Actas de las Jenui, vol. 5. 2020.
Páginas: 317-324

Una propuesta integral para desarrollo de proyectos en un curso de Compiladores con una metodología de aprendizaje basada en proyectos

José Miguel Benedí, Emilio Vivancos
Departament de Sistemes Informàtics i Computació
Universitat Politècnica de València
{jbenedi,vivancos}@dsic.upv.es

Resumen

En este trabajo se presenta una estrategia integral para el desarrollo y la evaluación de proyectos en una asignatura de *Compiladores*. En esta asignatura se ha optado por emplear una metodología activa basada en proyectos. Esta aproximación posee ventajas incuestionables; sin embargo, su puesta en marcha presenta dos importantes inconvenientes: sobrecarga de trabajo, tanto para los alumnos como para los profesores; y evaluación justa y realista del proyecto, tanto individual como grupal. Para mitigar el primer problema hemos propuesto un entorno de desarrollo de proyectos, de libre disposición y portable. En cuanto al sistema de evaluación hemos propuesto: un modelo de simulación de usuario, para permitir una evaluación realista del desempeño del proyecto; una prueba práctica individual, para considerar el trabajo individual; y un conjunto de actividades de seguimiento, para considerar el trabajo continuo en el desarrollo del proyecto. Esta estrategia se ha puesto en marcha en los últimos años, y el análisis de sus resultados parece avalar su implantación.

Abstract

This paper presents a comprehensive strategy for the development and evaluation of projects in a course of *Compilers*. In this course we have chosen to use an active methodology based on projects. This approach has unquestionable advantages. However, its implementation has two major drawbacks: work overload, both for students and teachers; and fair and realistic evaluation of the project, both individual and group. In order to mitigate the first problem we have proposed a project development environment, freely available and portable. Regarding the evaluation system we have proposed: a user simulation model, to allow a global evaluation of the final project performance; an individual practical test, to take into account the individual work of the components of the same team; and a set of monitoring activities, to consider continued work on the project. This strategy has been launched in recent years, and the analysis of its results seems to support its implementation.

Palabras clave

Compiladores, aprendizaje basado en proyectos, herramientas de ayuda al desarrollo de proyectos y evaluación de proyectos.

1. Introducción y motivación

En la Universitat Politècnica de València (UPV), el curso de Compiladores se imparte actualmente en la asignatura de *Lenguajes de Programación y Procesadores de Lenguajes* (LPyPL), del grado en Ingeniería Informática, en la Escuela Técnica Superior de Ingeniería Informática. LPyPL está enmarcada en el módulo de tecnología específica de Computación (4º curso, semestre A) y tiene asignados 6 créditos. El principal objetivo de la asignatura es que el alumnado conozca los fundamentos teóricos y prácticos, así como las técnicas y herramientas básicas, para el diseño y construcción de un compilador. Al mismo tiempo se pretende que el alumno sea capaz de aplicar las ideas y técnicas propias del diseño de compiladores en otros campos de la Informática.

Un compilador acepta como entrada un programa escrito en un cierto *Lenguaje Fuente*, usualmente un lenguaje de programación de alto nivel, y genera un programa escrito en un *Lenguaje Objeto*, usualmente un lenguaje máquina [2]. El diseño y construcción de un compilador es un perfecto ejemplo de maridaje entre teoría y práctica [1]; de hecho, constituye una de las primeras disciplinas de la Informática para la cual se desarrolló un importante aparato formal que actualmente se utiliza de forma rutinaria en la construcción

de compiladores. Históricamente, la mayor parte de los cursos de compilación ponían el acento en la teoría del análisis sintáctico y de la traducción dirigida por la sintaxis (*Fase de Análisis*) [2, 14]. Sin embargo, pronto se tomó conciencia de que centrarse sólo en los aspectos teóricos no ayuda a la comprensión del modo en que debe construirse un compilador útil [1, 7].

Esta observación fue positivamente reforzada por nuestra experiencia en la impartición de las sucesivas asignaturas de compiladores en la UPV [3, 13]. A lo largo de estos años, gradualmente hemos ido modificando nuestro punto de vista poniendo el énfasis en la parte de generación y optimización de código (*Fase de Síntesis*) [7]. Como resultado de todo ello, el objetivo de la actual asignatura de LPyPL es que los alumnos tengan la base teórica necesaria que les permita construir un compilador real y completo para un lenguaje de programación de alto nivel, sencillo, pero no trivial.

Para cumplir este objetivo, en LPyPL hemos optado por una estrategia metodológica activa orientada a la realización de un proyecto: aprendizaje basado en proyectos (ABP) [9, 12]. Este tipo de aproximación metodológica es muy atractiva ya que permite la superación de una enseñanza por simple *transmisión/recepción de conocimientos ya elaborados*. Esto sólo puede tener lugar si quien aprende participa, de alguna manera, en la (re)construcción de los conocimientos [8]. En este tipo de planteamientos, el aprendizaje resulta de la construcción de conocimientos a través del tratamiento colectivo de problemas significativos, donde los alumnos se enfrentan a situaciones y problemas realistas que deben solucionar, en un proceso colectivo de investigación dirigida a la consecución de un objetivo común: en nuestro caso, desarrollar un proyecto de compilación. Además, la labor del profesor ya no es solo la de transmisión de conocimientos, sino que también es la de preparación de las actividades que permitan a los alumnos participar en el proceso de aprendizaje y la de dirigir dichas actividades [8, 12].

Con la aplicación de esta estrategia de ABP se han constatado una serie de ventajas [6, 9, 10] que constituyen la motivación que justifica esta elección metodológica y que podemos resumir en los siguientes puntos:

- Conseguir que el alumno adquiera una visión más realista del funcionamiento de un compilador.
- Aumentar la motivación del alumno, ya que participa en la construcción de un compilador completo que funciona realmente.
- Facilitar la comprensión de algunos conceptos que difícilmente se entenderían solo en las sesiones de teoría.
- Incidir en que esta alternativa es la más cercana al tipo de trabajos que el ingeniero en informática se va a encontrar en sus labores profesionales.
- Desarrollar capacidades transversales como el trabajo en equipo, las relaciones interpersonales, la comunicación, toma de decisiones y manejo del tiempo.

Sin embargo, la puesta en marcha de una estrategia de ABP no está exenta de considerables dificultades [6, 9]. A lo largo de su implantación en la asignatura de LPyPL hemos detectado dos grandes fuentes de problemas:

- Sobrecarga de trabajo, tanto para los alumnos como para los profesores.
- Evaluación justa del proyecto, tanto individual como grupal.

La sobrecarga de trabajo para los alumnos es uno de los aspectos críticos asociados históricamente con el ABP y obliga a medir cuidadosamente el esfuerzo exigido estructurando adecuadamente el proyecto. Por otro lado, la sobrecarga de trabajo para los profesores está relacionado con el enorme esfuerzo y tiempo de dedicación que el profesorado debe invertir en el diseño y la preparación del proyecto de compilación para cada curso académico.

Para mitigar la sobrecarga de trabajo de los alumnos y de los profesores proponemos un entorno de desarrollo, de libre disposición y portable, para facilitar el diseño y la elaboración de nuevos proyectos de compilación. Una versión preliminar de este entorno fue presentada en [4], y ha sido utilizado con éxito en los últimos cursos de LPyPL. En la Sección 2, mostraremos el contexto docente y el modelo de implantación, y en la Sección 3, presentaremos nuestra propuesta de *entorno de desarrollo de proyectos de compilación*.

El segundo problema está relacionado con la evaluación del proyecto. Hay una gran cantidad de trabajos previos que abordan este tema [9, 11, 12] y, en general, existe un consenso general en la importancia de la evaluación del proyecto en una estrategia de ABP. Algunas de estas propuestas las hemos estado aplicando en las sucesivas asignaturas de compiladores [3, 13] con resultados dispares. En este trabajo vamos a proponer un sistema de evaluación de proyectos (de compilación) atendiendo a tres consideraciones:

1. *Evaluación objetiva del proyecto*. En el caso de LPyPL, el proyecto es un compilador real, por tanto, consideramos que su evaluación también debe ser lo más realista posible y sobre todo atendiendo al desempeño del mismo.
2. *Evaluación discriminativa individual del alumno*. Dado que el proyecto se realiza en grupos de 3 a 4 alumnos, el problema de la evaluación individual de la aportación de cada alumno al desarrollo final del proyecto cobra, a nuestro entender, una gran importancia en la motivación final del alumno.

3. *Evaluación continua del trabajo del grupo*. El proyecto esta dividido en etapas que los componentes del grupo deben cubrir hasta alcanzar el objetivo final. Por tanto, además de evaluar el resultado final, consideramos que se debe evaluar el proceso continuo de desarrollo del proyecto.

Una versión preliminar de esta propuesta de evaluación de proyectos fue presentada en [5]. En la Sección 4, enunciaremos los objetivos que pretendemos alcanzar y presentaremos el sistema de *evaluación de proyectos*, tanto individual como grupal, que proponemos. Finalmente, en la Sección 5, valoraremos el resultado de la aplicación de este sistema integral de desarrollo y evaluación de proyectos durante los últimos cursos académicos de implantación de la asignatura de LPyPL en la UPV.

2. Contexto docente

Como se ha mencionado anteriormente, el objetivo de la asignatura está orientado a la construcción de un compilador completo para un lenguaje de programación seleccionado. De hecho, tanto la planificación como el desarrollo de las clases de teoría están dirigidas a facilitar la construcción de dicho proyecto de compilación. Para la implantación de esta propuesta y para evitar, en lo posible, la *sobrecarga de trabajo para los alumnos*, que un proyecto de esta envergadura conlleva, se han tomado una serie de medidas correctoras:

1. Impulsar que el proyecto se realice en pequeños grupos, de tres a cuatro alumnos como máximo. Además, con ello se consigue fomentar las habilidades del trabajo en equipo, requisito imprescindible en todo ingeniero informático.
2. Proporcionar a los alumnos un adecuado material de ayuda que les permita reducir significativamente el trabajo de codificación para centrarse en los problemas típicos de la construcción de compiladores. En la Sección 3, daremos una descripción más detallada de este material desde el punto de vista del trabajo del profesor.
3. Planificar un conjunto de seminarios, en grupos reducidos, para la descripción pormenorizada del material y herramientas específicas del proyecto.
4. Dividir el proyecto en partes, para facilitar su elaboración y evaluación, que se corresponden con las fases de análisis (léxico, sintáctico y semántico) y síntesis (generador de código intermedio) en la construcción de un compilador.
5. Reforzar las tutorías en el laboratorio. La labor del tutor no solo debe ser la de resolver las dudas y problemas planteados sino también la de sugerir mejoras, detectar problemas, motivar hábitos de trabajo en equipo y enseñar a generar y documentar buenos programas.

La planificación del proyecto consiste en 10 sesiones de laboratorio de 90 minutos cada una. En 4 de estas sesiones se dedican los 20 minutos iniciales a impartir los seminarios específicos y el resto del tiempo se dedica al trabajo de los alumnos tutorizados por el profesor.

3. Entorno de desarrollo de proyectos de compilación

La preparación, por parte del profesorado, de un nuevo proyecto de compilación cada curso académico lleva asociadas varias e importantes tareas: definir formalmente el lenguaje fuente; definir el lenguaje objeto y desarrollar su máquina virtual; diseñar, implementar, probar y documentar todo el material de ayuda (librerías); y por último, elaborar toda la documentación necesaria. Además, el profesorado debe implementar el proyecto completo antes del comienzo de la asignatura. Este requisito es imprescindible, ya que solo una implementación preliminar completa nos permitirá garantizar que el proyecto es auto-consistente, completo y tratable por parte de los alumnos.

Esta descripción idealizada de la preparación de un proyecto, en realidad, tiene poco parecido con la realidad. Las presiones de tiempo pueden llevarnos a tomar atajos, o a imponer modificaciones en el diseño del proyecto en pleno desarrollo del mismo, o incluso a que el proyecto se diseñe *sobre la marcha*. Además, una vez que un profesor ha creado un proyecto, la gran inversión realizada proporciona un fuerte incentivo para reutilizar el proyecto una y otra vez, incluso más allá del punto en que el proyecto se convierte en obsoleto. Por lo tanto, muchos profesores crean sus propios proyectos de compiladores (cada curso) desde cero, repitiendo mucho del trabajo que ya se ha realizado otras muchas veces.

En otro sentido, hay un problema de índole práctico bastante común en el desarrollo de proyectos que es el de la reutilización de los mismos. La reutilización de un proyecto más de una vez estimula, de alguna forma, el desarrollo de una cierta *memoria* del proyecto que puede durar varios años. Además, los posibles estudiantes deshonestos podrían presentar el trabajo de cursos anteriores como propios. De hecho, solo este problema podría explicar porqué se inventan tantos nuevos proyectos de compiladores con la consiguiente multiplicación de esfuerzos.

Todo esto hace que, si queremos mitigar estos inconvenientes, el esfuerzo y dedicación del profesorado sea mucho mayor que con un enfoque clásico [6, 10]. Para intentar dar respuesta a estos problemas hacemos es-

ta propuesta de un *entorno de desarrollo de proyectos de compiladores* (EDPC), de libre disposición y portable, que facilite y simplifique el proceso de creación de nuevos proyectos de compilación. El EDPC permite a los profesores una fácil modificación de los proyectos de compilación, favoreciendo igualmente la elaboración de proyectos de distinta naturaleza y amplitud. Una característica adicional importante del EDPC es que está pensado para diseñar proyectos modulares, incrementales y portables. Este EDPC ha sido utilizado en los últimos cursos de la asignatura de LPyPL. Por ello consideramos que está lo suficientemente maduro como para que pueda ser útil a otros profesores de otras instituciones. El EDPC completo se compone de las siguientes partes:

3.1. Lenguaje fuente

La elección del lenguaje fuente incorpora todos los problemas de diseño e implementación de un lenguaje de programación y debe satisfacer dos restricciones de diseño previas: que esté bien definido, es decir, que se conozcan las especificaciones léxicas, sintácticas y semánticas; y que sea tratable, es decir, que sea suficientemente reducido como para que los alumnos puedan implementar su compilador en un semestre.

En esta trabajo, nos decantamos por un lenguaje fuente que sea un subconjunto reducido de un lenguaje de programación existente. Esta elección tiene como limitación el propio lenguaje seleccionado pero nos permite reducir la curva de aprendizaje y dota al proyecto de una coherencia que hace que, en nuestra opinión, sea didácticamente más atractivo. El lenguaje elegido, al que denominaremos `MenosC`, es un lenguaje basado en el lenguaje `C`, con algunas restricciones de tipos del lenguaje `C++` [4].

3.2. Lenguaje objeto y máquina virtual

El EDPC nace con la voluntad de ser independiente de la plataforma, portable y fácil de instalar en cualquier máquina `unix` con herramientas estándar de software `gnu`. Para cumplir estos objetivos nos hemos decantado por un código intermedio 3-direcciones (lenguaje objeto), independiente de la máquina. La introducción de un código intermedio independiente de la máquina es un práctica común en la construcción de compiladores, ya que mejora la fase de optimización de código (intermedio), aumenta la portabilidad y, en general, facilita la división en fases en el diseño de un compilador [7]. La elección del código intermedio 3-direcciones es similar al propuesto en [2]. Este código intermedio es similar al que se emplea en las clases de teoría; además, es simple e intuitivo y se adapta perfectamente a las necesidades del proyecto de compilación para la asignatura de LPyPL.

Para poder evaluar en condiciones reales el comportamiento del compilador se ha diseñado y construido una *máquina virtual* que es capaz de ejecutar el código intermedio generado por el compilador del proyecto [4].

3.3. Material de apoyo

En la construcción de un compilador es necesario codificar muchas estructuras de datos (p.ej. la tabla de símbolos) y generar código repetitivo de bajo nivel (p.ej. las plantillas para la generación de código intermedio). Su implementación requiere una meticulosa atención a los detalles y cuando no se hace con cuidado los errores son a veces difíciles de encontrar. Los estudiantes pueden pasar más tiempo tratando de solucionarlos que aprendiendo cómo se construye un compilador. Para atenuar este problema, el EDPC proporciona todo este código adicional, debidamente documentado, en forma de dos librerías [4]:

- `libtds`. Librería con las operaciones para la correcta manipulación de la tabla de símbolos.
- `libgci`. Librería con las operaciones para la gestión de memoria y la generación de código intermedio.

Estas librerías son de propósito general y, en principio, no cambian con la especificación del proyecto de cada curso. Además, facilitar este material de apoyo tiene otras ventajas adicionales: proporciona un nivel moderado de abstracción, elimina en gran medida los posibles errores y permite a los estudiantes centrarse en los aspectos más importantes del proyecto.

3.4. Compilador de referencia

El EDPC también aporta un ejemplo o *caso de estudio* sencillo y derivado directamente de los ejercicios específicos realizados en las clases de teoría. Este compilador de referencia desempeña varias funciones:

- Sirve como ejemplo de uso del EDPC para resolver un problema conocido. Los alumnos pueden ver cómo se emplea el diverso material de apoyo y, sobre todo, el uso de las librerías.
- Las partes en las que se descompone el compilador pueden compilarse y analizarse por separado, lo que apoya el proceso incremental del desarrollo del compilador.
- Sirve como un caso de estudio que los profesores pueden utilizar para guiar a los estudiantes en el desarrollo de su propio compilador.
- Los estudiantes pueden comparar sus soluciones adoptadas frente a un compilador de referencia para un ejemplo específico.

3.5. Documentación

La documentación que se aporta a los alumnos para la adecuada elaboración del proyecto se puede resumir:

- Especificación formal (léxica, sintáctica y semántica) del lenguaje fuente (`MenosC`) del curso actual.
- Descripción completa del material de apoyo, así como de las restricciones de su uso.
- Guía para la elaboración del proyecto de compiladores.
- Documentación adicional de todas las herramientas software utilizadas en el proyecto: Generador Automático de Analizadores Léxicos (`FLEX`)[1] y Generador Automático de Analizadores Sintáctico-Semánticos (`BISON`)[2].
- Documentación completa del compilador de referencia para el caso de estudio.

3.6. Programas de prueba

Estos programas de prueba permiten a los alumnos la correcta autoevaluación de cada una de las etapas en las que se divide su compilador: análisis léxico-sintáctico, análisis semántico y generación de código (intermedio). En la Sección 4, incidiremos más en este aspecto de la autoevaluación.

4. Sistema de evaluación

La evaluación es una pieza fundamental para el buen funcionamiento del método de ABP y, en general, para cualquier trabajo realizado en equipo. Debido a ello, la evaluación de un proyecto ha sido tratada con profusión en la literatura [9, 11, 12]. En este trabajo presentamos nuestra modesta aportación sobre este difícil problema. A nuestro juicio, dos son los grandes retos que comporta la evaluación de un proyecto: cómo realizar una evaluación realista del proyecto, y cómo evaluar individualmente a los diferentes componentes de un mismo equipo. A menudo, estos retos se abordan con una cierta falta de realismo, lo que conduce a una pérdida de motivación en los alumnos. En el contexto de las metodologías activas orientadas a la realización de un proyecto, nuestra propuesta de un sistema de evaluación de proyectos (de compilación) para la asignatura de LPyPL considera los siguientes objetivos principales:

- Debe permitir una evaluación global del desempeño final del proyecto.
- Debe tener en cuenta el trabajo individual de los componentes de un mismo equipo.
- Debe considerar el trabajo continuo en el desarrollo del proyecto.

A estos objetivos principales se le añaden también otros objetivos adicionales:

- Debe responder a los objetivos formativos planteados, tanto los específicos de cada asignatura como los transversales.
- Debe ser claro e informativo para los estudiantes, de modo que éstos sepan los aspectos concretos por los que se les va a evaluar.
- Debe fomentar la capacidad de evaluación y autoevaluación de los estudiantes.

Un sistema de evaluación que esté bien diseñado emplea diversas evidencias para determinar si los estudiantes han cumplido con los objetivos del proyecto. A continuación se describen las evidencias de evaluación para el proyecto de compiladores de la asignatura de LPyPL.

4.1. Evaluación final del proyecto

Tal y como se apuntó en las secciones anteriores, uno de los requisitos primordiales en la metodología de ABP es que el proyecto sea lo más realista posible; ya que, entre otras cosas, incrementa la motivación de los alumnos para desarrollarlo [9]. Por esas mismas razones, la evaluación del resultado final del proyecto también debería ser realista; es decir, debería evaluarse por medio de un *usuario final* que verifique objetivamente si el desempeño final del proyecto cumple con las especificaciones de diseño.

Dada la naturaleza de la asignatura de LPyPL y dado que el proyecto es un compilador, el usuario final debería verificar su desempeño final por medio de la compilación y ejecución de un subconjunto de programas de prueba. Considerando todo ello, y teniendo en cuenta que los alumnos son usuarios habituales de compiladores para sus lenguajes de programación favoritos, en este trabajo proponemos un *modelo de simulación de usuario final* formado por dos elementos:

- *Un conjunto de programas de prueba*, seleccionados por los profesores, para que se pueda testear cada una de las partes que integran el compilador.
- *Una máquina virtual*, incluida en el EDPC, que permita ejecutar el código generado por el compilador para demostrar su corrección.

Además, este modelo favorece la autoevaluación de los diferentes grupos de trabajo, ya que ellos son los usuarios finales y tienen las herramientas para analizar el desempeño de las diferentes partes de su proyecto.

[1] `FLEX: https://github.com/westes/flex`
[2] `BISON: https://www.gnu.org/software/bison`

4.2. Evaluación individual del alumno

En la puesta en marcha del ABP para LPyPL, el proyecto de compilación se realiza en grupos. Esto plantea un reto a la hora de realizar una evaluación individualizada, justa y objetiva, de las aportaciones de cada alumno del grupo en el desarrollo del proyecto

En este trabajo proponemos una *prueba práctica individual*, realizada en el laboratorio, donde cada alumno debe modificar el código de su compilador para incorporar una muy pequeña modificación en el lenguaje fuente. Esta prueba individual valora efectivamente el grado de comprensión y de implicación del alumno en el proyecto. Además, dado que esta prueba es conocida desde el comienzo del curso, motiva a todos los alumnos a participar activamente en la elaboración de su compilador.

Sin embargo, existe la posibilidad de que el miedo a esta prueba pueda retraer a los alumnos menos implicados y provocar un abandono prematuro del proyecto. Para evitar este problema, se han tomado una serie de medidas correctoras: proponer una tutorización, tanto grupal como individual, en las clases de laboratorio y proporcionar una serie de pruebas de ejemplo para que los alumnos puedan evaluar con anterioridad la dificultad de las mismas. En Sección 5 mostraremos que estas medidas correctoras parecen haber funcionado satisfactoriamente.

4.3. Evaluación continua

En una estrategia de ABP, además de la evaluación global del proyecto, también se debe valorar el trabajo continuo de los alumnos en la búsqueda de información y resolución de los sucesivos problemas que se les van presentando en la resolución de su proyecto [9, 11, 12]. En LPyPL, para evaluar el trabajo continuo en el desarrollo del compilador, definimos un conjunto de actividades de seguimiento en el laboratorio:

- *Evaluación continua*, mediante los correspondientes entregables asociados con cada una de las partes del proyecto. Estos entregables solo constan del código de cada una de las partes del compilador.
- *Autoevaluación continua*, mediante la superación de los programas de prueba, proporcionados por los profesores, de cada una de las partes del compilador.
- *Tutorización*, que permite conocer la aportación en la evolución del compilador tanto del grupo en su conjunto como de cada uno de los integrantes del mismo.

5. Valoración de la experiencia

Esta experiencia la hemos llevado a cabo durante los cursos académicos que lleva impartiéndose la asignatura de LPyPL, y ha afectado tanto al trabajo de los profesores como al desempeño de los alumnos. Desde el punto de vista del profesorado, la valoración de la experiencia ha sido muy positiva ya que hemos podido constatar la facilidad y la considerable reducción del trabajo necesario para la elaboración anual de cada nuevo proyecto de compilación para LPyPL. Desde el punto de vista de los alumnos, el EDPC también ha supuesto una considerable ayuda como lo demuestran algunos indicadores que vamos a presentar a continuación.

5.1. Dificultad del proyecto

Nuestra primera preocupación está relacionada con la dificultad del proyecto y cómo esta pueda retraer la participación activa de los estudiantes en el desarrollo del proyecto. Para ello, en la primera fila del Cuadro 1 se presenta el número y porcentaje de alumnos que entregan su proyecto respecto a los alumnos matriculados, para cada uno de los cursos analizados. Como se puede observar, hay un elevado porcentaje de alumnos que han completado su proyecto (90 % en promedio). En las cursos anteriores a la implantación del EDPC ese porcentaje fue del 70 % aproximadamente.

En la actualidad, los alumnos que no lo consiguen, típicamente son alumnos que, por diversos motivos, dejan la asignatura y no se presentan a ningún acto de evaluación. Esta información nos indica, de una manera indirecta, que el esfuerzo requerido a los alumnos para la elaboración de su proyecto es adecuado. Ya que, si éste fuera excesivo, los alumnos tenderían a abandonarlo en las primeras fases del proyecto. Mientras que si cuentan con la ayuda necesaria, hemos observado que los alumnos se implican más y se mantienen activos hasta la finalización del proyecto.

En la segunda fila del Cuadro 1 se presenta el número y porcentaje de alumnos con su proyecto aprobado respecto a los alumnos presentados. Como puede observarse, este porcentaje también es muy alto (95 %). Estos excelentes resultados nos muestran que, no solo la dificultad del proyecto está razonablemente ajustada, sino que el proceso de evaluación también considera el trabajo continuo en el desarrollo del proyecto, ya que la inmensa mayoría de los alumnos permanecen activos y completan todos los actos de evaluación asociados con el proyecto.

5.2. Comprensión del proyecto

El siguiente punto de interés es valorar el grado de comprensión que el alumno individualmente tiene del

	2014-15	2015-16	2016-17	2017-18	2018-19	2019-20
Entregados	51 (91 %)	66 (85 %)	59 (88 %)	66 (87 %)	65 (96 %)	63 (97 %)
Aprobados	51 (100 %)	63 (95 %)	53 (90 %)	60 (91 %)	64 (98 %)	61 (97 %)

Cuadro 1: Número y porcentaje de alumnos en proyectos entregados y evaluados positivamente

	2014-15	2015-16	2016-17	2017-18	2018-19	2019-20
Presentados	51 (91 %)	66 (85 %)	59 (88 %)	66 (87 %)	65 (96 %)	60 (92 %)
Aprobados	47 (92 %)	58 (88 %)	48 (81 %)	54 (82 %)	49 (75 %)	46 (77 %)

Cuadro 2: Número y porcentaje de alumnos presentados y aprobados en el examen individual de prácticas.

proyecto. Para ello, en la primera fila del Cuadro 2, se presenta el número y porcentaje de alumnos presentados al examen individual de prácticas. Como se puede observar, hay un alto porcentaje de alumnos presentados (90 %, en promedio). Esta medida, de alguna manera, da cuenta del grado de implicación en el desarrollo del proyecto. Solo los alumnos que han participado activamente en el desarrollo del proyecto de su grupo suelen presentarse a este examen individual del proyecto.

En la segunda fila del Cuadro 2, se muestra el número y porcentaje de alumnos aprobados en dicho examen individual. Este porcentaje es relativamente alto (82 %, en promedio), lo que nos indica el alto grado de comprensión del proyecto que tiene el alumno.

5.3. Interés del proyecto

En este último punto vamos a verificar la importancia del proyecto en la comprensión de la asignatura en su totalidad, y no solo en la parte práctica. Para ello vamos a analizar la correlación que existe entre los alumnos que hacen el proyecto correctamente y las notas que sacan de teoría. En primer lugar, si comparamos las calificaciones del examen individual de prácticas con las del examen de teoría, se obtiene un coeficiente de correlación de Pearson de $\rho_{x,y} = 0,41$, lo que indica una clara correlación positiva entre los resultados del examen individual de prácticas y el de teoría.

Estudiando con más detalle esta relación, en el Cuadro 3 se recogen los resultados del total de alumnos de estos últimos cursos agrupados en dos columnas: los alumnos que aprobaron el examen individual de prácticas, ≈81 % del total, y los que lo suspendieron, ≈19 % del total. Como se puede apreciar en el Cuadro 3, los alumnos que aprobaron el examen individual de prácticas, mayoritariamente aprobaron también el examen de teoría. Por último, indicar que un 97 % de alumnos

	Ex. individual de prácticas	
Ex. teoría	Aprobados	Suspendidos
Aprobados	228 (75 %)	37 (51 %)
Suspendidos	74 (25 %)	36 (49 %)

Cuadro 3: Resultados del examen de teoría (número de alumnos y porcentaje) en relación con el resultado del examen de prácticas.

que aprobaron el examen individual de prácticas aprobaron también la asignatura. Mientras que los alumnos que suspendieron o no se presentaron al examen individual de prácticas suspendieron también la asignatura en un 49 %.

Esta correlación entre los resultados del examen individual de prácticas y del examen de teoría se observa también en las calificaciones medias de la nota de teoría y la nota final de la asignatura, en función de si se ha aprobado o no el examen individual de prácticas (Figura 1). Como puede observarse, el grupo de alumnos que aprobaron el examen individual de prácticas obtuvo en el examen de teoría una calificación media superior en 1,7 puntos a la del grupo de alumnos que suspendió el examen de prácticas. En el caso de la nota final de la asignatura, la diferencia es mucho mayor (3,5 puntos). Lo que nos indica la importancia del proyecto en la comprensión final de la asignatura.

6. Conclusiones

En este trabajo hemos presentado un entorno de desarrollo de proyectos de compiladores y un sistema de evaluación de los mismos. Este EDPC no solo ha permitido reducir significativamente el esfuerzo docente en la preparación de un nuevo proyecto de com-

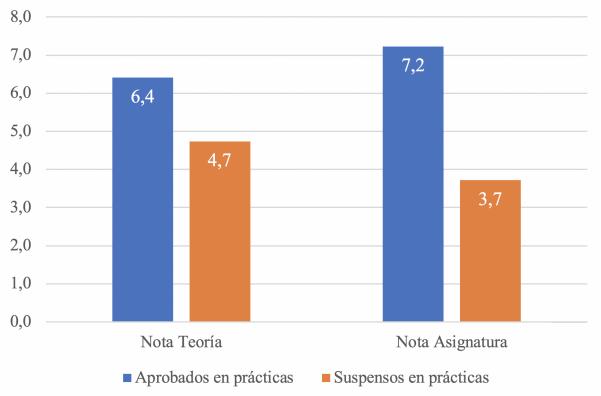

Figura 1: Nota media de teoría y final de la asignatura, de los alumnos aprobados y suspendidos en el examen individual de prácticas de los últimos cursos.

piladores, sino que también ha influido de manera positiva en los resultados obtenidos por nuestros estudiantes.

Del análisis de los resultados se puede destacar tres conclusiones: la dificultad y el esfuerzo exigido a los estudiantes es razonable, ya que un porcentaje muy elevado de los mismos completan el proyecto; el grado de compresión del proyecto es alto, ya que el porcentaje de estudiantes que se presentan al examen individual ronda el 90 % y el de los que finalmente lo aprueban el 80 %; y por último, la importancia del proyecto en la comprensión de la asignatura es muy destacada, como lo atestiguan las notas finales de los estudiantes aprobados en el examen individual de prácticas. Desde un punto de vista más cualitativo, también hemos detectado un aumento significativo en la motivación de los alumnos, una mejora en sus calificaciones y, en general, una mayor satisfacción en la experiencia docente por parte de los profesores.

No obstante, la puesta en marcha de este sistema de evaluación de proyectos también tiene sus limitaciones, dos son la mas importantes: se debe poder definir un modelo realista de usuario final, y esto no es siempre posible; y se precisa una importante infraestructura y una delicada logística para la adecuada gestión del examen individual de prácticas, y eso no es siempre factible.

Referencias

[1] Alfred Aho. *Teaching the compilers course*. ACM SIGCSE Bulletin, 40(4):6–8, 2008.

[2] Alfred Aho, Monica Lam, Ravi Sethi, and Jeffrey Ullman. *Compilers: Principles, Techniques, and Tools (2nd Edition)*. Addison Wesley, 2008.

[3] José Miguel Benedi, Lidia Moreno, Vicente Gisbert y Emilio Vivancos. *Procesadores de Lenguajes: una introducción a la fase de análisis*. Universitat Politècnica de València, 2008.

[4] José Miguel Benedí y Emilio Vivancos. *Un entorno para el desarrollo de proyectos en la enseñanza activa de un curso de compiladores*. Actas del II Congreso Nacional de Innovación Educativa y Docencia en Red, pp. 1–11, Valencia, 2016.

[5] José Miguel Benedí y Emilio Vivancos. *Una propuesta para la evaluación de proyectos en un curso de compiladores con una metodología de aprendizaje basada en proyectos*. Actas del V Congreso Nacional de Innovación Educativa y Docencia en Red, pp. 906–917, Valencia, 2019.

[6] Oscar Canovas, Imanol Usandizaga y Rafael Molina-Carmona. *Aprendizaje basado en proyectos entre asignaturas: tres experiencias, muchas preguntas y algunas respuestas*. Actas del XXV Jornadas de Enseñanza Universitaria de la Informática, pp. 79–86, Murcia, 2019.

[7] Keith Cooper and Linda Torczon. *Engineering a Compiler*. Morgan Kaufman, 2012.

[8] Daniel Gil-Pérez and Jaime Carrascosa-Alis. *Bringing pupils' learning closer to a scientific construction of knowledge: A permanent feature in innovations in science teaching*. Science Education, 78(3):301–315, 1994.

[9] John Larmer, John Mergendoller, and Suzie Boss. *Setting the Standard for Project Based Learning: A Proven Approach to Rigorous Classroom Instruction*. ASCD, 2015.

[10] Miguel Angel Martin. *Una experiencia docente basada en proyecto para la formación en competencias profesionales*. Actas del XXV Jornadas de Enseñanza Universitaria de la Informática, pp. 335–342, Murcia, 2019.

[11] Antonio Pérez-González, Julio Serrano Mira, Ignacio Peñarrocha y Emilio Pérez. *Un sistema para la evaluación del aprendizaje basado en proyectos*. Actas del XVI Congreso Universitario de Innovación Educativa en las Enseñanzas Técnicas, Cádiz, 2008.

[12] Harm-Jan Steenhuis and Lawrence Roland. *Project-Based Learning: How to Approach, Report, Present, and Learn From Course-Long Projects*. Business Expert Press, 2018.

[13] Emilio Vivancos, Lidia Moreno, José Miguel Benedí y Vicente Gisbert. *Metodología docente orientada a proyectos aplicada a las prácticas de compiladores*. Actas del IV Jornadas de Enseñanza Universitaria de la Informática, pp. 480–484, Principat d'Andorra, 1998.

[14] William M. Waite. *The compiler course in today's curriculum: Three strategies*. SIGCSE Bull., 38(1):87–91, 2006.