

UNIVERSIDAD NACIONAL DE EDUCACION A DISTANCIA
CENTRO ASOCIADO DE BARBASTRO

I JORNADAS NACIONALES SOBRE INFORMATICA EN LA ENSEÑANZA

VALORACION PEDAGOGICA DE LAS OPCIONES LINGÜISTICAS
DEL LENGUAJE EXPERIMENTAL UBL EN LA ENSEÑANZA DE
LA PROGRAMACION

E. Tubau, J. M. Sopena, J. M. Blasco, N. Sebastián, G. Alonso

Barbastro, 11-14 de julio de 1984

VALORACION PEDAGOGICA DE LAS OPCIONES LINGÜISTICAS DEL LENGUAJE EXPERIMENTAL UBL EN LA ENSEÑANZA DE LA PROGRAMACION

*E. TUBAU**, *J.M. SOPENA**, *J.M. BLASCO***,
*N. SEBASTIAN**, *G. ALONSO***

Conseguir que los sistemas informáticos sean fáciles de utilizar y entender (*user friendly*), es uno de los objetivos de importancia creciente tanto para los diseñadores de estos sistemas (*hardware* y *software*) como para los diseñadores y enseñantes de lenguajes de programación.

Cualquier ordenador es parte de un sistema persona-máquina. En esta situación interactiva entre la persona y el ordenador, podemos distinguir diferentes factores que en gran parte determinan su utilización y comprensión. Algunos de estos factores son de tipo físico (*physical interfaces*) y con el fin de mejorar este tipo de "interface", numerosos trabajos realizados por ingenieros en factores humanos, han incidido en el diseño de teclados, pantallas, etc. Sin embargo, los factores realmente importantes en los sistemas persona-ordenador son los que se refieren a "interfaces de tipo cognitivo", los cuales son más difíciles de estudiar ya que implican problemas psicológicos mucho más profundos.

Desde esta perspectiva, la Psicología del software intenta aplicar los conocimientos de la Psicología Cognitiva y las técnicas de la Psicología Experimental para resolver los problemas de los ordenadores y de la ciencia informática en general. Se trata de una nueva disciplina que estudia las habilidades y capacidades humanas en la utilización de ordenadores y sistemas informáticos. Su objetivo principal es facilitar la utilización de ordenadores (simplificando el aprendizaje, aumentando la capacidad de realización y disminuyendo la frecuencia de errores), incidiendo por un lado, en el diseño de lenguajes de programación y lenguajes de control de sistemas operativos y por otro lado, en el diseño de CAI, ordenadores personales, editores... Con ello se intenta adecuar lo más posible los lenguajes de programación y sistemas informáticos a los tipos de estrategias cognitivas propias de las personas.

Queremos destacar especialmente, como señala Shneiderman (1980), la valiosa contribución que la Psicología del Software puede hacer en el diseño y en la enseñanza de lenguajes de programación, no sólo verificando la utilidad de nuevas ideas, sino que aportando una comprensión del comportamiento del programador, podrá

* Departamento de Psicología Experimental. Universidad de Barcelona.

** Centro de Cálculo. Universidad de Barcelona.

ofrecer conocimientos que sugieran nuevas ideas en el diseño de lenguajes de programación.

Siguiendo esta línea de investigación, nuestra experiencia tuvo por objetivo valorar el aprendizaje del lenguaje UBL (lenguaje de la Universidad de Barcelona) en dos de sus versiones lingüísticas (catalán e inglés). Se trataba de ver si un lenguaje de programación con las palabras reservadas e identificadores predefinidos expresados en la lengua habitual del alumno, es más fácil de aprender y utilizar que el mismo lenguaje expresado en una lengua extraña. Según Dijkstra (1982), existe una estrecha relación entre el dominio de la propia lengua materna y la habilidad en programar: "... la base más esencial del programador competente es el tener un dominio excepcional de la propia lengua materna".

Como punto de partida para esta valoración, nos hemos basado en un modelo que intenta explicar los distintos procesos psicológicos que intervienen en la comprensión y producción de programas (adaptado del modelo de comprensión del lenguaje natural explicado en Arnau, Sebastián y Sopena, 1982 y Sopena y Sebastián, 1981). Como puede observarse en la figura 1, en primer lugar hemos diferenciado un nivel léxico donde se identifican las palabras del programa. En segundo lugar, estaría el nivel sintáctico donde estas palabras se estructuran en forma de sentencias. Paralelamente, existe un nivel semántico donde se atribuye significado a las palabras

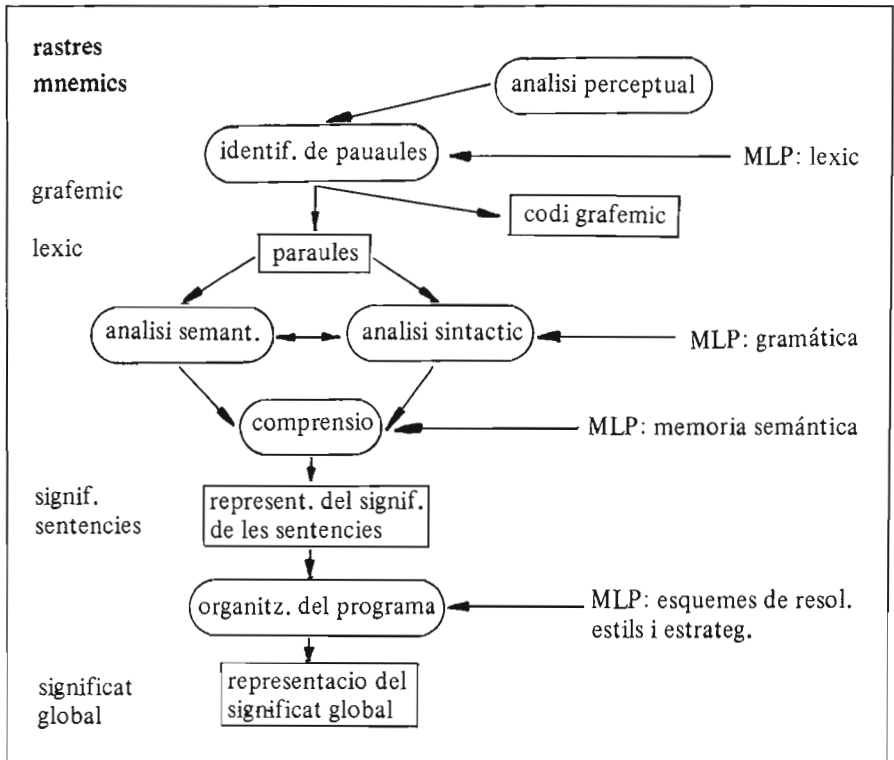


Figura 1: Diagrama dels processos psicològics implicats en la comprensió de llenguatges de programació. (Veure les explicacions del text).

y sentencias y finalmente, el nivel lógico (esquemas de resolución de problemas, estilos y estrategias de programación) da coherencia y significado global al programa entero. En el análisis de la producción y comprensión de programas realizados por los alumnos del curso, hemos tenido en cuenta los distintos errores que se producen en los diferentes niveles de análisis (errores léxicos, errores sintácticos, errores semánticos y errores lógicos).

Nuestra experiencia se ha centrado principalmente en analizar las connotaciones que puede tener la utilización de un léxico perteneciente al lenguaje habitual del programador frente a un léxico totalmente desconocido por el programador, en la rapidez del aprendizaje, en el número de errores cometidos en todos los niveles y en la capacidad de realización (habilidad en producir programas).

La experiencia fue dividida en dos fases: En la primera seguimos el curso de programador en aplicaciones científicas del Centro de Cálculo de la Universidad de Barcelona, donde se enseñaba por primera vez el lenguaje UBL. Ello nos permitió, aparte de conocer el lenguaje, hacer una primera valoración de la utilidad del lenguaje y los problemas docentes, a partir del análisis de los programas realizados por los alumnos del curso y de la observación de su actitud en las sesiones prácticas. Además en esta fase previa elaboramos técnicas y métodos para poder valorar el curso experimental de la segunda fase.

Con esta valoración inicial, empezamos la segunda fase, es decir la experiencia propiamente dicha.

Participaron en este curso experimental, 32 alumnos procedentes de segundo curso de la facultad de Psicología de la Universidad de Barcelona. A la mitad de ellos se les enseñó UBL versión inglés y a la otra mitad UBL versión catalán. La selección de los alumnos se hizo a través de un cuestionario en el que entre otros datos, se pedían las optativas hechas en COU, el nivel de catalán, el nivel de inglés y la lengua materna. Los alumnos del grupo inglés, no habían estudiado nunca este idioma y su desconocimiento era total. Los alumnos del grupo catalán, tenían un buen dominio de este idioma. A parte, se homogeneizaron los dos grupos según los estudios previos y la lengua materna.

Para valorar el aprendizaje en los dos grupos, se realizaron por un lado, una serie de pruebas en situaciones controladas (prueba de recuerdo, prueba de composición y prueba de ordenación) y por otro se analizaron todas las compilaciones y ejecuciones de los programas realizados por los alumnos durante el curso. A continuación explicaremos en qué consistía cada una de estas pruebas así como los resultados obtenidos en ellas:

1. Prueba de recuerdo: Después del primer mes de curso, se mostró a los alumnos de los dos grupos un programa (cada grupo en su versión lingüística). El programa como puede observarse en la tabla 1, pide una fecha (día de la semana, número del día, mes y año) y halla la del día siguiente. Los alumnos debían observar el programa e intentar comprenderlo durante 15 minutos. Media hora después se les pidió que recordaran el programa estudiado. Elaboramos unas categorías para valorar el recuerdo de cada línea del programa según el tipo de error cometido. Las puntuaciones medias obtenidas en cada línea del programa se hallan expresadas también en la tabla 1, así como la media total de líneas recordadas. Los resultados nos muestran que el grupo UBL inglés cometía significativamente más errores ($T = 1.92$, $gl = 68$, $p < 0.05$). Por otra parte el grupo UBL catalán recordaba más líneas que el grupo UBL inglés ($T = 2.52$, $gl = 19$, $p < 0.05$).

	CAT.	ANG.
<i>Programa</i> Seguent-data;	4	4
<i>TiOUS</i> dia-semmana <i>es</i> (dilluns, dimarts, dimecres, dijous, divendres, dissabte, diumenge);	3.83	2 + *
<i>TiOUS</i> mes <i>es</i> (gener, febrer, marc, abril, maig, juny, juliol, agost, sept, oct, novem, decem);	3.83	2 + *
<i>Tipus</i> dia <i>es</i> 1.. 31;	2.5	1.77 +
<i>ds</i> : dia-semmana;	3.25	2.77 +
<i>d</i> : dia;	3.41	2.77 +
<i>m</i> : mes;	3.41	2.77 +
<i>a</i> : enter;	1.83	2.77
<i>condicio</i> <i>es</i> -ultim-dia: logic <i>es</i>	3.25	2.33 + *
<i>inici</i>	1.33	2.66
<i>si</i> <i>d</i> 31 <i>llavors</i> <i>val</i> cert;	2.16	1.33 +
<i>sino</i> <i>val</i> fals;	3.41	2 + *
<i>fi</i> <i>sj</i> ;	2.66	2.66
<i>fi</i> <i>es</i> -ultim-dia;	1.83	1.77 +
<i>inici</i>	3.66	2.22 + *
<i>escriu</i> ('quin dia <i>es</i> avui?'); <i>salta</i> - <i>linia</i> ;	2.83	2.22 +
<i>llegeix</i> (<i>ds</i>);	2.66	1.77 +
<i>escriu</i> (' <i>escriu</i> la data d'avui'); <i>salta</i> - <i>linia</i> ;	3	1.77 +
<i>llegeix</i> (<i>d</i> , <i>m</i> , <i>a</i>);	2.58	1.44 +
<i>si</i> <i>ds</i> diumenge <i>llavors</i> <i>ds</i> : dilluns;	1.83	1.33 +
<i>sino</i> <i>ds</i> : suc (<i>ds</i>);	2	1.1 +
<i>fi</i> <i>si</i> ;	1.33	0.66 +
<i>si</i> <i>es</i> -ultim-dia <i>llavors</i>	1.25	0.33 + *
<i>si</i> <i>m</i> decembre <i>llavors</i>	2.41	1.33 +
<i>a</i> : <i>a</i> - 1; <i>d</i> : 1; <i>m</i> : gener;	2.5	1.5 +
<i>sino</i>	2.33	1.33 +
<i>d</i> : 1; <i>m</i> : suc (<i>m</i>);	1.75	1.11 +
<i>fi</i> <i>si</i> ;	1.75	0.44 +
<i>sino</i> <i>d</i> : <i>d</i> - 1;	1.33	0 + *
<i>fi</i> <i>si</i> ;	1.5	0.44 +
<i>escriu</i> (' <i>dema</i> sera ', <i>ds</i> , ', 'd', 'de', <i>m</i> , 'de', <i>a</i>);	2.33	1.22 +
<i>fi</i> <i>programa</i> ;	2	1.33 +
mitjana	2.5	1.7 + *
mitjana de línies recordades	23	17 + *

Taula 1: Puntuació mitjana de cada grup en les línies del programa de la prova de record. El signe + indica les línies on la puntuació del grup català va ser més alta que la puntuació del grup anglesès. L'asterisc indica les puntuacions estadísticament significatives al 0.05.

Se ha demostrado que esta prueba es un buen instrumento de medida de la habilidad de programar (Mc Keithen et al., 1981). Los programadores no almacenan en memoria el nivel léxico y sintáctico de un programa sino que extraen el significado de la información agrupando conjuntos de sentencias en esquemas de resolución de problemas representando el funcionamiento de este grupo de sentencias. Como prueba de validez hemos de señalar que las personas que recordaron más líneas fueron las que realizaron programas más complejos y obtuvieron puntuaciones más altas en las otras pruebas.

2. Prueba de composición: Con la finalidad de analizar la evolución en los dos grupos en la habilidad de resolver problemas, se les pidió la solución de un mismo problema en tres momentos distintos del curso. El problema que escogimos era uno de los problemas típicos que se enseñan en un curso introductorio. A partir de un ejemplo concreto, los alumnos tenían que modificarlo para poder solucionar el problema que se les planteaba. El programa 'ejemplo' y el programa 'prueba' (solución que debían realizar los alumnos) pueden observarse en las figuras 2 y

<pre> Programa Exemple; c: caracter; contador: enter; inici escriu ('escriu una frase', ' acabada per un punt'); salta-linia; repeteix llegeix (c); si (c = 'a') o (c = 'A') llavors contador := contador + 1; fi si; fins-que c = '.' ; escriu ('el nombre de as es', contador); fi programa; </pre>
<pre> Program Exemple; c: char; contador: integer; begin write ('Escriu una frase acabada per un punt '); writeln; repeat read (c); if (c = 'a') or (c = 'A') then contador := contador + 1; end if; until c = '.' ; write ('El nombre de as es ', contador); end program; </pre>

Figura 2: Programa exemple de la prova de composicio.

<pre> Programa Prova; n, contador, suma:enter; mitjana: real; inici escriu (' escriu els nombres '); salta-linia; repeteix llegeix (n); contador: = contador + 1; suma: = suma + n; fins-que suma > 100; mitjana: = suma/contador; escriu (' la mitjana es ', mitjana); fi programa; </pre>
<pre> Program Prova; n, contador, suma: integer; mitjana: real; begin write ('Escriu nombres enters '); writeln; repeat read (n); contador: = contador + 1; suma: = suma + n; until suma > 100; mitjana: = suma/contador; write ('La mitjana es ', mitjana); end program; </pre>

Figura 3: Programa Prova de la prova de composicio.

3 respectivamente. Los resultados mostraron que, especialmente en las sentencias que exigían una modificación del ejemplo inicial (contador y condición) los alumnos del grupo UBL inglés cometían más errores y sólo llegaban a solucionarlos en momentos más avanzados del curso (ver figuras 4, 5, 6). En el análisis de varianza de las puntuaciones obtenidas en la acción 'contar los números', se observa que tanto la variable grupo (UBL inglés o catalán) como la variable prueba (momento del curso) son significativas al 0.05 ($F = 3.61$ y $F = 8.53$). Una de las conclusiones que podemos sacar es que se tarda más en aprender cuando se utiliza un léxico desconocido por el principiante (como ya veremos en otros resultados). El hecho de que, en algunas de las pruebas, las rectas confluyan se debe a que el problema planteado era excesivamente sencillo.

3. Prueba de ordenación: Esta prueba está considerada especialmente como un test lógico (Coombs M.J. et al., 1982). Se trata de presentar un programa con las sentencias desordenadas y la tarea de realizar consiste en ponerlas en el orden co-

FIGURA 4
Evolución de la programación correcta de la instrucción
(prueba de composición)

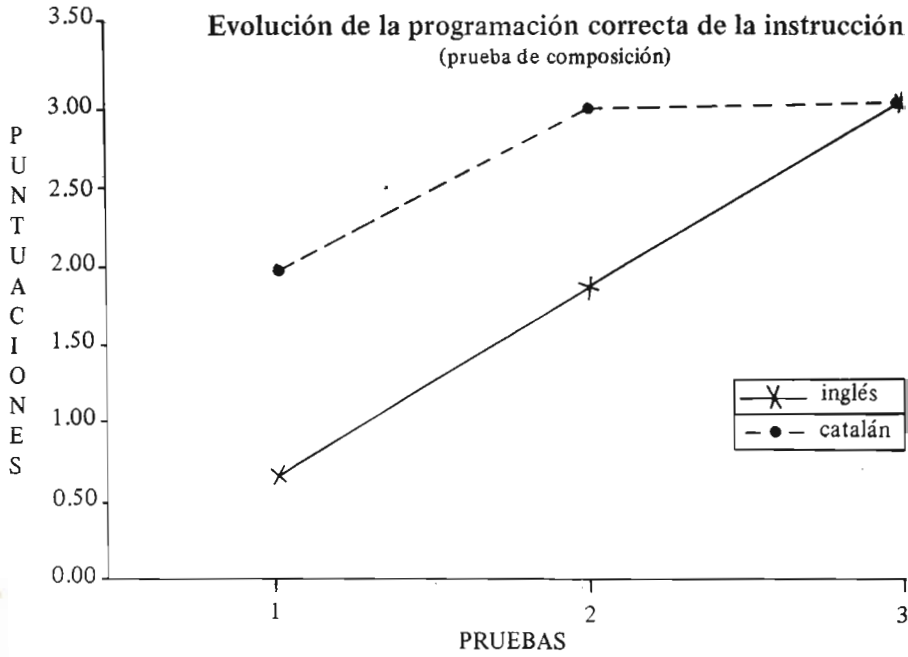
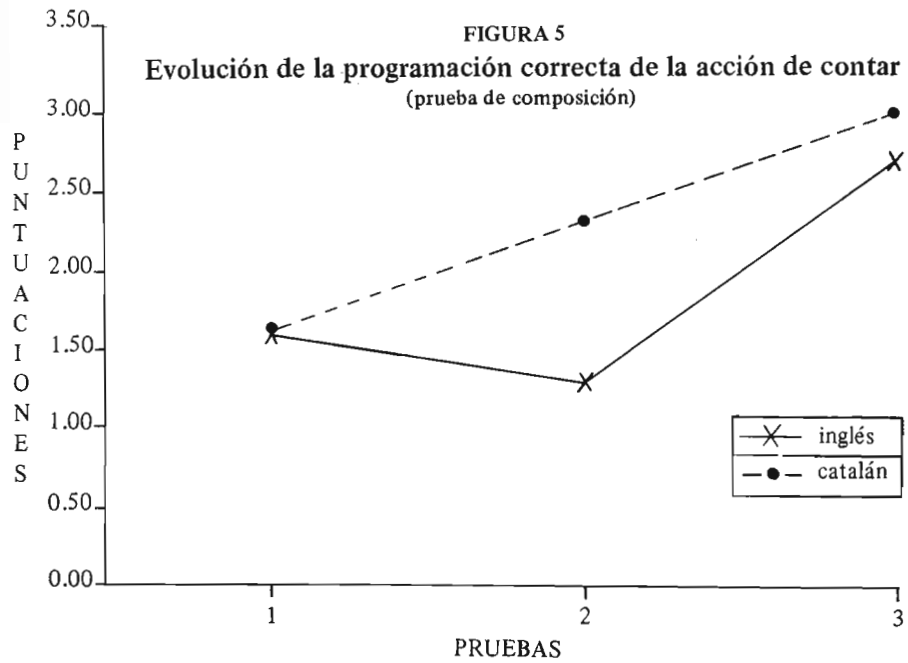
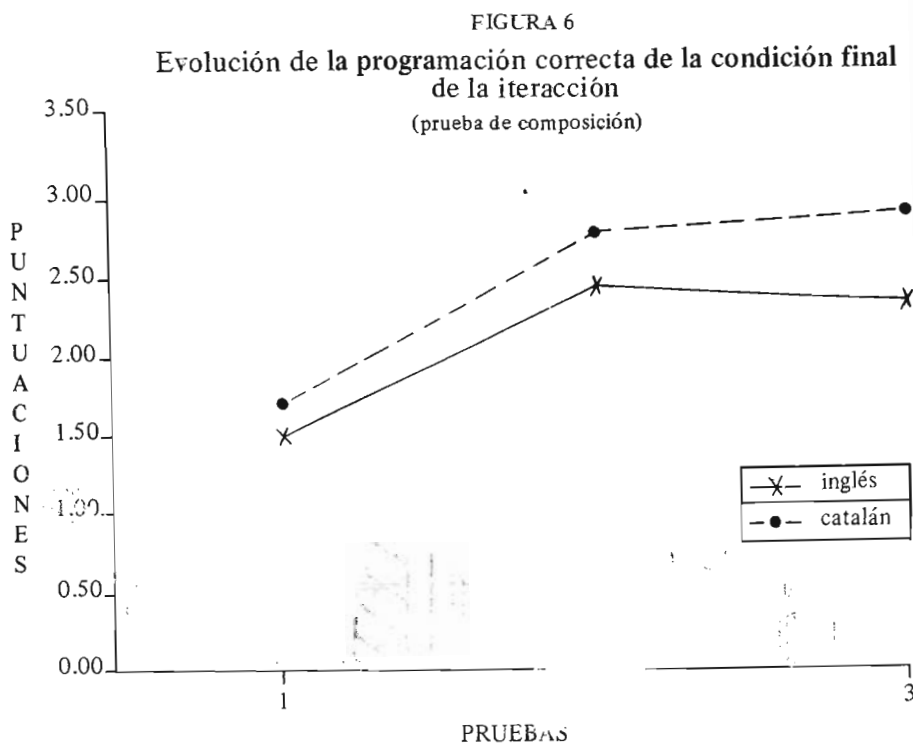


FIGURA 5
Evolución de la programación correcta de la acción de contar
(prueba de composición)





respondiente. La ventaja de esta prueba es que pueden analizarse con facilidad los errores de tipo lógico ya que no hay problemas de tipo sintáctico o léxico pues las sentencias ya están escritas. Como muestran los resultados ($T = 2.65$, $gl = 19$, $p < 0.05$) también en esta prueba, el grupo UBL catalán era más eficiente que el grupo UBL inglés.

4. Análisis de los programas. Finalmente, el análisis de los programas realizados durante el curso, complementa los resultados expuestos hasta ahora. Se analizaron las compilaciones y ejecuciones realizadas en todas las sesiones prácticas. Para evaluar el nivel alcanzado en los dos grupos, utilizamos los siguientes indicadores: 1. Número de líneas por programa, 2. programas originales o variaciones de programas solucionados en clase y 3. complejidad lógica. En todos estos indicadores el grupo UBL catalán fue superior al grupo UBL inglés (ver figuras 10 y 11).

Estos indicadores, deben tenerse en cuenta al interpretar el número de errores léxicos, sintácticos y semánticos y su evolución a lo largo del curso. Es de esperar que al aumentar la complejidad de los programas, aumenten el número de errores. Sin embargo y a pesar de que el grupo UBL catalán hacía programas más complejos, los resultados muestran, como puede observarse en la figura 7, que el grupo UBL inglés, cometía mayor número de errores de los tres tipos ($F = 8.01$, $gl = 1$, $p < 0.05$). El hecho de que en las figuras que muestran la evolución de los errores (figuras 8 y 9) parezca que los grupos converjan al final del curso, debe considerarse en función de la longitud del programa. Teniendo en cuenta el número de errores

FIGURA 7

Mesa de errores en el conjunto de programas a lo largo de la experiencia
(media de errores / número de programas)

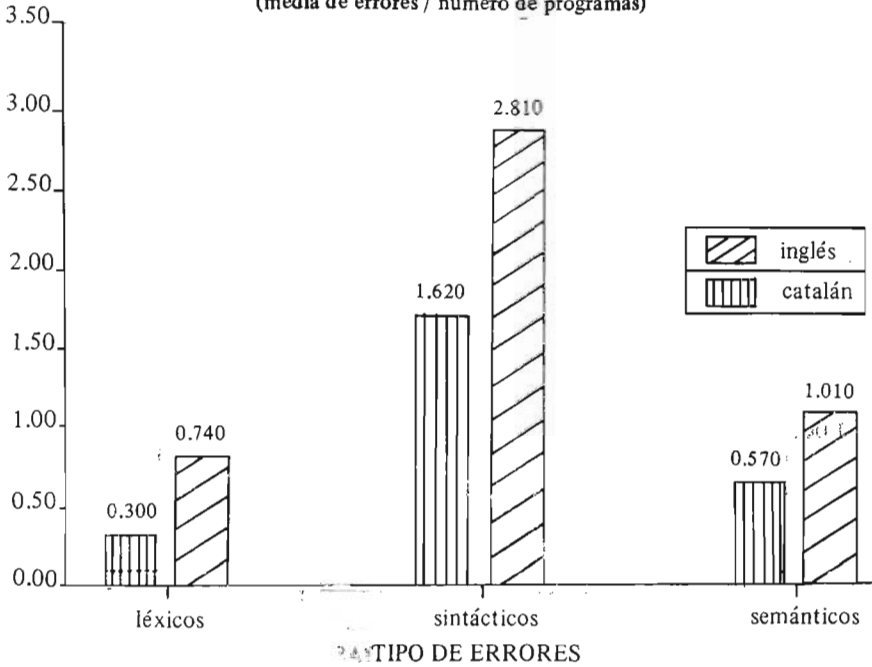
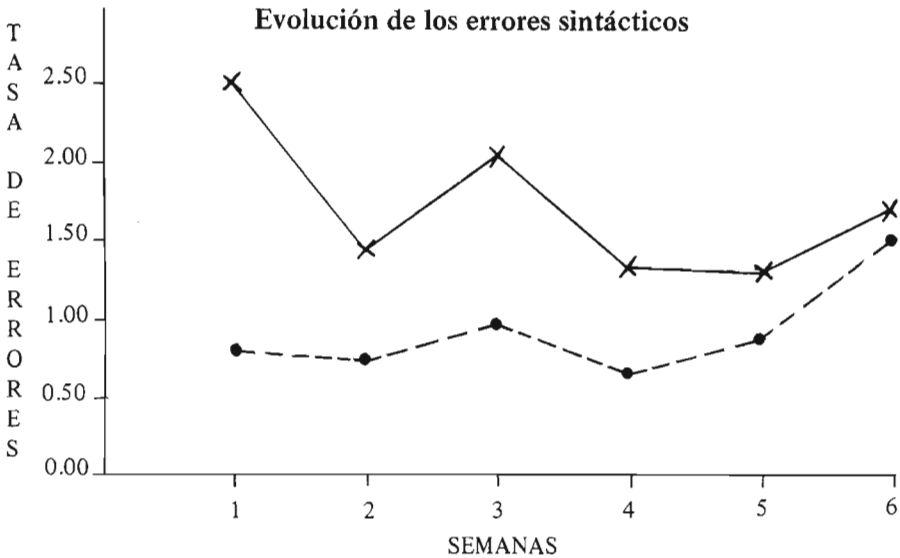


FIGURA 8

Evolución de los errores sintácticos



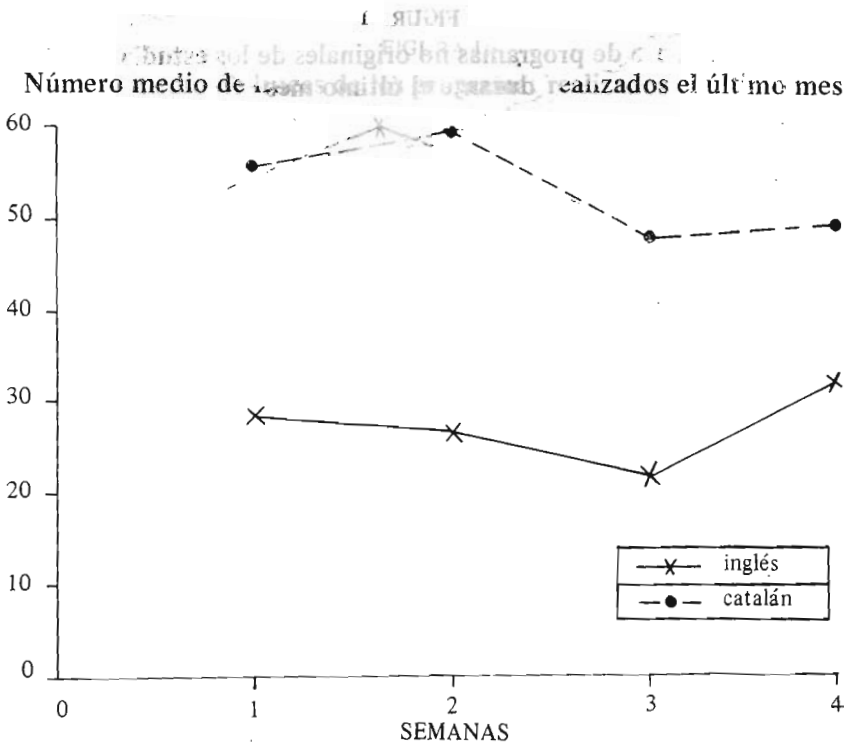
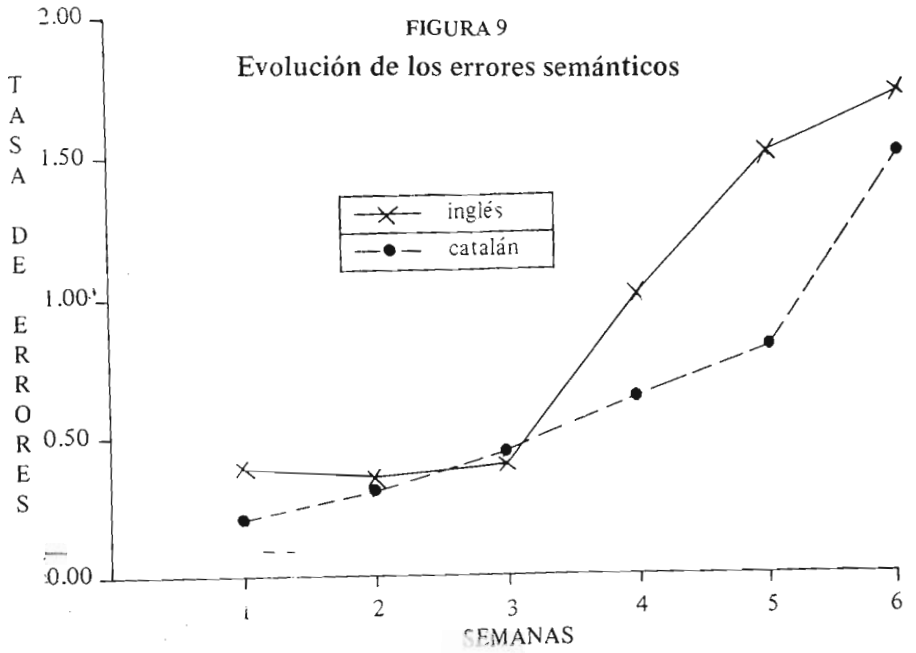
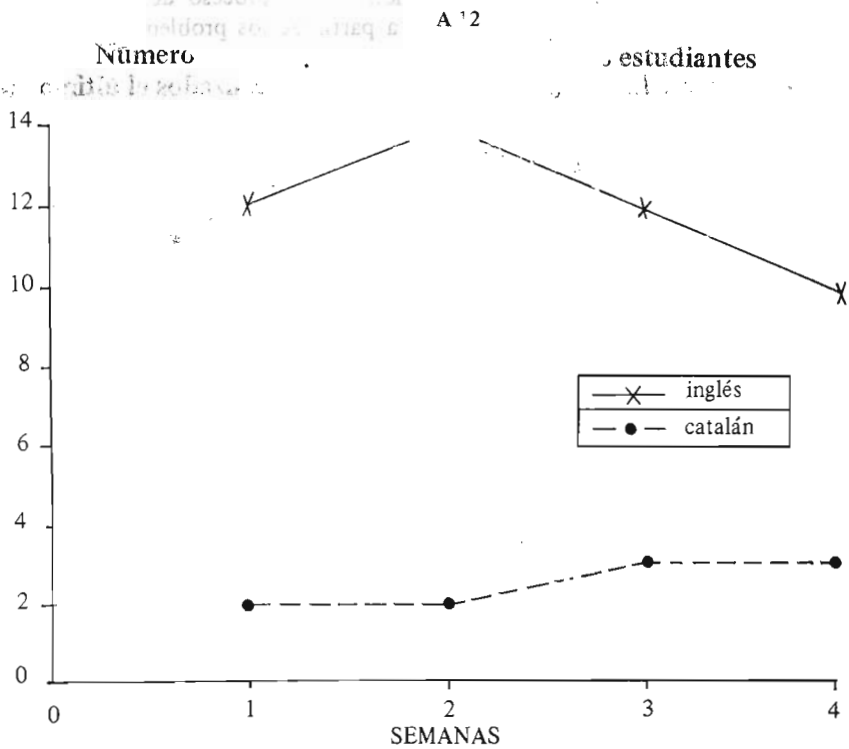
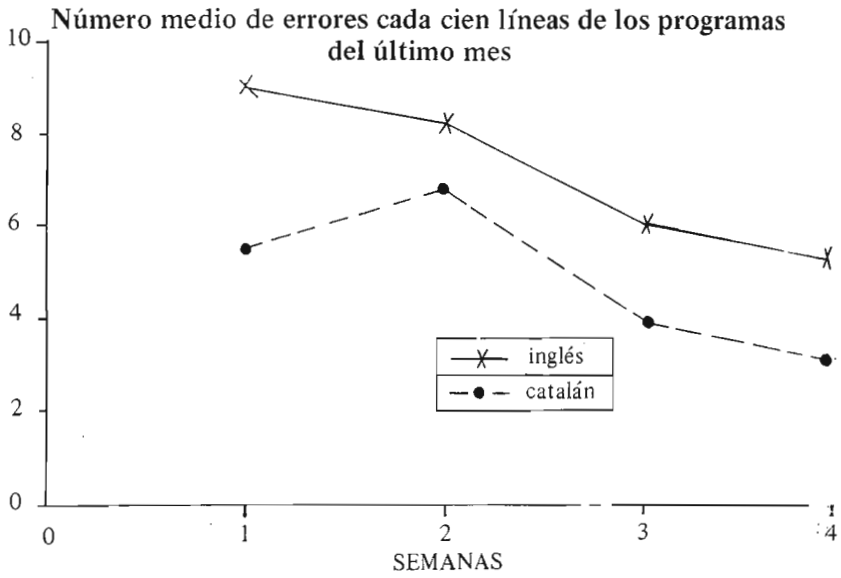


FIGURA 11



según número de líneas (en lugar de número de errores por programa), la diferencia de errores se mantiene a lo largo del último mes del curso (ver figura 12).

Ante estos resultados, podemos concluir que una codificación del lenguaje familiar para el alumno, es más fácil de utilizar y entender que una codificación extraña y que por lo tanto, facilita el aprendizaje de la programación. Por ello pensamos que especialmente para introducir la programación, parece conveniente que las palabras reservadas e identificadores predefinidos de los lenguajes de programación, estén traducidos a la lengua habitual de los estudiantes. Este aspecto, ha sido recientemente señalado por Manuel González Dávila en un artículo publicado en El País, en el que insiste en la importancia de realizar cualquier aprendizaje en la lengua habitual de las personas y por lo tanto "resulta necesario crear un lenguaje informático para la enseñanza redactado en cada una de las lenguas de nuestro país".

Sin embargo, pensamos que el seguimiento de un curso introductorio no es suficiente para valorar con profundidad la habilidad en programar. Sería interesante hacer esta valoración con programadores expertos para ver si la habilidad en los dos grupos llega a igualarse o bien si, por el contrario, se mantiene siempre esta diferencia. Para el próximo año, estamos organizando una experiencia de larga duración donde tenemos previsto realizar varios cursos para poder valorar con mayor profundidad la influencia del lenguaje habitual en la programación así como la influencia de diferentes estructuras de lenguajes de programación.

Algunos de los proyectos que tenemos actualmente planteados es comparar el Pascal y el BASIC y en un futuro próximo el UBL con otros lenguajes. De momento mantenemos una estrecha relación con el proceso de definición del lenguaje UBL, desarrollando nuevas ideas a partir de los problemas hallados en esta experiencia.

REFERENCIAS BIBLIOGRAFICAS

- ARNAU, J., SEBASTIAN, N., SOPENA, J.M. (1982): *Revisión histórica sobre el bilingüismo*. Anuario de Psicología.
- COOMBS, M.J., GIBSON, R., ALTY, J.L. (1982): *Learning a first computer language: strategies for making sense*. International Journal of Man-Machine studies, 16, 449-486.
- DIJKSTRA, E.W. (1980): *Selected Writings on Computing: a personal perspective*. Spinger-Verlag, New York.
- GONZALEZ, M. (1984): *Una informática en castellano para la enseñanza*. El País, martes 29 de mayo.
- McKEITHEN, K., REITMAN, J.S., RUEYER, H.H., HIRTLE, S.C. (1981): *Knowledge organization and Skill differences in Computer Programmers*. Cognitive Psychology, 13, 307-325.
- SHNEIDERMAN, B. (1980): *Software psychology. Human factors in computer and information systems*. Winthrop publishers, inc. Cambridge, Massachusetts.
- SOPENA, J.M. y SEBASTIAN, N. (1981): Tesina de licenciatura. Facultad de Psicología. Universidad de Barcelona.

Este proyecto se ha realizado con el soporte del Instituto de Ciencias de la Educación (ICE), Centro de Cálculo y el Departamento de Psicología Experimental de la Universidad de Barcelona.