Metodología docente orientada a proyectos aplicada a las prácticas de compiladores

Emilio P. Vivancos, Lidia Moreno, José M. Benedí y Vicente Gisbert

Facultad de Informática
Universidad Politécnica de Valencia
{vivancos,lmoreno,jbenedi,vgisbert}@dsic.upv.es

Resumen

En esta ponencia se presenta una experiencia en la organización de las prácticas de dos asignaturas ("Compiladores I" y "Compiladores II") en las que se afronta un problema de programación complejo. Las prácticas de estas dos asignaturas están organizadas en torno a un proyecto común: la construcción de un compilador para un lenguaje de programación similar a un subconjunto del lenguaje Pascal. Esta organización de las prácticas orientada a la elaboración de un proyecto facilita que el alumno aprenda a enfrentarse a un problema de programación complejo, al mismo tiempo que proporciona una visión global de la asignatura difícil de lograr de otra forma. Pero al tratarse de un único proyecto de una envergadura considerable que abarca las prácticas de dos asignaturas, aparecen problemas de coordinación y evaluación. En esta ponencia presentamos las soluciones adoptadas.

1 Introducción

En esta ponencia se presenta una experiencia en la que se han orientado las prácticas de las asignaturas de "Compiladores I" y "Compiladores II" de la Facultad de Informática de la Universidad Politécnica de Valencia, a la realización de un único proyecto: la construcción de un compilador para un lenguaje de programación similar a Pascal. Esta propuesta docente en la que predomina el "saber hacer", se enmarca dentro del Plan de Innovación Educativa de la Universidad Politécnica de Valencia [7], y es fácilmente extensible a otras asignaturas en las que se requiera poner en práctica conocimientos de programación.

Las asignaturas de "Compiladores I" y "Compiladores II" de la Facultad de Informática de Valencia se imparten en los cuatrimestres séptimo y octavo. La asignatura "Compiladores I" tiene asignados 3 créditos de teoría y 1.5 de prácticas. Su objetivo principal es enseñar al alumno los aspectos fundamentales del proceso de traducción, para que sea capaz de aplicar los conocimientos adquiridos sobre gramáticas formales y autómatas al proceso de diseño de lenguajes y a su traducción. Como resultado de ello, el alumno debe ser capaz de diseñar y desarrollar analizadores léxicos, sintácticos y semánticos. El contenido teórico de la asignatura es muy similar al de las asignaturas equivalentes de la mayoría de centros de España y del resto del mundo: Tras una introducción general al proceso de compilación, se dedica un tema a cada una de las fases de análisis de un compilador: análisis léxico, sintáctico y semántico [1, 2, 3, 5]. La asignatura "Compiladores II" tiene la misma distribución de créditos (3+1.5). El objetivo principal de la asignatura es que el alumno conozca las técnicas básicas empleadas durante los procesos de generación y optimización de código para que sea capaz de diseñar y desarrollar los módulos correspondientes de un compilador [1, 4, 6].

En esta ponencia nos vamos a centrar en la experiencia que estamos poniendo en práctica para organizar las prácticas de estas dos asignaturas en torno a un único proyecto. El resto de esta ponencia esta organizada de la siguiente manera: La siguiente sección está dedicada a presentar la organización y el contenido del proyecto. En la sección tercera se comenta la importancia de la

labor de tutorización del proyecto. La sección cuarta muestra los resultados de una pequeña encuesta realizada a los alumnos. Finalizamos el artículo presentado a modo de resumen las conclusiones más importantes de la experiencia presentada.

2 Organización del proyecto

Como se ha comentado en la introducción, en nuestro centro las asignaturas de compiladores se imparten en el cuarto año de los estudios de ingeniería informática. Esto permite que el alumno llegue a nosotros tras haber estudiado en otras asignaturas materias cuyo conocimiento es fundamental para el desarrollo de un compilador: Programación, algoritmos y estructuras de datos, teoría de lenguajes y gramáticas, ingeniería del software,...

A pesar de que el alumno lleva varios años empleando técnicas de programación para realizar las prácticas de otras asignaturas, cuando llega a cuarto curso en la mayoría de los casos todavía no se ha enfrentado a un problema de programación medianamente complejo. Las causas de esto pueden ser varias. Por un lado, pocos alumnos toman la iniciativa y dedican tiempo a desarrollar algún proyecto informático por su cuenta. Por otro lado, en la mayoría de las asignaturas de los estudios de ingeniería informática aparecen problemas de tiempo para impartir todos los contenidos teóricos y prácticos que se desean. Con el objetivo de que los alumnos puedan practicar el mayor número de "técnicas" distintas, una alternativa muy empleada es la de diseñar pequeñas prácticas muy centradas en la resolución de los problemas más típicos de cada asignatura. Desgraciadamente, cuando se adopta esta alternativa en la mayoría de asignaturas, el alumno llega a los cursos superiores sin haberse enfrentado al problema del desarrollo de un proyecto de programación de cierta envergadura.

Al plantearnos el diseño del contenido de las prácticas de las asignaturas de "Compiladores I" y "Compiladores II" tuvimos que decidir entre dos alternativas: preparar un conjunto de pequeñas prácticas en las que se resuelvan los problemas más significativos de la construcción de un compilador, o plantear al alumno la construcción de todo un proyecto paso a paso. Elegimos está segunda alternativa porque es una forma de conseguir que el alumno adquiera una visión global del funcionamiento de un compilador. Además, esta segunda alternativa es la más cercana al tipo de trabajos que el ingeniero en informática se va a encontrar en sus labores profesionales. Sin embargo, la limitación de recursos materiales y temporales, dificulta su puesta en práctica. Nuestra metodología está altamente orientada a la realización de un proyecto práctico donde el alumno puede poner en práctica los conocimientos del funcionamiento de un compilador aprendidos en las sesiones teóricas. Debido a la fuerte relación que hay entre los contenidos de las dos asignaturas de compiladores, decidimos plantear un único proyecto para las dos asignaturas.

Esta propuesta nos ha obligado a medir cuidadosamente el esfuerzo exigido a los alumnos. a estructurar adecuadamente el proyecto para evitar incompatibilidades entre los módulos desarrollados en cada una de las dos asignaturas. y a realizar modificaciones en el contenido de las clases teóricas. Entre los problema más típicos que hemos tenido que resolver se encuentra el de los alumnos que cursan un año "Compiladores I" pero no cursan "Compiladores II" hasta el siguiente. Para facilitar que un alumno realice la parte del proyecto correspondiente a "Compiladores II" cuando no dispone de los módulos desarrollados en "Compiladores I", los profesores de la asignatura hemos desarrollado estos módulos y se los proporcionamos.

Finalmente, como la construcción individual de todos los módulos de un compilador es un trabajo inabordable, se propone realizar el proyecto en pequeños grupos de como máximo cuatro alumnos. Además, con ello se consigue fomentar las habilidades de trabajo en equipo, requisito imprescindible en todo ingeniero informático.

2.1 Material elaborado

Para luchar contra las limitaciones temporales (3 créditos para las prácticas de las dos asignaturas) proporcionamos a los alumnos material de ayuda. Entre el material que suministramos a los alumnos se encuentra una librería que contiene gran parte de los tipos abstractos de datos necesarios para construir un compilador. Los alumnos ya han implementado estos tipos abstractos de datos en otras asignaturas de

la carrera, pero la adaptación (y la corrección de errores) de estas funciones les hacía perder un tiempo que necesitaban para centrarse en los problemas típicos de la construcción de compiladores. Entre las funciones que contiene esta librería se encuentran las necesarias para manipular listas de identificadores y de referencias no satisfechas.

También se proporciona a los alumnos un conjunto de librerías de funciones específicas para la construcción de compiladores. Así por ejemplo, les proporcionamos una librería con la implementación de una tabla de símbolos. A través de las funciones de modificación de esta estructura de datos, los alumnos implementan la gestión de la tabla de símbolos de su compilador. Entre estas funciones hemos incluido una para visualizar el contenido de la tabla de símbolos, lo que les es de mucha ayuda en la fase de depuración de su compilador.

Para evitarles el manejo de ficheros, también les proporcionamos una librería que les permite generar código intermedio directamente con el formato requerido. El código que los compiladores deben generar está basado en el propuesto por Aho et al. en [1], pero con algunas modificaciones. Entre las modificaciones podemos destacar la inclusión de instrucciones para el manejo de la pila de activación y de una estructura tipo display. Gracias a esto los alumnos pueden implementar el módulo encargado de generar el código para la carga y descarga de procedimientos y funciones recursivas con parámetros y variables locales. El código generado puede ser ejecutado en una máquina virtual que hemos desarrollado para el entorno Windows. Esta máquina virtual, a la que hemos llamado *Malpas*, facilita enormemente a los alumnos la depuración del generador de código. Este máquina permite ejecutar paso a paso el código, a la vez que muestra el contenido de la memoria, pila de activación y del display. Tal y como se puede apreciar en la figura 1, cuando los alumnos cargan la secuencia de código que ha generado su compilador, la pantalla de Malpas queda dividida en cuatro partes. En la parte superior izquierda se representa el código cargado, resaltando las instrucciones sobre las que se ha definido un punto de ruptura para facilitar la depuración del programa. En la ventana inferior derecha se muestra el resultado de la ejecución

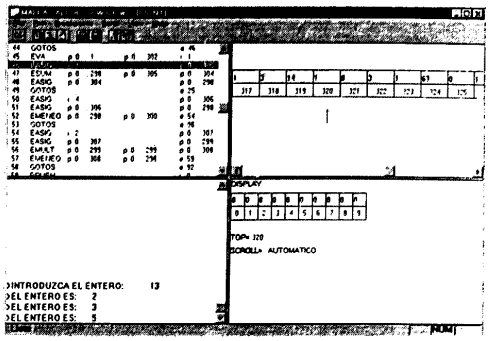

Figura 1: Aspecto de la máquina virtual *Malpas*.

del programa. Finalmente, en la parte derecha se muestra el contenido de la memoria, de la pila de activación y del display.

Para facilitar que los alumnos puedan verificar la corrección de cada módulo de su compilador se les entrega un conjunto de programas de prueba en el lenguaje propuesto. Para que el lector pueda hacerse una idea de la complejidad del compilador desarrollado, en la figura 2 se muestra un ejemplo de programa de prueba del compilador construido por los alumnos.

2.2 Evaluación

La evaluación de un proyecto de estas características es uno de los temas sobre los que más discutimos. Actualmente el proyecto representa el 30% de la nota de la asignatura de "Compiladores I", y entre un 30 y un 50% de la de "Compiladores II". Los grupos deben realizar obligatoriamente tres revisiones del proyecto. En la revisión que se realiza al final del séptimo cuatrimestre se comprueba el funcionamiento del analizador léxico y sintáctico. En la revisión segunda, realizada a mitad del octavo cuatrimestre, se comprueba el funcionamiento del analizador semántico. Al finalizar el octavo cuatrimestre se realiza una revisión final de todo el proyecto.

La calificación del proyecto está basada en tres parámetros:

- Una entrevista personal de aproximadamente 20 minutos con el grupo. Esta entrevista final con el tutor sirve para revisar la corrección del proyecto, y detectar a los alumnos de cada grupo que más han destacado en la realización del proyecto.

```
program Primos;
var
   a: array [3..5, 2..100] of boolean;
   n, m, max: integer;
function divisor(d, n: integer):boolean;
begin
  while d < n do n := n - d;
  divisor := (n = d) or (n = 0);
end;
begin
  read(max);
  n := 2;
  while n <= max do begin
    a[3,n] := true;
    n := n + 1;
    end;
  n := 4;
  while n <= max do begin
    m := 2;
    while m*m <= n do begin
      if divisor(m,n) then begin
        a[3,n] := false;
        m := n;
        end
      else m := m + 1;
      end;
    n := n + 1;
    end;
  n := 2;
  while n <= max do begin
    if a[3,n] then write(n);
    n := n + 1;
    end;
end.
```

Figura 2: Ejemplo de programa de prueba del compilador.

- Una prueba objetiva sobre el proyecto incluida en el examen final de la asignatura, con cuestiones relacionadas con la implementación de los módulos del compilador.

- Una prueba voluntaria de ampliación del compilador. En esta prueba, se plantea a los alumnos que amplíen o modifiquen un módulo concreto de su compilador.

3 Tutorías y lista de distribución

Debido a la complejidad del proyecto es muy importante la labor de tutorización del trabajo de los alumnos. Para facilitar que los alumnos comiencen a construir su compilador, a lo largo del curso se imparten en los laboratorios de programación seminarios en grupos reducidos. Cada uno de estos seminarios está dedicado a introducir un problema concreto del diseño y programación de un módulo del compilador: implementación del analizador léxico, implementación de un analizador sintáctico descendente recursivo,...

Pero además de estos seminarios, hemos considerado adecuado realizar un seguimiento más directo del proyecto. Para ello, hemos reforzado la labor de tutoría asignando a cada proyecto un profesor que realiza las labores de tutor-experto. Este tutor es el responsable de resolver todas las dudas y problemas que se le planteen a cada grupo. Además, es el encargado de sugerir mejoras, plantear ampliaciones y detectar problemas en cada uno de los proyectos que dirige.

Para facilitar el intercambio de información profesor-alumno y alumno-alumno, hemos creado una la lista de distribución de correo electrónico de temas relacionados con compiladores. La lista está abierta a todo tipo de colaboraciones, especialmente a dudas sobre la realización del proyecto y problemas surgidos al emplear las herramientas suministradas, aunque también se reciben consultas sobre cuestiones teóricas (un 30% aproximadamente). Las colaboración son enviadas a todos los alumnos que libremente se han subscrito a ella. En la medida de lo posible intentamos que sean los propios alumnos los que colaboren para resolver las dudas de sus compañeros, aunque los profesores de la signatura también intervenimos cuando nadie es capaz de responder adecuadamente, o cuando la naturaleza de la consulta así lo requiere. Esta lista no solo anima a los alumnos a cooperar y discutir entre sí, sino que también les ayuda a introducirse en canales y formas de intercambio de conocimiento habituales en el mundo científico.

4 Opinión de los alumnos

Al finalizar el octavo cuatrimestre, se ha realizado una pequeña encuesta anónima a 64 alumnos que han cursado "Compiladores I" y "Compiladores II". Los alumnos consideran mayoritariamente que la realización del proyecto ha sido muy importante para entender el funcionamiento de un compilador (Fig. 3). Un 59% considera que el proyecto ha sido bastante útil y un 22% dice que ha sido fundamental para entender el funcionamiento de un compilador.

En general, una mayoría considera que el proyecto ha satisfecho las expectativas creadas en su presentación a principio de curso. Un 56% considera que ha satisfecho bastantes de las expectativas que tenía, un 13% totalmente satisfecho, un 25% se siente solo un poco satisfecho, y

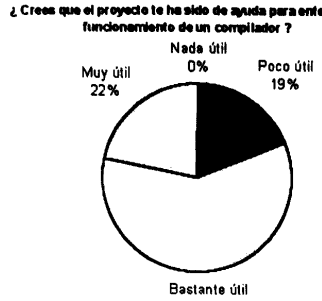

Figura 3: Utilidad del proyecto según los alumnos.

un 6% poco satisfecho.

Merece la pena destacar que para un 32% de los alumnos encuestados, este proyecto ha sido el proyecto de programación de mayor envergadura que ha desarrollado, y para un 57% es uno de los tres mayores proyectos que han desarrollado.

5 Conclusiones

En este trabajo presentamos la organización de las prácticas de las asignaturas de "Compiladores I" y "Compiladores II". A diferencia de otras asignaturas donde se suele plantear el desarrollo de varias prácticas de pequeño tamaño, planteamos la realización de un único proyecto que abarca a las dos asignaturas. En esta metodología orientada a un proyecto hemos encontrado varias ventajas. Por un lado proporciona una visión global mucho más real que cuando se realizan pequeñas prácticas por separado. El alumno se siente más motivado al saber que terminará el curso habiendo participado en la elaboración de un compilador completo que funciona "realmente". También facilita la comprensión de algunos conceptos que difícilmente entendería solo en las sesiones teóricas: distinción entre tiempo de compilación y tiempo de ejecución, generación de código para las llamadas a procedimientos y funciones, gestión de memoria, gestión de la tabla de símbolos en un lenguaje de alto nivel con estructura de bloque, etc.

Uno de los problemas más importantes que hemos tenido que resolver es la falta de tiempo para que los alumnos puedan llevar a cabo la construcción de todo el compilador. Para facilitar esto hemos tomado varias medidas, como organizar el trabajo en grupos de cuatro alumnos o proporcionar a los alumnos un conjunto de librerías de funciones que les descarguen del trabajo menos interesante para los objetivos de la asignatura. Además, la labor de un tutor asignado a cada proyecto que sigue de cerca el trabajo y resuelve las dudas que van surgiendo, supone un apoyo constante en su desarrollo.

Finalmente, queremos destacar que este proyecto de construcción de un compilador se ha convertido en uno de los trabajos de programación de mayor envergadura que realiza el futuro ingeniero informático durante sus estudios en nuestra facultad.

Referencias

[1] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compiladores: Principios, Técnicas, y Herramientas.* Addison-Wesley, 1990.

[2] Alfred V. Aho and Jeffrey D. Ullman. *The Theory of Parsing, Translation, and Compiling. Volume I:Parsing.* Prentice Hall, 1972.

[3] Bernald Teufel, Stephanie Schmidt, and Thomas Teufel. *Compiladores: Conceptos fundamentales.* Addison-Wesley Iberoamericana, 1995.

[4] Jean-Paul Tremblay and Paul G. Sorenson. *The Theory and Practice of Compiler Writing.* Mc. Graw-Hill, 1985.

[5] Emilio Vivancos, José M. Benedí, and Vicente Gisbert. Unidades temáticas de Compiladores I (PID 8033). Instituto de Ciencias de la Educación. Universidad Politécnica de Valencia, 1997.

[6] Emilio Vivancos, José M. Benedí, and Vicente Gisbert. Unidades temáticas de Compiladores II (PID 8033). Instituto de Ciencias de la Educación. Universidad Politécnica de Valencia, 1998.

[7] Emilio Vivancos, Lidia Moreno, José M. Benedí, and Vicente Gisbert. Compiladores: Diseño e implementación. Proyectos de Innovación Educativa (PID8033). Comisión de Calidad. Universidad Politécnica de Valencia, Septiembre 1997.