Evaluación de Niveles de Implicación de los Estudiantes con Animaciones de Programas Funcionales

Jaime Urquiza Fuentes[1]

[1] Departamento de Lenguajes y Sistemas Informáticos,
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 tuvo lugar a lo largo de un curso de programación funcional, dividimos a los estudiantes en tres grupos: los que no usaron animaciones, los que las usaron como material de consulta, y los que se dedicaron a construirlas. Comparando la opinión de los estudiantes, los datos de presentación al examen y las calificaciones obtenidas, concluimos que: el uso de animaciones, ya sea construyéndolas o como herramienta de consulta, mejora significativamente el aprendizaje de los conceptos estudiados, además la opinión de los alumnos coincide con este resultado; sin embargo, la construcción de las animaciones motiva significativamente más a los estudiantes, mejorando su interés por la asignatura.

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 proporcionar 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 revisaremos 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 detallaremos el diseño y los resultados de la evaluación. Finalmente, expondremos 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.
- *Contestación*, 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 di-

chas 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 *contestación* 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 una evaluación piloto del uso de estas animaciones desde el punto de vista de los algoritmos [21]. Los 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. El resto del capítulo presenta una evaluación continuación natural de la anteriormente mencionada, donde analizamos tres usos diferentes de las animaciones: no-uso, visión, construcción.

4 Diseño de la evaluación

En esta evaluación investigamos los efectos 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 la actitud de los alumnos hacia la asignatura, su calificación en el examen, 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.

4.1 Participantes

Los participantes son estudiantes de la asignatura antes mencionada, del primer año de la titulación. El número total es de 132, 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 con las respuestas al examen de la asignatura sobre el paradigma funcional; la actitud de los estudiantes hacia la asignatura y las animaciones, medida con los datos de presentación de los alumnos al examen y visitas a las animaciones disponibles en la página web de la asignatura; y la opinión de los estudiantes, recogida mediante dos cuestionarios uno para GV y otro para GC.

4.3 Protocolo

Está evaluación se ha realizado con vistas a analizar los efectos del uso de las animaciones a largo plazo. La idea principal es que los alumnos se familiaricen con una forma de usar las animaciones, y las utilicen durante todo del desarrollo de la materia.

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* [21]. En *HOPE* se usa la estrategia de evaluación impaciente, más fácil de entender, otros lenguajes como *HASKELL* [22] usan la estrategia de evaluación perezosa; no se mezclan características de otros paradigmas, como hace *ml* [23] con el paradigma imperativo, o *scheme* [24] y *lisp* [25] 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 [26].

Las clases magistrales recibidas por los estudiantes del GT seguían una metodología docente típica con utilización de transparencias (PowerPoint) 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 WinHIPE[2] y eran explicadas por el profesor durante su exposición. Las sesiones de laboratorio de ambos grupos eran totalmente distintas.

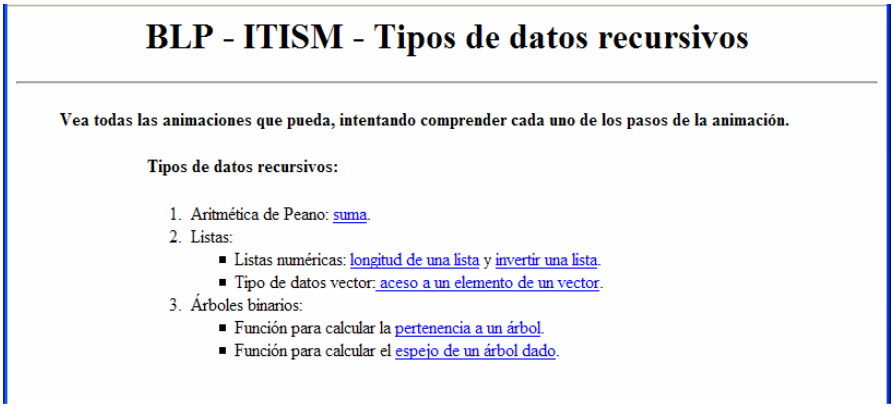
Las sesiones de laboratorio del GV consistían en el estudio de un conjunto de animaciones generadas con WinHIPE, la figura 1 muestra el ejemplo de enunciado. Dichas animaciones implementaban problemas del libro de ejercicios antes mencionado.

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

[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.

solución; al tener que generar la animación, ellos tenían que escribir la descripción de la solución y generar las visualizaciones correspondientes.



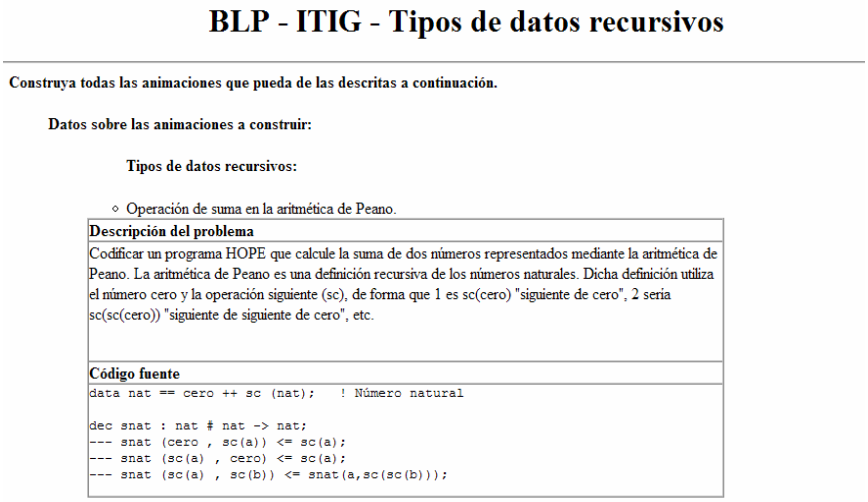
BLP - ITISM - Tipos de datos recursivos

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: aceso 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.

Descripción del problema
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.
Código fuente

```
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)));
```

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

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, y durante los siguientes 4 temas empezamos a aplicar los diferentes tratamientos a cada grupo en sus respectivas sesiones de laboratorio. 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.

Una semana más tarde se celebró el examen de la asignatura, de donde obtuvimos los datos para medir el conocimiento de los estudiantes sobre la materia.

5 Resultados de la evaluación

A continuación exponemos los resultados de la evaluación según los tres aspectos antes mencionados: actitud de los estudiantes respecto a la asignatura y las animaciones, eficacia de pedagógica de los tres enfoques, y opinión subjetiva de los estudiantes.

5.1 Actitud de los estudiantes

Hemos medido la actitud de los estudiantes en función de: si los estudiantes se presentaron al examen de junio de la asignatura o no, y el uso real de las animaciones publicadas en las páginas web de la asignatura.

En la tabla 1 mostramos los datos de presentación al examen por parte de los alumnos de la asignatura. Como se puede ver en la tabla, los datos de presentación al examen de *todos los estudiantes matriculados* son muy similares en los tres grupos, e independientes según el test de X^2 de Pearson. Sin embargo, las diferencias aparecen cuando únicamente estudiamos los *alumnos que participaron en la evaluación*. El test de X^2 de Pearson detecta dependencias prácticamente significativas (p<0,058), lo que nos indica que es posible la existencia de diferencias entre los grupos, pero no entre todos, de hecho, podemos observar que el grupo GC se ha presentado en mayor medida que los otros dos.

Para profundizar en el estudio de la presentación de los estudiantes al examen de la asignatura, hemos analizado los datos de los grupos por parejas; mostramos los resultados en la tabla 2. Se puede ver que sí existen diferencias significativas entre GC y GT, el porcentaje de presentación de GC fue un 19,12% mayor que el de GT; así como diferencias prácticamente significativas entre GC y GV, donde el porcentaje de presentación de GC fue un 14,29% mayor que el de GV; finalmente, no encontramos diferencias significativas entre GV y GT.

Tabla 1. Datos de presentación al examen. Analizamos, divididos por cada grupo, el porcentaje de estudiantes presentados al examen de la asignatura, respecto de todos los estudiantes matriculados en la asignatura, o sólo aquellos que hayan participado en la evaluación.

	Matriculados en el grupo	Participantes en la evaluación
Estudiantes GC	66,39% (79/119)	83,93% (47/56)
Estudiantes GV	63,44% (59/93)	69,64% (39/56)
Estudiantes GT	56,6% (90/159)	64,81% (35/54)
Test de dependencia	X^2 (2, 371) = 2,956 , p=0,228	X^2 (2, 167) = 5,73 , p=0,057

Tabla 2. Análisis de presentación al examen por parejas. Analizamos diferencias significativas, utilizado el test *U* de Mann-Whitney, entre los grupos, contando únicamente con los participantes en la evaluación

	Diferencia en % de presentación	Test de diferencias significativas
GC vs GT	**19,12%**	***U*=1223 , *p*=0,022**
GC vs GV	14,29%	U=1348,5 , p=0,054
GV vs GT	4,83%	U=1348,5 , p=0,688

En cuanto al uso real de las animaciones, monitorizamos los accesos realizados a estas durante la semana anterior al examen de la asignatura, problemas técnicos nos impidieron monitorizar los accesos durante más tiempo, aún así, la cantidad de accesos fue muy numerosa, 542 accesos en total.

5.3 Eficacia pedagógica

Hemos medido la eficacia pedagógica con las calificaciones de los estudiantes en el examen de la asignatura. Sólo hemos considerado a los participantes en la evaluación no repetidores, de esta forma eliminamos la posibilidad de que algunos estudiantes tengan conocimientos previos.

El paradigma funcional representa la mitad de esta asignatura, por ello sólo consideraremos los resultados obtenidos en las preguntas y ejercicios relacionados con el paradigma funcional. Todas las calificaciones están normalizadas al rango [0,1]. Para valorar la calificación global en el paradigma funcional, utilizaremos los mismos pesos que se asignan en el cómputo de la nota final de la asignatura. En la calificación global se contaba la práctica obligatoria (O), cuyo peso es de 25% de la nota, y el examen de teoría cuyo peso es de 75% ; dentro del examen de teoría había un cuestión teórica (C) cuyo peso es de 33,3% y un problema (P), cuyo peso es el 66,6% restante de la nota del examen. La fórmula de cálculo para saber la nota final en el paradigma funcional de los alumnos es:

$$\text{Calificación total } T(O,C,P) = 0{,}25*O + 0{,}75*((C+2*P)/3)$$

En la tabla 3 mostramos el análisis de las calificaciones, por cada ejercicio y la calificación total, de los tres grupos. Se puede observar que hay dependencias prácticamente significativas en *C* y *T*; además, podemos percibir un cierto patrón en estos resultados, GC y GV obtienen calificaciones similares entre si, y mejores que GT; este mismo patrón también aparece en *P*, aunque menos significativo (p=0,141); así que haremos un análisis más detallado por pares de los tres *C*, *P* y *T*.

Tabla 3. Datos de calificaciones obtenidas en el examen por los participantes no repetidores. Analizamos, por cada ejercicio, la nota media de cada grupo y el análisis de dependencia asociado. *O*, *C* y *P* no son normales por lo que usaremos el test de Krukal-Wallis; mientras que *T* sí es normal, por lo que usaremos un análisis ANOVA.

	GC	GV	GT	Dependencia X^2
Práctica obligatoria(*O*)	0,5486	0,65	0,525	$X^2(2, 116) = 1,77$, p=0,412
Cuestión teórica (*C*)	0,7122	0,7233	0,5194	X^2 (2, 116) = 5,74, p=0,057
Problema (*P*)	0,3954	0,3707	0,2808	X^2 (2, 116) = 3,92, p=0,141
Calificación total (*T*)	0,5129	0,5287	0,4015	$F(2,113)= 2,528$ p=0,084

Analizando los resultados de la cuestión teórica (*C*), el problema (*P*) y la calificación total (*T*), por parejas, ver tabla 4, descubrimos que sí existen diferencias significativas entre GV y GT; GV mejoró los resultados de GT un 20,4% en la cuestión teórica y un 12,7% en la calificación total. También existen diferencias casi-significativas ($p<0,075$) entre GC y GT; GC mejoró los resultados de GT un 19,3% en la cuestión teórica, un 11,5% en el problema y un 11,1% en la calificación total . Mientras que entre GC y GV no hay diferencias significativas.

Tabla 4. Análisis de calificaciones en el examen por parejas. Analizamos diferencias entre significativas, utilizado los test *U* de Mann-Whitney, y t-Student, entre los grupos, contando únicamente con los participantes en la evaluación. Destacamos en negrita las diferencias significativas.

	GC-GV	GC-GT	GV-GT
C	U=765 , p=0,743	U=505 , p=0,060	**U=565,5 , p=0,028**
P	U=743,5 , p=0,615	U=492,5 , p=0,052	U=630,5, p=0,153
T	$t(78) = -0,268$, p=0,789	$t(71) = 1,813$, p =0,074	**$t(77) = 2,028$, p =0,046**

Finalmente, hemos hecho un análisis de las calificaciones cualitativas en función de la calificación total (*T*). Las calificaciones cualitativas serían: suspenso, aprobado, notable y sobresaliente. Detectamos una dependencia significativa entre los grupos y estas calificaciones, $X^2(6, N=116)= 13,408$, p=0,037. En la tabla 5 mostramos los rangos de calificación cuantitativa correspondientes, así como los porcentajes para cada grupo. Como se puede ver, el grupo GT es el que más suspensos tiene, el GC aprobados y notables, y el GV el que más sobresalientes tiene.

Tabla 5. Análisis de calificaciones cualitativas. Para los cuatro niveles de calificación mostramos el rango asociado de las calificaciones cuantitativas, así como el porcentaje de estudiantes que obtuvieron dicha calificación en cada uno de los tres grupos, en negrita destacamos el grupo que obtuvo mayor porcentaje en cada calificación.

	Rango	GC	GV	GT
Suspenso	x < 0,5	43,2 %	46,5 %	**61,1 %**
Aprobado	0,5 <= x < 0,7	**35,1 %**	25,6 %	22,2 %
Notable	0,7 <= x < 0,9	**21,6 %**	14,0 %	16,7 %
Sobresaliente	x > 0,9	0,0 %	**14,0 %**	0,0 %

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 3 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.

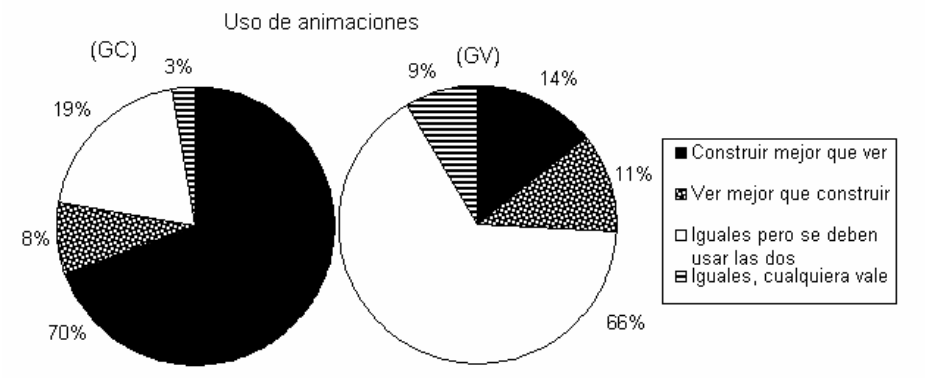


Fig. 3. Opinión de los estudiantes de los grupos GC y GV sobre el modo de uso de las animaciones, ver o construir, ¿que favorece más el aprendizaje?

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 de los distintos niveles de implicación en: la actitud de los estudiantes hacia la asignatura, la actitud hacia las animaciones, los resultados académicos, y la opinión de los estudiantes sobre las animaciones.

6.1 Análisis de factores externos a la evaluación

La situación ideal de esta evaluación sería un entorno aséptico donde el único factor influyente fuera el grado de implicación de los estudiantes con las animaciones. Sin embargo, los estudiantes son personas individuales con personalidad propia, sobre los que pueden afectar condicionantes externos al tratamiento, como la preparación académica previa a la universidad, el hábito de estudio, o la actitud hacia la titulación que están estudiando.

Para tratar de aislar al máximo el factor, detectando la posible influencia de estos condicionantes externos, hemos estudiado los datos de las asignaturas del mismo

curso y examinadas en la misma convocatoria que la que es objeto de la evaluación, *Bases de Lenguajes de Programación*. En concreto, hemos verificado si existe alguna relación entre los datos de presentación y calificaciones obtenidas por los alumnos en dicha asignatura, con respecto a las otras, donde no se ha utilizado el tratamiento anteriormente descrito.

Existían cuatro posibles asignaturas, dos anuales, *Metodología y Tecnología de la Programación* (MTP), y *Estructura y Tecnología de los Computadores* (ETC); y dos cuatrimestrales, *Cálculo*, y *Álgebra*. Las dos primeras no sirven de referencia debido a su alto grado de complejidad. Al ser las dos asignaturas más difíciles del curso, si un estudiante no ha sido capaz de aprobar la asignatura evaluada, es seguro que tampoco habrá aprobado ninguna de las dos anuales. Por el contrario, si un estudiante es capaz de aprobar (incluidos notables y sobresalientes) las asignaturas anuales, es prácticamente seguro que habrá podido aprobar la asignatura evaluada.

Así que utilizaremos las asignaturas cuatrimestrales, *Cálculo* y *Álgebra*, que tienen la misma duración que la signatura de la evaluación, y además son similares en cuanto a complejidad. Desafortunadamente, no hemos podido contar con toda la información necesaria en la asignatura de *Álgebra*, por lo que finalmente, trabajaremos con *Cálculo* como asignatura de referencia.

Hemos utilizado el coeficiente de correlación φ para verificar la independencia de los datos de presentación al examen de las asignaturas. Los indicadores estadísticos (ver tabla 7 del Anexo A) verifican la inexistencia de correlación entre los datos de presentación tanto a nivel general como a nivel particular por cada grupo (p>0,05). Sólo en el caso de los estudiantes que vieron las animaciones (GV) existe la posibilidad de una correlación estadísticamente casi significativa (p=0,052), sin embargo la fuerza de esta correlación es muy débil (φ<0,3).

Las fórmulas de cálculo de las calificaciones numéricas varían entre las asignaturas, por ello no haremos ningún análisis al nivel de las calificaciones cuantitativas. El análisis de dependencia lo haremos con las calificaciones cualitativas desde dos puntos de vista: con las cuatro calificaciones posibles - suspenso, aprobado, notable o sobresaliente - y con la calificación como suspenso o no suspenso (aprobado, notable y sobresaliente). Para estos análisis hemos usado respectivamente el coeficiente τ–b de Kendall, y el coeficiente de correlación φ. De nuevo, los indicadores estadísticos (ver tabla 8 Anexo A) verifican la inexistencia de correlación (p>0,123) entre las calificaciones, tanto a nivel general como a nivel particular por cada grupo.

Según estos resultados, podemos asegurar que los datos recogidos en la asignatura evaluada, tanto de presentación como de calificaciones, no tienen correlación alguna con los datos de la asignatura de referencia, donde no se han utilizado animaciones.

Sabemos que existen factores, como la preferencia, o aptitudes intelectuales, que pueden favorecer a alguna de las asignaturas, por ejemplo, un estudiante puede ser bueno programando pero costarle mucho entender cuestiones de más abstracción como las existentes en *Cálculo*. También puede darse el caso de planificaciones globales del curso, seleccionando las asignaturas para aprobarlas en convocatorias concretas, por ejemplo, un estudiante puede dedicarse a estudiar álgebra en junio y programación funcional en septiembre.

Sin embargo, la mayoría de los factores externos que pudieran influir en la presentación a un examen o en la calificación obtenida, deberían reflejarse en las califica-

ciones de las otras asignaturas. Si un estudiante es muy bueno, debería obtener buenas calificaciones, si no es buen estudiante debería obtener malas calificaciones o no presentarse a los exámenes. Por lo tanto, podemos suponer que a nivel de los tres grupos de estudiantes, las diferencias existentes en cuanto a presentación al examen y calificaciones obtenidas en la asignatura evaluada están principalmente causadas por el nivel de implicación con las animaciones.

6.2 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, 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 la 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. Sin embargo, a primera vista esto no es así. No hemos encontrado diferencias significativas en las calificaciones, entre los grupos GC y GV, pero sí en el porcentaje de presentados al examen. Vamos a calcular cuántos aprobados más hay en GC por el hecho de haberse presentado en un 14,3% más que GV. Repartiendo esta diferencia de presentados según los suspensos y aprobados de GC (36,2% y 47,7% respectivamente), recuérdese que hay más presentados en GC, tenemos que existe un 8,1% más de aprobados debido ala construcción de animaciones.

Además, la presentación al examen es un indicador de la actitud de los estudiantes hacia la asignatura. La construcción de animaciones ha influido claramente en esta

actitud. En concreto, la construcción de animaciones (83,9% de presentados) aumenta la presentación al examen con respecto a la simple visión de las mismas (69,6% de presentados).

Finalmente, comparando los resultados con el grupo de control que no usó animaciones de ninguna forma, comprobamos que estas han tenido un efecto positivo en GC y GV en conjunto. Los grupos que utilizaron animaciones han obtenido mejores resultados, mejoras de un 11,9% en la nota cuantitativa, que los que no utilizaron animaciones. En lo que se refiere a las calificaciones cualitativas, la diferencia principal se encuentra en la proporción suspensos/aprobados (incluyendo a los notables y sobresalientes junto con los aprobados), donde los grupos que usaron animaciones obtuvieron entre un 16,3% más de aprobados que el grupo que no usó animaciones. Además, los estudiantes que no utilizaron animaciones fueron los que menos se presentaron al examen (64,8% de presentados).

Podemos concluir que, en general, las animaciones, ya sean usadas al nivel de visión o de construcción, son una herramienta pedagógica mejora el aprendizaje de la programación funcional con respecto a un enfoque típico sin animaciones. Los estudiantes las reconocen como herramientas útiles que les ayudan a comprender los conceptos explicados. Además, el nivel de construcción parece dar mayor confianza en los conocimientos obtenidos o infundir mayor interés por la asignatura, tanto por la presentación al examen, como por la opinión de los estudiantes, 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 hablado tanto de la visión de animaciones como de su construcción partiendo de problemas resueltos. Sin embargo, hacer el programa que resuelve el problema propuesto es el fin de las asignaturas de programación.

La continuación natural de esta evaluación sería investigar el impacto de añadir la generación de animaciones a la tarea de codificar el problema que resuelve un problema propuesto. Este nuevo uso de las animaciones obligaría a los estudiantes, no sólo a codificar el programa y probar a sí mismos que funciona, sino a generar una animación que demuestre y explique su funcionamiento. Así conseguiríamos que la implicación de los estudiantes fuera todavía más activa, y en términos constructivistas, esto debería mejorar el aprendizaje.

Esta evaluación se ha centrado en el uso de las animaciones por parte de los estudiantes. Sin embargo, las animaciones generadas para su consulta, fueron generadas por profesores. Queda pendiente realizar una evaluación de la usabilidad de esta herramienta desde el punto de vista de los profesores.

Agradecimientos

Este trabajo se ha realizado gracias los fondos proporcionados por el proyecto TIN2004-07568 del Ministerio de Educación y Ciencia del Reino de España. Agra-

decemos 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. También agradecemos la colaboración prestada por la universidad a la hora de recolectar datos de otras asignaturas, en concreto al V*icerrectorado de Alumnos*, al *Vicerrectorado de Relaciones Internacionales, Nuevas Tecnologías y Promoción Informática*, a la *Dirección del Servicio de Informática*, y a la *Dirección de la Escuela del Campus de Móstoles* y su *Servicio de Alumnos*. Finalmente agradecemos a los profesores Mofdi El Anjoumi El Amrani, Isidoro Hernán Losada, Begoña Jiménez Martín, Luis Rincón Córcoles, Sergio Sánchez García y Luis Villa Conde su ayuda a la hora de interpretar las calificaciones de sus respectivas asignaturas.

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, Findland (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. Rößling, 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 Objetives: 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 Sceince 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, WarwirckPrint, 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
21. http://www.soi.city.ac.uk/~ross/Hope/ (2007)
22. http://www.haskell.org/ (2007)
23. Milner, R., Tofte, M., y Harper, R.: The definition of Standard ML. MIT Press, (1990)
24. Rees, J., y Clinger, W.: Revised report on the algorithmic language scheme. ACM SIGPLAN Notices 21(12) (1986) 37 - 79
25. http://www.lisp.org (2007)
26. Velázquez Iturbide, J.Á., Sánchez Calle, Á., Duarte Muñoz, A., y Lázaro Carrascosa, C.A.: Ejercicios Resultos de Bases de Lenguajes de Programación. Editorial universitaria Ramón Areces, Madrid, España, (2005)

Anexo A: Análisis de dependencias entre los datos de la asignatura evaluada y la asignatura de referencia

La tabla 7 muestra los resultados del análisis de dependencia según el coeficiente de correlación entre los presentados al examen de la asignatura evaluada y los presentados a la asignaturas de referencia. De forma análoga, la tabla 8 muestra los resultados de los análisis para calificaciones suspensos/aprobados, respectivamente.

Tabla 7. Análisis de correlacción en los datos de presentación, en general y por cada uno de los grupos. (*N; φ; p*) *N* representa el número de casos tratados, *φ* representa el coeficiente de correlación (rango entre 0 y 1, cuanto más cercano a 1 mayor correlacción) y *p* representa la significatividad de la correlación.

	Análisis de correlación
General	(*N*=132 ;*φ*=0,099 ; *p*=0,255)
Estudiantes GC	(*N*=34 ;*φ*=0,026 ; *p*=0,881)
Estudiantes GV	(*N*=44 ;*φ*=0,293 ; *p*=0,052)
Estudiantes GT	(*N*=54 ;*φ*=-0,093 ; *p*=0,496)

Tabla 8. Análisis de correlacción en los datos de calificaciones cualitativas y en la relación suspensos/aprobados. Para las calificaciones cualitativas y por cada uno de los grupos. (*N; τ-b; p*) *N* representa el número de casos tratados, *τ-b* representa el coeficiente de correlación (rango entre 0 y 1, cuanto más cercano a 1 mayor correlacción) y *p* representa la significatividad de la correlación. Para la relación suspensos/aprobados y por cada uno de los grupos. (*N; φ; p*) *N* representa el número de casos tratados, *φ* representa el coeficiente de correlación (rango entre 0 y 1, cuanto más cercano a 1 mayor correlacción) y *p* representa la significatividad de la correlación.

	Calificaciones cualitativas	*Suspensos/Aprobados*
General	(*N*=48 ; *τ-b* =0,108 ; *p*=0,397)	(*N*= 48 ;*φ*= 0,013 ; *p*=0,930)
Estudiantes GC	(*N*=15 ; *τ-b* =0,179 ; *p*=0,383)	(*N*=15 ;*φ*=0,189 ; *p*=0,464)
Estudiantes GV	(*N*=24 ; *τ-b* =0,102 ; *p*=0,572)	(*N*=24 ;*φ*=-0,063 ; *p*=0,759)
Estudiantes GT	(*N*=9 ; *τ-b* =-0,360 ; *p*=0,123)	(*N*=9 ;*φ*=-0,060 ; *p*=0,858)