Una herramienta para la enseñanza y aprendizaje de la asignatura Procesadores de Lenguajes

Pedro Antonio Santos, José Jesús Castro-Schez
Dpto. Tecnologías y Sistemas de Información
Escuela Superior de Informática
Universidad de Castilla-La Mancha
Pº Universidad, 4 - 13071 Ciudad Real
JoseJesus.Castro@uclm.es

Resumen

Uno de los problemas o preocupaciones que existen en el contexto universitario actual, manifestado en la Ley Orgánica de Universidades (Ley Orgánica 6/2001 de 21 de diciembre), es la búsqueda de la excelencia en la calidad de la formación impartida. Para hacer frente a este objetivo, nos encontramos con el problema de la elaboración, diseño y desarrollo de materiales y aplicaciones que de algún modo ayuden a lograr la mejora en la formación. En este trabajo se presentan las experiencias realizadas en este sentido en la asignatura Procesadores de Lenguajes de la Escuela Superior de Informática (ESI) de la Universidad de Castilla-La Mancha (UCLM).

1. Motivación

La finalidad de este trabajo es doble, por un lado se pretende hacer hincapié en la necesidad de desarrollar herramientas docentes que puedan ser empleadas en la enseñanza y estudio de una materia para buscar la excelencia y la mejora de la calidad en las instituciones universitarias dentro del campo de la informática y por otro presentar las experiencias realizadas en este sentido dentro de la asignatura Procesadores de Lenguajes (PL) de la ESI de la UCLM.

La calidad de la formación impartida en la asignatura de PL podría verse afectada positivamente con el uso de herramientas que se puedan emplear como apoyo tanto a la enseñanza como al estudio de las técnicas de análisis sintáctico, tema de importancia en el estudio de la materia y que causa confusión en los alumnos. En la actualidad encontramos que se han propuesto, diseñado y usado una serie de herramientas con este objetivo, como ejemplo citar ANAGRA[2], JavaPars[5], JFLAP[6], SEFALAS [9] y Caburé [10].

En este artículo se va a presentar una herramienta que permite la introducción de una o varias gramáticas mediante el uso de un determinado lenguaje y genera un analizador sintáctico LL1, SLR1, LR1 o LALR1 según se le indique. También se permite la simulación de un análisis de una frase de entrada por el analizador generado.

En el diseño de la herramienta propuesta se ha intentado incorporar las características más destacadas de cada una de las herramientas antes mencionadas (ejecución vía web[5], entornos fáciles de usar [2][6][9], gráfica[2][6], completa[2][9]) añadiendo nuevas características tales como la posibilidad de ejecutarla sin necesidad de instalación (utilizando para ello una arquitectura cliente/servidor), recoger datos de uso por parte de los usuarios (utilizando bases de datos) y generar varios analizadores en una sola ejecución.

Otra característica importante es su diseño basado en un procesador de un lenguaje que permite escribir gramáticas. La herramienta sirve de ejemplo de diseño de una solución a un problema real basado en el uso de un procesador de un lenguaje, lo cual puede motivar a los estudiantes viendo ejemplos reales de aplicaciones.

Las ideas y conclusiones que se presentan en este trabajo son fruto del Proyecto Fin de Carrera realizado para la asignatura Procesadores de Lenguajes [8] y gracias a un programa de becas de colaboración para participar en el desarrollo y mejora de acciones relacionadas con el funcionamiento de la ESI en la UCLM.

El hecho de que la herramienta haya sido realizada por un compañero anima al resto de alumnos a “conseguir” lo que otros ya han logrado

y a aumentar su capacidad crítica y creativa en la solución del problema propuesto.

Este trabajo está estructurado de la siguiente manera: En la siguiente sección se presenta y analizará el problema que se pretende resolver. En la Sección 3 se estudiará como se ha diseñado y desarrollado la aplicación *Proletool*. A continuación, se mostrarán unos datos sobre el uso de la herramienta en los cursos académicos 04/05 y 05/06 (hasta el momento) y para finalizar se mostrarán las conclusiones y futuros trabajos.

2. Problemática a la que se hace frente

El objetivo fundamental del perfil de *Procesadores de Lenguajes* (PL) es introducir al alumno en el estudio de los problemas que se plantean durante la construcción de procesadores, traductores, compiladores e intérpretes de los lenguajes de programación y la aplicación de técnicas para su solución.

Los contenidos del perfil objeto de este trabajo están recogidos en las directrices establecidas por el Consejo Superior de Universidades para los planes de estudio de las Ingenierías Informáticas, dentro de la troncalidad obligatoria de segundo ciclo, y en diversas propuestas de currícula internacionales, p.e. ACM/IEEE-CS [1]. Así mismo, en el ambiente en el que se encuentra inmersa la universidad española de adaptación al Espacio Europeo de Educación Superior (EEES), también encontramos en el Libro Blanco de la Ingeniería Informática dentro de la propuesta para contenidos troncales del Título de Grado en Ingeniería Informática los contenidos de este perfil [3]. Con esta reflexión se quiere destacar la importancia que tiene y tendrá esta asignatura en el diseño de los futuros planes de estudio y la necesidad de continuar trabajando en la mejora de la calidad docente en este perfil.

En las asignaturas del perfil se requieren fundamentalmente conocimientos 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.

Los contenidos de la asignatura por lo general se estructuran en seis unidades temáticas, que agrupan uno o varios temas. Estas unidades están 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.

A la hora de construir un procesador de un lenguaje, saber como afrontar el diseño y construcción de un analizador sintáctico del mismo, es un tema principal. Esto es debido a que muchas otras fases posteriores a ella dependen del enfoque que se la haya dado a la misma. Esta importancia puede comprobarse observando las planificaciones docentes de asignaturas del perfil. La mayoría de ellas dedican gran parte del tiempo al análisis sintáctico, estudiándose en detalle los métodos de análisis descendentes (LL1) y ascendentes (SLR1, LR1 y LALR1).

Para aprender estos métodos de forma sólida y eficiente, y adquirir la habilidad necesaria para su aplicación es muy importante la realización y corrección de un gran número de ejercicios prácticos. Sin embargo esto no es posible hacerlo de manera colectiva en clase, debido a la extensión del temario y al tiempo limitado de las mismas, y más en el contexto en el que nos movemos para la adaptación al EEES y hacia el sistema de créditos europeos de transferencia (ECTS). Este sistema *pretende reflejar el esfuerzo real requerido por el estudiante para conseguir una serie de objetivos* [4] valorando el trabajo realizado por este en el estudio de la materia PL. Por ello, se hace necesario el establecimiento de medidas encaminadas a proporcionar a los alumnos una batería de ejercicios junto con sus soluciones de manera que les permita asimilar y aprender las técnicas de análisis sintáctico de una manera individual y autónoma.

Como alternativa a esta solución, en este trabajo se propone el diseño y desarrollo de una aplicación, a la que se ha nombrado *Proletool*, para su uso en la enseñanza y aprendizaje de las técnicas de análisis sintáctico LL1, SLR1, LR1 y LALR1.

3. Diseño de la herramienta

A la hora de diseñar la aplicación *Proletool* se han tenido en cuenta los servicios que ofrecen una serie de herramientas que existen en la actualidad

para la enseñanza y el estudio de las técnicas de análisis sintáctico (Proyecto SEPa (concretamente la aplicación Caburé! actualmente en desarrollo) [10], JFLAP[6], SEFALAS[9], ANAGRA[2], JavaPars[5]).

La primera reflexión que cabe hacerse después de analizar estas herramientas y antes de comenzar el diseño de la aplicación es la existencia de dos niveles o tipos de usuarios (docente y alumno) y que cada uno de estos tendrá una importancia en el éxito de la aplicación. Si queremos que la aplicación desarrollada sea útil, debe serlo en ambos niveles por lo que no se puede ignorar las necesidades, los deseos y las prioridades de los usuarios en cada uno de estos niveles. Por todo esto, también se han considerado las necesidades y expectativas de los alumnos del curso 04/05 y profesorado actual y pasado de PL de la ESI en la UCLM, que han mostrado interés por el problema y por participar en el diseño de la aplicación y que son los potenciales usuarios de la misma.

Después de un análisis detallado se alcanzaron las siguientes conclusiones acerca de que funcionalidad que debía ofrecer la herramienta desde un punto de vista común: Aceptar ejercicios sobre aplicación de técnicas de análisis sintáctico a gramáticas y generar soluciones a los mismos. Se quiere hacer especial hincapié en el plural de "ejercicios", cuyas ventajas se analizarán más adelante en la Sección 3.3. También debe permitir la simulación de un análisis sintáctico de una cadena dada como entrada para cualquiera de los ejercicios introducidos.

Dentro de las características deseables en la aplicación para ambos niveles se señalaron las siguientes:

- Disponibilidad, se debe hacer todo lo posible por facilitar el uso de la herramienta. Para ello se ha de establecer un procedimiento de instalación lo más fácil posible.
- Actualizable, se necesita una instalación que permita una fácil actualización del software, ya que se pretende que sea una aplicación en continúa mejora. La aplicación se va a usar en las presentaciones de clase, laboratorios y ordenadores personales de los alumnos, un cambio en la aplicación debe ser fácil de trasladar a cada uno de estos dispositivos si no puede llevar al rechazo en su uso en ambos niveles.
- Accesibilidad, debe presentar una interfaz que facilite su uso. La entrada de ejercicios debe ser lo más sencilla posible y la generación de soluciones a los mismos debe ser dada en un formato sencillo y amigable.
- Correcta, debe proporcionar una información precisa y correcta.
- Integrada, aparte del resultado final debe proporcionar toda la información que sea necesaria para que se comprendan las soluciones ofrecidas.
- Plataforma libre, se debe poder ejecutar en cualquier plataforma.

Desde el punto de vista del docente la herramienta debe tener además la siguiente funcionalidad: Establecer que es lo que se puede hacer con la herramienta y a quién. Además debe permitir el seguimiento de la actividad que lleva a cabo cada uno de los alumnos. Esta información le permite hacer un seguimiento del interés de los alumnos por la asignatura. Para los docentes sería deseable que la aplicación tuviera las siguientes características generales:

- Ejemplar, es decir la aplicación misma debe ser un ejemplo para futuros alumnos sobre como emplear los conocimientos teóricos y prácticos de la asignatura en la resolución de un problema real.
- Agradable, debe permitir el acceso a la aplicación de una forma rápida, sencilla y cómoda para facilitar su uso en cualquier ordenador ya sea propio, de clase (tanto teórica como práctica) o de un alumno.
- Funcional, debe presentar una interfaz que permita acceder a los elementos importantes de la aplicación e interactuar con los contenidos presentados en la solución, facilitando la navegación en ellos.
- Gráfica, que muestre los autómatas que reconocen prefijos viables en los análisis ascendentes (SLR1, LR1, LALR1).
- Software Libre, se requiere que sea desarrollado bajo licencia GPL con el objetivo de que permita la continua mejora de la misma por otros docentes o alumnos de futuros cursos de esta universidad u otras.

Los alumnos no añadieron ninguna funcionalidad nueva a las ya mencionadas, sin embargo si

especificaron las siguientes características como expectativas propias sobre la herramienta:

- Eficiente, debe realizar las operaciones de una manera rápida y con efectos inmediatos.
- Amigable y cercana, se requiere que exista un sistema de ayuda y consulta sobre la aplicación así como un sistema de asistencia o soporte técnico para resolver dudas.
- Uniforme, que se utilice la misma notación y terminología que se emplea en clase.
- Memoria, que le permita guardar el trabajo realizado para su análisis y estudio posterior así como para su reconocimiento por parte del profesorado.

Una vez definidos los requisitos de la herramienta a desarrollar, en las siguientes secciones se detallarán las decisiones de diseño que han conducido a la aplicación final.

3.1. Especificación de la entrada y salida de la herramienta

Como ya ha sido destacado en las secciones anteriores el objetivo principal de la herramienta es tomar un conjunto de ejercicios de aplicación de técnicas de análisis sintáctico a unas gramáticas y obtener la solución a los mismos. Veamos en que consisten los ejercicios.

Un ejercicio es una especificación de una gramática libre de contexto, **`G=(V,T,P,S)`** donde **`V`** es el conjunto de símbolos no terminales, **`T`** el conjunto de símbolos terminales, **`P`** el conjunto de producciones de la gramática y **`S`** es el símbolo inicial de la gramática, a la que se le asocia uno o varios tipos de análisis sintáctico (**`LL1`**, **`SLR1`**, **`LR1`** o **`LALR1`**).

La solución consiste en la generación de la tabla de análisis que corresponda en función del tipo de análisis asociado a la gramática. Además de las tablas de análisis se generará:

- La información adicional utilizada en el proceso de resolución del ejercicio o lo que es lo mismo de construcción de la tabla a partir de la gramática.
- Los mensajes importantes del proceso de generación de la solución.
- Si no es posible la resolución del ejercicio, se muestra información sobre los conflictos que aparecen (entre iniciales, iniciales-seguidores, o de desplazamiento-reducción, reducción-reducción). Además, se indicará claramente la posición de la tabla de análisis donde ha ocurrido el conflicto.

3.2. Solución basada en un procesador de lenguajes

En los requisitos de la herramienta se hizo especial hincapié por parte del profesorado en que el diseño ya fuera un ejemplo para futuros alumnos del uso de los conocimientos de la asignatura en la resolución de problemas reales. Es por esto por lo que el diseño está basado en un procesador de lenguajes.

Se ha creado un lenguaje específico para la introducción, usando cualquier editor de textos, de los ejercicios. Para procesar dicho lenguaje se ha construido un procesador que comprueba si es correcto sintáctica y semánticamente conforme a las reglas del lenguaje de entrada creado, a la vez que recopila la información necesaria para la generación de la tabla de análisis.

Otro de los requisitos de la herramienta era que debía facilitar su instalación, actualización, disponibilidad y uso. Además, se debía crear algún sistema para recoger el trabajo realizado por los alumnos con la herramienta, así como para establecer las operaciones que se podían hacer con la misma. Una forma de cumplir todos estos requisitos es mediante una aplicación centralizada con una arquitectura cliente/servidor ejecutable vía Internet. Con ello se consigue:

- Que la aplicación no necesite instalaciones locales, lo único que se necesita para usarla es un navegador web. Se facilita de este modo su disponibilidad y uso en cualquier ordenador que tenga conexión a Internet. También se soluciona el problema de la actualización ya que bastaría con actualizar la aplicación en el servidor.
- Que toda la carga de trabajo caiga en el servidor, haciendo así al cliente muy ligero.
- Que todo el trabajo realizado por los alumnos se pueda memorizar, al ser un sistema centralizado desde el servidor se pueden registrar todas las operaciones de los usuarios. Para ello la aplicación contará con un acceso autenticado, permitiendo también el uso anónimo de la herramienta (esto no lo permiten las herramientas antes mencionadas).

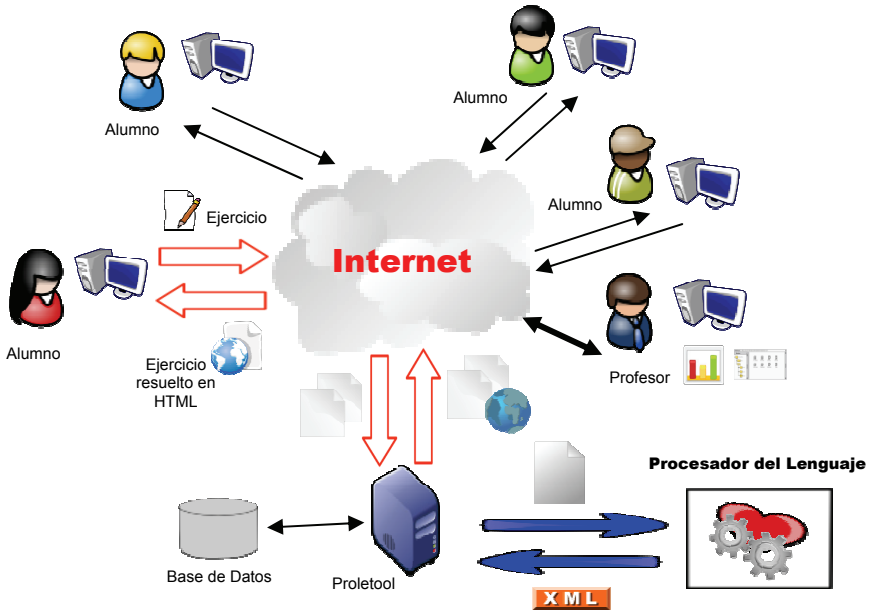

Figura 1. Arquitectura centralizada cliente/servidor

En la Figura 1 puede observarse la arquitectura de la herramienta cuyo núcleo o corazón es un procesador de lenguajes, así como su funcionamiento. Lo primero que hace el usuario es conectarse con su cliente web al servidor y enviarle los ejercicios a resolver. Una vez le han llegado al servidor, los scripts del servidor llaman al procesador del lenguaje, que es el que analiza el texto de entrada ve si es correcto, obtiene la información relevante, la asimila y genera la solución en un lenguaje que permita la visualización, navegación por la misma y su almacenamiento para un estudio posterior. Este lenguaje será HTML. El profesor además tiene capacidades de administración y generación de informes.

Para facilitar el uso de la aplicación por parte de alumnos y profesorado, la entrada de ejercicios se va a poder realizar de dos formas:

- A partir de archivos previamente escritos usando un editor de textos, lo cual permite trabajar fuera de línea y aprovechar los tiempos de conexión a Internet para resolver.
- A partir de un espacio en la propia herramienta reservado para la escritura de ejercicios, para facilitar la interacción con la herramienta.

3.3. El lenguaje de entrada y salida

El diseño del lenguaje de entrada se ha realizado con mucho detenimiento, ya que si no se define correctamente repercutirá negativamente en la utilidad de la herramienta. El lenguaje diseñado tiene las siguientes características:

- Se parece a los lenguajes existentes empleados para expresar gramáticas libres de contexto (los que usan las herramientas CUP, YACC, Bison). De esta forma, con la experiencia que ya tenga el usuario en este tipo de lenguajes, le será más fácil aprender a utilizarlo.
- Permite escribir una o varias gramáticas a la vez. Con esto se consigue que un alumno pueda escribir en un único archivo todos los ejercicios que quiere realizar y conectarse una única vez al servidor para obtener la resolución (muy útil cuando el tiempo de conexión a Internet es limitado). Además hay veces en las que varias gramáticas están relacionadas y se quiere mostrar la relación entre las soluciones obtenidas.

- El lenguaje permite asociar a cada gramática el tipo de análisis que se quiere hacer sobre ella, pudiendo ser este uno o varios de los tipos permitidos (LL1, SLR1, LR1, LALR1).
- Existen dos ámbitos uno global donde se pueden hacer declaraciones comunes y utilizables por varias gramáticas y otro local de aplicación a una gramática particular. Los símbolos que se pueden declarar globalmente son los Terminales y No terminales. Además, se pueden definir grupos de producciones identificadas por un nombre, para usar estas producciones posteriormente en una gramática basta con poner dentro de su definición local la llamada a este conjunto (poniendo el identificador).
- Se permite la definición de un análisis global, que se realizará para todas las gramáticas dadas a la entrada.

De lo dicho anteriormente, se puede deducir que en el lenguaje diseñado hay dos partes, una primera zona opcional, que determinará el ámbito global y una segunda parte, que corresponde a los ámbitos locales, donde aparecerán las definiciones de cada una de las gramáticas, ver Figura 2.

A continuación se describe su sintaxis empleando notación EBNF:

```
RELACION   ::= GLOBAL LOCAL | LOCAL
GLOBAL     ::= global '{' C_GLOBAL '}'
C_GLOBAL   ::= [DEC_ANA] [DEC_SIM] [DEC_CJTOP]
DEC_ANA    ::= analysis T_ANA [',' T_ANA][',' T_ANA][',' T_ANA] ';'
T_ANA      ::= LL1|SLR1|LR1|LALR1
DEC_SIM    ::= D_SIMBOLO {D_SIMBOLO}
D_SIMBOLO::= TIPO L_SIMBS
TIPO       ::= terminal | nonterminal
L_SIMBS    ::= ID {',' ID} ';'
DEC_CJOP   ::= CJO_PROD {CJO_PROD}
CJO_PROD   ::= ID '=' '{' L_PROD '}'
L_PROD     ::= PROD {PROD}
PROD       ::= ID ':=' (PRT_DCHA | ';')
PRT_DCHA   ::= DCHA { '|' DCHA} (';'| '|' ';')
DCHA       ::= ID {ID}
LOCAL      ::= GRAM {GRAM}
GRAM       ::= grammar ID '{' C_LOCAL '}'
C_LOCAL    ::= [DEC_ANA] [DEC_SIM] [DEC_PROD]
DEC_PROD ::= (PROD | ID) {PROD | ID';'}
```

El símbolo ID corresponde al lexema identificador cuyo patrón de construcción es el siguiente:

$$ID = [\text{A-Za-z}][\_\text{A-Za-z0-9}]^* \qquad (1)$$

En las partes derechas de las producciones también se permiten la declaración de caracteres especiales (;,+,*,/,-,…) entre comillas (ver la ayuda de la herramienta para una enumeración más detallada [7]).

```
//- zona global, comentario
global
{
}
//- fin zona global
//- zona local
grammar ejercicio1
{
}
grammar ejercicio2
{
}
//- fin zona local
```

Figura 2. Estructura general de un archivo de entrada a la aplicación

El símbolo inicial de cada gramática será el primer símbolo no terminal de la primera producción declarada que sea válida (después de expandir los conjuntos de producciones insertados en la gramática).

En la Figura 3, se muestra un ejemplo de un archivo escrito según el lenguaje diseñado. Veamos algunas consideraciones semánticas sobre dicho ejemplo:

- Antes de usar cualquier símbolo en una producción este debe ser declarado, exceptuando los símbolos especiales que van entre comillas simples que son considerados terminales (p.e. en `EP := '+' T EP` el '+' es considerado como Terminal al ir entre comillas y no es necesario declararlo).
- El tipo de un símbolo vendrá determinado por la declaración local si existiera y en caso contrario por la global (p.e. el tipo del símbolo `F` de la gramática `recursiva_der` en el ejemplo de la Figura 3 será No Terminal, como se especifica en la zona global).
- No está permitida la redeclaración de un símbolo en un mismo ámbito.
- Las producciones deben ser de una gramática libre de contexto (p.e. TP := '*' F TP | '/' F TP | ;).
- La utilización de un identificador de un conjunto de producciones localmente en una gramática requiere su previa definición en la zona global (p.e. `comunes` en Figura 3).

El formato elegido como lenguaje de salida en nuestra aplicación ha sido XML por ser un

lenguaje fácil de generar y de utilizar por aplicaciones externas.

```
/*
 * Ejemplo con dos gramaticas
 */

global
{
    analysis    LL1;
    nonterminal E, T, F;
    terminal    num;
    comunes={
              F := num | '(' E ')';
            }

}
grammar recursiva_izq
{
    E := E '+' T | E '-' T | T;
    T := T '*' F | T '/' F | F;
    communes;
}
grammar recursiva_der
{
    nonterminal EP, TP;

    E  := T EP;
    EP := '+' T EP | '-' T EP | ;
    T  := F TP;
    TP := '*' F TP | '/' F TP | ;
    comunes;
}
```

Figura 3. Archivo escrito en el lenguaje diseñado

3.4. El procesador del lenguaje, el motor de Proletool

El procesador del lenguaje presentado en la sección anterior es el componente más importante de la aplicación, por dicha razón se ha tenido mucho cuidado en el diseño del mismo. El procesador tiene las siguientes características:

- Portable. Se ha implementado en Java, sin emplear ningún paquete externo que no tenga la distribución estándar.
- Fácil acoplamiento. Puede usarse de manera aislada vía línea de comandos, lo cual hace más rápida su ejecución y facilita su acoplamiento o interconexión con otras aplicaciones. Esto es lo que se ha hecho en *Proletool* (ver Figura 1)
- Extensible. Está diseñado de manera que puedan añadirse otro tipo de análisis en un futuro, p.e. análisis de precedencias, con el menor tiempo de desarrollo posible.
- Integro. El procesador informa detalladamente y con precisión de todos los errores (léxicos, sintácticos y semánticos) que encuentra durante el proceso de análisis de la entrada. Intenta recuperarse lo antes posible de los mismos.

Para que la aplicación misma sea un ejemplo para los futuros estudiantes de la asignatura PL, se ha realizado un diseño basado en fases. Primero se desarrollo el analizador léxico empleando para ello la herramienta *JFlex*, posteriormente se utilizó *Cup* para desarrollar el analizador sintáctico. Finalmente, se añadieron acciones semánticas a la especificación *Cup* con el objeto de realizar las comprobaciones semánticas así como la de extraer la información relevante para resolver los ejercicios y generar una representación interna con toda esa información. Una vez finalizada la etapa de análisis nos enfrentamos a la etapa de síntesis, en la que se trata la información recopilada y almacenada en las estructuras internas, generando la salida en formato HTML.

3.5. Proletool

La herramienta *Proletool* además de resolver los ejercicios propuestos por el usuario permite:

- Gestión y administración de usuarios (alta, baja, modificación de datos).
- Administrar las opciones de la herramienta, estableciendo que es lo que se puede hacer con ella y a quién se le permite.
- Publicar noticias relacionadas con la herramienta, así como su envío a través del correo electrónico a los usuarios registrados.
- Registrar las estadísticas de uso y generar informes textuales y gráficos.
- Permitir la asistencia a los usuarios de la herramienta con las dudas que estos puedan tener, así como la recopilación de sus comentarios y sugerencias.
- Gestionar y administrar una batería de ejemplos útiles para las explicaciones de los métodos.

4. Resultados observados

La herramienta está en funcionamiento desde el Curso 04/05 y en vista de los resultados obtenidos seguirá a disposición de la comunidad universitaria en los próximos años. En la Tabla 1 se muestra un resumen con los datos más

interesantes sobre el uso que se hizo de *Proletool* durante el curso 04/05, también se muestran datos sobre el curso actual 05/06 incluyendo alumnos de la convocatoria de diciembre (7 alumnos) hasta la fecha actual, principios de febrero de 2006.

Variable	Curso 04/05	Curso 05/06
Alumnos registrados	40	55
Sobrecargas del servidor	0	0
Archivos sometidos	678	551
Gramáticas compiladas	718	566
Total Análisis realizados	899	675
Análisis LL1	282(31,4%)	351(52%)
Análisis SLR1	253(28,1%)	101(15%)
Análisis LR1	302(33.6%)	169(25%)
Análisis LALR1	62(6.9%)	54(8%)
Total Simulaciones realizadas	74	127
Simulaciones LL1	16(21,6%)	66(52%)
Simulaciones SLR1	9(12,2%)	43(33,9%)
Simulaciones LR1	38(51,4%)	14(11%)
Simulaciones LALR1	11(14,9%)	4(3,1%)
Total comentarios realizados	33	11
Utilidad media (1-5)	4,2	4,5
Facilidad de uso media (1-5)	3,9	4,5

Tabla 1. Resumen con las estadísticas de uso

Los datos muestran no sólo que los usuarios han utilizado y utilizan la herramienta, sino que también la consideran útil y fácil de usar lo cual nos permite afirmar que se ha cumplido con el principal objetivo: ser una herramienta de apoyo en el estudio de la asignatura de Procesadores de Lenguajes que sea utilizada por los alumnos.

5. Conclusión

En este trabajo se han presentado los trabajos realizados en la ESI (UCLM) para la mejora de la calidad docente de la asignatura de PL. Desarrollando una herramienta que utiliza Internet para facilitar la enseñanza y aprendizaje de las técnicas de análisis sintáctico. El uso de la arquitectura cliente/servidor para su uso a través de Internet es la principal ventaja de *Proletool* frente a otras herramientas diseñadas con el mismo propósito.

Por motivos de espacio no se ha incluido ninguna pantalla de la aplicación realizada. Para la prueba de la misma u obtener una descripción más detallada sobre el proceso de construcción del procesador, así como de la aplicación *Proletool* se recomienda la consulta de la documentación en [8] o en la ayuda de la aplicación [7].

Agradecimientos

A la ESI y a la UCLM, a los alumnos del curso 04/05 y profesorado presente y pasado de la asignatura PL por su apoyo a este proyecto y sugerencias que han permitido su desarrollo y mejora. Trabajo financiado por los proyectos DIMOCLUST TIC2003-08807-C02-02 y PREDACOM PBC-03-004.

Referencias

[1] ACM-IEEE. *Computing Curricula.* 2001. http://www.sigce.org/cc2001/

[2] Ezpeleta, J. ANAGRA: *Una herramienta visual para estudiar técnicas de análisis sintáctico. Univ. de Zaragoza.* http://webdiis.unizar.es/~ezpeleta/COMPI/compiladoresI.htm

[3] ANECA. *Libro blanco del Título de grado en Ingeniería Informática.* 2005. http://www.aneca.es/modal_eval/docs/libroblanco_jun05_informática.pdf

[4] ECTS. *European Credit Transfer System.* http://europa.eu.int/comm/education/programmes/Sócrates/ecos_en.html

[5] Diez, J.L., Diaz J. JavaPars. *Proyecto Java Parser 1.0. Univ. de Deusto.* http://paginaspersonales.deusto.es/josuka/jparser/parser.html

[6] Rodger, S., Finley, T. JFLAP. *An Interactive Formal Languages and Automata Package. Duke University.* http://www.jflap.org

[7] Santos, P.A., Castro-Schez, J.J. PROLETOOL. *Univ. de Castilla-La Mancha.* http://oreto.inf-cr.uclm.es/ proletool/

[8] Santos, P.A. *Una herramienta para el apoyo en el estudio y la enseñanza de la asignatura Procesadores de Lenguajes.* 2005. http://personal.oreto.inf-cr.uclm.es/jjcastro/ProyectosFindeCarrera/ProyectosRealizados/ proletool-pasantos.pdf

[9] Jodar, J.F., Revelles, J. SEFALAS. *Software para la Enseñanza de las Fases de Análisis Léxico y Sintáctico en Procesadores de Lenguajes. Univ. de Granada.* http://lsi.ugr.es/~pl/software.php

[10] SEPa!. *Proyecto SEPa! Software para la Enseñanza de Parsing. Univ. Católica de Santiago del Estero (Argentina).* http://www.ucse.edu.ar/fma/sepa