J. Ángel Velázquez Iturbide

Nuevo Análisis de la Evaluación de un Método Instruccional para el Aprendizaje de la Recursividad

Número 2015-03

Serie de Informes Técnicos DLSI1-URJC
ISSN 1988-8074
Departamento de Lenguajes y Sistemas Informáticos I
Universidad Rey Juan Carlos

Índice

1 Introducción 1
2 Descripción de la Evaluación 2
2.1 Población y Asignatura 2
2.2 Preguntas y Diseño de la Investigación 3
2.3 Tratamiento 3
2.4 Tareas 4
3 Modelos Mentales Identificados 4
3.1 Método de Análisis 4
3.2 Rastros, Evaluaciones y Soluciones 5
3.2 Modelos Mentales 8
4 Resultados 17
4.1 Resultados de P2 y P3 17
4.2 Coherencia o Fragilidad de los Modelos Mentales 20
4.3 Resultados de Diseño en P4 21
5 Conclusiones 23
Agradecimientos. 24
Referencias 24
Apéndice: Enunciado del Cuestionario 25

Nuevo Análisis de la Evaluación de un Método Instruccional para el Aprendizaje de la Recursividad

J. Ángel Velázquez Iturbide

Departamento de Informática y Estadística, Universidad Rey Juan Carlos,
C/ Tulipán s/n, 28933, Móstoles, Madrid
angel.velazquez@urjc.es

Resumen. En dos informes anteriores se ha presentado una evaluación preliminar de un nuevo método instruccional para mejorar la comprensión de la recursividad, basado en enseñar la eliminación de la recursividad lineal. Presentamos aquí un nuevo análisis de la evaluación. Se describe el diseño de la evaluación, incluyendo la población y la asignatura, el protocolo seguido y las tareas realizadas. Los resultados presentados incluyen los modelos mentales identificados para algoritmos lineales y múltiples, su evolución tras aplicar el método instruccional citado, un análisis de la utilidad de otros datos secundarios y un análisis de la comprensión de la recursividad (y su evolución) por cada alumno. En resumen, los resultados de aprendizaje del método instruccional son positivos para la comprensión de la ejecución de algoritmos recursivos, pero no parece haber variado su capacidad para diseñar algoritmos recursivos.

Palabras clave: Aprendizaje de la programación, recursividad, método instruccional, malentendidos, modelos mentales.

1 Introducción

La recursividad es uno de los conceptos de programación más difíciles de aprender, habiéndose realizado numerosas propuestas para facilitar su aprendizaje [1]. En diversos estudios se ha conjeturado que es distinta la dificultad de comprender la ejecución de algoritmos recursivos y de diseñarlos. En este informe se presenta una su evaluación realizada en la Escuela Superior Politécnica del Chimborazo (ESPOCH) del uso de la eliminación de la recursividad lineal [2] como método instruccional para mejorar la comprensión de la ejecución de los algoritmos recursivos.

La estructura del informe es la siguiente. El apartado 2 describe la evaluación, incluyendo población, asignatura, protocolo aplicado y tareas realizadas. El apartado 3 presenta el método de análisis y los modelos mentales identificados en los alumnos. En el apartado 4 se presentan los resultados obtenidos, tanto en el estudio principal como en una actividad secundaria. En el quinto apartado se comentan los resultados y en el sexto se incluyen nuestras conclusiones. Finalmente, los Apéndices A y B contienen los enunciados de las dos pruebas utilizadas en la evaluación.

2 Descripción de la Evaluación

Se presenta la evaluación estructurada en tres subapartados: población y asignatura, protocolo aplicado y tareas realizadas por los participantes.

2.1 Población y Asignatura

La evaluación se realizó en la asignatura "Programación Avanzada", optativa de 2 créditos de 6º nivel de Ingeniería de Sistemas, con 25 alumnos matriculados.

La asignatura iba a tratar el diseño y análisis de algoritmos. Sin embargo, consultado el coordinador de la titulación (y después algunos profesores) sobre los conocimientos de los alumnos, resultó evidente que la instrucción de los alumnos en recursividad había sido muy baja. Se rediseñó el contenido de la asignatura para aumentar sus conocimientos de recursividad y así poder tratar posteriormente la complejidad de algoritmos y la técnica de divide y vencerás.

Debido a las restricciones de presencia del profesor en la ESPOCH y a diversas incidencias con el calendario, la asignatura se impartió durante cinco semanas con tres sesiones semanales de dos horas cada una. La Tabla 1 presenta las sesiones de clase impartidas. El tratamiento dado a los alumnos duró 3 semanas aproximadamente, resultando relevantes para este estudio sólo las sesiones 1-6 y 9. Se ha marcado en negrita las actividades relevantes de evaluación de los alumnos.

Tabla 1. Sesiones impartidas en la asignatura

Sesión	Contenidos
1.	Presentación de la asignatura Presentación informal de la recursividad
2.	Repaso de diseño y ejecución de algoritmos recursivos
3.	**Pretest** Motivación de la eliminación de la recursividad lineal
4.	Eliminación de la recursividad lineal
5.	Eliminación de la recursividad lineal
6.	**Práctica sobre eliminación de la recursividad**
7.	Conceptos de eficiencia y análisis complejidad de algoritmos iterativos
8.	Análisis de complejidad de algoritmos iterativos y notaciones asintóticas
9.	**Posttest** Análisis de complejidad de algoritmos recursivos
10.	Análisis de complejidad de algoritmos recursivos
11.	Test sobre complejidad
12.	Técnica de divide y vencerás
13.	Técnica de divide y vencerás
14.	Práctica sobre divide y vencerás

En el subapartado siguiente se revisa en detalle el contenido de las sesiones relevantes.

2.2 Preguntas y Diseño de la Investigación

La investigación tiene como objetivo principal evaluar la eficacia educativa del nuevo método instruccional. Sin embargo, el diseño que se describe a continuación nos permite plantear el siguiente conjunto de preguntas:

- ¿Se obtienen los mismos modelos mentales que en estudios anteriores?
- ¿Existen diferencias en los modelos mentales desarrollados para comprender la recursividad lineal y la múltiple?
- ¿Los alumnos comprenden mejor el rastro de la recursividad como resultado de utilizar el nuevo método instruccional?
- ¿Los alumnos diseñan algoritmos recursivos mejor como resultado de utilizar el nuevo método instruccional?

El diseño adoptado ha sido de medidas repetidas, ya que no había otra posibilidad. La medida se tomó mediante un cuestionario formado por cuatro problemas, que se usó como pretest y como posttest. Para evitar un efecto de repetición, tras el pretest, los alumnos fueron atendidos en tutoría sobre cualquier duda que tuvieran salvo dudas sobre el cuestionario: se argumentó que se les atendería en sus dudas sobre el cuestionario tras conocer sus resultados en la práctica sobre eliminación de la recursividad. Además, entre la primera realización del cuestionario y la segunda se dejó transcurrir el máximo tiempo admisible para que no se distorsionara el ritmo de la asignatura, que fueron dos semanas.

2.3 Tratamiento

La asignatura comenzó con dos sesiones de repaso de la recursividad. En la primera sesión, se presentó la asignatura y se realizó una introducción informal a la recursividad con ejemplos visuales, de texto [3] y de dramatización [4]. Se tuvieron en cuenta los problemas de algunas metáforas a la hora de seleccionarlas [5].

En la segunda sesión se recordaron los elementos de los algoritmos recursivos, ilustrado con ejemplos numéricos y sobre vectores, así como las distintas clases de recursividad. Se propuso a los alumnos que diseñaran algunas variantes de los ejemplos presentados. Por último, se mostraron algunas formas de analizar y representar los procesos recursivos: pila de control, rastros (o “trazas”), reescritura y árbol de recursión, con ayuda de los sistemas BlueJ, WinHIPE y SRec. Los sistemas BlueJ y SRec estuvieron disponibles en el campus virtual durante todo el periodo de impartición de la asignatura.

En la tercera sesión se entregó en papel un cuestionario de conocimientos a los alumnos. Este cuestionario sería parte de las pruebas de evaluación de la asignatura. Aunque no tuvieron límite de tiempo, los alumnos lo contestaron en hora y media. La media hora restante se dedicó a motivar la eliminación de la recursividad.

En los días siguientes se informó a los alumnos de la calificación obtenida en el cuestionario, pero se retrasó explicarles los errores cometidos.

Las sesiones cuarta y quinta se dedicaron al método instruccional, en el que se presentó un método “general” de eliminación de la recursividad lineal y cuatro casos particulares, más sencillos [2].

La sesión sexta fue una sesión de laboratorio en la que los alumnos debía iniciar la realización de una práctica sobre eliminación de la recursividad lineal. Los alumnos disponían del resto de la semana (al ser martes, el resto del día y cinco días más) para realizar la práctica y entregar un informe. Mientras tanto se impartieron dos sesiones del tema siguiente.

Al realizar la sesión novena, ya había acabado el plazo de entrega de la práctica. Se volvió a entregar a los alumnos el mismo cuestionario.

En los días siguientes y una vez corregido, se les informó de sus nuevas calificaciones y a los alumnos interesados ya se les pudo explicar en tutorías los errores que habían cometido.

2.4 Tareas

El cuestionario utilizado como pre- y post-test constaba de cuatro preguntas:

1. Traducir una definición recursiva a Java (Q9 de [6]).
2. Detallar la ejecución de una función recursiva lineal (suma de naturales).
3. Detallar la ejecución de una función recursiva múltiple (formas de subir una escalera, tarea 2 de [7]).
4. Diseñar un algoritmo iterativo y otro recursivo para un mismo problema (multiplicar dos naturales, tarea 1 de [7]).

Denotaremos las cuatro preguntas como P1, P2, P3 y P4. La pregunta P1 se usó para comprobar los conocimientos de los alumnos de Java imperativo. A su vez, P2 y P3 eran los elementos principales para nuestra evaluación, en las que se pedía dar un rastro detallado (*trace*, “traza” en español de España, “prueba de escritorio” en español de Ecuador) de la ejecución del algoritmo correspondiente. Por último, P4 se utilizó de forma complementaria, para ver si el método instruccional también afectaba a la capacidad de los alumnos para diseñar algoritmos recursivos; no se pedía que presentaran ningún rastro. El cuestionario se encuentra en el Apéndice A; puede verse que la operación de combinación en las definiciones recursivas P2 y P3 es una suma.

3 Modelos Mentales Identificados

En el apartado primero se resume el método de análisis utilizado y los modelos mentales identificados en este estudio como relevantes.

3.1 Método de Análisis

Primero, se corrigieron los cuestionarios del pretest para entregar las calificaciones a los alumnos. Esto sirvió de análisis preliminar ya que permitió tener una idea aproximada de las respuestas más frecuentes o más llamativas de los alumnos.

Una vez corregidos los tests, se analizaron en detalle las respuestas a las preguntas P2 y P3. Se propusieron como categorías iniciales para nuestro análisis varias categorías extraídas de otros estudios anteriores: modelo sólido [6, 8], modelo frágil

[8], modelo iterativo [6] y “otros”. También se utilizó un modelo no encontrado antes en la literatura, “definición recursiva”. A partir de estos modelos, dos colaboradores (véase la sección de agradecimientos) se repartieron las soluciones del pretest y del posttest y las clasificaron por separado. Celebraron tres reuniones hasta llegar a un consenso en su catalogación.

A continuación, el autor revisó la catalogación. Como resultado, se vio la conveniencia de introducir el modelo de avance [6, 8] y resolver algunas cuestiones, que presentamos más adelante. Posteriormente se realizaron dos rondas más de análisis, con frecuentes revisiones de casos particulares. Los resultados de la primera de estas rondas se ha presentado en otro informe técnico [11], mientras que los resultados de la segunda se presentan en este informe técnico.

3.2 Rastros, Evaluaciones y Soluciones

Götschi *et al.* [6] clasifican las respuestas de los alumnos en: rastros, evaluaciones y soluciones. Nosotros también encontramos estas tres clases de respuestas. Tratamos en este apartado las evaluaciones y soluciones, dejando los rastros para la subsección siguiente.

Una *evaluación* describe cómo se calcula una solución pero sin detallar la forma de calcularla. Encontramos *evaluaciones inductivas* en las que el alumno explica la ejecución del algoritmo utilizando su definición, pero sin desplegar la llamada recursiva (véase Fig. 1, A21 en P2). Este modelo recoge la explicación dada por los profesores sobre cómo diseñar una función recursiva, así como comprender su ejecución. Sin embargo, no debería usarse cuando se pide un rastro detallado. Encontramos 3 evaluaciones inductivas en el pre-test y 1 en el post-test de P2.

No tenemos información suficiente sobre la comprensión por el alumno de la ejecución de la recursión como para considerarlo un modelo viable. Por tanto, es preferible excluir estas respuestas de nuestro análisis.

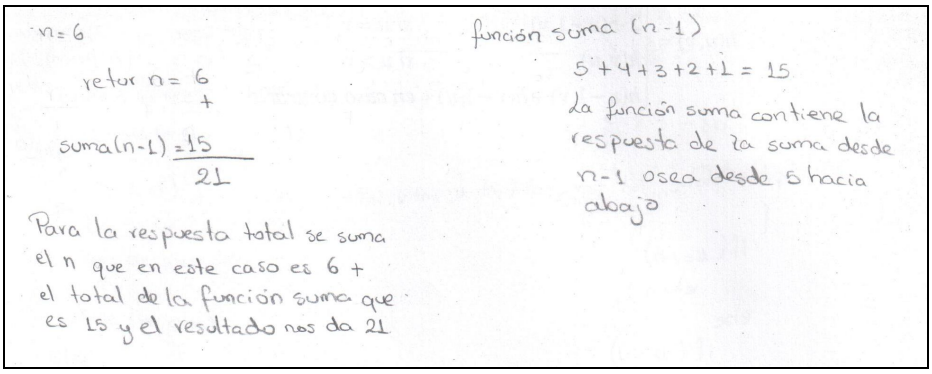
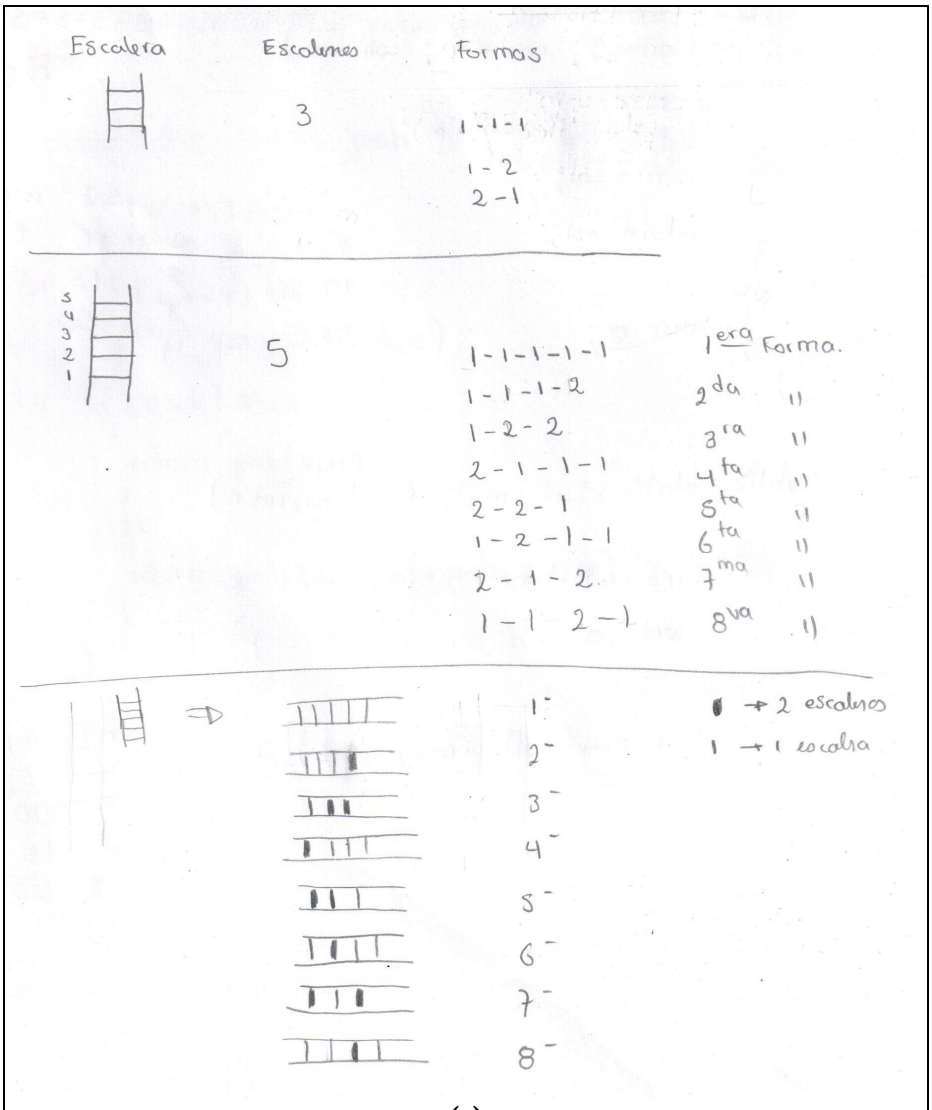


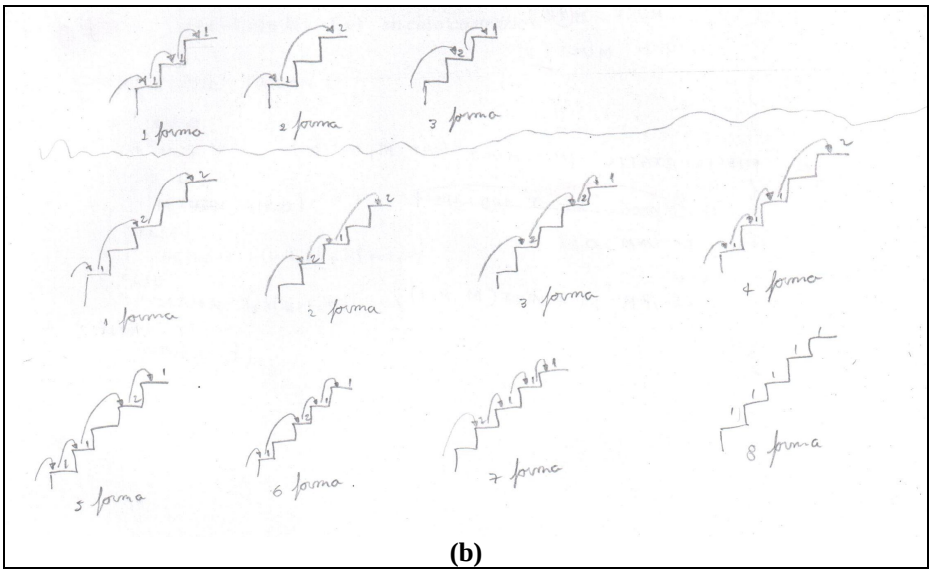
Fig. 1. Ejemplo de solución correspondiente al modelo de definición recursiva.

Por otro lado, bastantes alumnos incluyeron la *solución* a algún ejemplo de los problemas, aunque no se pedía en el enunciado. Estos alumnos dieron una solución

encontrada *ad hoc*, sin relación con ningún algoritmo. La Figura 2 contiene dos soluciones de P3, pertenecientes a A1 y A11 en el pretest.



(a)



(b)

Fig. 2. Dos ejemplos de soluciones *ad hoc* de P3.

La Tabla 2 contiene las soluciones descritas, junto con una indicación de si acompañaba un rastro, identificando su modelo mental (se verán todos en la sección 4.1).

Tabla 2. Alumnos que entregaron soluciones

Pregunta	Alumno	Solución	Modelo mental
P2-Pre	A16	Correcta	Ninguno
P3-Pre	A1	Correcta	Ninguno
	A3	Correcta	Ninguno
	A7	Correcta	Otros
	A11	Correcta	Ninguno
	A12	Correcta	Otros
	A15	Correcta	Frágil
	A16	Incorrecta	Ninguno
	A18	Incorrecta	Ninguno
	A22	Correcta	Ninguno
	A23	Correcta	Ninguno
P3-Post	A3	Correcta	Otros
	A10	Incorrecta	Avance
	A11	Correcta	Ninguno

En total se encontraron 14 soluciones: 1 solución para P2 en el pretest (A16), 10 soluciones para P3 en el pretest (A1, A3, A7, A11, A12, A15, A16, A18, A22, A23) y 3 soluciones para P3 en el posttest (A3, A10, A11).

Las soluciones dadas fueron correctas en 11 ocasiones y sólo fueron incompletas o incorrectas en 3. Asimismo, 9 veces fue la única respuesta de los alumnos, 3 veces se dio junto con soluciones incompletas, y sólo dos veces se utilizó con modelos relativamente viables. Por tanto, llegamos a dos conclusiones:

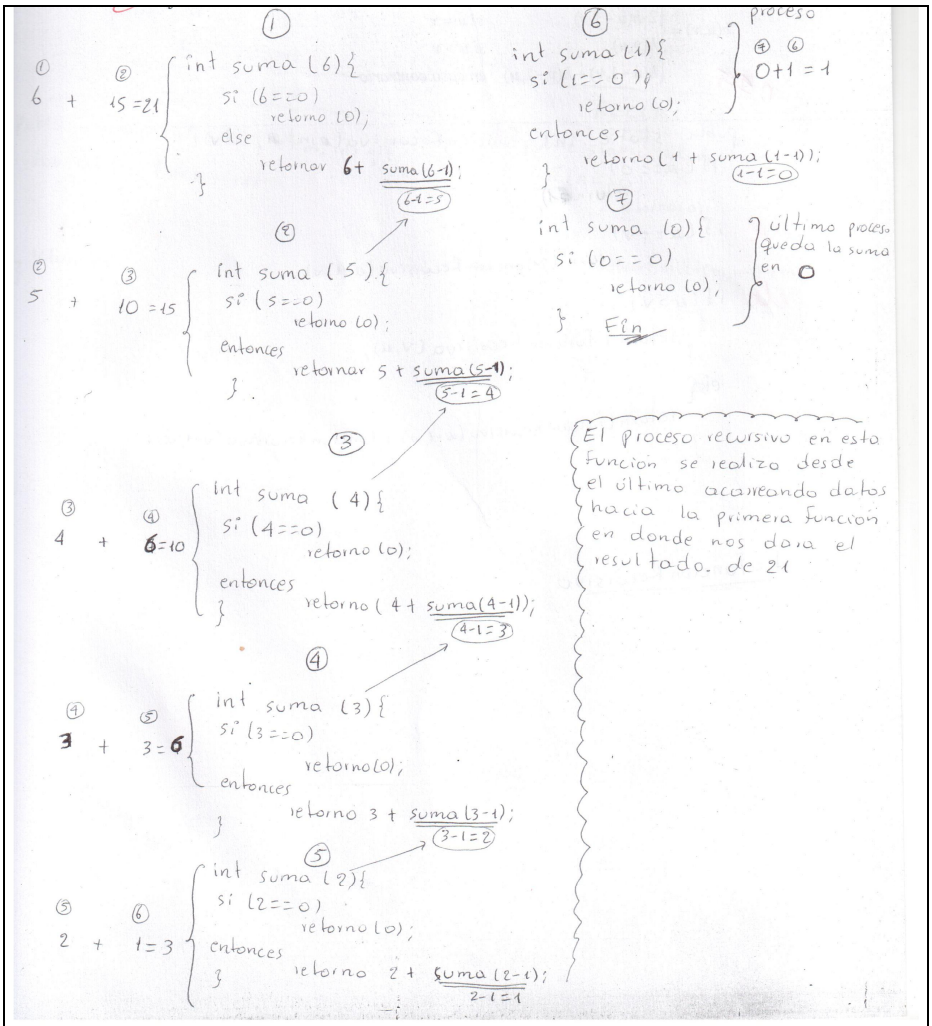
- Las soluciones son un indicador de que el problema se ha entendido.
- En general, los alumnos dieron esta respuesta ante su incapacidad de contestar a lo que realmente se les pedía.

Por tanto, el modelo mental del alumno queda más claro si suprimimos estas soluciones, dejando en su lugar el modelo mental correspondiente al resto de la solución o “ninguno”, según sea el caso.

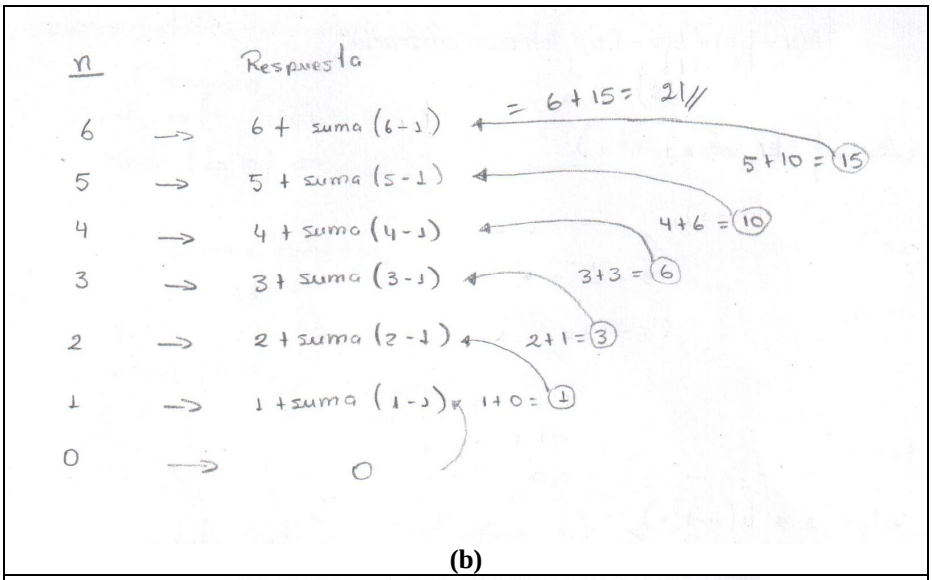
3.2 Modelos Mentales

Veamos los modelos mentales identificados, ilustrados ilustramos con las respuestas dadas por algunos alumnos en el pretest.

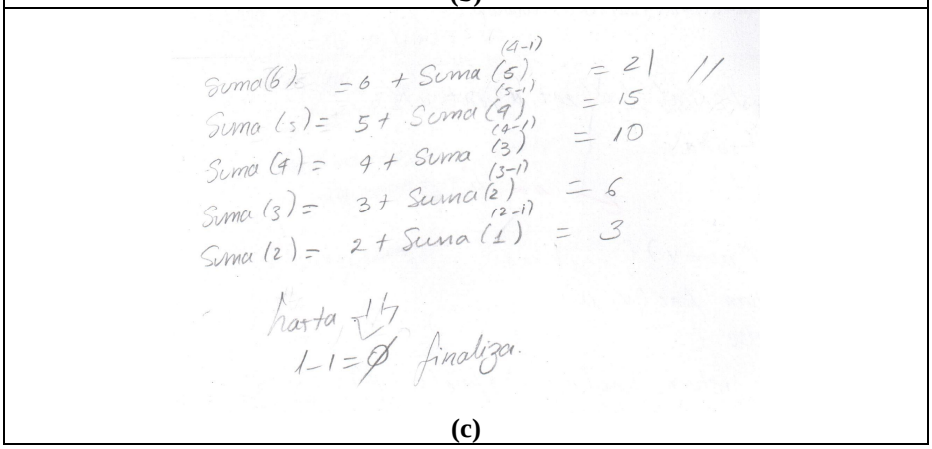
- Modelo sólido [8]. Coincide con el modelo de copias [6]. A veces se hace explícito el proceso de vuelta (Fig. 3.a, A4 en P2); en otros casos queda implícito mediante el uso de flechas (Fig. 3.b, A2 en P2), de paréntesis o de resultados parciales (Fig 3.c, A13 en P2). Para la pregunta P3, a veces se utilizan árboles de recursión con diversos formatos (Fig. 3.d, A10, y 3.e, A14).


1
① ② 6 + 15 = 21
int suma (6){
si (6==0)
retorno (0);
else
retornar 6+ suma(6-1);
6-1=5
2
② ③ 5 + 10 = 15
int suma (5){
si (5==0)
retorno (0);
entonces
retornar 5 + suma(5-1);
5-1=4
3
③ ④ 4 + 6 = 10
int suma (4){
si (4==0)
retorno (0);
entonces
retorno (4 + suma(4-1));
4-1=3
4
④ ⑤ 3 + 3 = 6
int suma (3){
si (3==0)
retorno(0);
entonces
retorno 3 + suma(3-1);
3-1=2
5
⑤ ⑥ 2 + 1 = 3
int suma (2){
si (2==0)
retorno (0);
entonces
retorno 2 + suma(2-1);
2-1=1
6
int suma (1){
si (1==0){
retorno (0);
entonces
retorno (1 + suma (1-1));
1-1=0
proceso
④ ⑥
0+1 = 1
7
int suma (0){
si (0==0)
retorno (0);
Fin
último proceso queda la suma en 0
El proceso recursivo en esta función se realizo desde el último acarreando datos hacia la primera función en donde nos daria el resultado de 21

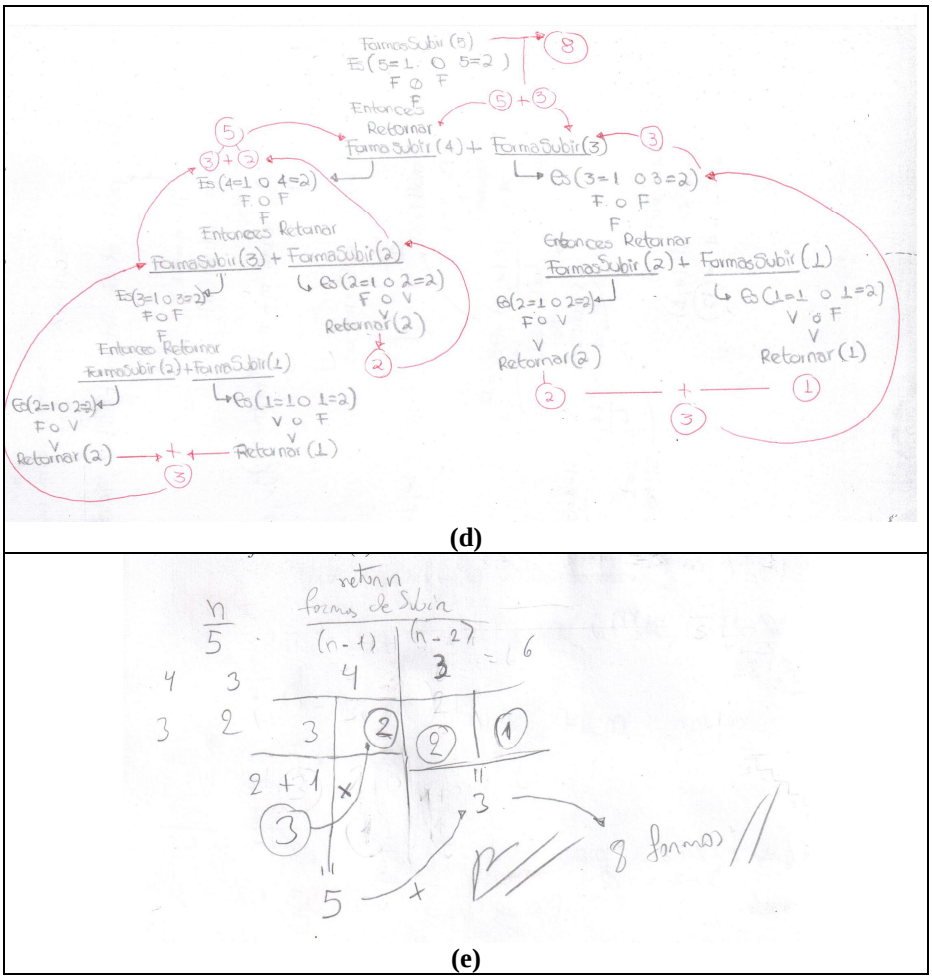
(a)


n
Respuesta
6 → 6 + suma (6-1)
5 → 5 + suma (5-1)
4 → 4 + suma (4-1)
3 → 3 + suma (3-1)
2 → 2 + suma (2-1)
1 → 1 + suma (1-1)
0 → 0
= 6 + 15 = 21
5 + 10 = 15
4 + 6 = 10
3 + 3 = 6
2 + 1 = 3
1 + 0 = 1

(b)


Suma(6) = 6 + Suma (5) = 21 //
Suma (5) = 5 + Suma (4) = 15
Suma (4) = 4 + Suma (3) = 10
Suma (3) = 3 + Suma (2) = 6
Suma (2) = 2 + Suma (1) = 3
hasta 1
1-1 = ∅ finaliza.

(c)



(d)

(e)

Fig. 3. Ejemplos de soluciones correspondientes al modelo sólido.

- Modelo frágil [8]. Es una variante del anterior. Consideramos un modelo frágil cuando la respuesta es correcta pero contiene falta de información (véase Fig. 4.a, A5 en P2), contiene pequeños errores o utiliza una notación inadecuada (Fig. 4.b A8, 4.c A25 en P2), que muestra lo frágil del conocimiento del alumno. En el caso de recursividad múltiple, hay respuestas donde no queda claro si se realizan independientemente las llamadas recursivas redundantes o si éstas se realizan una sola vez (Fig. 4.d, A2 en P3).

n	return Else	
6	6 +	⟹ 21
5	5 +	
4	4 +	
3	3 +	
2	2 +	
1	1 +	
0	0 +	

(a)

suma (6) = 6 + 5+4+3+2+1+0 = 21

suma (6) = 6 + suma(5) = 21

suma (6) = 6 + [5 + 4+3+2+1+0] = 21

Suma (6) = 6 + [5 + suma(4)] = 21

suma (6) = 6 + [5 + (4 + 3+2+1+0)] = 21

Suma (6) = 6 + [5 + (4 + suma (3))] = 21

Suma (6) = 6 + [5 + (4 + [3 + 2+1+0])] = 21

suma (6) = 6 + [5 + (4 + [3 + suma (2)])] = 21

suma (6) = 6 + [5 + (4 + [3 + (2+1+0)])] = 21

suma (6) = 6 + [5 + (4 + [3 + (2 + suma(1))])] = 21

suma (6) = 6 + [5 + (4 + [3 + (2 + (1+0))])] = 21

Suma (6) = 6 + [5 + (4 + [3 + (2 + (1 + sum(0)))])] = 21

suma (6) = 6 + [5 + (4 + [3 + (2 + (1 + (0)))])] = 21

caso general : suma(n) = n + suma (n-1);

caso base : n == 0 ; 0

(b)

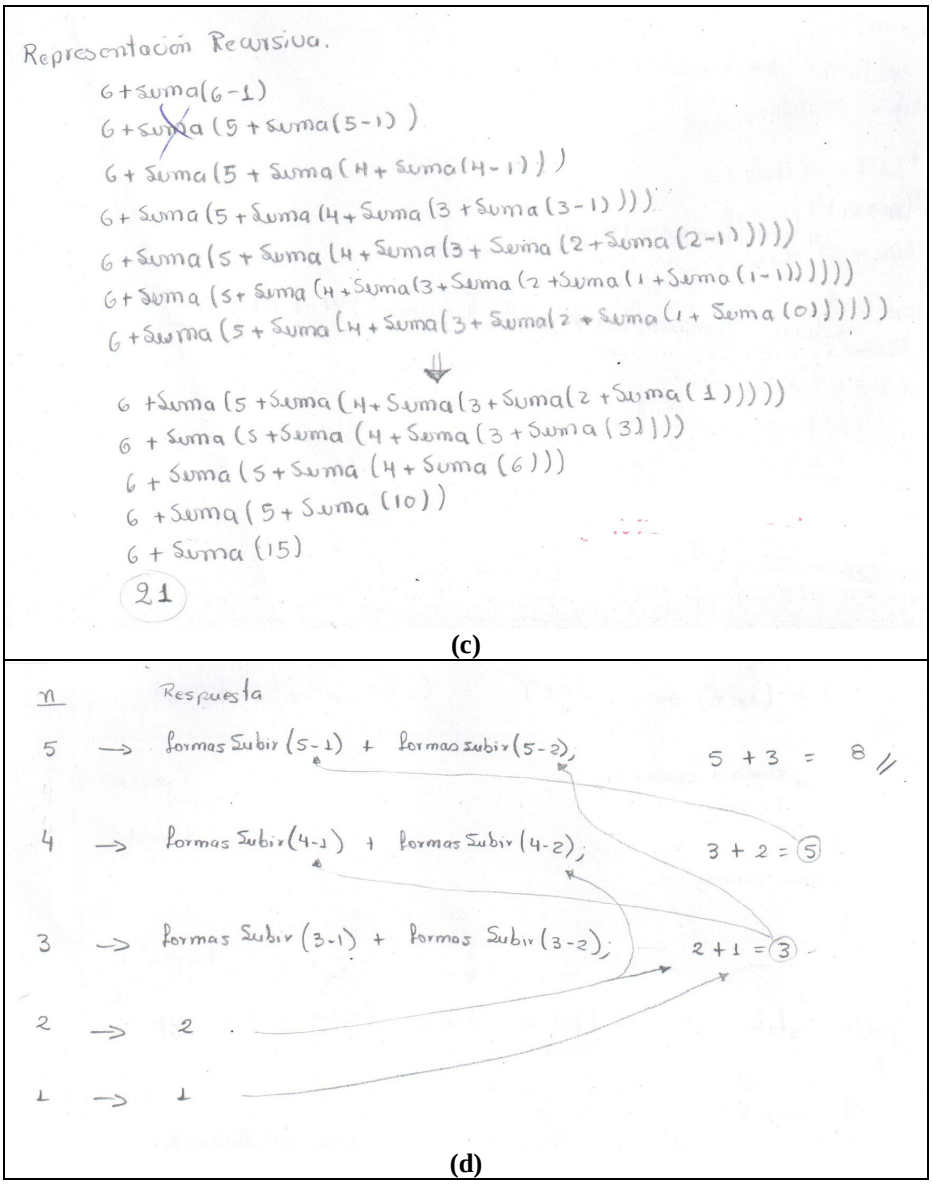


Fig. 4. Ejemplo de soluciones correspondientes al modelo frágil.

- Modelo de avance [8]. Corresponde al modelo activo [6]. Se manifiesta en que no se conoce el orden del cálculo de las sumas en el proceso de vuelta (véase Fig. 5, A7 y A11 en P2, A17 en P3).

n	n + suma (n-1)
6	6 + 5
5	5 + 4
4	4 + 3
3	3 + 2
2	2 + 1
1	1 + ∅
∅	0

1.- Ingresa el numero de elementos como parametro.
2.- Si no cumplir con la condición del If continua el proceso.
3.- En la condición del else se realizara la suma en este caso la sumatoria se llevara a cabo con los valores de la columna n como se aprecia en el cuadro de arriba.
4.- Una vez que el valor de n llega a ∅ terminara el ejercicio devolviendonos su respectivo resultado.

(a)

EJEMPLO

3 + SUMA(2)

3 + 2 + SUMA(1)

3 + 2 + 1

6 + SUMA(5)

6 + 5 + SUMA(4)

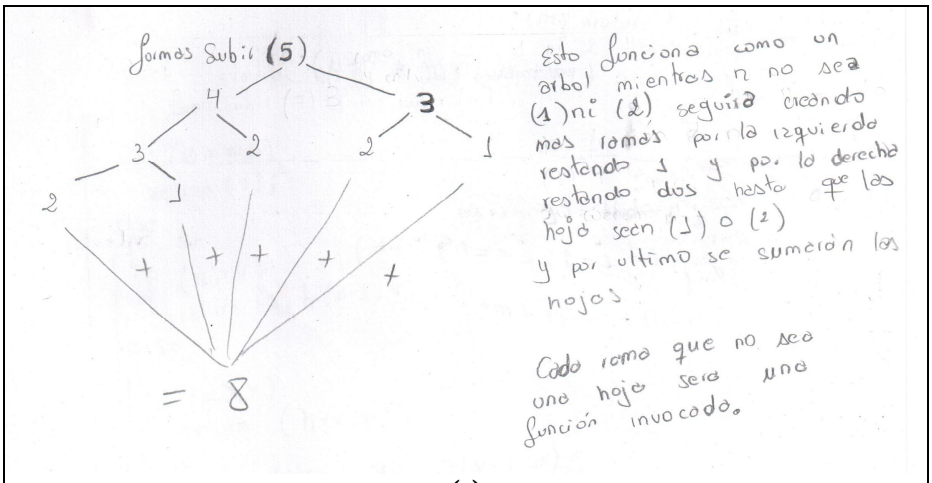
6 + 5 + 4 + SUMA(3)

6 + 5 + 4 + 3 + SUMA(2)

6 + 5 + 4 + 3 + 2 + SUMA(1)

6 + 5 + 4 + 3 + 2 + 1 + 0 = 21

(b)



(c)

Fig. 5. Ejemplos de soluciones correspondientes al modelo de avance.

- Modelo iterativo [6]. Se hace explícito el cálculo (suma) de los resultados parciales según avanza el proceso recursivo (véase Fig. 6, A1 en P2).

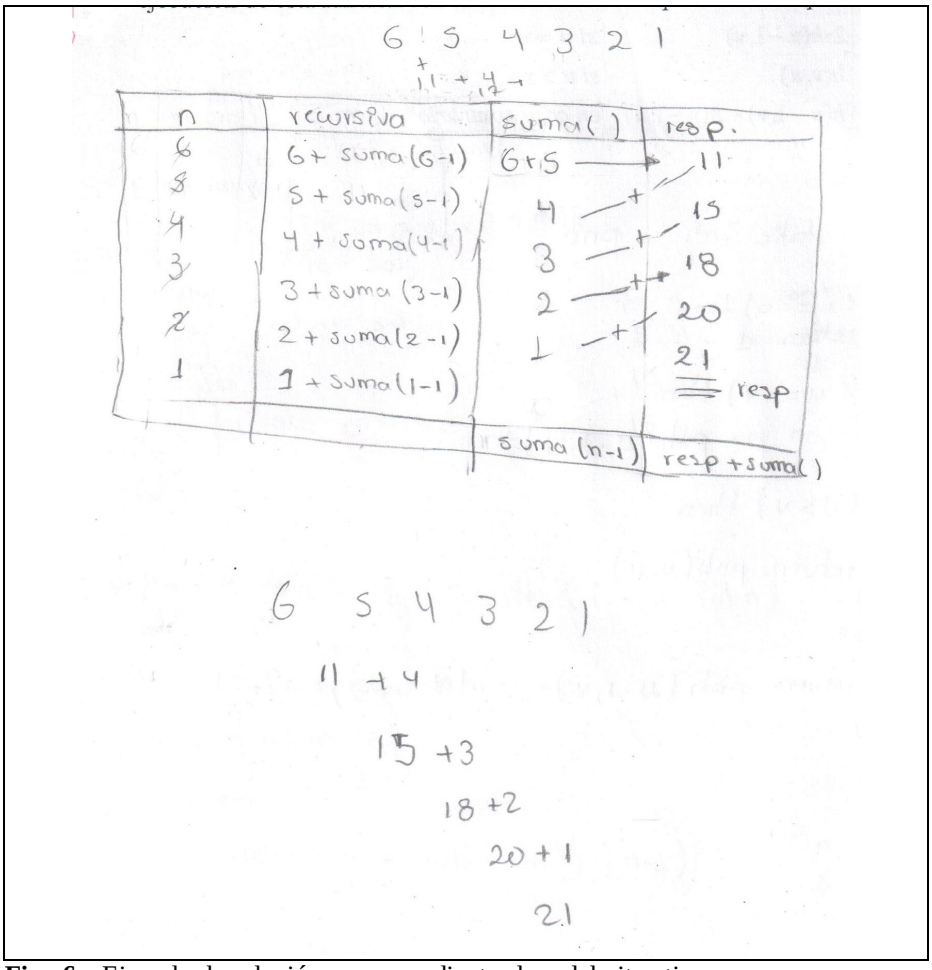


Fig. 6. Ejemplo de solución correspondiente al modelo iterativo.

- Otros modelos. Se recogen otros modelos mentales distintos y minoritarios. Normalmente se presentan incompletos y con partes difíciles de comprender.
- Categoría “ninguno”. No es un modelo mental propiamente dicho, sino que permite catalogar las respuestas dejadas en blanco.

En resumen, los modelos sólido y frágil son viables. Los modelos iterativo y “otros”, así como la categoría “ninguno” corresponden a modelos inviables. Finalmente, las categorías de avance y de definición recursiva recogen esas respuestas sobre las que no podemos juzgar su viabilidad sin más información, por tanto, ambas categorías recogen modelos dudosos para nuestro análisis.

En la sección siguiente, utilizaremos el siguiente orden parcial de la viabilidad de los modelos para analizar el progreso de los alumnos: “ninguno” y “otros” < “iterativo” < “avance” < “frágil” < “sólido”. La justificación de este orden parcial es

clara. Obviamente, las categorías “ninguno” y “otros” no contienen apenas ninguna información significativa sobre el rastro. El modelo iterativo es un modelo inviable, pero constituye un paso adelante en la construcción de este conocimiento. Las categorías de avance y definición recursiva recogen respuestas correctas pero incompletas. Finalmente, los modelos sólido y frágil son viables, donde el modelo frágil muestra algunos problemas menores. En cuanto al modelo de definición recursiva, no lo hemos tenido en cuenta porque no expresa un auténtico rastro.

4 Resultados

Presentamos en primer lugar los rastros de las preguntas P2 y P3 del pretest y del posttest. Posteriormente analizamos los datos secundarios, es decir, los rastros incluidos en el diseño de una función recursiva para P4, tanto en el pretest como en el posttest. Por último, analizamos la comprensión de las distintas formas de recursividad que presentan los alumnos.

4.1 Resultados de P2 y P3

Se recogieron 22 cuestionarios del pretest y 25 del posttest. En la Tabla 2 se recogen los resultados del análisis de todos los cuestionarios respondidos. Para el análisis detallado presentado en este apartado sólo se considerarán los 22 cuestionarios correspondientes a alumnos que contestaron ambos tests. Recuérdese que 3 alumnos también presentaron evaluaciones inductivas para P2, por lo que los cuestionarios a considerar se reducen a 19 para P2 y 22 para P3. En la Tabla 2 incluimos las 3 evaluaciones inductivas y los 3 resultados incompletos, pero con tinta gris para distinguirlos.

Tabla 3. Modelos mentales de los alumnos en las preguntas P2 y P3, tanto en el pretest como en el posttest (*N*=25)

Alumno	P2 pretest	P2 posttest	P3 pretest	P3 posttest
A1	Iterativo	Sólido	Ninguno	Sólido
A2	Sólido	Sólido	Frágil	Sólido
A3	Frágil	Avance	Ninguno	Otros
A4	Sólido	Sólido	Otros	Sólido
A5	Frágil	Frágil	Sólido	Avance
A6		Avance		Otros
A7	Avance	Avance	Otros	Otros
A8	Frágil	Sólido	Sólido	Avance
A9		Iterativo		Otros
A10	Avance	Sólido	Sólido	Avance
A11	Avance	Avance	Ninguno	Ninguno
A12	Frágil	Sólido	Otros	Sólido
A13	Frágil	Frágil	Frágil	Avance
A14	Iterativo	Sólido	Sólido	Sólido
A15	Iterativo	Sólido	Frágil	Sólido
A16	Ninguno	Avance	Ninguno	Otros
A17	Frágil	Frágil	Avance	Sólido
A18	Iterativo	Sólido	Ninguno	Sólido
A19		Sólido		Otros
A20	Avance	Avance	Avance	Avance
A21	Eval. inductiva	Eval. inductiva	Ninguno	Otros
A22	Eval. inductiva	Avance	Ninguno	Otros
A23	Iterativo	Avance	Ninguno	Avance
A24	Eval. inductiva	Sólido	Otros	Otros
A25	Frágil	Frágil	Otros	Frágil

La Tabla 3 contiene los resultados de P2, desglosados por modelos mentales, tanto en el pretest como en el posttest.

Puede observarse que hay una mejora en el número de respuestas correspondientes a los modelos viables (de 9 a 13) y, dentro de éstos, en su calidad (un aumento de 2 modelos sólidos a 9, y una disminución de 7 modelos frágiles a 4). También hay un pequeño incremento en el número de modelos de avance (de 4 a 6). Por último, desaparece el modelo iterativo y ningún alumno deja la pregunta sin dar un rastro.

Tabla 4. Número de respuestas de P2 por modelo mental (*N*=22)

Modelo mental	# respuestas en pretest	# respuestas en posttest
Sólido	2	9
Frágil	7	4
Avance	4	6
Iterativo	5	
Ninguno	1	

La Tabla 4 contiene los resultados obtenidos para P3, también desglosados por modelos mentales. Llama la atención que no hay ningún modelo iterativo para este algoritmo con recursividad múltiple.

No aumenta demasiado el número de modelos viables (de 7 a 9), pero sí lo hace el número de modelos sólidos (de 4 a 8). Disminuye mucho el número de alumnos que no responden nada significativo (de 8 a 1), aumenta el número de modelos de avance (de 2 a 6) y se mantienen las respuestas catalogadas en "otros".

Tabla 5. Número de respuestas de P3 por modelo mental (*N*=22)

Modelo mental	# respuestas en pretest	# respuestas en posttest
Sólido	4	8
Frágil	3	1
Avance	2	6
Otros	5	6
Ninguno	8	1

Los resultados obtenidos para la pregunta P3 (recursividad múltiple) son peores que para P2 (recursividad lineal). En P2 se obtienen más modelos viables y menos modelos de "ninguno" u "otros".

Podemos analizar la evolución de cada alumno del pretest al posttest utilizando el orden parcial presentado al final de la sección anterior: "ninguno" y "otros" < "iterativo" < "avance" < "frágil" < "sólido".

Las tendencias aparecen resumidas en la Tabla 5. Mejoran las respuestas de los alumnos a ambas preguntas. Los empeoramientos de A5, A8 y A10 en P3 revelan que algunas respuestas correspondientes al modelo de avance probablemente reflejen malas explicaciones escritas de los alumnos.

Tabla 6. Evolución de los modelos mentales para P2 (*N*=19) y P3 (*N*=22)

Evolución	P2	Alumnos	P3	Alumnos
Mejoran	9	A1, A8, A10, A12, A14, A15, A16, A18, A23	13	A1, A2, A3, A4, A12, A15, A16, A17, A18, A21, A22, A23, A25
Igual	9	A2, A4, A5, A7, A11, A13, A17, A20, A25	5	A7, A11, A14, A20, A24
Empeoran	1	A3	4	A5, A8, A10, A13

4.2 Coherencia o Fragilidad de los Modelos Mentales

La Tabla 6 contiene los modelos obtenidos para los tres ejercicios (P2, P3 y P4), tanto en el pretest como en el posttest. Utilizaremos estos datos para comparar las soluciones presentadas por los alumnos a diferentes problemas. Conviene recordar que P4 pedía el diseño de algoritmos, no su rastro. Los alumnos que han presentado rastros lo han hecho con algún fin clarificador, bien para aclararse ellos mismos, bien para mostrar al profesor que el algoritmo es correcto.

Tabla 6. Modelos mentales de los alumnos en todas las preguntas de ambos tests

Alumno	P2 pretest	P3 pretest	P4 pretest	P2 posttest	P3 posttest	P4 posttest
A1	Iterativo	Ninguno	Iterativo	Sólido	Sólido	Avance
A2	Sólido	Frágil	Sólido	Sólido	Sólido	Avance
A3	Frágil	Ninguno		Avance	Otros	Avance
A4	Sólido	Otros		Sólido	Sólido	
A5	Frágil	Sólido		Frágil	Avance	Frágil
A6				Avance	Otros	
A7	Avance	Otros		Avance	Otros	
A8	Frágil	Sólido		Sólido	Avance	
A9				Iterativo	Otros	
A10	Avance	Sólido	Sólido	Sólido	Avance	Iterativo
A11	Avance	Ninguno		Avance	Ninguno	
A12	Frágil	Otros	Avance	Sólido	Sólido	Sólido
A13	Frágil	Frágil		Frágil	Avance	Iterativo
A14	Iterativo	Sólido	Avance	Sólido	Sólido	Sólido
A15	Iterativo	Frágil	Iterativo	Sólido	Sólido	Sólido
A16	Ninguno	Ninguno		Avance	Otros	
A17	Frágil	Avance		Frágil	Sólido	
A18	Iterativo	Ninguno		Sólido	Sólido	Frágil
A19				Sólido	Otros	Sólido
A20	Avance	Avance		Avance	Avance	
A21	Eval. inductiva	Ninguno		Eval. inductiva	Otros	
A22	Eval. inductiva	Ninguno		Avance	Otros	
A23	Iterativo	Ninguno		Avance	Avance	
A24	Eval. inductiva	Otros		Sólido	Otros	
A25	Frágil	Otros	Frágil	Frágil	Frágil	Avance

P2 y la solución de P4 son algoritmos recursivos lineales, estructuralmente parecidos. Si las comparamos, para aquellos alumnos que han incluido rastros, encontramos las situaciones presentadas en la Tabla 7. Son 19 casos, 7 del pretest y 12 del posttest. En este análisis no tiene importancia si un rastro se presenta en el pretest o en el posttest, por lo que los presentamos unidos. Cada caso se identifica con la notación A*nn* más el sufijo "–Pre" o "–Post" que indica si la respuesta fue elaborada por dicho alumno en el pretest o en el posttest, respectivamente.

Puede observarse que en 10 casos el rastro de P4 repite el modelo de P2, mientras que en 3 presenta un modelo contradictorio con el de P2. Sin embargo, los casos más

interesantes son los 6 en los que interviene algún modelo de avance. Como razonamos a continuación, en 5 de los 6 casos, el modelo de avance puede representar falta de detalle de un modelo viable.

En un caso, todos los rastros son de avance, así que el rastro de P4 no aporta nada nuevo (A3-Post). En 4 casos se presenta un rastro de avance en P4, cuando el alumno presentó un modelo viable en P2. Puede ser que el rastro para P4 se elaboró sin cuidar los detalles porque el alumno simplemente pretendía asegurarse de que el algoritmo funcionaba correctamente. O puede ser que el alumno utilice diferentes modelos mentales según el caso.

Esta fragilidad la muestra A10 de forma aún más acentuada. En el pretest, presentó un modelo de avance para P2 y un modelo sólido para P3 y P4. El rastro de P4 es completo, pero también incluye un resumen sin paréntesis; si no lo diera tan detallado, con dos rastros paralelos, lo habríamos catalogado como de avance. Por tanto, su modelo de avance para P2 parece deberse a una precipitación o descuido, no a un malentendido. En el posttest vuelve a repetirse esta situación contradictoria, pero más grave: viable sólido en P2 e iterativo en P4, cuando el proceso recursivo en ambas funciones es similar. Por tanto, cambia de modelo mental o simplemente lo simplifica en ciertas situaciones.

Tabla 7. Rastro de P4 según rastro de P2 (*N*=19)

Interpretación	**# grupos**	**Grupos**
Reitera modelo viable	8	A2-Pre, A25-Pre, A5-Post, A12-Post, A14-Post, A15-Post, A18-Post, A19-Post
Reitera modelo iterativo	2	A1-Pre, A15-Pre
Permite interpretar P2 como viable	1	A10-Pre
Avance como simplificación de viable	4	A12-Pre, A1-Post, A2-Post, A25-Post
Sin información	1	A3-Post
Contradicción	3	A14-Pre, A10-Post, A13-Post

4.3 Resultados de Diseño en P4

Veamos si hubo algún cambio apreciable en los alumnos en su capacidad de diseñar funciones recursivas. La Tabla 9 muestra los resultados de los alumnos en P4, tanto el pretest como en el posttest, así como la categoría del rastro para aquellos alumnos que los realizaron.

Llaman la atención los alumnos A7, A8 y A22, que han empeorado sustancialmente su solución. Por el contrario, A11, A16 y A21 experimentan avances claros. En muchos otros alumnos, puede observarse una mejoría o empeoramiento, pero en detalles que no reflejan un cambio cualitativo entre ambas soluciones.

Tabla 9. Respuestas de los alumnos a la pregunta P4 y modelo de su rastro, en caso de realizarlo

Alumno	P4 pretest	Rastro P4 pretest	P4 posttest	Rastro P4 posttest
A1	Correcto	Iterativo	Correcto	Avance
A2	Falta caso *n*=0	Sólido	Falta caso *n*=0	Avance
A3	Resultado 1 para *n*=0		Correcto	Avance
A4	Correcto		Falta caso *n*=0	
A5	Falta caso 0 Caso básico sobre *m*, pero recursión sobre *n*		Falta caso *n*=0	Frágil
A6			Correcto	
A7	Correcto (caso *m*=0 redundante)		Mal intercambio de parámetros y caso básico	
A8	Correcto		Falta caso *n*=0 Parámetro acumulador mal usado	
A9			Correcto	
A10	Falta caso *n*=0	Sólido	Correcto	Iterativo
A11	Caso básico "*m* mod 2 == 0 AND *n*=1"		Falta caso *n*=0	
A12	Correcto	Avance	Correcto	Sólido
A13	Correcto (una condición y un caso que sobran)		Falta	Iterativo
A14	Correcto (caso *n*=0 redundante)	Avance	Correcto (casos *n*=0 y *m*=1 redundantes)	Sólido
A15	Caso *m*=1 debe dar 0 Caso *n*=0 redundante	Iterativo	Correcto	Sólido
A16	Incompleto		Caso *n*<1 debe dar 0 Falta *m* en llamada recursiva	
A17	Falta caso *n*=0		Falta caso *n*=0	
A18	Correcto (4 casos básicos)		Correcto	Frágil
A19			Falta caso *n*=0	Sólido
A20	Correcto (caso *m*=0 redundante)		Correcto	
A21	Condición compuesta mal escrita Recursión doble (sobre *m*, *n*)		Caso *m*=0 redundante Falta -1 en llamada recursiva	
A22	Correcto (caso *n*=0 redundante)		Caso básico compuesto Recursión sobre *n*+*n*	
A23	Correcto		Caso *n*=0 debe dar 0	
A24	Falta caso *n*=0		Falta caso *n*=0	
A25	Caso *n*=1 debe ser 0	Frágil	Correcto	Avance

En la Tabla 10 se muestra si hay mejoras entre ambos tests en las respuestas a P4. Aunque en la Tabla 9 hemos recogido todos los resultados, en la Tabla 10 sólo comparamos los resultados para los 19 alumnos que realizaron el pretest y el posttest y que no presentaron evaluaciones inductivas. Hay un número ligeramente mayor de

alumnos que mejoran, pero no parece que sea una diferencia significativa (8 que mejoran frente a 5 iguales o 5 que empeoran).

Para la clasificación de la Tabla 10, hemos seguido los siguientes criterios. No penalizamos que haya un caso básico redundante (suele ser comprobar que el otro parámetro sea igual a cero), aunque sí que haya más casos básicos. También consideramos peor un algoritmo incompleto que uno completo, que haya errores graves de diseño de la recursividad, un algoritmo incomprensible frente a un algoritmo incorrecto pero "encaminado", un algoritmo sin el caso básico 0 o que devuelva un resultado equivocado para un caso básico (aunque es probable que sea un error "tonto") frente al algoritmo correcto. El caso del alumno A13 no lo consideramos porque no incluye algoritmo pero incluye un rastro: ¿podría ser que se le olvidara escribir el algoritmo si lo pasó a limpio?

Tabla 10. Evolución del rendimiento en P4 del pretest al postest (*N*=19)

Cambio	#alumnos	Alumnos
Mejoran	8	A3, A5, A10, A11, A15, A16, A18, A25
Igual	5	A1, A2, A12, A17, A20
Empeoran	5	A4, A7, A8, A14, A23
Indeterminado	1	A13

5 Conclusiones

Hemos realizado una evaluación de un nuevo método instruccional para la comprensión de la recursividad. Resumimos los resultados más relevantes obtenidos:

- Encontramos los modelos mentales mayoritarios de investigaciones previas. Sin embargo, también encontramos otro modelo, de definición recursiva. Aunque refleja la forma de pensar más adecuada sobre el funcionamiento de los algoritmos recursivos, no es adecuado en este contexto porque no incluye detalles suficientes de su ejecución.
- Hay una diferencia clara en los resultados obtenidos para recursividad lineal (P2) y múltiple (P3). Con la recursividad lineal es más corriente encontrar modelos iterativos, mientras que con la recursividad múltiple es más común encontrar respuestas vacías o del modelo mental "otros".
- Para P2, la mejora de los alumnos en el posttest ha sido clara. Todos los alumnos han logrado responder con algún rastro. Además, dos tercios de los alumnos han desarrollado un modelo viable, siendo sólido para la mayoría. Ha desaparecido el modelo iterativo y casi desaparece el modelo de llamada recursiva.
- Para P3, no aumenta demasiado el número de modelos viables, pero sí lo hace en su solidez: aumenta el número de modelos viables sólidos y disminuye el número de modelos viables frágiles. De nuevo, disminuye mucho el número de alumnos que no responden nada significativo, aumentando el número de modelos de avance.

- Encontramos evidencia de que los alumnos pueden utilizar varios modelos mentales. Puede ser que manejen varios modelos mentales, eligiendo en cada situación el que les resulta más adecuado, o que en algunos casos simplifican sus explicaciones.
- Los resultados de P4 no muestran una mejora significativa en sus habilidades para diseñar algoritmos recursivos.

Agradecimientos. Este trabajo se ha financiado con el proyecto "Becas Prometeo" de la SENESCYT y con el proyecto TIN2011-29542-C02-01 del Ministerio de Economía y Competitividad de España. Quiero agradecer a Jorge Huilca y Lorena Aguirre su ayuda en las primeras rondas de clasificación de los tests en modelos mentales, recogido en un informe técnico anterior [10].

Referencias

1. Rinderknecht, C.: A survey on teaching and learning recursive programming. Informatics in Education 13, 1 (2014) 87-119
2. Scholl, P.C.: Algorítmica y representación de datos, recursividad y árboles. Masson (1986)
3. Levy, D., Lapidot, T.: Recursively speaking: Analyzing students' discourse of recursive phenomena. En: Proceedings of the 31st SIGCSE Technical Symposium on Computer Science Education (2000) 315–319
4. Ben-Ari, M.: Recursion: From drama to program. Computer Science Education 11, 3 (1997) 9–12
5. Forišek, M., Steinová, M.: Metaphors and analogies for teaching algorithms. En: Proceedings of the 43rd SIGCSE Technical Symposium on Computer Science Education (2012) 15-20
6. Mirolo, C.: Learning (through) recursion: A multidimensional analysis of the competences achieved by CS1 students. En: Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education (2010) 160–164
7. Ginat, D., Shifroni, E.: Teaching recursion in a procedural environment – How much should we emphasize the computing model? En: Proceedings of the 30th SIGCSE Technical Symposium on Computer Science Education (1999) 127–131
8. Götschi, T., Sanders, I., Galpin, V.: Mental models of recursion. En: Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (2003) 346-350
9. Velázquez-Iturbide, J.Á., Menéndez-Verdecia, J., Santillán-Castillo, J., Veloz-Remache, G., Villa-Villa, E.: Replication in Ecuador of a multinational study on performance of programming students. En: 2014 Frontiers in Education Conference - Proceedings (2014) 2.605-2.612
10. Velázquez-Iturbide, J.Á., Aguirre, G.L., Huilca, J.: Evaluación preliminar de un nuevo método instruccional para el aprendizaje de la recursividad. Serie de Informes Técnicos DLSI1-URJC 2014-08. Departamento de Lenguajes y Sistemas Informáticos I, Universidad Rey Juan Carlos
11. Velázquez-Iturbide, J.Á.: Evaluación definitiva de un nuevo método instruccional para el aprendizaje de la recursividad. Serie de Informes Técnicos DLSI1-URJC 2014-10. Departamento de Lenguajes y Sistemas Informáticos I, Universidad Rey Juan Carlos

Apéndice: Enunciado del Cuestionario

Ingeniería de Sistemas
Asignatura *Programación Avanzada*
Curso 2013/2014
Test

1. Codifica en Java la siguiente función recursiva *h*, definida sobre todos los pares de naturales:

$$h(u,v) = \begin{cases} 1 & si\ u = 0 \\ 2 \cdot h(u-1,v) & si\ u = v \\ h(v,u) & si\ u > v \\ h(u-1,v) + h(v-1,u) & en\ caso\ contrario \end{cases}$$

2. La siguiente función recursiva calcula la suma de todos números naturales hasta *n*

```
public static int suma (int n) {
   if (n==0)
      return 0;
   else
      return n + suma(n-1);
}
```

Por ejemplo, *suma*(3) = 6 porque es el resultado de sumar, 1, 2 y 3.

Ilustra la ejecución de esta función con todo detalle, de forma que se muestre cómo se calcula el resultado *suma*(6) = 21.

3. La siguiente función recursiva calcula de cuántas formas es posible subir una escalera de *n* peldaños (*n*>0), suponiendo que en cada paso se puede subir uno o dos peldaños.

```
public static int formasSubir (int n)
   if ((n==1) || (n==2))
      return n;
   else
      return formasSubir(n-1) + formasSubir(n-2);
}
```

Por ejemplo, *formasSubir*(3) = 3 porque una escalera de 3 escalones puede subirse de tres maneras: 1-1-1, 1-2 y 2-1.

Ilustra la ejecución de esta función con todo detalle, de forma que se muestre cómo se calcula el resultado *formasSubir*(5) = 8.

4. Diseña dos funciones en Java, una iterativa y otra recursiva, ambas con la siguiente cabecera:

```
public static int mult (int m, int n)
```

Ambas funciones deben hallar el producto de dos números naturales, es decir, deben calcular *m***n*. Además, deben calcular el producto mediante sumas sucesivas.