos que importen la clase Cuenta.
   c. no porque en JAVA no se permite la herencia múltiple.
   d. a) y b) son correctas.

4. Se tiene una interface denominada Libro, que como atributos tiene todos los datos y los métodos sin implementar referentes a un libro genérico (numeroPaginas, tamaño, escribir( )… etc.). Si queremos crear una Novela, que es un tipo de libro ¿Qué afirmación es verdadera?

   a. Novela será una clase que herede por extensión a la interface Libro, a la que únicamente se le añadirán los atributos particulares de las novelas.
   b. Novela será una clase que use la interface Libro, para usar sus atributos y sus métodos.
   c. Novela será una clase que herede por implantación a la interface Libro, definiendo al menos todos los métodos de la interface Libro.
   d. Novela será un objeto de la interface Libro.

5. Dado el siguiente código:

```
class Ropa {
   private int precio=100;
   public int getPrecio() {

        return precio;

   }
}

class RopaTemporada extends Ropa {
   public int getPrecio(){

      return ((precio*120)/100);

   }
}
```

   Si hacemos la siguiente llamada:

```
Ropa ropa = new Ropa();
int pagado = ropa.getPrecio();
```

   a. pagado será igual a (100*120)/100.
   b. Existe un error, ya que no se puede usar ropa.getPrecio debido a que precio es privado.
   c. Existe un error, pues el método ropa.getPrecio() está sobrecargado.
   d. pagado será igual a 100.

6. Dado el siguiente código::

```
class Automovil {
   int puertas;
   boolean verificarCapacidad( )
   { ... }
}
class Coche extends Automovil {
   private boolean gasolina;
   ...
}
```

   ¿Qué sentencias son válidas?

   a. Coche coche = new Coche();
      coche.puertas = 4;
   b. Automovil auto = new Automovil();
      auto.gasolina = false;

c. Para que la opción b) sea correcta la propiedad gasolina debería ser pública.
d. a) y c) son correctas.

7. Dado el siguiente código, marque la opción correcta:

```
class Utensilio {
   private int tamaño;
    protected String comida;
   public Utensilio (int tamaño) {
      this.tamaño = tamaño;
   }
   public String getUso() {
      return this.comida;
   }
     ...
}
class Tenedor extends Utensilio {
  public Tenedor (int tamaño) {
     super(tamaño);
  }
  ...
}
```

a. Si ejecuta Utensilio cuchillo=new Utensilio(5) y Tenedor tenedor=new Tenedor(2) (en este orden) entonces cuchillo.tamaño valdrá 2.
b. Si ejecuta Utensilio cuchillo=new Utensilio(5) y Tenedor tenedor=new Tenedor(2) (en este orden) entonces cuchillo.tamaño valdrá 5.
c. El compilador dará un error, debido a que la clase Tenedor no puede cambiar la propiedad tamaño, pues es privada a Utensilio.
d. El compilador dará un error, debido a que es necesario llamar al constructor de la clase Utensilio dentro del constructor de la clase Tenedor.

8. Si la clase Tenedor del ejercicio anterior se encuentra en otro paquete diferente al de la clase Utensilio, marque la afirmación correcta:

a. La clase Tenedor, solo tendrá un atributo, que es comida, ya que tamaño es privado y un método (getUso())
b. La clase Tenedor heredará los dos atributos de Utensilio, aunque no podrá acceder a ninguno de ellos, ya que se encuentra en otro paquete. Tambien hereda getUso() pues es público.
c. La clase Tenedor heredará los dos atributos de Utensilio, aunque solo podrá acceder al atributo comida.
d. La clase Tenedor no heredará ningún atributo de Utensilio, ya que se encuentra en otro paquete.

9. Sea el siguiente código ¿Qué saldrá en pantalla?

```
class Luz {

  protected lumens=0;
  public Luz() {
    System.out.println("Creada una luz");

    lumens++;

  }
```

a. Creada una luz

   Creada una bombilla

   Creada una bombilla de ahorro de energia

   Lumens = 3

b. Creada una bombilla de ahorro de energia.

```
   public void mostrar () {

       System.out.println(“Lumens = “+lumens);
}
class Bombilla extends Luz {
   public Bombilla() {
      System.out.println("Creada una bombilla");

      lumens++;
   }
}

class BombillaAhorro extends Bombilla {
   public BombillaAhorro() {
      System.out.println("Creada una bombilla de ahorro
de energia");

      lumens++;
   }
}



BombillaAhorro b = new BombillaAhorro();

b.mostrar();
```

Creada una bombilla

Creada una luz

Lumens = 3

c. Lumens = 3

d. El código tiene errores pues las clases Bombilla y BombillaAhorro no pueden acceder al atributo lumens.

B.3. Pretest de CreOO

APELLIDOS:

NOMBRE:

1. Escoge la afirmación correcta:

a) El nombre del constructor de una clase debe coincidir con el nombre de la clase.
b) El nombre del constructor de una clase nunca debe coincidir con el nombre de la clase.
c) El nombre del constructor de una clase puede coincidir con el nombre de la clase, pero no es obligatorio.

2. Escoge la afirmación correcta:

a) Una clase puede tener varios constructores, y todos tienen el mismo nombre.
b) Una clase puede tener varios constructores, pero deben tener diferentes nombres.
c) Una clase no puede tener varios constructores.

3. Escoge la afirmación correcta:

a) Los diferentes constructores de una clase deben tener distinto número de argumentos.
b) Los diferentes constructores de una clase pueden tener el mismo número de argumentos, pero deben ser de diferentes tipos.

c) Una clase no puede tener varios constructores.

4. El constructor por defecto de una clase:

a) Es aquel cuyos argumentos son siempre tipos primitivos.
b) Es aquel que no tiene argumentos.
c) Es aquel que realiza una copia del objeto en cuestión.

5. Un constructor sirve para:

a) Reservar la cantidad necesaria de memoria para las clases.
b) Generalmente, asignar a los atributos de la clase los argumentos de entrada, si es que los tiene.
c) a) y b) son correctas.

6. ¿Cuántos comentarios hay en el programa?

```
/*This program is used to test for knowledge about comments

*/



public class Teram {

       public static void main(String[] args) {

              int i=3, j=4, k=5;
       // initialize some variables

              System.out.println("product="+(i*j*k)); // output their product

  } // end of the main method

}
```

a) 4
b) 3
c) 2
d) 1

7. Las palabras *public class static void* son palabras reservadas. Hay otra más en el programa. ¿Cuál?

```
public class Tercf {

       public static void main(String[] args) {

              double d = 12.787;

  }

}
```

a) Tercf
b) String
c) main
d) double
e) args

8. Dentro del programa, ¿cuál de las siguientes es una variable local?

```
public class Tere {

	double num = 17.3;

	static double calc(double radius) {

		return Math.PI * (radius * radius)

	}

	public static void main(String[] args) {

		double area = calc(3.0);

	}

}
```

a) num
b) area
c) calc
d) radius

9. ¿Qué array se crea en este programa?

```
public class Terhw {

	public static void main(String[] args) {

		int j = 21;

		int[] ia, ib = new int[3], ic;

		float diameter = 1.8;

	}

}
```

a) ib
b) ia
c) ic
d) j

10. ¿Cuántos métodos se definen en el programa?

```
	public class Terjb {

	static void m() {

		System.out.println("bienvenue");

	}

	public static void main(String[] args) {

		System.out.println("welcome");

		m();
```

```
                System.out.println("creoso");

        }

}
```

a) 1
b) 2
c) 0
d) 3

11. ¿Cuál es el valor de x en la salida?

```
public class Sjb {

        public static void main(String[] args) {

                int i=0, j=i+3;

                int[] ia = {7,2,4};

                int x = ia[j-2] + j + i;

                System.out.println("x="+x);

        }


}
```

a) 8
b) 5
c) 10

12. ¿Cuál es el valor de x en la salida?

```
public class Eflowm {

        public static void main(String[] args) {

                int x = 3;

                if(x > 3)

                        x = x * x;

                        x = x + 1;

                System.out.print("x = " + x);

        }

}
```

a) 3
b) 4
c) 10
d) 9

13. ¿Qué imprime este programa por pantalla?

```
public class Eflownd {

	public static void main(String[] args) {

		int i = 0;

		do {

			i++;

			System.out.print(i);

		}

		while (i < 3);

	}

}
```

a) 123
b) 012
c) 12
d) 0123

14. ¿Cuántos rectángulos dibuja este programa?

```
import javax.swing.*;

import java.awt.*;

public class Eflowng extends JApplet {

	public void paint(Graphics g) {

		super.paint(g);

		g.setColor(Color.yellow);

		Graphics2D g2 = (Graphics2D)g;

		g2.setStroke(new BasicStroke(10));

		for(int i=0; i<4 ; ++i) {

			if(i==1) continue;

			int w = 40*(i+1);

			int x = 150 - w/2;

			g2.drawRect(x,x,w,w);

		}

	}

}
```

a) 3
b) 4
c) 5
d) 2

15. ¿Cuántos círculos dibuja este programa?

```
import javax.swing.*;

import java.awt.*;

public class Eflownh extends JApplet {

	public void paint(Graphics g) {

		super.paint(g);

		g.setColor(Color.green);

		Graphics2D g2 = (Graphics2D)g;

		g2.setStroke(new BasicStroke(10));

		for(int i=0; i<4 ; ++i) {

			if(i==0) continue;

			if(i==3) break;

			int w = 40*(i+1);

			int x = 150 - w/2;

			g2.drawOval(x,x,w,w);

		}

	}

}
```

a) 1
b) 0
c) 3
d) 2

16. ¿Qué imprime este programa por pantalla?

```
public class Eybp {

	void m() {

		System.out.print("A");

	}

	public Eybp(int x) {

		this();

		System.out.print("B");

	}

	public Eybp() {

		System.out.print("C");

	}

	static public void main(String[] args) {

		Eybp dd = new Eybp(3);
```

```
                Eybp ee = new Eybp();


                dd.m();
        }
}
```

a) CBCA
b) BCA
c) ABC

17. ¿Qué valor de f se imprimirá?

```
public class Clqc {
        static float bb = 1.5;
        int ff;
        public static void main(String[] args) {
                int ff=3;
                Clqc obj = new Clqc();
                float f = obj.ff + bb;
                System.out.println("f=" + f);
        }
}
```

a) 4.5
b) 1.5

18. ¿Qué valor de x se imprimirá?

```
class ClassH {
        protected int ff = 17;
        protected int m() {
                return ff + 3;
        }
}
public class Clqr extends ClassH {
public int m() {
                return ff - 3;
        }
        public static void main (String arg[]) {
                Clqr obj = new Clqr();
```

```
                int x = obj.m();

                System.out.println("x = "+x);

        }

}
```

a) -3
b) 20
c) 14

19. Un constructor sirve para:

a) Garantizar la inicialización del objeto
b) Únicamente para reservar memoria para la clase
c) Dar valor a todos los atributos
d) Construir una clase.

20. El nombre del constructor es:

a) inicializar( );
b) el mismo que el nombre de la clase
c) cualquiera definido por el programador
d) construct( );

21. Para crear un objetos de la clase Prueba con el constructor por defecto se usa la instrucción:

a) Prueba( );
b) new Prueba( );
c) inicializar( );
d) Prueba.Prueba();

22. Si únicamente escribimos un constructor con un parámetro de tipo int y queremos crear un objeto podemos:

a) usar este constructor o el constructor por defecto
b) únicamente usar este constructor
c) no hace falta usar un constructor
d) usar el constructor por defecto

23. Si declaro un constructor privado

a) puedo crear objetos desde cualquier clase externa
b) no se pueden crear objetos
c) puedo crear objetos únicamente desde dentro de mi clase
d) puedo crear objetos desde cualquier clase que herede de mi clase

24. Dos constructores de la misma clase pueden

a) tener el mismo número, orden y tipo de argumentos
b) tener distintos nombres
c) no tener argumentos
d) tener el mismo número de argumentos, pero de distintos tipos

25. Una clase tiene

a) siempre varios constructores, para hacer distintas inicializaciones
b) al menos un constructor denominado constructor por defecto
c) atributos y métodos, pero puede no tener ningún constructor
d) herencia y polimorfismo

26. Dado el siguiente código:

```
class Banco {

       private int r;

       Banco (){

         r = 46;

       }

       Banco (int n1){

         r = n1;

       }

       static void main(String[ ] args){

         Banco nuevo = new Banco(36);

       }

}
```

El valor de r es:

a) r=36
b) r=0
c) r=46
d) r=72

27. Dado el siguiente código:

```
class Cuenta {

       private int d;

       private int q;

       Cuenta (){

         d = 64;

         q = 30;

       }

       Cuenta (int n1){

         d = n1;

       }

       Cuenta (int n1,int n2){

         d = n1;
```

```
        q = n2;

      }

      static void main(String[ ] args){

        Cuenta nuevo = new Cuenta(100);

      }

}
```

Los valores de d y q son:

a) d=100 y q=0
b) d=0 y q=0
c) d=15 y q=20
d) d=20 y q=16

28. Dado el siguiente código:

```
class Examen {

      private int q;

      private int w;

      private int c;

      Examen (){

        q = 22;

        w = 36;

        c = 56;

      }

      Examen (int n1){

        q = n1;

      }

      Examen (int n1,int n2){

        q = n1;

        w = n2;

      }

      Examen (int n1,int n2,int n3){

        q = n1;

        w = n2;

        c = n3;

      }

      static void main(String[ ] args){

        Examen nuevo = new Examen(57);
```

```
        }

}
```

Los valores de q, w y c es:

a) q=0, w=0, c=0
b) q=31, w=15, c=10
c) q=57, w=0, c=0,
d) q=56, w=98, c=63

B.4. Postest de CreOO

APELLIDOS:

NOMBRE:

1. Escoge la afirmación correcta:

a) El nombre del constructor de una clase debe coincidir con el nombre de la clase.
b) El nombre del constructor de una clase nunca debe coincidir con el nombre de la clase.
c) El nombre del constructor de una clase puede coincidir con el nombre de la clase, pero no es obligatorio.

2. Escoge la afirmación correcta:

a) Una clase puede tener varios constructores, y todos tienen el mismo nombre.
b) Una clase puede tener varios constructores, pero deben tener diferentes nombres.
c) Una clase no puede tener varios constructores.

3. Escoge la afirmación correcta:

d) Los diferentes constructores de una clase deben tener distinto número de argumentos.
a) Los diferentes constructores de una clase pueden tener el mismo número de argumentos, pero deben ser de diferentes tipos.
b) Una clase no puede tener varios constructores.

4. El constructor por defecto de una clase:

a) Es aquel cuyos argumentos son siempre tipos primitivos.
b) Es aquel que no tiene argumentos.
c) Es aquel que realiza una copia del objeto en cuestión.

5. Un constructor sirve para:

a) Reservar la cantidad necesaria de memoria para las clases.
b) Generalmente, asignar a los atributos de la clase los argumentos de entrada, si es que los tiene.
c) a) y b) son correctas.

6. ¿Qué imprime este programa por pantalla?

```
public class Eybp {

       void m() {

              System.out.print("A");

       }
```

```
        public Eybp(int x) {

                this();

                System.out.print("B");

        }

        public Eybp() {

                System.out.print("C");

        }

        static public void main(String[] args) {

                Eybp dd = new Eybp(3);

                Eybp ee = new Eybp();


                dd.m();

        }
}
```

a) CBCA
b) BCA
c) ABC

7. Un constructor sirve para:

e) Garantizar la inicialización del objeto
a) Únicamente para reservar memoria para la clase
b) Dar valor a todos los atributos
c) Construir una clase.

8. El nombre del constructor es:

a) inicializar( );
b) el mismo que el nombre de la clase
c) cualquiera definido por el programador
d) construct( );

9. Para crear un objetos de la clase Prueba con el constructor por defecto se usa la instrucción:

a) Prueba( );
b) new Prueba( );
c) inicializar( );
d) Prueba.Prueba();

10. Si únicamente escribimos un constructor con un parámetro de tipo int y queremos crear un objeto, podemos:

a) usar este constructor o el constructor por defecto
b) únicamente usar este constructor
c) no hace falta usar un constructor
d) usar el constructor por defecto

11. Si declaro un constructor privado

a) puedo crear objetos desde cualquier clase externa

b) no se pueden crear objetos
c) puedo crear objetos únicamente desde dentro de mi clase
d) puedo crear objetos desde cualquier clase que herede de mi clase

12. Dos constructores de la misma clase pueden

a) tener el mismo número, orden y tipo de argumentos
b) tener distintos nombres
c) no tener argumentos
d) tener el mismo número de argumentos, pero de distintos tipos

13. Una clase tiene

a) siempre varios constructores, para hacer distintas inicializaciones
b) al menos un constructor denominado constructor por defecto
c) atributos y métodos, pero puede no tener ningún constructor
d) herencia y polimorfismo

14. Dado el siguiente código:

```
class Banco {

       private int r;

       Banco (){

         r = 46;

       }

       Banco (int n1){

         r = n1;

       }

       static void main(String[ ] args){

         Banco nuevo = new Banco(36);

       }

}
```

El valor de r es:

e) r=36
a) r=0
b) r=46
c) r=72

Bibliografia

[Ainsworth 99] Ainsworth, A. (1999). The functions of multiple representations. *In Computers & Education,* 33(2-3), 131-152.

[Ala-Mutka 05] Ala-Mutka, K.M. (2005). A Survey of Automated Assessment Approaches for Programming Assignments. *In Computer Science Education Vol. 15, No. 2*, Tampere University of Technology, Tampere, Finland. 83–102.

[Aliaga 00] Aliaga Abad, F. (2000). *Bases epistemológicas y Proceso de Investigación psicoeducativa.* Universidad de Valencia: Ed. C.S.V.

[Alice] http://www.alice.org/. Consultada en diciembre-2007.

[Allen 02] Allen, E., Cartwright, R. & Stoler, B. (2002). DrJava: a lightweight pedagogic environment for java. *In SIGCSE '02: Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education*, New York, NY, USA: ACM Press, 137–141.

[Almeida 09] Almeida, F., Urquiza, J. & Velázquez, J. A. (2009). Visualization of syntax trees for language processing courses, *In Journal of Universal Computer Science, Vol 15, N 7*, 1.546-1.561.

[Anderson 01] Anderson, L.W. & Krathwohl, D.R. (2001). *A taxonomy for learning, teaching and assessing. A revision of Bloom's Taxonomy of educational objectives*. Complete Edition. New York: Addison Wesley Longman Inc.

[Apple 02] Apple, D. K., Nygren, K. P., Williams, M. W. & Litynski, D. M. (2002). Distinguishing and elevating levels of learning in engineering and technology instruction, *In Proc. of the 32nd ASEE/IEEE Frontiers in Education Conference*: IEEE Computer Society Press, T4B 7-11.

[Ausubel 63] Ausubel, D. P. (1963). *The psychology of meaningful verbal learning*. Oxford, England: Grune & Stratton.

[Ausubel 69] Ausubel, D. P. & Robinson, F. G. (1969). *School learning: An introduction to educational psychology*. New York: Holt, Rinehart and Winston.

[Barrows 80] Barrows, H. S. & Tamblyn, R. M. (1980). *Problem-based learning: An approach to medical education*. Nueva York: Springer-Verlag.

[Bergin 02] Bergin J., Stehlik, M., Roberts, J. & Pattis, R. (2002). Karel J. Robot: A gentle Introduction to the Art of Object-Oriented Programming in Java. Consultado en 17 Feb. 2012 desde http://csis.pace.edu/~Ebergin/KarelJava2ed/Karel++JavaEdition.html

[Ben-Bassat 11] Ben-Bassat Levy, R. & Velázquez Iturbide, J.A. (2011). A problem solving teaching guide based on a procedure intertwined with a teaching model. *In Proceedings of the 16th annual joint conference on Innovation and technology in computer science education (ITiCSE '11)*. New York, NY, USA: ACM, 374-374.

[Biggs 82] Biggs, J.B. & Collis, K.F. (1982). *Evaluating the quality of learning: The SOLO taxonomy (Structure of the Observed Learning Outcome)*. Academic Press, New York.

[Bloom 56] Bloom, B.S., Engelhart, M.D., Furst, E.J., Hill, W.H. & Krathwohl, D.R. (1956). *Taxonomy of Educational Objectives, The Classification of Educational Goals, Handbook I, The Cognitive Domain*. New York: David McKay Company, Inc.

[BlueJ] http://www.bluej.org/ consultada en dic-2007.

[Bologna 03] Bologna Process: Lifelong Learning EEES Bologna Process. Consultado en 17 Septiembre del 2010 desde http://www.bologna-berlin2003.de/pdf/Prague_communiquTheta.pdf.

[Bourque 03] Bourque, P., Buglione, L., Abran, A. & April, A. (2003). Bloom's Taxonomy Levels for Three Software Engineer Profiles, *Eleventh Annual International Workshop on Software Technology and Engineering Practice* (STEP'03), 123-129.

[Bridgeman 00a] Bridgeman S., Goodrich M.T., Kobourov S.G. & Tamassia R. (2000). PILOT: An interactive tools for learning and grading. *In Proceedings of SIGCSE 2000*, 139-143.

[Bridgeman 00b] Bridgeman S., Goodrich M.T., Kobourov S.G. & Tamassia R. (2000). SAIL: A system for generating, archiving and retrieving specialized assignments using LaTex. *In Proceedings of SIGCSE 2000*, 300-304.

[Bruce 81] Bruce, R.L. (1981). Programming for intangibles. *Cornell information bulletin 179, Extension publication 9/81*. Ithaca, NY: New York State College of Agriculture and Life Sciences and New York State College of Human Ecology at Cornell University.

[Bruner 66] Bruner, J. (1966). *Toward a Theory of Instruction*. Cambridge, MA: Harvard University Press.

[Brusilovsky 96] Brusilovsky, P. & Weber, G. (1996). Collaborative example selection in an intelligent example-based programming environment. *In Proceedings of the 1996 international conference on Learning sciences (ICLS '96)*, Daniel C. Edelson and Eric A. Domeshek (Eds.). International Society of the Learning Sciences, 357-362.

[Brusilovsky 01] Brusilovsky, P. (2001). WebEx: Learning from examples in a programming course, In: Fowler, W. and Hasebrook, J. (eds.) *Proc. of WebNet'2001, World Conference of the WWW and Internet* , Orlando, FL, AACE, 2001, 124-129.

[Buck 00] Buck, D. & Stucki, D. (2000). Design Early Considered Harmful: Graduated Exposure to Complexity and Structure Based on Levels of Cognitive Development. *In Proceedings SIGCSE 2000*, ACM Press, 75-79.

[Buck 01] Buck, D. & Stucki, D. (2001). JKarelRobot: a case study in supporting levels of cognitive development in the computer science curriculum. *In Proceedings of the 32nd SIGCSE Technical Symposium on Computer Science Education*, 16–20.

[Buckley 03] Buckley J. & Exton C. (2003). A framework for assessing programmers' knowledge of software systems. *In Proceedings of the 11th IEEE International Workshop on Program Comprehension* (IWPC'03), 165-175.

[Budd 00] Budd, T. (2000). *Understanding object-oriented programming with Java*. Ed. Addison-Wesley.

[Carmona 77] Carmona Guillén, J.L. (1977). *Los indicadores sociales hoy.* Madrid Ed. CIS.

[Carro 99] Carro, R., Pulido, E. y Rodríguez, P. (1999). TANGOW: Un Sistema de Enseñanza Adaptativa a través de Internet. *Congreso Internacional de Informática Educativa CONIED '99*, Puertollano.

[Carro 00] Carro, R., Pulido, E. y Rodríguez, P. (2000). How adaptivity affects the development of TANGOW web-based courses. *In*: P. Brusilovsky, O. Stock (Eds.), *Adaptive hypermedia and adaptive web-based systems, Proceeding of international conference, AH2000*, Springer-Verlag, Berlin, 280–283.

[Carter 99] Carter, J. & Jenkins, T. (1999). Gender and programming: What's going on?. *In Proc. 4th Annual Conf. Innovation and Technology in Computer Science Education (ITiCSE'99)*, ACM Press, 1-4.

[Cooper 03] Cooper, S., Dann, W. & Pausch, R. (2003). Teaching objects-first in introductory computer science. *In Proceedings of the 34th SIGCSE technical symposium on Computer science education (SIGCSE '03)*. ACM, New York, NY, USA, 191-195.

[Dancik 03] Dancik, G. & Kumar, A.N. (2003). A Tutor for Counter-Controlled Loop Concepts and Its Evaluation. *In Proceedings of Frontiers in Education Conference (FIE 2003)*, Boulder, CO, Session T3C.

[De Miguel 67] De Miguel, A. (1967). *Tres estudios para un sistema de indicadores sociales (Estudio primero)*, Fundación Foessa, Madrid: Ed. Euramérica.

[Deimel 90] Deimel, L.E. & Naveda, J.F. (1990). *Reading Computer Programs: Instructor's guide and exercises*. Educational Materials CMU.

[Dillon 98] Dillon, A. & Gabbard, R. (1998). Hypermedia as an educational technology: A review of the quantitative research literature on learner comprehension, control, and style. *In Review of Educational Research, Vol. 68, No. 3*, 322-349.

[Doran 95] Doran, M. V. & Langan, D. D. (1995). A cognitive-based approach to introductory computer science courses: lesson learned. *In Proceedings of the twenty-sixth SIGCSE Technical Symposium on Computer Science Education*, Nashville, Tennessee, United States, ACM Press.

[Dorn 03] Dorn, B. & Sanders, D. (2003). Using Jeroo to Introduce Object-Oriented Programming, *In Proceedings of the 2003 IEEE/ASEE Frontiers in Education Conference, Vol.1*, T4C 22-T4C 27.

[Douce 05] Douce, C., Livingstone, D. & Orwell, J. (2005) Automatic test-based assessment of programming: A review, *ACM Journal of Educational Resources in Computing, v.5 n.3 4-es*, 1-13.

[DrJava] http://www.drjava.org/. Consultada en mayo 2008.

[Eclipse] http://www.eclipse.org/. Consultada en febrero 2006.

[Edwards 08] Edwards, S.H., Börstler, J., Cassel, L.N., Hall, M.S. & Hollingsworth, J. (2008). Developing a common format for sharing programming assignments, *ACM SIGCSE Bulletin*, 40(4), 167-182.

[Farnsworth 94] Farnsworth, C. C. (1994). Using computer simulations in problem-based learning. *In Proc. Thirty Fifth ADCIS Conference*, Omni Press, Nashville, TN, 137-140,.

[Felder 96] Felder R.M. (1996). Matters of style. *ASEE Prism*, 6(4), 18-23.

[Felder 00] Felder, R.M., Woods, D.R., Stice, J.R. & Rigarcia, A. (2000). The future of engineering education. Teaching methods that work. *Chemical Engineering Education, 34(1)*, 26–39.

[Fernández 02] Fernández Muñoz, L., Peña Ros, R., Nava García, A. y Velázquez Iturbide, J. Á. (2002). Enfoque diacrónico para la enseñanza de la programación imperativa. *En Actas de las VIII Jornadas de la Enseñanza Universitaria de la Informática, JENUI'02*, Universidad de Extremadura, 407-414.

[Freund 96] Freund, S. N. & Roberts, E. S. (1996): THETIS: an Ansi C programming environment designed for introductory use. *In Proc. 27th SIGCSE Technical Symp. On Computer Science Education,* Philadelphia, ACM Press, New York, 300- 304.

[Fuller 07] Fuller, U., Johnson, C.G., Ahoniemi, T., Cukierman, D., Hernán-Losada, I., Jackova, J., Lahtien E., Lewis, T.L., McGee Thompson, D., Riedesel, C. & Thompson, E. (2007). Developing a Computer Science-specific learning taxonomy. *ACM SIGCSE Bulletin 34(4)*, New York: ACM Press, 152-170.

[Gagné 70] Gagné, R. M. (1970). *The conditions of learning*. (2nd ed.), New York: Holt, Rinehart and Winston.

[Gallant 08] Gallant, R. J. & Mahmoud, Q. H. (2008). Using Greenfoot and a Moon Scenario to teach Java programming in CS1. *In Proceedings of the 46th Annual Southeast Regional Conference*. New York, NY, USA: ACM Press, 118-121.

[Garcia 03] García Ferrando, M. (2003). *El análisis de la realidad social métodos y técnicas de investigación*. Ed. Alianza.

[Garrido 08] Garrido Abenza, P. P., Grediaga Olivo, A. & Ledesma Latorre, B. (2008). VisualJVM: a visual tool for teaching Java technology, *IEEE Transactions on Education*, *Vol. 51, no.1*, 86-92.

[Gediga 02] Gediga, G., Hambor, K. & Düntsch, I. (2002). Evaluation of software systems. *In Encyclopedia of Computer Science and Technology, Vol. 45*, Ed. Marcel Dekker, 127-153.

[Gerlach 67] Gerlach, V. & Sullivan, A. (1967). *Constructing statements of outcomes*. Inglewood, CA. Southwest Regional Laboratory for Educational Research and Development.

[Greenfoot] http://www.greenfoot.org/. Consultado en enero-2009.

[Gregorio-Rodríguez 01] Gregorio-Rodríguez, C., Llana-Díaz, L., Martínez-Unanue, R., Palao-Constanza, P., Pareja-Flores, C. & Velázquez-Iturbide J.Á. (2001). eXercita: Automatic Web publishing of programming exercises. *In Proc. 6th Annual Conference on Innovation and Technology in Computer Science Education*, ACM Press, 161-164.

[Gregorio 2002] Gregorio-Rodríguez, C., Llana-Díaz, L., Martínez-Unanue, R., Pareja-Flores, C., Velázquez-Iturbide J.Á. & Palao-Constanza, P. (2002). A system to generate electronic books on programming exercises, *The Electronic Library, Vol. 20, N. 4*, 314-321.

[Gómez 05] Gómez-Albarrán, M. (2005). The Teaching and Learning of Programming: A Survey of Supporting Software Tools. *The Computer Journal, 48(2),* 130-144.

[Gomez-Martín 03] Gómez-Martín, P.P., Gómez-Martín, M.A. & González-Calero, P.A. (2003). Javy: Virtual Environment for Case-Based Teaching of Java Virtual Machine. *In Proceedings of the 7th International Conference on Knowledge-Based Intelligent Information & Engineering Systems.* University of Oxford, United Kingdom: Springer Verlag, 906-913.

[Hauenstein 98] Hauenstein, A. D. (1998). *A conceptual framework for educational objectives: A holistic approach to traditional taxonomies.* Ed. Lanhman, MD: University Press of America.

[Healy 11] Healy, W.J., Taran, Z. & Betts S. C. (2011). Sales course design using experiential learning principles and Bloom's taxonomy. *En Journal of Instructional Pedagogies*, *Vol. 6*. Consultado en mayo- 2011 desde http://www.aabri.com/jip.html.

[Hernán 03] Hernán-Losada, I., Lázaro-Carrascosa, C. y Velázquez-Iturbide, J. Á. (2003). Hacia el diseño de herramientas educativas de programación basadas en la taxonomía de Bloom. *In Proc. of 5º Simposio Internacional en Informatica Educativa (SIIE)*, Braga ,Portugal, 761-765

[Hernán 04a] Hernán-Losada, I., Lázaro-Carrascosa, C. & Velázquez-Iturbide, J. Á. (2004). On the use of Bloom's taxonomy as a basis to design educational software on programming. *In Proc. of World Conference on Engineering and Technology Education, WCETE 2004*, COPEC, Brazil, 351-355.

[Hernán 04b] Hernán-Losada, I. y Lázaro-Carrascosa, C. (). Un prototipo de herramienta educativa basada en la taxonomía de Bloom, *In Proc. of 6º Simposio Internacional de Informática Educativa (SIIE)*, Cáceres, España, 284.

[Hernán 05] Hernán-Losada, I., Lázaro-Carrascosa, C. y Velázquez-Iturbide, J. Á. (2005). Una Aplicación Educativa Basada en la Jerarquía de Bloom para el Aprendizaje de la Herencia de POO, *In Proc. of 7º Simposio Internacional de Informática Educativa (SIIE)*, Leiria, Portugal, 107-112.

[Hernán 06a] Hernán-Losada, I., Velázquez-Iturbide, J. Á. & Lázaro-Carrascosa, C. (2006). Programming Learning Tools Based on Bloom's Taxonomy: Proposal and Accomplishments, *In Proc. of 8º Simposio Internacional de Informática Educativa (SIIE)*, León, España, 243-252.

[Hernán 06b] Hernán-Losada, I., Velázquez-Iturbide, J. Á. y Lázaro-Carrascosa, C. (2006). Dos Herramientas Educativas para el Aprendizaje de Programación: Generación de Comentarios y Creación de Objetos, *En Actas del 7º Congreso Internacional de Interacción Persona-Ordenador*. Puertollano, España, 325-334.

[Hernán 07a] Hernán-Losada, I. y Velázquez-Iturbide, J. Á. (2007). Hacia el nivel de aplicación en la taxonomía de Bloom: Generación de ejercicios semiabiertos sobre POO. *En Actas del VIII Simposio Nacional De Tecnologías De La Información Y Las Comunicaciones En La Educación (Sintice 2007)*. Zaragoza, España, 27-34.

[Hernán 07b] Hernán-Losada, I., Lázaro-Carrascosa, C. & Velázquez-Iturbide, J. Á. (2007). An educative application based on Bloom's taxonomy for the learning of inheritance in oriented-object programming. *Computers and Education: Towards Educational Change and Innovation*, António Mendes, Isabel Pereira y Rogério Costa (eds.), Berlín: Springer. 157-166

[Hernán 07c] Hernán-Losada, I., Velázquez-Iturbide, J. Á. & Lázaro-Carrascosa, C. (2007). Diseño y evaluación de una herramienta para el aprendizaje de la creación de objetos en Java. *In Proc. of IX Simposio Internacional de Informática Educativa SIIE,* Escola Superior de Educação do Instituto Politécnico do Porto, Portugal, 91-96.

[Hernán 08] Hernán-Losada, I., Pareja-Flores, C. & Velázquez-Iturbide, J. Á. (2008). Testing-Based Automatic Grading: A Proposal from Bloom's Taxonomy. *In Proc. of 8th IEEE International Conference on Advanced Learning Technologies (ICALT 2008)*, Santander, España, 847-849.

[Hernán 09a] Hernán-Losada, I., Lázaro-Carrascosa, C.A., Paredes-Velasco, M. y Velázquez-Iturbide, J.A. (2009). Tutores interactivos basados en objetivos pedagógicos para el aprendizaje de la programación. *En I Encuentro de Intercambio de Experiencias en Innovación Docente*, Félix Labrador Arroyo y Rosa Santero Sánchez (eds.), 27.

[Hernán 09b] Hernán-Losada, I. (2009). Conclusiones sobre la aplicación de la Taxonomía de Bloom al diseño de herramientas pedagógicas. *Actas del I Seminario de Investigación en Tecnologías de la Información Aplicadas a la Educación*; Ed. Dykinson, 139-150.

[Hernán 10a] Hernán-Losada, I. (2010). GenPR: generación aleatoria de problemas, *Actas de las I Jornadas en Innovación y TIC Educativas – JITICE 2010,* Estefanía Martín Barroso, Manuel Rubio Sánchez y Jaime Urquiza Fuentes (eds.), Serie de Informes Técnicos DLSI1-URJC, número 2010-05, Departamento de Lenguajes y Sistemas Informáticos I, Universidad Rey Juan Carlos, 53-54.

[Hernán 10b] Hernán-Losada, I., Pareja-Flores, C. & Velázquez-Iturbide, J. Á. (2010): Pedagogical use of automatic graders. *Advances in learning processes*. Editorial: In-tech. Vukovar, Croatia, 265-274

[Hernán 11a] Hernán-Losada, I. y Velázquez-Iturbide, J. Á. (2011). Aplicación de la investigación social a la evaluación y su relación con la taxonomía de Bloom. *In Old Meets New: Media in Education – Proceedings of the 61st International Council for Educational Media and the XIII International Symposium on Computers in Education (ICEM&SIIE'2011).* Joint Conference, António Moreira, Maria José Loureiro, Ana Balula, Fernanda Nogueira, Lúcia Pombo, Luís Pedro y Pedro Almeida (eds.), Universidade de Aveiro, Portugal, 457-466.

[Hernán 11b] Hernán-Losada, I. (2011). Comparación de la taxonomía de Bloom con otras teorías del aprendizaje. *Actas del III Seminario de Investigación en Tecnologías de la Información Aplicadas a la Educación (SITIAE 2009)*. Editorial Dykinson, S.L. Madrid., 203-213

[Hernán 11c] Hernán-Losada, I. y Martin-Barroso, E. (2011). Experiencia Docente Innovadora en la Enseñanza. Aplicación a la Programación Orientada a Objetos. *En el I Congreso Internacional sobre aprendizaje, innovación y competitividad (CINAIC 2011)*. Madrid: Ed. UPM, 254-259.

[Hernán 11d] Hernán-Losada, I., Velázquez-Iturbide, J. Á.: Aplicación de la investigación social a la evaluación y su relación con la taxonomía de Bloom. En Indagatio Didactica, 3(3):141-158, diciembre 2011

[Hernández 97] Hernandez Rojas, G. (1997). *Módulo Fundamentos del Desarrollo de la Tecnología Educativa (Bases Psicopedagógicas).* Coord. Frida Díaz Barriga Arceo. México: Ed. ILCE- OEA.

[Hestenes 92] Hestenes, D., Wells, M. & Swackhamer, G. (1992). Force concept inventory, *The Physics Teacher, Vol. 30,* 141-158.

[Howard 96] Howard, A. R., Carver, C. A. & Lane, W. D. (1996). Felder's learning styles, Bloom's taxonomy, and the Kolb learning cycle: Tying it all together in the CS2 course. *In Proc. 27th SIGCSE Technical Symp. Computer Science Education (SIGCSE'96)*, ACM Press, 227-231.

[JCreator] http://www.jcreator.com/. Accedido en enero-2009

[Jeliot3] http://cs.joensuu.fi/jeliot/. Accedido en marzo-2010.

[Jeroo] http://home.cc.gatech.edu/dorn/jeroo. Accedido en enero-2012

[JGrasp] http://www.jgrasp.org/. Accedido en enero-2010.

[Johnson 07] Johnson, C.G. & Fuller, U.D. (2006). Is Bloom's taxonomy appropriate for computer science?. *In 6th Baltic Sea Conference on Computing Education Koli Calling 2006*, Koli Calling, Berglund, A. and Wiggberg, M., Eds. Department of Information Technology, University of Uppsala, Stockholm, 120-123.

[Karavirta 04] Karavirta, V., Korhonen, A., Malmi, L. & Stalnacke, K. (2004). MatrixPro - A tool for on-the-fly demonstration of data structures and algorithms. *In Proceedings of the 3rd Program Visualization Workshop*. The University of Warwick, UK, 26–33.

[Kemmis 77] Kemmis, S., Atkin, R. & Wright, E. (1977). How do students learn? *Working papers on computer assisted learning*. Norwich, Centre for Applied Research in Education, UEA.

[Khairuddin 08] Khairuddin, N.N. & Hashim, K. (2008). Application of Bloom's taxonomy in software engineering assessments, *ACS'08 Proceedings of the 8th conference on Applied computer science.* Ed. World Scientific and Engineering Academy and Society (WSEAS), 66-69.

[Kolling 03] Kolling, M., Quig, B., Patterson, A. & Rosenberg, J. (2003). The BlueJ System and its Pedagogy, *Journal of Computer Science Education, Special issue on Learning and Teaching Object Technology*, *vol. 13,* 249-268.

[Kolling 08] Kölling, M. (2008). Greenfoot: a highly graphical ide for learning object-oriented programming. *In Proceedings of the 13th annual conference on Innovation and technology in computer science education (ITiCSE '08).* New York: ACM, 327-327.

[Krathwohl 02] Krathwohl D.R. (2002). A revision of Bloom's taxonomy: An overview. *Theory into Practice, Vol. 41, No. 4,* 212-218.

[Krishna 01] Krishna, A. & Kumar A.N. (2001). A Problem Generator to Learn Expression Evaluation in CS I and Its Effectiveness, *The Journal of Computing in Small Colleges, Vol. 16, No. 4*, 34-43.

[Kumar 00] Kumar A.N. (2000). Dynamically Generating Problems on Static Scope, *Proc. of The Fifth Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2000)*, Helsinki, Finland, 9-12.

[Kumar 02] Kumar A.N. (2002). A tutor for using dynamic memory in C++. *In Proc. of the 32rd ASEE/IEEE Frontiers in Education Conference*, Boston, MA. Session T4G, 12-16.

[Lahtinen 05] Lahtinen, E. & Ahoniemi, T. (2005). Visualizations to Support Programming on Different Levels of Cognitive Development. *In Proc. of the Fifth Koli Calling Conference on Computer Science Education*, 87-94.

[Lahtinen 07] Lahtinen, E. (2007). A Categorization of Novice Programmers: A Cluster Analysis Study. *In Proc. of the 19th annual Workshop of the Psychology of Programming Interest Group*, Joensuu, Finland, Eds. Sajaniemi, J. and Tukiainen, M., University of Joensuu Department of Computer Science and Statistics, 32-41.

[Lauer 08] Lauer, T. (2008). Reevaluating and refining the engagement taxonomy. *In Proc. of the 13th annual conference on Innovation and technology in computer science education (ITiCSE '08)*. New York: ACM, 355-355.

[Lázaro 05] Lázaro-Carrascosa C., Velázquez-Iturbide, J. Á., Hernán-Losada, I., Gortázar Bellas, F. & Gallego Carrillo, M. (2005). TextOO: Learning object oriented modeling using the enunciate. *In Proc. of the Fifth Koli Calling Conference on Computer Science Education*, Tapio Slakoski, Tomi Mäntylä y Mikko Laakso (eds.), Turku Centre for Computer Science, TUCS General Publications Series, nº. 41, 181-182.

[Lázaro 06] Lázaro-Carrascosa C., Velázquez-Iturbide, J. Á., Hijón-Neira, R & Hernán-Losada, I. (2006). TextOO: An Object-Oriented learning tool based on enunciates. *Proc of 8º Simposio Internacional de Informática Educativa (SIIE).* León, España, 132-140.

[Lister 01] Lister, R. (2001). Objectives and Objective Assessment in CS1. *In Proc. of SIGCSE Technical Symp. On Computer Science Education*. ACM Press, 292-296.

[Lister 03] Lister, R. & Leaney, J. (2003). First year programming: Let all the flowers bloom. *In Proc. of the 5th Australasian Computer Education Conference (ACE2003)*, 221–230.

[LITE] Laboratory of Information Technologies in Education, http://www.lite.etsii.urjc.es/. Consultado abril-2010.

[Mcdaniel 07] Mcdaniel, M., Roediger, H. & Mcdermott, K. (2007). Generalizing test-enhanced learning from the laboratory to the classroom. *Psychonomic Bulletin & Review Vol. 14 , Issue 2*, New York: Springer, 200-206.

[Machanick 00] Machanick, P. (2000). Experience of applying Bloom's Taxonomy in three courses. *In Proc. Southern African Computer Lecturers' Association Conference*, Strand, South Africa, 135-144.

[Manaris 04] Manaris, B. & McCauley, R. (2004). Incorporating HCI into the undergraduate curriculum: Bloom's taxonomy meets the CC'01 curricular guidelines, *Frontiers in Education 2004. FIE. Vol. 1,* 20-23.

[Marquès 99] Marquès Graells, P. (1999). *Software educativo multimedia: tipologías*. Accedida en enero-12 desde http://www.peremarques.net/tipologi.htm.

[Martín 08] Martín, E. (2008). *Creación de entornos adaptativos móviles: recomendación de actividades y generación dinámica de espacios de trabajo basadas en información sobre usuarios, grupos y contextos*. Tesis Doctoral, Departamento de Ingeniería Informática, Escuela Politécnica Superior, Universidad Autónoma de Madrid.

[Martín 10] Martín, E., Lázaro, C. & Hernán-Losada, I (2010). Active learning in Telecommunication Engineering: A case study. *IEEE Engineering Education (Educon 2010)*, Madrid, España, 1555-1562.

[Martínez 02] Martínez Unanue, R., Paredes Velasco, M., Pareja Flores, C., Urquiza Fuentes, J. & Velázquez Iturbide, J.Á. (2002). Electronic books for programming education: A review and future prospects. *In Proc. 7th Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE),* ACM Press, 34-38.

[Maslow 54] Maslow, A. (1954). *Motivation and Personality*. New York: Addison-Wesley Educational Publishers Inc.

[McConnell 96] McConnell, J.J. (1996). Active learning and its use in Computer Science. *In Proc. Conference on Integrating Technology into Computer Science Education (ITiCSE)*, ACM Press, 52-54.

[Metfessel 69] Metfessel, N. S., Michael, W. G. & Kirsner, D. A. (1969). Instrumentation of Bloom's and Krathwohl's taxonomies for the writing of educational objectives. *Psychology in the Schools, Vol. 6*, 227-231.

[Moreno 04] Moreno, A., Myller, N., Sutinen, E. & Ben-Ari, M. (2004). Visualizing Program with Jeliot 3. *In Proc. of the International Working Conference on Advanced Visual Interfaces*, Gallipoli (Lecce), Italy, 373–380.

[Naps 03a] Naps, T., Roessling, G. , Almstrum, V., Dann, W., Fleischer, R., Hundhausen, C., Korhonen, A., Malmi, L., McNally, M., Rodger, S. & Velazquez-Iturbide, J. A. (2003). Exploring the role of visualization and engagement in computer science education. *SIGCSE Bulletin, Vol. 35, No. 2*, 131-152.

[Naps 03b] Naps, T., Cooper, S., Koldehofe, B., Roessling, G., Dann, W., Korhonen, A., Malmi, L., Rantakokko, J., Ross, R.J., Anderson, J., Fleischer, R., Kuittinen, M. & McNally, M. (2003). Evaluating the educational impact of visualization. *ACM SIGCSE Bulletin 35*, *No. 4*, 124-136.

[Naps 05] Naps.,T. L. (2005). JHAVE – Supporting Algorithm Visualization. *IEEE Computer Graphics and Applications*, 25(5), 49–55.

[Netbeans] www.netbeans.org/. Consultado en marzo 2007.

[Oliver 04] Oliver, D., Dobele, T., Greber, M. & Roberts, T. (2004). This course has a Bloom Rating of 3.9. *In Proc. of the sixth conference on Australasian computing education, Vol. 30.* Dunedin, New Zealand, Australian Computer Society, Inc., 227-231.

[Pattis 95] Pattis, R.E. (1995). *Karel the Robot: A Gentle Introduction to the Art of Programming*. 2nd ed., Wiley.

[Pavlov 27] Pavlov, I.P. (1927). *Conditioned Reflexes: An Investigation of the Physiological Activity of the Cerebral Cortex*. Translated and Edited by G. V. Anrep. London: Oxford University Press.

[Pareja-Flores 00] Pareja-Flores, C. & Velázquez-Iturbide, J.Á. (2000): Local vs. comprehensive assignments: two complementary approaches, *ACM SIGCSE Bulletin, 32(4)*, 48-52.

[Pears 07] Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M. & Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *SIGCSE Bull. 39, 4*, 204-223.

[Pérez 06] Pérez Marín, D., Alfonseca Cubero, E. y Rodríguez Marín, P. (2006). ¿Pueden los ordenadores evaluar automáticamente preguntas abiertas? *Novática, 183*, 50-54.

[Piaget 52] Piaget, J. (1952). *The Origins of Intelligence in Children*. New York: International University Press.

[Piaget 54] Piaget, J. (1954). *The Child's Construction of Reality*. London: Routledge and Kegan Paul Ltd.

[Piaget 72] Piaget, J. (1972) *Psychology and Epistemology: Towards a Theory of Knowledge.* Harmondsworth. Penguin (Non-Classics).

[Piaget 77] Piaget, J. (1977) Algunas reflexiones sobre la pedagogía. *Cuadernos de Pedagogía. N. 27.*

[Piombo 03] Piombo, C., Batatia, H. & Ayache, A. (2003). A framework for adapting instruction to cognitive learning styles. *In Proc. 3rd IEEE International Conference on Advanced Learning Technologies*. IEEE Computer Society Press. 434-435.

[Prince 04] Prince, M. (2004). Does Active Learning Work? A Review of the Research. *Journal of Engineering Education, vol. 93 (3),* 223-231.

[Quellmalz (87] Quellmalz, E. (1987). Developing reasoning skills. *In Teaching thinking skills*. New York: W. H. Freeman. 86-105.

[Rademacher 99] Rademacher, R. (1999). Applying Bloom's taxonomy of cognition to knowledge management systems. *ACM SIGCPR conference on Computer Personnel Research*, New Orleans, LA, New York, NY: ACM Press, 276-278.

[Reigeluth 99] Reigeluth, C. M. & Moore, J. (1999). Cognitive education and the cognitive domain. *In Instructional-design theories and models: A new paradigm of instructional theory.* 2nd ed. NJ: Lawrence Erlbaum Associates, 51-68.

[Reynolds 96] Reynolds, C. & Fox, C. (1996). Requirement for a Computer Science Curriculum Emphasizing Information Technology Subject Area: Curriculum Issues. *SIGSE Bulletin, Vol 28, No.1*. 247-251.

[Rhoads 99] Rhoads, T.R. & Roedel, R.J. (1999). The wave concept inventory: A cognitive instrument based on Bloom's taxonomy. *In Proc. of the 29th ASEE/IEEE Frontiers in Education Conference*. IEEE Computer Society Press. 13C 1-14.

[Romizowski 81] Romizowski, A. J. (1981). *Designing instructional systems: Decision making in course planning and curriculum design*. London: Kogan Page/New York Nichols Publishing.

[Ruiz 96] Ruiz, F., Ortega, M., Bravo, J. y Prieto, M. (1996). Nuevas herramientas tecnológicas para la realización de cursos por computador. *Informatica Educativa Comunicaciones, 1(1-5)*. Consultado el febrero-2012, desde http://161.67.140.29/iecom/index.php/IECom/article/view/61/55.

[Rutkowski 10] Rutkowski, J., Moscinska, K. & Jantos, P. (2010). Application of Bloom's taxonomy for increasing teaching efficiency – case study. *In Proc. of the International Conference on Engineering Education ICEE-2010*, Gliwice, Poland.

[Sanders 00] Sanders, I. & Mueller, C. (2000). A fundamentals-Based Curriculum for First Year Computer Science. *In Proc. 31st SIGCSE Technical Symposium on Computer Science Education*, ACM Press, 227-231.

[Satratzemi 01] Satratzemi, M., Dagdilelis, V. & Evagelidis, G. (2001). A system for program visualization and problem-solving path assessment of novice programmers. *ACM SIGCSE Bulletin (2001) Vol. 33, Issue: 3*, 137-140.

[Schatzberg 02] Schatzberg, L. (2002). Applying Bloom's and Kolb's Theories to Teaching Systems Analysis and Design. *In The Proc. of the Information Systems Education Conference (ISECON 2002), vol. 19* (San Antonio). § 342b.

[Scott 03] Scott, T. (2003). Bloom's taxonomy applied to testing in computer science classes. *Journal of Computing Sciences in Colleges. Vol 19 (1),* 267-274.

[Shneider 06] Shneider, E. & Gladkikh, O. (2006). Designing questioning strategies for information technology courses. *The 19th Annual Conference of the National Advisory Committee on Computing Qualifications: Preparing for the Future – Capitalising on IT*. Wellington, National Advisory Committee on Computing Qualifications: Mann, S. and Bridgeman, N. eds. 243- 248.

[Skinner 38] Skinner, B.F. (1938). *The behaviour of organisms*. Nueva York: Appleton-Century. Traducción española: La conducta de los organismos: un análisis experimental. (1979). Barcelona: Fontanella.

[Skinner 74] Skinner, B.F. (1974). *About behaviorism*. Nueva York, USA: Alfred A. Knopf. Traducción española: Sobre el conductismo, Barcelona, Orbis, 1987.

[Slavin 90] Slavin, R.E. (1990). *Cooperative Learning-Theory, Research and Practice*. Englewood Cliffs, NJ: Prentice Hall.

[Stahl 81] Stahl, R.J. & Murphy, G.T. (1981). The domain of cognition: An alternative to Bloom's cognitive domain within the framework of an information processing model. *ERIC Document reproduction service No. ED 208 511*.

[Stasko 98] Stasko, J., Domingue, J., Brown, M. H., & Blaine, B. A. (eds.). (1998). *Software Visualization*. Cambridge, MA, USA: The MIT Press.

[Thomas 02] Thomas L., Ratcliffe M., Woodbury M. & Jarman E. (2002). Learning styles and performance in the introductory programming sequence. *In Proc. of SIGCSE'02 Technical Symposium on Computer Science Education.* Covington, Kentucky, USA, 33-37.

[Thompson 08] Thompson, E., Luxton-Reilly, A., Whalley, J., Hu, M. & Robbins, P. (2008). Bloom's taxonomy for CS assessment. *Proceedings of the Tenth Australasian Computing Education Conference (ACE2008)*, Wollongong, Australia, 155-161.

[Van Haaster 04] Van Haaster, K. & Hagan, D. (2004). Teaching and learning with BlueJ: An evaluation of a pedagogical tool. *In Information Science and Information Technology Education Joint Conference*, Rockhampton, QLD, Australia, 455-470.

[Velázquez 08] Velázquez, J.A., Pérez Carrasco, A. & Urquiza, J. (2008). SRec: An animation system of recursion for algorithm courses, *In 13rd Annual Conference on Innovation and Technology in Computer Science Education (ITiCSE 2008)*, 225-229.

[Velázquez 09] Velázquez-Iturbide, J.Á., Lázaro-Carrascosa, C.A., Hernán-Losada, I. (2009). Asistentes interactivos para el aprendizaje de algoritmos voraces, *IEEE Revista Iberoamericana de Tecnologías del Aprendizaje*, *IEEE-RITA, 4(3)*, 213-220.

[Venables 05] Venables, W.N., Smith, D.M. & the R Development Core Team (2005). *An Introduction to R.* Consultada el 26-05-2005 desde http://www.r-project.org/.

[Vygostky 78] Vygotsky, L.S (1978). *Mind in society: The development of higher psychological processes*. Cambridge MA: Harvard University Press.

[Watson 13] Watson, J.B. (1913). Psychology as the behaviorist views it. *Psychological Review. Vol 20*, 158–277.

[Weber 01] Weber, G. & Brusilovsky, P. (2001). ELM-ART: An Adaptive Versatile System for Web-based Instruction. *In International Journal of Artificial Intelligence in Education (IJAIED), 12*, 351-384