Actas XVIII JENUI 2012, Ciudad Real, 10-13 de julio 2012
I.S.B.N. 10: 84-615-7157-6 | I.S.B.N. 13:978-84-615-7157-4
Páginas 49-56

Aprendizaje basado en intereses: Una experiencia en la asignatura de Procesadores de Lenguajes

José Jesús Castro-Sánchez
Departamento de Tecnologías y Sistemas de Información
Universidad de Castilla-La Mancha
Paseo de la Universidad, 4
13071 Ciudad Real
josejesus.castro@uclm.es

Jesús Gallardo
Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
Ciudad Escolar, s/n
44003 Teruel
jesus.gallardo@unizar.es

Resumen

La materia de Procesadores de Lenguajes tradicionalmente ha estado presente como materia obligatoria u optativa en la mayoría de titulaciones relacionadas con la Informática. Esta materia trata sobre el desarrollo y la comprensión de los compiladores e intérpretes. En este trabajo se presenta el enfoque que se ha dado a dicha materia en el contexto EEES, y fundamentalmente a la parte práctica, para potenciar un aprendizaje significativo. La propuesta se basa en la realización de trabajos que impliquen dar solución a problemas que requieren del diseño y desarrollo de un lenguaje específico de un dominio así como de su procesador asociado en lugar de la aproximación clásica de diseñar un compilador para un subconjunto de un lenguaje de programación conocido. El trabajo explica el desarrollo de este planteamiento, así como algunos ejemplos significativos de prácticas desarrolladas en los años en los cuales se lleva aplicando.

Summary

The Language Processors subject is present as a core or optional subject in most degrees related with the field of Computer Science. Usually, this subject has dealt with the development and comprehension of compilers and interpreters. In this paper we introduce the approach we have followed in the subject, mainly in the practical part, to promote a significant learning. In order to achieve this goal, students have solved problems that require designing a domain-specific language and its related language processor instead of just designing a compiler for a subset of a given programming language. The paper explains the development of this approach in detail, and it also shows some outstanding works among those carried out by the students in this time.

Palabras clave

Procesadores de Lenguajes, Aprendizaje Significativo, Lenguajes específicos del dominio.

1. Introducción

Históricamente, *Procesadores de Lenguajes* (en ocasiones con el nombre de *Compiladores*) ha sido una materia presente como materia obligatoria u optativa en la mayoría de titulaciones relacionadas con la Informática. En España, fue recogida como una materia troncal de segundo ciclo con 9 créditos LRU en los estudios de Ingeniería Informática (Real Decreto 1459/1990, BOE de 20 de Noviembre de 1990). La remodelación de las titulaciones y los planes de estudio debidos a la adaptación de la titulación al Espacio Europeo de Educación Superior (EEES) ha traído consigo que esta materia no se pueda integrar tal y como estaba en los nuevos Grados en Ingeniería Informática. Esto se ha de aprovechar para replantearnos la materia y sus contenidos, de manera que se consiga aumentar el interés de los alumnos por la asignatura y mejorar el aprendizaje de los mismos, a la vez que volvemos a poner en valor la utilidad de la materia en la formación de nuestros alumnos.

Usualmente, esta materia se ha asociado al desarrollo y la comprensión de los compiladores e intérpretes, introduciendo al alumno en el estudio de los problemas que se plantean durante la construcción de los mismos y

de las técnicas que se aplican para su solución. Sin embargo, la utilidad de las técnicas de procesamiento del lenguaje va mucho más allá de su aplicación para la creación de un compilador o intérprete. En la actualidad existen numerosos problemas en los que para la construcción de una solución basada en software es necesario establecer un mecanismo de comunicación propio, con un lenguaje específico de dominio. En este contexto, se hace necesario que el software que soluciona el problema posea como interfaz de comunicación un procesador de dicho lenguaje específico del dominio.

Por otra parte, hay que destacar que el Espacio Europeo de Educación Superior (EEES) ha implicado un cambio de paradigma educativo en el que se pasa de una educación basada en la enseñanza a otra basada en el aprendizaje. Para que este cambio se realice con ciertas garantías de éxito es necesario un cambio en la forma de actuar del profesor en clase y la planificación de actividades por parte de este que permitan al alumno un aprendizaje autónomo y significativo.

Para que las actividades de aprendizaje planificadas permitan la consecución de los objetivos antes mencionados se requiere que los alumnos estén motivados para aprender. Esto conduce al Aprendizaje Significativo [2]. El aprendizaje significativo es aquel que proviene del interés del individuo, lo que se aprende es significativo si le sirve y utiliza porque es valorado para él como primordial y útil.

En este trabajo se presenta el enfoque que se ha dado a la asignatura de Procesadores de Lenguajes, y fundamentalmente a la parte práctica, para potenciar un aprendizaje significativo. Para ello en la parte práctica nos hemos apartado de la normalidad, que básicamente consiste en el diseño de un compilador para un subconjunto de un lenguaje de programación existente o para algún lenguaje de programación *ad hoc* basado en uno conocido. Nosotros proponemos la realización de unas prácticas que impliquen dar solución a problemas que requieren del diseño y desarrollo de un lenguaje específico de un dominio así como de su procesador asociado. En estas prácticas se otorga libertad a los alumnos para que desarrollen trabajos prácticos sobre cualquier temática.

Este tipo de prácticas que sugerimos permiten trabajar igualmente los contenidos del perfil recogidos en propuestas de currícula internacionales, p.e. ACM/IEEE-CS, que básicamente implican la adquisición de conocimiento y habilidades sobre gramáticas y lenguajes formales que permitan entender y afrontar la resolución de problemas que aparecen en el análisis y diseño de procesadores de lenguajes.

El resto del trabajo se estructurará de la siguiente manera: la siguiente sección tratará acerca de la asignatura de Procesadores de Lenguajes y del método tradicionalmente empleado para su enseñanza. En la Sección 3, se relacionará dicha asignatura con el contexto universitario actual del EEES. En la Sección 4, se presentará la experiencia fundamental propuesta en este trabajo, que es la planificación de las prácticas potenciando el aprendizaje significativo. A continuación, se mostrarán algunos ejemplos de trabajos prácticos realizados por alumnos siguiendo la filosofía propuesta en diferentes ámbitos. Finalmente, se presentan unas conclusiones obtenidas durante el trabajo expuesto.

2. Procesadores de Lenguajes en la Ingeniería Informática de la UCLM

El perfil *Procesadores de Lenguajes* recoge todas aquellas materias/conocimientos que tratan la teoría del diseño y procesamiento de lenguajes, así como la metodología, ideas fundamentales y técnicas usadas en el diseño de procesadores de lenguajes, traductores, compiladores e intérpretes. El objetivo principal de las materias de este perfil es que el alumno conozca la organización de los programas procesadores, traductores, compiladores e intérpretes de un lenguaje y que sea capaz de aplicar las técnicas necesarias para la resolución de los problemas que se puedan plantear a la hora de construir un programa de este tipo. Como ya se ha comentado, dado su carácter Troncal, todas las titulaciones oficiales de Ingeniero en Informática impartidas en España, debían poseer alguna/s asignatura/s cuyo contenido se ajustase al perfil de Procesadores de Lenguajes.

Actualmente, y con la reestructuración de las titulaciones para su adaptación al EEES, la situación ha cambiado y el perfil ha perdido el peso que se le había reconocido por Real Decreto.

El objetivo de este trabajo es plantear cómo debería ser reenfocada la asignatura, sobre todo en la parte práctica, para, sin apartarnos de los

UNIDAD	TEMAS	HORAS
U1: Introducción	Tema 1. Introducción a los traductores.	4
U2: Análisis Léxico	Tema 2. Análisis Léxico.	4
U3: Análisis Sintáctico	Tema 3. Introducción al Análisis Sintáctico. Tema 4. Análisis Sintáctico Descendente. Tema 5. Análisis Sintáctico Ascendente.	24
U4: Análisis Semántico y Generación de Código Intermedio	Tema 6. Traducción Dirigida por la Sintaxis. Tema 7. Comprobación de Tipos. Tema 8. Lenguajes Intermedios.	16
U5: Entorno de Ejecución	Tema 9. Organización y gestión de la memoria.	4
U6: Etapa de Síntesis	Tema 10. Generación de Código. Tema 11. Optimización de Código.	8

Tabla 1. Programa de teoría resumido y presencialidad de PL en la titulación Ingeniería Informática.

principios de la misma, conseguir un mayor interés por parte de los alumnos y aumentar el aprendizaje de sus contenidos.

En el plan de estudios de la Ingeniería Informática de la Universidad de Castilla-La Mancha (UCLM), la asignatura Procesadores de Lenguajes aparecía como una materia troncal de carácter anual en el cuarto curso con 9 créditos LRU (unas 90 horas de clase). En la fase de adaptación a los EEES a la asignatura se le asignó una carga de 7,5 créditos ECTS o lo que es lo mismo unas 187,5 horas de esfuerzo, de las cuales 75 horas correspondían a actividades presenciales (50 de teoría y 25 de laboratorio).

Los contenidos teóricos de dicha materia se estructuraban en seis unidades temáticas, que agrupaban uno o varios temas. Estas unidades estaban condicionadas por las etapas en el diseño y desarrollo de una aplicación de este tipo: introducción a los traductores; análisis léxico; análisis sintáctico; análisis semántico y generación de código; entorno de ejecución; etapa de síntesis. En la Tabla 1 se presentan las unidades junto con los temas que componen cada unidad.

En la primera unidad temática se presentaba la materia, se definía el proceso de traducción y se ofrecía una visión de las áreas donde se utilizan las ideas de la traducción. Se presentaban las distintas fases que conducen a la construcción de un procesador, traductor, compilador o intérprete.

En la segunda unidad nos ocupábamos del análisis léxico, estableciendo su relación con los lenguajes regulares, las expresiones regulares y las máquinas de estados finitos. Se establecían las conexiones con la materia Teoría de Autómatas y Lenguajes Formales.

En la tercera unidad se estudiaba la fase de análisis sintáctico, presentando inicialmente los conceptos más importantes que son necesarios para comprender los distintos métodos de análisis sintáctico. Después se estudiaban en detalle los métodos de análisis sintácticos descendentes (LL1) y ascendentes (SLR1, LR1 y LALR1). Esta parte tiene un gran peso en tiempo puesto que a la hora de construir un procesador de un lenguaje, la cuestión principal a afrontar es el diseño y construcción de un analizador sintáctico del mismo.

En la cuarta unidad temática se introducían las ideas principales de la traducción dirigida por la sintaxis, que eran empleadas tanto para dotar de significado y realizar comprobaciones semánticas (prestando especial atención a la verificación de tipos) como para realizar generación de código. Se estudiaban también distintos tipos de lenguaje intermedios que servían de nexo de unión entre las etapas de análisis y síntesis.

Dentro de la quinta unidad, se trataban en detalle temas relacionados con las implicaciones que tenía para un compilador el almacenamiento y manejo de la memoria del ordenador, durante una compilación y su gestión en tiempo de ejecución.

Para finalizar, en la unidad sexta, se abordaban los aspectos relacionados con la etapa de síntesis de la construcción de un compilador. Se estudiaban métodos de generación de código objeto y se introducían ideas sobre la necesidad de realizar optimizaciones sobre el código generado.

El programa práctico de la materia se dividía en seis prácticas, en las que se pretendía que el alumno aplicara los conocimientos y técnicas aprendidas en las clases de teoría para construir un compilador, es por ello que la organización de las mismas guardaba una fuerte dependencia con la organización de la parte teórica. Las prácticas de laboratorio eran las siguientes:

PRACTICA	HORAS
P1: Presentación del Lenguaje	2
P2: Analizadores Léxicos: Generadores automáticos	4
P3: Analizadores Sintácticos: Generadores automáticos	8
P4: Semántica: Atributos y reglas semánticas en los generadores automáticos	8
P5: Generando de Código	6
P6: Conclusiones	2

Tabla 2. Distribución de tiempos del programa de prácticas de PL en la Ingeniería Informática.

- P1.- Presentación del Lenguaje. Se presentaba el lenguaje con el que se iba a trabajar, y para el que se quería construir el procesador.
- P2.- Diseño e implementación del Analizador Léxico. Se introducían los generadores automáticos de analizadores léxicos, y se diseñaban y desarrollaban los analizadores léxicos.
- P3.- Diseño e implementación del Analizador Sintáctico. Aquí se introducían las herramientas para la generación automática de analizadores sintácticos, se diseñaba y construía el analizador sintáctico para el lenguaje, se conectaba con el analizador léxico construido en la práctica anterior y se incluían mecanismos de tratamiento de error al analizador desarrollado.
- P4.- Diseño e implementación del Analizador Semántico. Se explicaba cómo introducir atributos y reglas semánticas en los generadores automáticos de analizadores sintácticos, se diseñaba el analizador semántico del lenguaje y se introducían los atributos y reglas semánticas al analizador sintáctico.
- P5.- Generación de Código. Se presentaba la máquina para la que se quería generar el código y se diseñaba y generaba código para ella.
- P6.- Conclusiones. Era una sesión que se dedicaba a exponer los resultados y presentar las conclusiones obtenidas.

Esta forma de organizar la asignatura, tanto en la parte teórica como práctica ha provocado que la asignatura haya sido considerada por los alumnos como una de las más complicadas de la titulación. Si a esto le añadimos otros dos factores fundamentales: el tiempo disponible y la desmotivación de los alumnos (debida en gran parte a que muy pocos alumnos se dedicará de forma profesional al desarrollo de compiladores) nos lleva a buscar planteamientos alternativos sobre cómo enfocar la asignatura de manera que se capte el interés de los alumnos trabajando los contenidos del perfil y pensar en metodologías docentes alternativas para la enseñanza-aprendizaje de la materia.

3. La materia en el Grado en Ingeniería Informática de la UCLM

La adaptación de las titulaciones al EEES nos ha proporcionado el marco perfecto en el que replantear la materia. La asignatura Procesadores de Lenguajes ha sido incluida como una asignatura obligatoria de tecnología específica de carácter cuatrimestral en el cuarto curso, dentro de la intensificación de Computación en el Grado en Ingeniería Informática. La carga de trabajo total asignada a la materia es de 6 créditos ECTS o 150 horas de esfuerzo del alumno, lo cual supone una reducción con respecto a la situación de partida (7,5 créditos ECTS o 187,5 horas de esfuerzo) y hace imposible que se integre tal y como está actualmente. La cantidad de horas presenciales también se ha visto afectada, pasando de 90 horas en la Ingeniería Informática a 75 horas en la fase de adaptación para finalmente llegar a las 60 horas presenciales en el Grado.

Las competencias específicas de la materia son: Capacidad para conocer los fundamentos

UNIDAD	TEMAS	HORAS
U1: Introducción	Tema 1. Introducción	2
U2: Análisis Léxico	Tema 2. Análisis Léxico.	2
U3: Análisis Sintáctico	Tema 3. Introducción al Análisis Sintáctico. Tema 4. Análisis Sintáctico Descendente. Tema 5. Análisis Sintáctico Ascendente.	20
U4: Análisis Semántico y Generación de Código Intermedio	Tema 6. Traducción Dirigida por la Sintaxis: Significado y Comprobaciones	14
U5: Etapa de Síntesis	Tema 7. Etapa de Síntesis del Compilador	2

Tabla 3. Programa de teoría resumido y presencialidad de PL en la titulación Grado en Ingeniería Informática.

teóricos de los lenguajes de programación y las técnicas de procesamiento léxico, sintáctico y semántico asociadas, y saber aplicarlas para la creación, diseño y procesamiento de lenguajes.

En vista de las competencias y de los créditos establecidos para la materia se ha realizado una planificación de la materia de manera que el alumno gane confianza en la comprensión del proceso de compilación. Sin embargo, y como ya ha sido resaltado anteriormente, la utilidad de los procesadores de lenguajes va mucho más allá del desarrollo de los compiladores e intérpretes. A priori podría parecer que la implementación de dichos programas es el objetivo fundamental del procesamiento del lenguaje para un futuro profesional de la informática. Pero la cantidad de alumnos que van a acabar trabajando desarrollando compiladores puede ser pequeña en comparación con la cantidad de alumnos que en su vida profesional futura pueden llegar a utilizar técnicas de procesamiento del lenguaje para otros fines. De este modo, se hace patente la necesidad de enfocar la asignatura, y en particular la parte práctica, a otro tipo de sistemas que no sean sólo los compiladores o intérpretes de lenguajes de programación.

El enfoque que se va a seguir a la hora de planificar la materia de Procesadores de Lenguajes ha sido el usual, es decir basándonos en las distintas fases por las que hay que pasar para construir un programa de este tipo (ver Tabla 3). De este modo los contenidos teóricos se han estructurado en cinco unidades temáticas, que agrupan un total de 7 temas.

En la unidad primera se presentan en qué consiste un procesador de lenguajes y el tipo de aplicaciones para el que son útiles, se destaca el caso de los compiladores y se recuerdan conceptos sobre lenguajes y gramáticas de la asignatura de Teoría de Autómatas y Lenguajes Formales.

En la unidad segunda se hace hincapié en los conceptos relacionados con los lenguajes y gramáticas regulares, expresiones regulares, y autómatas finitos y su utilidad en el diseño de analizadores léxicos.

En la tercera y cuarta unidad no hay cambios significativos con respecto a las mismas unidades de la asignatura en la Ingeniería Informática. La diferencia en horas a que se dedicarán menos horas a la realización de ejercicios en clase.

La quinta y última unidad tiene como objetivo introducir al alumno en los aspectos y conceptos básicos relacionados con la etapa de síntesis en el proceso de compilación: Entorno de Ejecución, Generación de Código y Optimización de Código.

Como se puede observar en la Tabla 3, las fases de análisis sintáctico y semántico tendrán más peso específico, y coparán la mayor parte del temario.

En la siguiente sección se presentará cómo se ha planificado la parte del laboratorio dentro de este contexto y buscando sobre todo motivar al alumno en la asignatura por medio del aprendizaje significativo.

4. Planteamiento para la parte práctica

La parte práctica tradicionalmente ha implicado el desarrollo de un compilador o intérprete para algún subconjunto de un lenguaje de programación existente o en uno diseñado para ese fin. Por ejemplo, en el curso académico 2011/12, las prácticas de la asignatura en la Universidad de Huelva y en la Universidad Politécnica de Madrid han seguido este patrón. En el primer caso se trabaja con un lenguaje basado en C, mientras que en el segundo se propone

hacerlo sobre un subconjunto del lenguaje JavaScript. Para ello, se suelen utilizar herramientas como *Lex* o *JFlex* para el desarrollo del analizador léxico y *Yacc* o *CUP* para el desarrollo del analizador sintáctico/semántico y la generación de código, aunque en algunos casos se exige el desarrollo desde cero de los programas procesadores de lenguajes, sin utilizar herramientas de generación automática. En cuanto a la parte de generación de código, hay quien propone el uso de herramientas como ANTLR [5]. También existen ejemplos de investigación en la cual se han planteado lenguajes específicos para la enseñanza de los compiladores, como COOL [1] y MINIML [3], que se basan en lenguajes de propósito general, KLX [4], que es un lenguaje para especificación de gráficos, o CHIRP [6], que es un lenguaje de alto nivel que genera código que se ejecuta en un cierto tipo de robot. Pero en general, todo este enfoque mantiene la idea de trabajar sobre compiladores o intérpretes de lenguajes de programación.

Así pues, y teniendo en cuenta lo ya comentado anteriormente acerca de las dificultades encontradas por los alumnos al afrontar la asignatura, nuestra idea al plantear la parte práctica de la asignatura ha sido alejarnos de los planteamientos tradicionales para potenciar el aprendizaje significativo del alumno.

De este modo, el trabajo que deben realizar los alumnos en la parte práctica de la asignatura es resolver un problema en el cual sea necesario el diseño de un lenguaje y de su procesador asociado, así como la realización de alguna acción en base a la entrada. El problema y el lenguaje son propuestos por los alumnos, pero teniendo en cuenta que hay una serie de características que debe cumplir el lenguaje, y que son las siguientes:

- Debe ser un lenguaje libre de contexto.
- Debe ser un lenguaje infinito.
- El lenguaje debe incluir la declaración y posterior uso de variables.
- Dichas variables deben ser de al menos dos tipos de datos.
- Habrá bloques anidados, con uno global que actúe como contenedor del resto.
- La declaración de variables podrá hacerse en cualquier bloque, incluido el global.
- Poseerá la sentencia de asignación para todos los tipos de datos.
- No debe ser sensible a las mayúsculas y minúsculas. Es decir, dos cadenas son la misma independientemente de las letras mayúsculas o minúsculas que posea.

Dicha libertad a la hora de elegir el problema y el lenguaje permite a los alumnos realizar algún trabajo que pueda ser útil para su investigación o para su trabajo, en el caso que estén implicados en alguna de dichas actividades, o en general algún tema que les resulte atractivo y que les motive a la hora de llevarlo a cabo. A los alumnos se les presenta el concepto de lenguaje específico del dominio, pero sin demasiada profundidad. Simplemente es necesario que conozcan que van a diseñar un lenguaje pensado para un campo y que, por tanto, se engloba dentro de ese tipo de lenguajes.

Para que los alumnos no se encuentren perdidos a la hora de elegir el tema de su trabajo práctico, la primera sesión de prácticas se dedica a que alumnos que realizaron prácticas destacadas el curso anterior y que permanecen en la Escuela las expongan a los alumnos que deben elegir ahora su temática. De este modo, los alumnos se hacen una idea del tipo de trabajos que pueden proponer. En la siguiente sección se comentarán algunas de esas prácticas destacadas. Así, un posible inconveniente que se puede encontrar a este enfoque con respecto a uno más dirigido, en el cual el lenguaje sobre el que se trabaja es fijo, es que aquellos alumnos que no puedan asistir a clase y tener una cierta interacción con el profesor pueden tener problemas para la elección del problema y el inicio del trabajo.

Los alumnos realizan los trabajos prácticos de forma incremental, pasando sucesivamente por las etapas de desarrollo del analizador léxico, del analizador sintáctico, del analizador semántico y de la parte de generación de código. Así, las sesiones de prácticas en el laboratorio se dedican sucesivamente a cada una de estas partes del procesador de lenguajes que los alumnos deben desarrollar. Las sesiones de prácticas acaban con la exposición voluntaria de prácticas para aquellos alumnos que deseen optar a la máxima nota posible. El trabajo práctico supone 2 puntos sobre los 10 de la nota final de la asignatura, pero en

casos excepcionales, prácticas muy destacadas han sido valoradas con hasta 3 puntos.

SESIONES	HORAS
Sesiones dedicadas al problema y al lenguaje	4
Sesiones dedicadas al analizador léxico	4
Sesiones dedicadas al analizador sintáctico	4
Sesiones dedicadas al analizador semántico	4
Sesiones dedicadas a la generación de código	2
Exposición voluntaria de prácticas	2

Tabla 4. Distribución de tiempos de las sesiones de prácticas en la asignatura de Procesadores de Lenguajes del Grado en Ingeniería Informática.

Para la realización del procesador se puede utilizar cualquier lenguaje de programación y las herramientas de generación de analizadores léxicos y sintácticos que se consideren oportunas. Las explicaciones y ejemplos que se hacen a lo largo de la asignatura se hacen utilizando las herramientas *JFlex* y *CUP* por trabajar estas sobre *Java*, que es el lenguaje vehicular de la titulación.

El entregable que deben elaborar los alumnos y presentar al final del curso incluye, por un lado, el código generado, y por el otro, la documentación del trabajo. El código entregado no es sólo el código final del trabajo, sino que se pide el estado del trabajo después de terminar cada una de las fases del procesador de lenguajes. En cuanto a la documentación, cada fase tiene sus propios requisitos en ese sentido. Por ejemplo, en cuanto a la elección del problema y lenguaje y la definición del mismo se pide, entre otras cosas, la explicación de dichos aspectos mediante diagramas en forma de T, mientras que para la parte de análisis léxico lo fundamental es la tabla con los *tokens* identificados en el lenguaje con el que se va a trabajar.

5. Ejemplos de prácticas realizadas

Durante el tiempo que lleva aplicándose este enfoque al desarrollo de la parte práctica en la asignatura, han sido varias las prácticas que han destacado por su calidad, su utilidad o ambas cosas. Inicialmente, cabe destacar una serie de prácticas que posteriormente han encontrado una utilidad como material de apoyo de la propia asignatura o de otras asignaturas de la titulación. Se trata de *Selfa* (y su continuación *Selfa-Pro*) y de *Proletool* en sus versiones 1 y 2. *Selfa*[1] es una herramienta *online* que resuelve ejercicios sobre autómatas finitos, expresiones regulares, gramáticas regulares, autómatas con pila y gramáticas libres de contexto. Además permite realizar simulaciones sobre los autómatas finitos y los autómatas con pila. De este modo, resulta de gran ayuda en la impartición de las clases de la asignatura de Teoría de Autómatas y Lenguajes Formales y similares, y se ha utilizado en la propia Escuela Superior de Informática de la UCLM como material de apoyo.

Por su parte, Proletool[2] es también una herramienta *online* que resuelve ejercicios sobre generación de analizadores léxicos y analizadores sintácticos descendentes y ascendentes, mostrando las tablas de análisis generadas, así como la información que permite construirlas. Además, permite realizar simulaciones de los analizadores construidos. Así, es una herramienta muy útil para la asignatura de Procesadores de Lenguajes tanto para que los profesores enseñen las distintas técnicas de análisis como para que los alumnos practiquen los ejercicios planteados.

Otro ejemplo de práctica con fines educativos que está encontrando una utilidad más allá de la asignatura es la práctica Electronics[3], que se plantea como un sistema para la enseñanza de los fundamentos de los circuitos electrónicos y de la resolución de problemas utilizando la teoría de mallas. El programa permite definir circuitos y mallas con componentes electrónicos y posteriormente plantear problemas y posibles soluciones a los mismos. Esta práctica implicó la definición, por un lado, de un lenguaje compilado para la definición de los circuitos, y por el otro, de un lenguaje interpretado para las preguntas.

Otras prácticas se han orientado más a aspectos relacionados con el ocio. Un ejemplo

[1] http://portal.esi.uclm.es/selfa/

[2] http://portal.esi.uclm.es/proletool/

[3] http://www.esi.uclm.es/jjcastro/electronics

sería la práctica GDL, en la cual se definía un lenguaje para diseñar niveles de un videojuego de plataformas. El procesador de lenguajes asociado a dicho lenguaje se integraba en el desarrollo de ese pequeño videojuego, desarrollado también por los alumnos. Así, utilizando el lenguaje se definían las distintas características del nivel (obstáculos, enemigos, etc.), que luego tenían su reflejo en el videojuego generado.

También resulta interesante en este sentido la práctica YakiMusic, en la cual se planteaba un lenguaje de definición de composiciones musicales en texto plano con una serie de características que le hacían cumplir los requisitos exigidos en la práctica, como por ejemplo uso de ritmos predefinidos, creación de sonidos como variables, etc. Posteriormente, el procesador de dicho lenguaje desarrollado por los alumnos permitía que la melodía compuesta se exportase a formato MIDI, con lo cual esta podía escucharse en cualquier reproductor que soportase dicho formato.

Finalmente, una práctica con una aproximación intermedia entre lo educativo y lo lúdico es un generador de *circuitos* de pasatiempos cuyo lenguaje asociado permite definir distintos tipos de pasatiempos (crucigramas, sopas de letras, etc.) a partir de una base de conocimiento. Posteriormente, el lenguaje proporciona también la posibilidad de crear circuitos de pasatiempos de manera más o menos adaptativa, de forma que dependiendo del resultado de un pasatiempo (tiempo invertido, número de intentos, etc.) el siguiente pasatiempo a ejecutar es uno u otro. Esto permite que, además del fin meramente lúdico, la práctica pueda tener una aplicación educativa al poder utilizarse como herramienta de enseñanza.

De esta forma, puede verse cómo los alumnos abarcan una gran variedad de tipos de problemas con las prácticas que vienen realizando, la mayor parte de las veces motivados por temas que ellos eligen y que les resultan interesantes.

6. Conclusiones

En este trabajo se ha presentado un enfoque para el desarrollo de la asignatura Procesadores de Lenguajes, y en concreto para su parte práctica, basado en el aprendizaje significativo. Lo que se ha planteado es que los alumnos sean capaces de proponer prácticas sobre cualquier temática que impliquen diseñar un lenguaje y su procesador. De este modo, los alumnos se ven más motivados por estar implicados en la elección del tema y además se les hace ver que la utilidad de los procesadores de lenguajes va más allá del desarrollo de compiladores o intérpretes para lenguajes de programación.

La recepción por parte de los alumnos de este enfoque ha sido positiva, y en general los alumnos están satisfechos con la manera de llevar las prácticas, a pesar de que no se disponga de una comparativa formal entre una manera y otra de llevarlas a cabo. Además, mediante este enfoque se han desarrollado prácticas bastante interesantes que en ocasiones han sido puntos de partida para trabajos fin de grado o se han integrado en proyectos de investigación. En cuanto a puntos débiles, el principal que se puede identificar es el problema que se puede plantear si los alumnos no son capaces de encontrar un tema que les resulte atractivo para realizar el trabajo práctico. En este sentido, una idea para el futuro puede ser desarrollar una lista de temas de prácticas entre los cuales puedan elegir los alumnos que no sean capaces de proponer un tema ellos mismos.

Referencias

[1] Aiken, A., *Cool: A portable project for teaching compiler construction*. ACM SIGPLAN Notices, 31(7):19–24.ACM, 1996.

[2] Ausubel, D. , *The Psychology of Meaningful Verbal Learning*. Grune & Stratton, 1963.

[3] Baldwin, D., *A compiler for teaching about compilers*. Proceedings of SIGCSE 2003. ACM, 2003.

[4] Ruckert, M., *Teaching compiler construction and language design: making the case for unusual compiler projects with postscript as the target language*. Proceedings of SIGCSE 2007. ACM, 2007.

[5] Troyano, J.A. et al., *Prácticas de la asignatura Procesadores de Lenguaje con la herramienta ANTLR*. XI Jornadas de Enseñanza Universitaria de la Informática (JENUI 2005). Thomson, 2005.

[6] Xu, L., Martin, F.G., *Chirp on crickets: teaching compilers using an embedded robot controller*. Proceedings of SIGCSE 2006. ACM, 2006.