

Aprendizaje activo en informática educativa, el caso de la construcción de animaciones de programas

Jaime Urquiza Fuentes¹

¹ Departamento de Lenguajes y Sistemas Informáticos I,
Universidad Rey Juan Carlos
C/ Tulipán, s/n, 28933 Móstoles, Madrid, España
jaime.urquiza@urjc.es

Abstract. En este capítulo analizamos los distintos niveles de implicación de los estudiantes con las animaciones de programas, en el marco de la enseñanza de la programación funcional. La evaluación mide los resultados a corto plazo de tres niveles de implicación distintos: sin visión, visión y construcción. Desde un enfoque general, tanto ver como construir mejoran significativamente los resultados de los alumnos, sobre todo en los niveles superiores de la taxonomía de Bloom, análisis y síntesis. Sin embargo, entrando más en detalle con los conceptos tratados, parece que los temas más abstractos se benefician más de explicaciones añadidas, ya sea en las propias animaciones o del profesor, que de una mayor implicación construyendo animaciones. En temas con menor grado de abstracción las animaciones, verlas o construirlas, implican una mejora significativa. Además, según la opinión de los alumnos, la construcción de animaciones es una tarea que la gran mayoría utilizaría como ejercicio educativo, ya sea sola o acompañada de la consulta de animaciones.

1 Introducción

Las animaciones de software se usan en la enseñanza de la informática desde principios de los 80 [1,2]. Estas animaciones se pueden usar en múltiples escenarios [3]: clases, laboratorios, estudio fuera de clase, etc. Sin embargo, a pesar del potencial educativo que poseen, su uso no se ha incorporado al día a día de la enseñanza de la informática.

Una de las posibles definiciones de visualización es: “representación visual que ayuda a la formación de modelos mentales sobre conceptos abstractos”. Las animaciones son visualizaciones dinámicas, por lo tanto, las animaciones de programas deberían ayudar a la formación de modelos mentales sobre el funcionamiento de los programas, ya que los conceptos involucrados en el funcionamiento de los programas son abstractos; así que parece lógico intentar utilizar las visualizaciones para enseñar programación.

Aunque sea poco riguroso, en principio, para cualquier estudiante es más atractivo poder ver el funcionamiento de un programa, que atender a una explicación de un profesor. Por otro lado, no estamos hablando de sustituir al profesor, sino de propor-

cionar nuevos medios que ayuden a los alumnos en el aprendizaje de las materias. Podríamos mencionar diferentes ventajas del uso de la Visualización del Software en la enseñanza de programación, tanto desde el punto de vista del alumno, como del profesor:

1. Los conceptos abstractos visualizados se asimilan más fácilmente. La visualización aumenta la motivación.
2. El profesor puede elegir qué aspectos de la materia hay que mostrar en la visualización resaltando aquellos más importantes para el objetivo final.
3. Ayuda a la explicación del comportamiento dinámico de programas y algoritmos, permitiendo entre otros la detección de problemas y errores.
4. El ritmo de la visualización se puede adaptar al ritmo de aprendizaje de los alumnos.
5. Se pueden ofrecer distintas vistas del funcionamiento de un algoritmo, los alumnos pueden descubrir ellos mismos características del algoritmo.

Pero no debemos confiar en que estas ventajas son intrínsecas a las visualizaciones, tomando como dogma de fe cuando no lo es, la frase “*una imagen vale más que mil palabras*”. La mayoría de los autores que han trabajado con Visualización del Software en la enseñanza, véase [1,2,4,5,6,7,8,9], concluyen que la eficacia de su uso depende del cumplimiento de ciertas condiciones como:

- *Explicaciones añadidas a la visualización.* El hecho de ver una visualización no implica la comprensión de la misma, hay que explicar cómo funciona la visualización, qué muestra, cuál es su objetivo y cómo lo conseguirá.
- *Capacidad de abstracción.* La visualización debe resaltar la información que atañe a su objetivo, ocultando aquellos detalles que puedan confundir.
- *Diseño gráfico apropiado.* El diseño gráfico debe ser atractivo y claro, si los estudiantes no conocen las convenciones visuales, estas se deben explicar con anterioridad. La idea es minimizar el tiempo invertido en entender la visualización frente al tiempo invertido en comprender el algoritmo.
- *Complejidad variable.* La complejidad de lo que se muestra debería poder variar. Las visualizaciones para la primera vez que se explica un algoritmo no son las mismas que aquellas que muestran posibles optimizaciones o casos particulares de funcionamiento.
- *Temporización y vuelta atrás.* Tratándose de animaciones (visualizaciones dinámicas), la temporización es fundamental, la velocidad de la visualización al principio será mucho menor que al final. Al principio cuesta entender el algoritmo, pero al final, una vez comprendido puede resultar incluso aburrido. También es importante ofrecer la posibilidad de cambiar el sentido de ejecución del programa o algoritmo, pudiendo volver a ver la visualización de un momento anterior de la ejecución.
- *Multiplicidad de vistas.* Es conveniente poder ofrecer al usuario de la visualización varias vistas, de forma que pueda elegir cuál le interesa más. Por ejemplo, en un algoritmo de ordenación, podría interesar tanto la línea del algoritmo que se está ejecutando en ese momento, como el estado del conjunto que hay que ordenar.
- *Integración con el entorno de programación.* La generación de una visualización podría ser una acción más de las posibles, como ejecutar, compilar o depurar un

programa, permitiendo al alumno experimentar con el sistema creando sus propias visualizaciones.

Como se puede observar, algunas de estas condiciones no están relacionadas directamente con la visión pasiva de animaciones. Desde el punto de vista del estudiante, las tres últimas condiciones le obligan a interactuar más con las animaciones. En este capítulo estudiamos el impacto pedagógico de diferentes formas de interactuar con las animaciones de programas funcionales.

El resto del capítulo se estructura como sigue. En la siguiente sección revisamos los distintos modos de interacción entre las animaciones y el alumno, también conocidos como niveles de implicación. A continuación describimos las animaciones de programas funcionales utilizadas para este estudio. Después detallamos el diseño y los resultados de la evaluación. Finalmente, exponemos nuestras conclusiones, así como los trabajos futuros.

2 Niveles implicación con las animaciones

Como se ha explicado en la introducción, la eficacia pedagógica de las animaciones no se obtiene directamente con la visión de las mismas, sino que es posible que se necesite una forma diferente de interactuar con ellas. Desde el punto de vista del constructivismo, la visión pasiva de una animación no aporta mucho, sin embargo una implicación mayor mejoraría el aprendizaje. Por lo tanto parece interesante analizar formas diferentes de interactuar con las animaciones.

2.1 Taxonomía de niveles de implicación

La interacción con las animaciones se ha interpretado como la implicación del alumno en su uso. Así, Naps et al [10] crearon una taxonomía para diferenciar los distintos niveles de implicación de los alumnos con las visualizaciones, cada nivel de implicación supone una forma diferente de interactuar con ellas. A continuación describimos brevemente cada uno de estos niveles:

- *Sin visión*, que caracteriza a aquellos entornos donde no se usa ninguna tecnología relacionada con la visualización.
- *Visión*, es el nivel más básico, donde el alumno ve la animación. Existen dos clases de visión: la pasiva donde el alumno no hace nada más, y la activa donde el alumno es capaz de controlar propiedades básicas como el sentido, la velocidad o seleccionar diferentes vistas.
- *Respuesta*, implica que el alumno, además de ver la animación, debe contestar a ciertas preguntas planteadas durante la misma.
- *Cambio*, permite al alumno aportar datos de entrada para el algoritmo, de forma que controla la ejecución del mismo.
- *Construcción*, requiere que el alumno genere la animación, ello implica que el alumno podría seleccionar qué partes de la animación se muestran y cuáles no. Esto se puede hacer de dos formas: usando la codificación del algoritmo (propia o ajena), aportando los datos de entrada y obteniendo los resultados en imágenes

o animaciones; o utilizando una herramienta de dibujo que le permita generar dichas animaciones, incluso un generador de presentaciones valdría. Nótese por tanto, que en este nivel no se requiere la codificación del algoritmo.

- *Presentación*, requiere que se exponga una animación a una audiencia, para después obtener opiniones sobre ella o celebrar debates.

2.2 Eficacia pedagógica de los niveles de implicación con las animaciones

La idea subyacente a esta taxonomía es: *cuanta más implicación con la animación, mejor es el aprendizaje*. Sin embargo, no hay estudios que respalden esta afirmación en toda su amplitud. Sí existen experiencias alentadoras, que muestran beneficios de las animaciones en situaciones concretas, p.e. uso de las animaciones en conjunción con otras características como evaluación automática, o con cierto tipo de estudiantes; o mejoras sólo en ciertos niveles de aprendizaje, p.e. niveles en la taxonomía de Bloom [11].

Grissom et al [12] estudian el nivel *respuesta* con animaciones de algoritmos, detectando mejoras en los dos primeros niveles de la taxonomía de Bloom.

Moskal et al [13] muestran resultados positivos con el nivel de *cambio*, pero sólo en estudiantes con poca experiencia en programación o pocos conocimientos matemáticos, y alta probabilidad de abandonar el estudio de la materia. Ben-Bassat et al [14] también analizan el nivel de *cambio*, obteniendo resultados positivos en los estudiantes medios, según ellos, los estudiantes buenos no necesitaban de las visualizaciones, y los malos las encontraban demasiado complejas para ellos.

Laakso et al [15] estudian el nivel de *construcción*, junto con herramientas de simulación y evaluación automática; los resultados muestran mejoras en el porcentaje de aprobados/suspensos.

A nivel general, Hundhausen et al [16] hicieron un meta-estudio sobre la eficacia pedagógica de las animaciones de algoritmos, concluyendo que el uso de las animaciones desde el punto de vista constructivista permitía obtener mejoras en el aprendizaje. Por ello, aumentando la implicación de los estudiantes con las animaciones de algoritmos, se puede mejorar el aprendizaje de estos. Aunque la implicación mencionada en su estudio no estaba directamente relacionada con los niveles de implicación antes mencionados, es decir, los resultados de Hundhausen et al no prueban la relación de orden implícita en los distintos niveles de implicación expuestos por Naps et al [10].

Existen otros trabajos que no se relacionan directamente con las visualizaciones pero sí las utilizan como un valor añadido, un ejemplo son los *problets* de Kumar. Tutores capaces de generar, de forma aleatoria, problemas según unas plantillas; el énfasis lo pone en el suministro de feedback tras las respuestas de los alumnos. Kumar [17] describe los resultados de usar visualizaciones, solas y acompañadas de explicaciones textuales, como feedback. El aprendizaje de los alumnos mejora cuando se usan las visualizaciones acompañadas de explicaciones textuales.

3 Animaciones de programas funcionales

Existe una cantidad trabajo significativa sobre la visualización del paradigma de programación funcional [18]. Los esfuerzos más importantes se han realizado a nivel de depuración, aunque también existen esfuerzos en el ámbito educativo. Sin embargo no conocemos ningún resultado empírico que pruebe el beneficio pedagógico de estas visualizaciones.

Nuestro trabajo se ha centrado en crear animaciones de programas funcionales con fines educativos. Así, hemos desarrollado un IDE con capacidades de visualización integradas [19], siguiendo algunos de los requisitos mencionados en la primera parte de este capítulo.

Una descripción detallada sobre las animaciones generadas se puede encontrar en [20]. A modo de resumen podemos decir que hemos diseñado un proceso de construcción que requiere poco esfuerzo. Además desde el punto de vista del profesor, se pueden usar en clase, así como reutilizarlas fácilmente. Por otro lado los alumnos pueden utilizarlas como material complementario a su estudio.

Hemos realizado dos evaluaciones sobre estas animaciones. Una evaluación piloto desde el punto de vista de los algoritmos [21], cuyos resultados, aunque tienen una generalización limitada, son prometedores ya que muestran mejoras de los estudiantes que se dedicaron a construir animaciones con respecto a los que únicamente las vieron. También hemos realizado una evaluación a largo plazo [22], los resultados muestran que el uso de animaciones, ya sea construyéndolas o como herramienta de consulta, mejora significativamente el aprendizaje de los conceptos estudiados. Además, la construcción de las animaciones motiva significativamente más a los estudiantes, mejorando su interés por la asignatura.

El resto del capítulo presenta una evaluación del impacto a corto plazo de tres usos diferentes de las animaciones: no-uso, visión, construcción.

4 Diseño de la evaluación

En esta evaluación investigamos los efectos a corto plazo de la visión y la construcción de animaciones de programas funcionales como herramientas educativas. Usando la taxonomía de niveles de implicación con las animaciones de Naps et al [10], hemos comparado el nivel de “construcción” –implementado con nuestro enfoque de construcción sin esfuerzo [20]–, con los niveles “sin visión” y “visión”. Mediremos dichos efectos en términos de los conocimientos adquiridos por los alumnos y su opinión sobre las animaciones.

El contexto de esta evaluación es la asignatura de Bases de Lenguajes de Programación de las titulaciones de Ingeniería Técnica en Informática Gestión y Sistemas, de la Universidad Rey Juan Carlos. La evaluación se centró en la segunda mitad de la asignatura, que está dedicada al paradigma de programación funcional, donde se utiliza el lenguaje *HOPE* [23].

4.1 Participantes

Los participantes son estudiantes de la asignatura antes mencionada, del primer año de la titulación. El número total es de 132 estudiantes divididos en tres grupos: los que siguen un enfoque típico de enseñanza de la asignatura sin utilizar animaciones (GT, $n=42$), los que usan las animaciones sólo para verlas (GV, $n=50$), y los que se dedican a construir sus propias animaciones (GC, $n=40$).

La participación de los estudiantes es voluntaria e incentivada. La asignatura consta de dos partes, una práctica y otra teórica, cuyo peso en la nota final es 25% y 75% respectivamente. La parte práctica se califica sobre 10 puntos, y la participación en la evaluación se premia como máximo con 1 punto extra en la parte práctica.

Hemos filtrado aquellos estudiantes repetidores, de forma que ninguno de los participantes tenga conocimientos previos de programación funcional.

4.2 Variables

La variable independiente de la evaluación es el nivel de implicación con las animaciones. Las variables dependientes son la eficacia pedagógica, medida según la taxonomía de Bloom [11] con las respuestas a tres test de conocimientos sobre el paradigma funcional (véase anexo A) y la opinión de los estudiantes, recogida mediante dos cuestionarios uno para GV y otro para GC.

4.3 Protocolo

Esta evaluación se ha realizado con vistas a analizar los efectos del uso de las animaciones a corto plazo.

En primer lugar describiremos la estructura de la materia, después explicaremos la metodología didáctica empleada con cada grupo, y finalmente detallaremos el desarrollo temporal de la evaluación.

Estructuración de la materia

La materia pretende dar una visión básica del paradigma de programación funcional, sin entrar en profundidad con características avanzadas – como funciones de orden superior, estrategia de evaluación perezosa –, utilizando el lenguaje funcional *HOPE* [23]. En *HOPE* se usa la estrategia de evaluación impaciente, más fácil de entender, otros lenguajes como *HASKELL* [24] usan la estrategia de evaluación perezosa; no se mezclan características de otros paradigmas, como hace *ml* [25] con el paradigma imperativo, o *scheme* [26] y *lisp* [27] con el lógico; y tiene una sintaxis más literal, al contrario que los mencionados *scheme* o *lisp*. La materia está dividida en ocho unidades temáticas:

1. *El paradigma funcional*, donde se exponen los conceptos básicos y las propiedades del paradigma, sin entrar en detalles del lenguaje utilizado.

2. *Características básicas del lenguaje HOPE*, donde se explican las propiedades básicas del lenguaje: operadores, precedencia y asociatividad, tipos básicos de datos, y expresiones sencillas.
3. *Conceptos básicos de recursividad*. El mecanismo de repetición de procesos en HOPE es la recursividad, por ello se da un tema enteramente dedicado a ella.
4. *Operadores prefijos e infijos*. Por defecto, las funciones definidas por el programador se usan de forma prefija, mientras operadores “básicos”¹ del lenguaje, como los aritméticos, los lógicos, o los de comparación, se usan de forma infija. Sin embargo, en HOPE se pueden definir funciones para usar de forma infija, además se hace hincapié en la definición de la precedencia de operadores.
5. *Tipos de datos simples definidos por el usuario*. Declaración y uso de tipos de datos simples creados por el programador. Son simples puesto que sus valores, o se definen por extensión (y por lo tanto son finitos, p. ej., tipos enumerados), o se definen en función de otros tipos declarados previamente, como los básicos.
6. *Definiciones locales*. Declaración y ejecución de definiciones locales. Optimización por medio de declaraciones locales.
7. *Tipos de datos recursivos*. Declaración y uso de tipos de datos recursivos creados por el programador. Son recursivos ya que su dominio se puede definir de forma inductiva. Tipos recursivos estándar como listas, o árboles binarios.
8. *Polimorfismo*. Ventajas de los tipos polimórficos, uso de listas polimórficas.

Metodología didáctica utilizada. Los estudiantes de los tres grupos recibieron tanto clases magistrales, como sesiones de laboratorio donde se les proponían un conjunto de prácticas optativas sobre los conceptos explicados anteriormente. El contenido de los materiales didácticos utilizados era común: teoría en transparencias y ejercicios del libro [28].

Las clases magistrales recibidas por los estudiantes del GT seguían una metodología docente típica con utilización de presentaciones de ordenador y pizarra. Las sesiones de laboratorio consistían en la codificación de problemas propuestos por el profesor, utilizando el entorno WinHIPE, y la posterior explicación de la solución.

Las clases magistrales recibidas por los estudiantes de GV y GC seguían una metodología docente similar a GT, junto con la exposición de animaciones por parte del profesor. Dichas animaciones estaban hechas con WinHIPE² y eran explicadas por el profesor durante su exposición. Las sesiones de laboratorio de ambos grupos eran totalmente distintas.

Las sesiones de laboratorio del GV consistían en el estudio de un conjunto de animaciones generadas con WinHIPE, la figura 1 muestra el ejemplo de enunciado. Dichas animaciones implementaban problemas del libro de ejercicios antes mencionado.

¹ En HOPE, todo son funciones, incluidos los operadores, que en otros lenguajes parecerían básicos, como los citados en el texto.

² La flexibilidad de configuración de la apariencia gráfica de las visualizaciones, nos permitió generar animaciones con un tamaño de letra, mucho más grande, conveniente para su uso en clase.

Las sesiones de laboratorio del GC consistían en la construcción de un conjunto de animaciones con WinHIPE, la figura 2 muestra el ejemplo de enunciado. Dichas animaciones implementaban problemas del libro de ejercicios antes mencionado, a los estudiantes se les proporcionaba la descripción del problema y el código fuente de la solución; al tener que generar la animación, ellos tenían que escribir la descripción de la solución y generar las visualizaciones correspondientes. En la figura 3 resumimos la metodología didáctica utilizada en cada grupo.

BLP - ITISM - Tipos de datos recursivos

Vea todas las animaciones que pueda, intentando comprender cada uno de los pasos de la animación.

Tipos de datos recursivos:

1. Aritmética de Peano: [suma](#).
2. Listas:
 - Listas numéricas: [longitud de una lista](#) y [invertir una lista](#).
 - Tipo de datos vector: [acceso a un elemento de un vector](#).
3. Árboles binarios:
 - Función para calcular la [pertenencia a un árbol](#).
 - Función para calcular el [espejo de un árbol dado](#).

Fig. 1. Ejemplo de enunciado de sesión de prácticas del GV. Se proporcionaba a los alumnos un conjunto de animaciones para su estudio

BLP - ITIG - Tipos de datos recursivos

Construya todas las animaciones que pueda de las descritas a continuación.

Datos sobre las animaciones a construir:

Tipos de datos recursivos:

◦ Operación de suma en la aritmética de Peano.

<p>Descripción del problema</p> <p>Codificar un programa HOPE que calcule la suma de dos números representados mediante la aritmética de Peano. La aritmética de Peano es una definición recursiva de los números naturales. Dicha definición utiliza el número cero y la operación siguiente (sc), de forma que 1 es sc(cero) "siguiente de cero", 2 sería sc(sc(cero)) "siguiente de siguiente de cero", etc.</p>
<p>Código fuente</p> <pre>data nat == cero ++ sc (nat); ! Número natural dec snat : nat # nat -> nat; --- snat (cero , sc(a)) <= sc(a); --- snat (sc(a) , cero) <= sc(a); --- snat (sc(a) , sc(b)) <= snat(a,sc(sc(b)));</pre>

Fig. 2. Ejemplo de enunciado de sesión de prácticas del GC. Se proporcionaba a los estudiantes la descripción de un problema y el código fuente que lo resolvía, teniendo que construir la animación

Grupo	Clases teóricas	Laboratorios
GT	Exposición + ejemplos	Ejercicios + test conocimientos
GV	Exposición + ejemplos + Animaciones	Ver animaciones + test conocimientos
GC		Construir animaciones + test conocimientos

Fig. 3. Resumen de las metodologías utilizadas en cada grupo diferenciando clases teóricas y laboratorios.

Desarrollo temporal de la evaluación. Durante el primer tema no hubo sesión de laboratorio. Las sesiones de laboratorios del segundo y tercer tema se dedicaron a familiarizar a los estudiantes con el entorno de programación WinHIPE, así como con el lenguaje de programación HOPE.

A partir de aquí, se realizaron tres sesiones de laboratorio con los distintos tratamientos coincidiendo con los temas 4, 5 y 7. Al finalizar cada sesión de laboratorio, los estudiantes tenían que responder a un test de conocimientos sobre los conceptos que habían estado practicando, algunos de los ejercicios propuestos en este test implicaban el uso del entorno para codificar programas.

En la última sesión de tratamiento se facilitó a los estudiantes un cuestionario de opinión sobre el uso de las animaciones como herramientas educativas.

El último tema no tuvo un tratamiento diferenciado entre los grupos, y la última sesión de laboratorio del curso se dedicó a la resolución de dudas de los estudiantes.

5 Resultados de la evaluación

A continuación exponemos los resultados de la evaluación según los dos aspectos antes mencionados: eficacia de pedagógica de los tres enfoques y opinión subjetiva de los estudiantes.

5.1 Eficacia pedagógica

Hemos medido la eficacia pedagógica con las calificaciones de los estudiantes en los tres test de conocimientos. Dichos test estaban organizados según la taxonomía de Bloom [11]. Sólo hemos considerado a los participantes en la evaluación no repetidores, de esta forma eliminamos la posibilidad de que algunos estudiantes tengan conocimientos previos.

Hemos considerado esta variable desde cuatro puntos de vista distintos:

- global: donde combinamos todas las calificaciones de los distintos niveles de Bloom de los tres test en un solo valor.
- por niveles de Bloom: donde por cada nivel de Bloom combinamos todas las calificaciones de los tres test. De esta forma obtenemos 5 valores por cada estudiante.
- por tema: donde por cada test combinamos todas las calificaciones de los distintos niveles de Bloom. De esta forma obtenemos 3 valores por cada estudiante.
- por tema y nivel de Bloom: donde por cada test combinamos todas las calificaciones de preguntas y apartados asociadas a un mismo nivel de Bloom. De esta forma obtenemos 15 valores por cada estudiante (3 test x 5 niveles de Bloom).

Combinación de valores de distintos niveles de Bloom. Los puntos de vista *global* y *por tema* tienen que combinar en un solo valor las calificaciones de los cinco niveles de Bloom. Sin embargo, no hay una correspondencia clara entre los niveles de Bloom y la complejidad o importancia de cada uno de ellos. Por un lado la estructura jerárquica original de la taxonomía asignaba más importancia a cada nivel, siendo el nivel de conocimiento el menor de ellos y el nivel de evaluación el mayor. Algunos autores han usado esta ordenación de los niveles como marco de evaluación de los estudiantes [29]. Por otro lado existen estudios sobre la estructura de la taxonomía que concluyen que la estructura jerárquica original no es tan clara [30]. Los niveles de comprensión, aplicación y análisis se ajustan bien a la ordenación jerárquica. Por encima de estos se encuentran los niveles de síntesis y evaluación pero sin una ordenación clara entre ambos. Finalmente, no encuentran una ubicación clara para el nivel de conocimiento.

Nosotros hemos utilizado dos enfoques distintos para combinar los resultados de los distintos niveles de Bloom. En primer lugar hemos considerado la ordenación jerárquica original, de forma que la combinación de los cinco niveles de Bloom en un solo valor sigue la siguiente fórmula:

$$(\text{conocimiento} + 2*\text{comprensión} + 3*\text{aplicación} + 4*\text{análisis} + 5*\text{síntesis})/15 \quad (1)$$

Y en segundo lugar damos la misma importancia para todos los niveles, calculando la combinación de estos como la media aritmética de los cinco valores. Para el resto del capítulo nos referiremos dichos enfoques como valores ponderados o no ponderados respectivamente.

A continuación comparamos la eficacia pedagógica de los tres grupos utilizando los cuatro puntos de vista mencionados anteriormente.

Eficacia pedagógica global. En este enfoque calculamos un valor para cada estudiante. Este valor es el promedio de los valores correspondientes a cada test, siendo cada uno de estos la combinación de los valores de cada nivel de Bloom. Realmente tenemos dos valores por estudiante según hayamos combinado los valores de los distintos niveles de Bloom de forma ponderada o no. La tabla 1 muestra el análisis ANOVA de los tres grupos, y la tabla 2 muestra la comparación de los tres grupos de estudiantes dos a dos.

Tabla 1. Valoración media de cada grupo y análisis ANOVA sobre la eficacia educativa de los tres grupos³.

	GC	GV	GT	ANOVA
No ponderado	.6075	.6068	.5140	F(2; 120) = 3,274; p = .041
Ponderado	.5887	.5996	.4651	F(2; 120) = 6,061; p = .003

Tabla 2. Estudio comparativo dos a dos de la eficacia pedagógica global.

Grupos	No ponderado	Ponderado
GC vs GV	t(78,450) = ,018; p = ,986	t(78,784) = -,256; p = ,799
GC vs GT	t(74,360) = 2,311; p = ,024	t(75,975) = 3,020; p = ,003
GV vs GT	t(85) = 2,102; p = ,038	t(85) = 3,015; p = ,003

Por lo tanto, desde el punto de vista global, tanto ver como construir animaciones mejoran significativamente el aprendizaje de los conceptos en un 13% (valores no ponderados) o 9% (valores ponderados).

Eficacia pedagógica por niveles de Bloom. En este enfoque calculamos cinco valores para estudiante, uno para cada nivel de la taxonomía presente en los test. Los valores son el promedio de los resultados de cada test. La tabla 3 muestra los análisis ANOVA o X^2 de los tres grupos para cada nivel de la taxonomía de Bloom y la tabla 4 muestra la comparación de los tres grupos de estudiantes dos a dos para los niveles de Bloom donde se han encontrado diferencias significativas previamente.

Tabla 3. Valoración media de cada grupo y análisis ANOVA o X^2 de la eficacia pedagógica por niveles de Bloom.

Nivel	GC	GV	GT	Análisis ANOVA o X^2
Conocimiento	.6019	.5348	.5837	F(2; 120) = 1,084; p = ,342
Comprensión	.6525	.6794	.6316	F(2; 120) = ,561; p = ,572
Aplicación	.6623	.6988	.5972	X2(2 , 123) = 2,899, p = ,235
Análisis	.6675	.6009	.4487	X2(2 , 123) = 19,175, p < ,01
Síntesis	.4532	.5200	.3087	F(2; 120) = 7,268; p = ,001

³ Para el resto de la sección, destacamos los resultados estadísticos significativos en negrita.

Tabla 4. Estudio comparativo dos a dos de la eficacia pedagógica por niveles de Bloom.

Grupos	Análisis	Síntesis
GC vs GV	$t(78,450) = ,018; p = ,986$	$t(78,784) = -,256; p = ,799$
GC vs GT	U=318,0, p < ,01	t(76) = 2,667; p = ,009
GV vs GT	U=596,5, p = ,003	t(79,191) = 3,827; p < ,01

Como se puede ver no aparecen nuevos resultados respecto del análisis global, tanto ver como construir animaciones mejora el aprendizaje. Sin embargo hemos podido comprobar que los niveles de Bloom donde se detecta la mejora son análisis con un 18,6% y síntesis con un 16,9%.

Eficacia pedagógica por tema. En este enfoque calculamos tres valores por estudiante, uno para cada tema tratado: operadores infijos, tipos de datos definidos por el usuario y tipos de datos recursivos. Los valores son la combinación de los distintos niveles de Bloom presentes en cada test hecho para los tres temas. La tabla 5 muestra los análisis ANOVA o X^2 de los tres grupos para cada tema y la tabla 6 muestra la comparación de los tres grupos de estudiantes dos a dos para los temas donde se han encontrado diferencias significativas previamente.

Tabla 5. Valoración media de cada grupo y análisis X^2 de la eficacia pedagógica por temas. Las celdas muestran los resultados tanto para valores ponderados como no ponderados (resultado no ponderado / resultado ponderado).

Temas	GC	GV	GT	Análisis ANOVA o X^2
Operadores infijos	,5842 / ,6764	,5296 / ,5856	,4840 / 4902	$F(2; 123) = 1,153; p = ,224 / X^2(2, 126) = 3,830, p = ,147$
Tipos de datos definidos por el usuario	,6319 / ,5244	,6289 / ,5676	,3983 / ,2714	$X^2(2, 123) = 23,260, p < ,01 / F(2; 123) = 15,983; p < ,01$
Tipos de datos recursivos	,7055 / ,6561	,8529 / ,8306	,7923 / ,7600	$F(2; 98) = 8,019; p = ,001 / F(2; 98) = 7,161; p = ,001$

Tabla 6. Estudio comparativo dos a dos de la eficacia pedagógica por temas. Los nombres correspondientes a los temas son: TDDU para tipos de datos definidos por el usuario y TDR para tipos de datos recursivos. Las celdas muestran los resultados tanto para valores ponderados como no ponderados (resultado no ponderado / resultado ponderado).

Grupos	TDDU	TDR
GC vs GV	$t(79) = 1,161; p = ,249 / U=310,0, p < ,246$	t(64) = -4,322; p < ,01 / t(64) = -4,184; p < ,01
GC vs GT	U=688,0, p < ,01 / t(76) = 5,688; p < ,01	t(64) = -2,225; p = ,030 / t(64) = 2,113; p = ,038
GV vs GT	U=415,0, p < ,01 / t(76,878) = 5,963; p < ,01	$t(68) = 1,673; p = ,099 / t(62,974) = 1,535; p = ,130$

Con este enfoque podemos observar que las mejoras educativas se han producido en los temas de tipos de datos definidos por el usuario y los tipos de datos recursivos.

En el primero, de nuevo tanto ver como construir mejoran el aprendizaje un 23,2% (no ponderado) o 27,5% (ponderado). Sin embargo, en el segundo tema tanto ver animaciones como no verlas mejoran los resultados de los que las construyen en un 12% (no ponderado) o 14% (ponderado).

Eficacia pedagógica por tema y nivel de Bloom. En este enfoque calculamos 15 valores por estudiante, uno para cada nivel de Bloom de cada tema tratado. Las tablas de la 7 a la 15 muestran los análisis ANOVA o X^2 de los tres grupos para cada nivel de Bloom y tema así como las comparaciones de los tres grupos de estudiantes dos a dos para los niveles y temas donde se han encontrado diferencias significativas previamente.

Tal y como muestra la tabla 7, en el nivel de *conocimiento* no hemos detectado ninguna mejora entre ninguno de los grupos.

Tabla 7. Valoración media de cada grupo y análisis X^2 de la eficacia pedagógica por temas para el nivel de Bloom *conocimiento*.

Temas	GC	GV	GT	Análisis X^2
Operadores infijos	,3264	,2722	,3896	$X^2(2, 126) = 3,027, p = ,220$
Tipos de datos definidos por el usuario	,7569	,6278	,6190	$X^2(2, 126) = 1,613, p = ,446$
Tipos de datos recursivos	,8387	,9057	,8943	$X^2(2, 101) = 1,399, p = ,497$

En las tablas 8 y 9 se pueden observar las mejoras detectadas en el nivel de *comprensión* para cada tema. En este caso, las mejoras son variadas. Ver animaciones mejora la comprensión de operadores infijos respecto a construirlas en un 15,6%. Además, tanto ver como construir mejoran la comprensión de tipos de datos definidos por el usuario respecto a no usar animaciones en un 22,2%. Finalmente, en la comprensión de tipos de datos recursivos, ver animaciones mejora respecto a no usarlas en un 5,7% y no usar animaciones mejora respecto a construirlas en un 3,28%.

Tabla 8. Valoración media de cada grupo y análisis X^2 de la eficacia pedagógica por temas para el nivel de Bloom *comprensión*.

Temas	GC	GV	GT	Análisis X^2
Operadores infijos	,2308	,3867	,4229	$X^2(2, 126) = 2,747, p = ,253$
Tipos de datos definidos por el usuario	,9722	,9000	,7143	$X^2(2, 126) = 9,347, p = ,009$
Tipos de datos recursivos	,8756	,9654	,9084	$X^2(2, 101) = 22,114, p < ,01$

Tabla 9. Estudio comparativo dos a dos de la eficacia pedagógica por cada tema para el nivel de Bloom *comprensión*. Los nombres correspondientes a los temas son: OI para operadores infijos, TDDU para tipos de datos definidos por el usuario y TDR para tipos de datos recursivos.

Grupos	OI	TDDU	TDR
GC vs GV	U=588,5, p = ,033	U=743,0, p = ,161	U=226,0, p < ,01
GC vs GT	U=627,5, p = ,193	U=561,0, p = ,002	U=384,5, p = ,039
GV vs GT	U=932,5, p = ,915	U=774,0, p = ,035	U=342,0, p = ,001

En las tablas 10 y 11 se pueden observar las mejoras detectadas en el nivel de *aplicación* para cada tema. Aquí ver animaciones mejora la aplicación de tipos de datos definidos por el usuario respecto a no usar animaciones en un 17%, mientras que ver animaciones o no usarlas mejoran la aplicación de tipos de datos recursivos respecto a construir animaciones en un 19,3%.

Tabla 10. Valoración media de cada grupo y análisis ANOVA y X^2 de la eficacia pedagógica por temas para el nivel de Bloom *aplicación*.

Temas	GC	GV	GT	Análisis ANOVA o X^2
Operadores infijos	,7964	,7702	,6664	$X2(2, 126) = 0,790, p = ,674$
Tipos de datos definidos por el usuario	,7694	,7622	,5926	$X2(2, 126) = 3,450, p = ,178$
Tipos de datos recursivos	,4892	,7248	,6390	$F(2, 98) = 5,403, p = ,006$

Tabla 11. Estudio comparativo dos a dos de la eficacia pedagógica por cada tema para el nivel de Bloom *aplicación*. Los nombres correspondientes a los temas son: OI para operadores infijos, TDDU para tipos de datos definidos por el usuario y TDR para tipos de datos recursivos.

Grupos	TDDU	TDR
GC vs GV	U=775,5, p = ,704	t(64) = -3,319; p = ,001
GC vs GT	U=606,0, p < ,103	t(64) = -2,141; p = ,036
GV vs GT	U=730,5, p = ,042	t(68) = 1,174; p = ,245

En las tablas 12 y 13 se pueden observar las mejoras detectadas en el nivel de *análisis* para cada tema. En este caso, tanto ver como construir animaciones mejoran el análisis de tipos de datos definidos por el usuario respecto a no usar animaciones en un 39,7%.

Tabla 12. Valoración media de cada grupo y análisis X^2 de la eficacia pedagógica por temas para el nivel de Bloom *análisis*.

Temas	GC	GV	GT	Análisis X^2
Operadores infijos	,8608	,6582	,6033	$X2(2, 126) = 3,522, p = ,172$
Tipos de datos definidos por el usuario	,4367	,4669	,0552	$X2(2, 126) = 42,746, p < ,01$
Tipos de datos recursivos	,8177	,8693	,7923	$X2(2, 101) = 0,449, p = ,799$

Tabla 13. Estudio comparativo dos a dos de la eficacia pedagógica por cada tema para el nivel de Bloom comprensión. TDDU representa al tema sobre tipos de datos definidos por el usuario.

Grupos	TDDU
GC vs GV	U=776,5, p = ,737
GC vs GT	U=256,0, p < ,01
GV vs GT	U=236,0, p < ,01

En las tablas 14 y 15 se pueden observar las mejoras detectadas en el nivel de *síntesis* para cada tema. Tanto ver como construir animaciones mejoran la síntesis de operadores infijos y tipos de datos definidos por el usuario respecto a no usar animaciones en un 29,1% y 29,6% respectivamente, mientras que tanto ver como no usar animaciones mejoran la síntesis de tipos de datos recursivos respecto a construir animaciones en un 24,1%.

Tabla 14. Valoración media de cada grupo y análisis X^2 de la eficacia pedagógica por temas para el nivel de Bloom *síntesis*.

Temas	GC	GV	GT	Análisis X^2
Operadores infijos	,7056	,5578	,3405	$X^2(2, 126) = 11,730, p = ,003$
Tipos de datos definidos por el usuario	,2222	,3856	,0083	$X^2(2, 126) = 23,552, p < ,01$
Tipos de datos recursivos	,5016	,7929	,6929	$X^2(2, 101) = 10,926, p = ,004$

Tabla 15. Estudio comparativo dos a dos de la eficacia pedagógica por cada tema para el nivel de Bloom comprensión. Los nombres correspondientes a los temas son: OI para operadores infijos, TDDU para tipos de datos definidos por el usuario y TDR para tipos de datos recursivos.

Grupos	OI	TDDU	TDR
GC vs GV	U=653,0, p = ,116	U=645,0, p = ,319	U=293,0, p = ,001
GC vs GT	U=446,0, p = ,001	U=519,0, p < ,01	U=377,5, p = ,030
GV vs GT	U=700,5, p = ,026	U=476,5, p < ,01	U=537,5, p = ,349

5.4 Opinión de los estudiantes

Hemos recogido la opinión subjetiva de los estudiantes sobre las animaciones con un cuestionario. Sólo trabajaremos con los grupos GC y GV, ya que GT no tuvo ningún contacto con ellas. En este cuestionario hicimos 4 preguntas sobre la experiencia de los estudiantes con las animaciones, en la tabla 6 resumimos sus resultados. Podemos apreciar que la opinión de los estudiantes es muy buena. El 93% piensan que las animaciones ayudan a la comprensión de los conceptos, con una ligera diferencia entre GC y GV. Además, existe total unanimidad en la facilidad de construcción y uso de las animaciones. Y el 91,5% de los estudiantes piensan que las animaciones son útiles, de nuevo con una ligera diferencia entre GC y GV. Finalmente les preguntamos

sobre la disponibilidad de distintos formatos para ver las animaciones, el 78,9% mostraron su acuerdo en la utilidad de esta característica.

Tabla 6. Opinión de los alumnos sobre su experiencia con las animaciones. Cada fila se refiere a un aspecto. Las columnas indican, por cada grupo y en general, el % de estudiantes que están de acuerdo o totalmente de acuerdo con la afirmación.

	GC	GV	Total
Ayuda a la comprensión de los conceptos	86,2 %	100 %	93 %
Facilidad de construcción/ uso de las animaciones	100 %	100 %	100 %
Utilidad de las animaciones	83,4 %	100 %	91,5 %
Utilidad sobre la disponibilidad en varios formatos	83,3 %	74,3 %	78,9 %

También les preguntamos sobre el modo de uso de las animaciones que más favorecería el aprendizaje, les propusimos cuatro opciones:

- Construir es mejor ver.
- Ver es mejor que construir.
- Ambas son iguales pero deberían usarse conjuntamente.
- Ambas son iguales, cualquiera vale.

En la figura 4 mostramos un resumen de las respuestas en cada grupo. Como se puede observar, la importancia asignada por los estudiantes al proceso de construcción de las animaciones depende claramente del grupo: los estudiantes de GC la dan mayor importancia, los de GV la dan la misma que a la visión de animaciones.

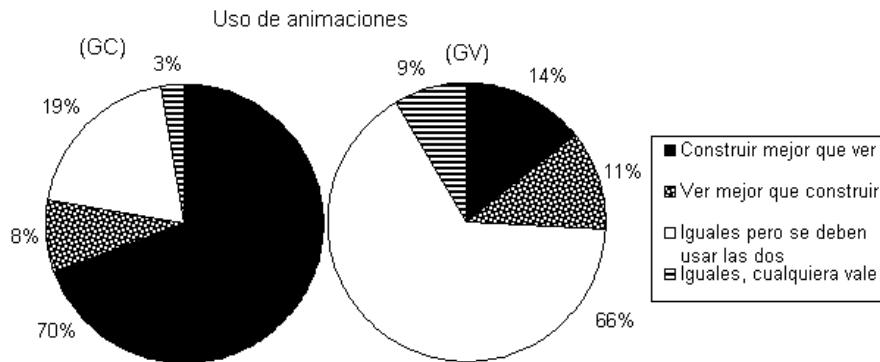


Fig. 4. Opinión de los estudiantes de los grupos GC y GV sobre el modo de uso de las animaciones, ¿que favorece más el aprendizaje?

6 Conclusiones y trabajo futuro

Esta evaluación sigue un diseño experimental controlado, donde hemos analizado el impacto en el aprendizaje de la programación funcional, del uso de animaciones de programas de dos formas distintas: mediante la construcción de animaciones que muestren el comportamiento del programa, o mediante el uso de las animaciones a modo de consulta para ver cómo funciona un programa. Además utilizamos un grupo de control que recibió las clases con el enfoque típico de la asignatura sin el uso animaciones. Estas tres formas distintas se corresponden con tres de los niveles de implicación descritos por Naps et al [10]: *construcción*, *visión* y *sin visión*, respectivamente.

El objetivo es comprobar el impacto a corto plazo de los distintos niveles de implicación en los conocimientos adquiridos durante las sesiones de trabajo, así como la opinión de los estudiantes sobre las animaciones.

6.1 Interpretación de los resultados

Los diferentes niveles de implicación requieren del estudiante diferentes formas de interactuar con las animaciones, y con lo que representan, ejecución de programas funcionales. En un campo similar como es la animación de algoritmos, Hundhausen et al [16] hicieron un meta-estudio sobre la eficacia de las animaciones, en cuanto a la eficacia de las animaciones de algoritmos, detectaron que la teoría pedagógica que mejor podía predecir la eficacia pedagógica era el constructivismo, cuanto mayor es la implicación de los estudiantes con las animaciones mayor es el aprendizaje.

La implicación de los estudiantes que construyen animaciones de programas, es mucho mayor que aquellos que solamente las ven. Construir una animación implica un conocimiento más profundo del programa ejecutado, saber qué pasos son los importantes en cada momento de la ejecución, de forma que la animación se suficientemente explicativa de la ejecución del programa. Además, los estudiantes que construyeron animaciones tenían que escribir una descripción de la solución al problema implementada en el programa que estaban animando, lo que implica una tarea de reflexión sobre el programa. Por otro lado, la visión de las animaciones requiere que los estudiantes sean capaces de interpretar el funcionamiento del programa, sabiendo qué pasos se han ejecutado para llegar a cada fotograma de la animación.

En general la construcción de la animación requiere de un estudio más detenido del programa a animar, consiguiendo que el estudiante dedique más tiempo y atención al programa. Viendo animaciones, el tiempo y grado de atención dedicado a estas es menor, ya que a los estudiantes les basta con interpretar para ellos mismos lo que están viendo, no tienen que explicarlo posteriormente a nadie. También podríamos decir que la construcción animaciones obliga a reflexionar más que la simple visión de estas.

Claramente, el nivel de implicación es mayor para los estudiantes que construyeron animaciones, que para los que sólo las vieron. Por lo tanto, los primeros deberían obtener mejores resultados que los segundos. Además, si el uso de animaciones implicara una mejora de aprendizaje los resultados de los dos grupos anteriores deberían

ser mejores que los del grupo que no usó animaciones. Sin embargo, a primera vista esto no es así, en la figura 5 mostramos un resumen de las mejoras detectadas en los análisis anteriormente realizados.

% de mejora	OI	TDDU	TDR	Total
Conocimiento				
Comprensión	15,6	22,2	5,7 – 3,28	
Aplicación		17	19,3	
Análisis		39,7		18,6
Síntesis	29,1	29,6	24,1	16,9
Total (NP/P)		23,2 / 27,5	12 / 14	13 / 9

GC/V > GT	GV > GT	GV > GC	GV > GT > GC	GV/T > GC
-----------	---------	---------	--------------	-----------

Fig. 5. Resumen de mejoras pedagógicas (en porcentaje) detectadas desde los cuatro puntos de vista: global (celda de la esquina inferior derecha), por nivel de Bloom (1ª columna por la derecha), por tema (fila inferior de la tabla) y, por tema y nivel de Bloom (resto de celdas). En la parte baja se encuentra la leyenda que identifica qué grupo/s han obtenido mejoras respecto a qué otro/s grupo/s, así X > Y implica que el grupo X obtuvo mejores resultados que el Y.

Desde una perspectiva general parece que tanto ver como construir animaciones mejoran los resultados del enfoque típico sin uso de animaciones. Dichas mejoras oscilan entre un 13% y un 9% con el punto de vista global. Además hemos detectado mejoras en los niveles de Bloom que supuestamente son más complejos e indican un mayor dominio de la materia, análisis y síntesis, con un 18,6% y 16,9% de mejora respectivamente.

Sin embargo, el análisis por temas arroja otros resultados distintos. En primer lugar, el factor común de todas las mejoras detectadas es la visión de animaciones con explicaciones. Además podemos detectar una clara diferencia entre los temas de tipos de datos. Así, mientras que en los tipos de datos definidos por el usuario el grupo con peores resultados es el que no usa animaciones (el de menor implicación), en los tipos de datos recursivos el peor grupo es el de construcción de animaciones (el de mayor implicación). La diferencia entre ambos temas estriba en los conceptos de recursividad e inducción, cuestiones más abstractas. A la luz de estos resultados, cuanto mayor es la abstracción mayor es la importancia de las explicaciones añadidas, ya sean en las animaciones (grupo de visión de animaciones) o del profesor (grupo sin uso de animaciones).

A pesar de estos resultados pedagógicos variados, los estudiantes reconocen a las animaciones como herramientas útiles que les ayudan a comprender los conceptos explicados. Más aún, el nivel de construcción, que requiere más esfuerzo por parte de los estudiantes, goza de una buena opinión ya que la gran mayoría no descarta la construcción como forma de usar las animaciones con fines pedagógicos.

6.3 Trabajo futuro

En este capítulo hemos tratado tres niveles de implicación concretos con las animaciones: no usar las animaciones, verlas (con explicaciones añadidas) y construirlas. Los resultados muestran que el nivel de implicación no es el único factor decisivo en la eficacia pedagógica del uso de las animaciones. Aunque existen experiencias con otros niveles [31], se hace necesario un estudio más detallado de la eficacia del resto de niveles de implicación así como la relación entre ellos.

Además, se ha visto como las explicaciones añadidas tienen cierta influencia sobre la eficacia de las animaciones. Por lo tanto parece lógico profundizar en qué características hacen de las animaciones herramientas educativas eficaces. En concreto nos proponemos estudiar el efecto de la narrativa asociada a las animaciones, así como la retroalimentación y pistas asociadas a preguntas planteadas durante las animaciones.

Agradecimientos

Este trabajo se ha realizado gracias los fondos proporcionados por el proyecto TIN2008-04103 del Ministerio de Ciencia y Tecnología del Reino de España. Agradecemos la participación de los alumnos matriculados en la asignatura de Bases de Lenguajes de Programación del 1er curso de las titulaciones de Ingeniería Técnica en Informática de Sistemas y Gestión de la Universidad Rey Juan Carlos, del año académico 2005/2006.

References

1. Baecker, R.: Sorting out sorting: A case study of software visualization for teaching computer science. In: Stasko, J.T., Domingue, J., Brown, M.H., Price, B.A. (eds.): *Software Visualization*. MIT Press, Cambridge MA (1998) 369-381
2. Bazik, J., Tamassia, R., Reiss, S. P., van Dam, A.: Software visualization in teaching at Brown University. In: Stasko, J.T., Domingue, J., Brown, M.H., Price, B.A. (eds.): *Software Visualization*. MIT Press, Cambridge MA (1998) 383-398
3. Pareja-Flores, C., Velázquez-Iturbide, J. Á.: Program execution and visualization on the Web. In: A. Aggarwal (ed.), *Web-based Learning and Teaching Technologies: Opportunities and Challenges*. Idea-Group Publishing, Hershey, PA (2002) 236-259
4. Anderson, J.M. and Naps, T.L.: A context for the assessment of algorithm visualization system as pedagogical tools. In *Proceedings of the First International Program Visualization Workshop*. University of Joensuu Press, Joensuu, Finland (2001) 121-130.
5. Ben-Ari, M.: Program visualization in theory and practice. *Informatik/Informatique*, 2 (2001) 8-11.
6. Gloor, P.A.: Animated Algorithms. In: Stasko, J.T., Domingue, J., Brown, M.H., Price, B.A. (eds.): *Software Visualization*. MIT Press, Cambridge MA (1998) 409-416.
7. Kehoe, C., Stasko, J.T. and Taylor, A.: Rethinking the evaluation of algorithm animations as learning aids: An observational study. *git-gvu-99-10*, Technical report. Georgia Institute of Technology, Georgia, USA (1999). <ftp://ftp.cc.gatech.edu/pub/gvu/tech-reports/1999/99-10.pdf> (2006)

8. Mulholland, P. and Eisenstadt, M.: Using software to teach computer programming: Past present and future. In: Stasko, J.T., Domingue, J., Brown, M.H., Price, B.A. (eds.): *Software Visualization*. MIT Press, Cambridge MA (1998) 399-408.
9. Böcker, H.D., Fisher, G. and Nieper, H.: The enhancement of understanding through visual representations. *Proceedings of the ACM SIGCHI '86 Conference on Human Factors in Computing*. ACM Press, New York, USA (1986) 44-50.
10. T. Naps, G. Röbling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J.Á. Velázquez-Iturbide.: *Iticse 2002 working group report: Exploring the role of visualization and engagement in computer science education*. *ACM SIGCSE Bulletin, Working group reports from ITiCSE on Innovation and technology in computer science education*, 35(2), (2002):131–152.
11. Bloom, B., Furst, E., Hill, W., y Krathwohl, D.R.: *Taxonomy of Educational Objectives: Handbook I, The Cognitive Domain*. Addison-Wesley (1959)
12. Grissom, S., McNally, M.F., y Naps, T.L.: Algorithm visualization in CS education: comparing levels of student engagement. *Proceedings of the 2003 ACM symposium on Software Visualization*. ACM Press New York, NY, USA (2003) 87-94
- 13 Moskal, B., Lurie, D., y Cooper, S.: Evaluating the Effectiveness of a New Instructional Approach. *Proceedings of the Technical Symposium on Computer Science Education, SIGCSE'04*. ACM Press New York, NY, USA (2004) 75-79
14. Ben-Bassat, R., Ben-Ari, M., y Uronen, P.A.: The Jeliot 2000 program animation system. *Computers & Education* 40(1) (2003) 1-15
15. Laakso, M-J., Salakoski, T., Grandell, L., Qiu, X., Korhonen, A., y Malmi, L.: Multi-Perspective Study of Novice Learners Adopting the Visual Algorithm Simulation Exercise System TRAKLA2. *Informatics in Education* 4(1) (2005) 49-68
16. Hundhausen, C.D., Douglas, S.A., y Stasko, J.T.: A Meta-Study of Algorithm Visualization Effectiveness. *Journal of Visual Languages and Computing* 13(3) (2002) 259-290
17. Kumar, A.N.: Results from the evaluation of the effectiveness of an online tutor on expression evaluation. *SIGCSE '05: Proceedings of the 36th SIGCSE technical symposium on Computer science education*. ACM Press New York, NY, USA (2005) 216-220
18. Urquiza-Fuentes, J., y Velázquez-Iturbide, J.Á.: Program Visualization for the Functional Paradigm. *Proceedings of the Third Program Visualization Workshop, WarwreckPrint, Coventry, UK, (2004)*, 2-9
19. Velázquez-Iturbide, J.Á., Pareja-Flores, C., y Urquiza-Fuentes, J.: An approach to effortless construction of program animations. *Computers & Education*. E imprenta.
20. Urquiza-Fuentes, J., y Velázquez-Iturbide, J.Á.: Effortless Construction and Management of Program Animations on the Web. *Advances in Web-Based Learning – ICWL 2005: Fourth International Conference. Lecture Notes in Computer Science*, vol. 3583, Springer-Verlag, Berlin, (2005) 163-173
21. Urquiza-Fuentes, J., y Velázquez-Iturbide, J.Á.: An evaluation of the effortless approach to build algorithm animations with WinHIPE. *Proceedings of the Fourth Program Visualization Workshop (PVW 2006)*, (2006) 29-33
22. Urquiza-Fuentes, J.: Evaluación de niveles de implicación de los estudiantes con animaciones de programas funcionales. In: Velázquez-Iturbide, J.Á., Lovillo Gil, A. (eds.): *Actas del I Seminario de Investigación en Tecnologías de la información Aplicadas a la Educación SITIAE 2007*. Universidad Rey Juan Carlos-Dykinson S.L., Madrid (2009) 151-172
23. <http://www soi.city.ac.uk/~ross/Hope/> (2009)
24. <http://www.haskell.org/> (2009)
25. Milner, R., Tofte, M., y Harper, R.: *The definition of Standard ML*. MIT Press, (1990)
26. Rees, J., y Clinger, W.: Revised report on the algorithmic language scheme. *ACM SIGPLAN Notices* 21(12) (1986) 37 - 79
27. <http://www.lisp.org> (2009)

28. Velázquez Iturbide, J.Á., Sánchez Calle, Á., Duarte Muñoz, A., y Lázaro Carrascosa, C.A.: Ejercicios Resultados de Bases de Lenguajes de Programación. Editorial universitaria Ramón Areces, Madrid, España, (2005)
29. Lister, R. and Leaney, J. First year programming: let all the flowers bloom. In Proceedings of the Fifth Australasian Conference on Computing Education - Volume 20 (Adelaide, Australia). T. Greening and R. Lister, Eds. Conferences in Research and Practice in Information Technology Series, vol. 140. Australian Computer Society, Darlinghurst, Australia, (2003) 221-230.
30. Anderson, L.W., & Krathwohl (Eds.). A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives. New York: Longman. (2001).
31. Urquiza-Fuentes, J. and Velázquez-Iturbide, J. A Survey of Successful Evaluations of Program Visualization and Algorithm Animation Systems. Transactions on Computing Education 9, 2 (2009) 1-21.

Anexo A: Test de conocimientos utilizados para medir la eficacia pedagógica

Cada test está estructurado según la taxonomía de Bloom [11]. Dentro de cada pregunta se especifica el nivel de la taxonomía correspondiente entre paréntesis y en negrita.

A.1 Test de conocimientos sobre operadores infijos y prefijos

1. **(Conocimiento)** Define el concepto operador infijo.
2. **(Conocimiento)** ¿Qué diferencia existe entre un operador infijo y uno prefijo?
3. **(Comprensión)** A la vista de la siguiente expresión:

```
ident(3=6 , 4<6) nor true and false imp xor(4<5
nand true , 2=0);
```

donde además de operadores predefinidos en HOPE, se usan los siguientes operadores definidos por el usuario: nand, nor, xor, imp, ident.

- a) copie solamente el(los) nombre(s) de el(los) operador(es) prefijo(s).
- b) copie solamente el(los) nombre(s) de el(los) operador(es) infijo(s).

4. **(Aplicación)** Dado el siguiente programa HOPE:

```
infix y : 6;
dec y : truval # truval -> truval;
--- false y _ <= false;
--- true y b <= b;

infix o : 5;
dec o : truval # truval -> truval;
```

```

--- false o b <= b;
--- true o _ <= true;

infix ny : 5;
dec ny : truval # truval -> truval;
--- true ny true <= false;
--- _ ny _ <= true;

infix no : 6;
dec no : truval # truval -> truval;
--- false no false <= true;
--- _ no _ <= false;

dec n : truval -> truval;
--- n true <= false;
--- n false <= true;

infix ox : 7;
dec ox : truval # truval -> truval;
--- a ox a <= false;
--- _ ox _ <= true;

```

- a) Escriba el resultado de ejecutar la siguiente expresión:
true ny false y true ny true;
- b) Escriba el resultado de ejecutar la siguiente expresión:
false no true o true no false ox true;
- c) Escriba el resultado de ejecutar la siguiente expresión:
true ox n false no true ny true no n false;

5. Dada la siguiente descripción de un problema:

Se pide implementar un operador infijo para la doble implicación lógica \leftrightarrow .
Sólo se usará con valores lógicos, su tabla de verdad es:

A	B	A \leftrightarrow B
T	T	T
T	F	F
F	T	F
F	F	T

Este operador es de alto nivel, lo que significa que cualquier otra operación lógica predefinida (and, or, not, comparaciones) debe evaluarse antes que esta.

- a) (**Análisis**) copie la parte de la descripción referente a la precedencia del operador.
- b) (**Análisis**) copie la parte de la descripción referente a la declaración del operador.

c) (**Análisis**) copie la parte de la descripción referente a la definición (ejecución o evaluación) del operador.

d) (**Síntesis**) codifique un programa HOPE que implemente el operador antes descrito.

A.2 Test de conocimientos sobre tipos de datos definidos por el usuario

1. (**Conocimiento**) ¿Para qué sirve la sentencia type?
2. (**Conocimiento**) ¿Para qué sirve la sentencia data?
3. Dado el siguiente programa HOPE:

```
data INTERVALO ==
  Ext (real # real) ++      ! Extremos
  Cen (real # real);      ! Centro y radio

type INTERVALO_ENTERO == num # num;

dec Dentro : INTERVALO # real -> truval;
--- Dentro(Ext(i,s) , p) <= (p >= i) and (p <= s);
--- Dentro(Cen(c,r) , p) <= (p >= (c-r)) and
                               (p <= (c+r));

dec incluye : INTERVALO # INTERVALO -> truval;
--- incluye(Ext(i1,s1) , Ext(i2,s2)) <=
    (Dentro(Ext(i1,s1),i2) and Dentro(Ext(i1,s1),s2));
```

a) (**Comprensión**) Copie aquí el trozo de código donde se definen tipos definidos por el usuario que sólo renombran a otros tipos.

b) (**Comprensión**) Copie aquí el trozo de código donde se definen tipos definidos por el usuario que crean nuevos tipos.

c) (**Aplicación**) Escriba el resultado de ejecutar la siguiente expresión:
`incluye(Ext(4.0*3.2 , 25.1) , Ext(13.0,20.9));`

d) (**Aplicación**) Escriba una expresión que cumpla las siguientes tres condiciones:

- d1.- durante su evaluación debe llamar a la función incluye,
- d2.- alguno de los cálculos lógicos realizados durante la evaluación de la llamada a la función incluye devuelva true, y
- d3.- como resultado final debe devolver false.

4. Dada la siguiente descripción de un problema:

Se trata de implementar en HOPE la posibilidad de utilizar números complejos. Los números complejos tienen dos componentes, la parte real y la parte imaginaria. Evidentemente, la parte real está representada por un número real. La parte

imaginaria es la multiplicación de un número real por el número i (resultado de la raíz cuadrada de -1).

Los números complejos se representan de la forma : $a + bi$. Donde a y b son números reales, y a es la parte real y bi es la parte imaginaria.

Además se pide implementar tres operaciones a realizar con los números complejos:

- suma de complejos: $(a+bi) + (c+di) = (a+c) + (b+d)i$
- multiplicación de complejos: $(a+bi) * (c+di) = (ac-bd) + (ad+bc)i$
- módulo de un complejo: raíz cuadrada de la suma de a al cuadrado más b al cuadrado

a) (**Análisis**) copiar la parte de la descripción referente a la definición de un nuevo tipo de dato a crear con la sentencia `data`.

b) (**Análisis**) copiar la(s) parte(s) de la descripción donde se haga referencia al uso del tipo de datos creado, como parámetros de llamada.

c) (**Análisis**) copiar la(s) parte(s) de la descripción donde se haga referencia al uso del tipo de datos creado, como valor(es) devuelto(s) por una función.

d) (**Síntesis**) codifique un programa HOPE que implemente el tipo de datos y las operaciones antes descritas.

A.3 Test de conocimientos sobre tipos de datos recursivos

1. (**Conocimiento**) Describe las diferencias que existen entre los tipos de datos recursivos y los no recursivos.
2. Dado el siguiente código fuente:

```
data Stack == EndOfStack ++ Elem(num # Stack);
```

a) (**Comprensión**) ¿Es este tipo de datos recursivo (SI/NO)?:

b) Si respondió que sí a la pregunta anterior:

(**Comprensión**) Copiar la parte del código que se refiere al(a los) caso(s) base(s).

(**Comprensión**) Copiar la parte del código que se refiere al(a los) caso(s) recursivo(s).

(**Aplicación**)Escriba un ejemplo de valor de este tipo de datos:

3. Dado el siguiente código fuente:

```
data Stack == EndOfStack ++ Elem(num # num);
```


- a) **(Comprensión)** ¿Es este tipo de datos recursivo (SI/NO)?:
- b) Si respondió que sí a la pregunta anterior:
- (Comprensión)** Copiar la parte del código que se refiere al(a los) caso(s) base(s).
- (Comprensión)** Copiar la parte del código que se refiere al(a los) caso(s) recursivo(s).
- (Aplicación)** Escriba un ejemplo de valor de este tipo de datos:

4. Dado el siguiente código fuente:

```
data Graph ==
  EndNode(num) ++
  InitNode( list(Graph) # num ) ++
  InternalNode( list(Graph) # num ) ++
  Emptygraph;
```

- a) **(Comprensión)** ¿Es este tipo de datos recursivo (SI/NO)?:
- b) Si respondió que sí a la pregunta anterior:
- (Comprensión)** Copiar la parte del código que se refiere al(a los) caso(s) base(s).
- (Comprensión)** Copiar la parte del código que se refiere al(a los) caso(s) recursivo(s).
- (Aplicación)** Escriba un ejemplo de valor de este tipo de datos:

5. Los conjuntos de números son estructuras de datos que sirven para definir agrupaciones de números. También existe el concepto del conjunto vacío, que es el conjunto que no tiene ningún elemento.

- a) **(Análisis)** Copiar la parte de la descripción que se refiere al(a los) caso(s) base(s) del tipo de datos:
- b) **(Análisis)** Copiar la parte de la descripción que se refiere al(a los) caso(s) recursivo(s) del tipo de datos:
- c) **(Síntesis)** Codificar el tipo de dato:
- d) **(Comprensión⁴)** Codificar la función cardinal de conjunto. El cardinal de un conjunto es el número de elementos que tiene ese conjunto.

6. Los árboles ternarios numéricos son aquellos que están compuestos por nodos que pueden tener como máximo 3 nodos hijos.

Cada nodo tiene un valor numérico asociado. Un árbol ternario puede ser un árbol vacío o un nodo con 0, 1, 2 o 3 nodos hijos. La idea básica es que cada nodo hijo es en sí otro árbol.

⁴ Aunque este ejercicio podría ser del nivel de síntesis, lo clasificamos en el de comprensión porque los estudiantes hicieron otro similar en clase de teoría.

- a) (**Análisis**) Copiar la parte de la descripción que se refiere al(a los) caso(s) base(s) del tipo de datos:
- b) (**Análisis**) Copiar la parte de la descripción que se refiere al(a los) caso(s) recursivo(s) del tipo de datos:
- c) (**Síntesis**) Codificar el tipo de dato árbol ternario:
- d) (**Síntesis**) Escribir la expresión que representa al árbol de la figura.

