Una herramienta para la enseñanza y aprendizaje de lenguajes formales y teoría de autómatas

Jose Jesus Castro-Schez, Ester del Castillo, Julian Hortolano
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 Ester.Castillo@uclm.es

Resumen

El objetivo de este trabajo es presentar una experiencia de innovación docente basada en el uso de una herramienta software que ha sido desarrollada empleando tecnologías Web. Este software ha sido diseñado con el propósito de mejorar la calidad de la formación impartida, facilitando el aprendizaje de los principales conceptos de la asignatura Teoría de Autómatas y Lenguajes Formales. Esta herramienta comprende desde aspectos puramente docentes hasta aspectos de seguimiento del trabajo realizado por los alumnos durante el periodo de estudio de la asignatura.

1. Motivación

El empleo de software docente puede ser útil para la enseñanza y aprendizaje de una materia y una ayuda en la búsqueda de la excelencia y mejora de la calidad formativa de cualquier institución universitaria. En este trabajo se describe el diseño y desarrollo de una herramienta creada para mejorar la calidad de la formación impartida en la asignatura Teoría de Autómatas y Lenguajes Formales (TALF) de la Escuela Superior de Informática (ESI) de la Universidad de Castilla-La Mancha (UCLM).

La hipótesis de partida es que la calidad de la formación que se imparte en dicha asignatura se podría mejorar usando herramientas software, ya que en dicha asignatura, los ejercicios son muy importantes a la hora de asimilar conocimientos teóricos y estos pueden ser resueltos siguiendo una serie de pasos perfectamente definidos.

En la actualidad se pueden encontrar una serie de herramientas que se han propuesto, diseñado y usado con este objetivo, como ejemplo citar JFLAP[12], SEFALAS[5] o el proyecto SEPa![13].

Estas herramientas permiten la realización de ejercicios sobre o con autómatas y gramáticas que son dados como entrada al sistema.

En el diseño de la herramienta (SELFA) que se propone en este trabajo se ha intentado incorporar las características más interesantes de las herramientas antes mencionadas (entornos fáciles de usar, visuales o con capacidad gráfica, uso de un lenguaje para la definición de autómatas y gramáticas así como para la especificación de ejercicios sobre esos elementos). Además, se han añadido características útiles, como la posibilidad de ejecutarla sin necesidad de instalación local (utilizando una arquitectura cliente/servidor) y la creación e integración de una base de datos para recoger datos de uso con los que generar estadísticas que permitan analizar el trabajo realizado por los alumnos. Otro aspecto a destacar es el uso de variables para almacenar resultados intermedios de manera que se generen soluciones en un solo paso a diversos tipos de ejercicios tanto sobre gramáticas como autómatas indistintamente.

La solución al problema presentado se fundamenta en la propuesta de un lenguaje que permite definir autómatas, gramáticas y ejercicios sobre ambos, así como usar variables para guardar resultados intermedios. De este modo, el diseño de la herramienta se basa en un procesador para dicho lenguaje.

Este último aspecto permite que la herramienta pueda emplearse como ejemplo de aplicación de los conocimientos vistos en otra asignatura de los planes de estudio de la ingeniería informática, Procesadores de Lenguajes, en la solución a un problema real.

La idea de desarrollar esta aplicación tiene su origen al estudiar los resultados obtenidos por experiencias previas realizadas para otras asignaturas del plan de estudio de la Ingeniería Informática, concretamente por la herramienta ProleTool [9][10]. Esta herramienta se usa en la

asignatura Procesadores de Lenguajes y permite definir gramáticas y resolver ejercicios de aplicación de técnicas de análisis sintáctico descendente (LL1) y ascendentes (SLR1, LR1 y LALR1) sobre dichas gramáticas.

El resto del trabajo está estructurado de la siguiente manera: En la siguiente sección se presenta y analiza el problema que se pretende resolver. En la Sección 3 se estudiará como se ha diseñado y desarrollado la aplicación *SELFA*. A continuación, se mostrarán unos datos sobre el uso de la herramienta en el curso académico 05/06 y, para finalizar, se mostrarán las conclusiones y futuros trabajos.

2. Problemática existente

La asignatura de Teoría de Autómatas y Lenguajes Formales tiene una gran importancia en la formación de los titulados en Informática ya que proporciona los fundamentos teóricos de la disciplina, lo cual permite comprender mucho mejor la informática y sus orígenes (tanto históricos como matemáticos) y explorar sus problemas y posibilidades [4][6][7][8].

En la actualidad, parece que existe una preocupación por una enseñanza eminentemente práctica. Es por esto por lo que esta asignatura ha dejado de ser troncal en los planes de estudios y no tiene presencia en el Libro Blanco de la Ingeniería Informática, dentro de la propuesta para contenidos troncales del Título de Grado en Ingeniería [1]. No obstante, creemos o confiamos que las universidades, dentro de las asignaturas optativas, ofrecerán a los alumnos la posibilidad de elegir formarse en esta disciplina.

En la asignatura TALF se pretende que los alumnos adquieran conocimientos y desarrollen habilidades sobre autómatas, gramáticas y lenguajes formales que les permitan analizar, entender y solucionar problemas.

La asimilación y aprendizaje de los conocimientos y conceptos de la asignatura de una manera sólida requieren de la realización de un gran número de ejercicios por parte del alumno.

Sin embargo, en la mayoría de los casos, esto no es posible hacerlo de forma intensiva en clase, debido a la extensión del temario y al tiempo limitado de las mismas.

Por ello, se hace cada vez más necesario establecer medidas encaminadas a minimizar este problema. Una primera posible medida consiste en proporcionar material a los alumnos para que estos lo estudien antes de cubrir el tema en clase, de esta forma se podría dedicar ese tiempo a la realización de ejercicios. El principal inconveniente de esta medida es que los conceptos que los alumnos tienen que asimilar son muchas veces demasiados abstractos, lo cual puede dificultar su aprendizaje sin la explicación del profesor.

Otra posible medida implica diseñar y proporcionar a los alumnos una batería de ejercicios junto con sus soluciones de manera que les permita resolver cada uno de los ejercicios propuestos por su cuenta y comprobar la solución obtenida. El principal inconveniente de esta medida es la necesidad de mantener actualizada constantemente esta batería.

Como una alternativa a esta medida, en este trabajo se propone el diseño y desarrollo de una aplicación, a la que se ha nombrado SELFA, que permite a los alumnos la propuesta de ejercicios de la asignatura y la generación de las soluciones a los mismos, pudiendo comprobar como SELFA ha generado los resultados y si estos coinciden con la resolución hecha por el alumno. Esto puede motivar a los estudiantes en el estudio de la asignatura proponiendo ellos sus propios ejercicios, así como potenciar su capacidad creativa y de análisis.

Por otro lado se pretende que la herramienta también pueda ser empleada en las clases magistrales que imparte el profesor.

De este modo, se pretende mejorar la enseñanza y aprendizaje de la asignatura de TALF.

Además, el contexto en el que se mueve la Universidad Europea hacia el establecimiento de un Espacio Europeo de Educación Superior (EEES) y hacia el sistema de transferencia de créditos europeos (ECTS) [2] que pretende reflejar el esfuerzo real requerido y realizado por el estudiante para conseguir una serie de objetivos, hace necesario incorporar a SELFA un sistema de control de uso asociado a cada alumno que permita valorar el trabajo realizado por este en el estudio de la materia de TALF.

3. Diseño de SELFA

En el diseño de la aplicación SELFA[11] se han tenido en cuenta otras herramientas existentes, como JFLAP[12], SEFALAS[5], el proyecto SEPa![13] y ProleTool[10] así como los requisitos de los potenciales usuarios de la misma.

Se pretende que SELFA sea usada por dos tipos de usuarios: alumnos y profesores. Estos dos tipos tienen necesidades, deseos y prioridades comunes y diferentes, por lo que hay que tener en cuenta todo ello para realizar un diseño que satisfaga a ambos.

La principal funcionalidad que debe ofrecer la herramienta, desde el punto de vista del alumno y del profesor, es la de *aceptar ejercicios y calcular las soluciones a los mismos, mostrándolas de una forma clara y sencilla de manera que se permita comprender como se han alcanzado*.

A este objetivo común se deben añadir otras características deseables para ambos tipos de usuarios de cara a conseguir que la herramienta sea práctica y se use:

- Disponibilidad temporal. Debería estar disponible las 24 horas de todos los días.
- Disponibilidad geográfica. Sería interesante que la herramienta tuviera una instalación lo más sencilla posible para conseguir su uso en el ordenador del alumno, de los laboratorios, de las clases teóricas y en el ordenador del profesor.
- Actualizable. Esta herramienta surge con la idea de que esté en continua mejora. Un cambio en la aplicación debe ser fácil de trasladar a cada uno de los ordenadores donde esté instalada. Por ello, se ha de establecer un sistema de actualización que facilite dicha tarea.
- Intuitiva. Debe tener una interfaz amigable, no sólo a la hora de introducir ejercicios, si no en todas las opciones que pueda realizar el usuario con la herramienta.
- Facilitar la entrada de tantos ejercicios como se desee e incluso utilizar resultados intermedios para realizar operaciones sobre ellos.
- Correcta. Debe proporcionar una información precisa y libre de errores.
- Completa. Debe proporcionar toda la información necesaria para que se comprendan las soluciones obtenidas.
- Plataforma libre, se debe poder ejecutar en cualquier plataforma.

Desde el punto de vista de los profesores de la asignatura sería deseable que la aplicación tuviera las siguientes características generales:

- Ejemplar. La aplicación debe ser un ejemplo para futuros alumnos sobre como emplear los conocimientos teóricos y prácticos de la asignatura y en general de la carrera en la resolución de un problema real.
- Accesible. Debe tener un acceso cómodo y rápido para que se pueda usar fácilmente como herramienta de ayuda a la enseñanza en cualquier sitio.
- Funcional. De manera que se pueda acceder a las opciones más importantes y mostrar la información generada de la manera más adecuada dependiendo del ambiente en la que se use.
- Transparente. Debe permitir ver el trabajo que se realiza con la herramienta y quién lo realiza, si el usuario lo permite. Esta información podría ser empleada para valorar el trabajo realizado por el alumno a la hora de aprender los conceptos de la materia.
- Software Libre. Que esté desarrollado con licencia GPL, para así permitir que personas interesadas, como alumnos o personal docente, puedan mejorar la herramienta.

La herramienta ha sido diseñada de una manera incremental. En cada paso del diseño, se ha tenido un producto que los alumnos y profesores han podido probar. A cada producto obtenido se le ha incorporado un mecanismo de seguimiento de uso de la misma. Además, se le incorporaba un sistema para recoger comentarios y sugerencias de los usuarios que pudieran ser empleados para mejorar la herramienta.

Durante este proceso aparecieron nuevos requisitos que fueron considerados en el diseño y desarrollo final de la herramienta SELFA.

Las siguientes características aparecieron como nuevos requisitos durante este proceso:

- Lenguaje fácil de aprender y usar.
- Eficiente. Las soluciones a los ejercicios deben calcularse de forma rápida.
- Ayuda útil. Debe existir un sistema de consulta con todo lo referente a la herramienta que sirva como manual de usuario.

- Uniforme. Debe emplear en la medida de lo posible la misma notación, tanto para autómatas como gramáticas. Además esta debe ser lo más parecida posible a la empleada en clase de teoría.
- Gráfica. Las soluciones deben mostrarse de una manera gráfica para facilitar la comprensión de la misma.

El implicar a los usuarios en el desarrollo de la herramienta, considerando sus comentarios, ha servido para fomentar su uso.

En la siguiente sección se detallan las decisiones de diseño tomadas para satisfacer los requisitos de los usuarios

3.1. Interacción con el usuario

La primera decisión de diseño que se ha tenido que tomar es la de establecer como debe ser la entrada de ejercicios a la herramienta. Para ello, antes se ha definido con precisión cual va a ser la entrada.

Una entrada consiste en la definición de uno o varios autómatas, una o varias gramáticas y la especificación de una serie de operaciones sobre esos elementos definidos, que pueden o no devolver resultados intermedios sobre los que se van a poder volver a operar.

La definición de un autómata $A=(Q,\Sigma,\delta,q,F)$, implica la especificación del conjunto de estados (Q), del alfabeto de entrada (Σ), de la función de transición (δ), del estado inicial (q) y del conjunto de estados finales (F).

La definición de una gramática, $G=(N,T,P,S)$, implica especificar el conjunto de símbolos No Terminales (N), el conjunto de Terminales (T), el conjunto de producciones (P) y el símbolo inicial (S).

La definición de un ejercicio implicará el uso de una palabra reservada asociada a dicho ejercicio y unos argumentos, entre los cuales se encontrará una gramática o un autómata previamente definido sobre el cual se va a aplicar la operación.

Para determinar como se va a proporcionar la entrada a SELFA, se han estudiado las distintas alternativas usadas en las herramientas existentes: vía formularios [3], gráficamente [12][13] o por medio de un lenguaje de entrada [5][10].

La alternativa gráfica es muy interesante para usuarios no especializados en el tema, pero por contra tiene como principales inconvenientes el ser lento a la hora de interactuar con la herramienta y obligar a tener que trabajar "on-line".

La entrada a través de formularios tiene la ventaja de ser un proceso guiado, lo cual es idóneo para usuarios noveles en el proceso pero tiene los mismos inconvenientes que la anterior.

Por estas razones, la forma elegida ha sido mediante un fichero de entrada escrito en un lenguaje en el que se especifiquen los autómatas, gramáticas y los ejercicios a resolver. Esta alternativa, aunque tiene el inconveniente de que obliga a los usuarios a aprender un lenguaje, tiene como principal ventaja el permitir trabajar "off-line" y proporcionar una forma rápida de introducir ejercicios. Para escribir ejercicios bastaría con tener un editor de texto e introducir el ejercicio o ejercicios que se quisiera solucionar. Además, el inconveniente se puede minimizar diseñando un lenguaje sencillo y fácil de aprender.

La solución que ofrece la herramienta será:

- Nuevos autómatas o gramáticas, más información adicional útil para comprender la solución y cómo se ha alcanzado, cuando la entrada es correcta.
- Una serie de mensajes útiles para detectar los errores que puedan aparecer en la entrada, cuando esta no es correcta.

La herramienta ofrecerá un mecanismo que permita navegar por la entrada y la información dada como salida.

3.2. Lenguaje de entrada a SELFA

El diseño del lenguaje de entrada es algo delicado, ya que es una de las partes más importantes de la herramienta y debe realizarse minuciosamente. El lenguaje diseñado tiene las siguientes características:

- Tiene cierta similitud con el lenguaje CUP, que se usa en clases prácticas de TALF, lo cual facilitará su aprendizaje.
- Permite la entrada en un solo archivo de todos los ejercicios que el usuario quiera y resolverlos en un solo paso. Esto es de utilidad cuando se tiene un tiempo de acceso limitado a Internet o se trabaja off-line.
- Los resultados intermedios de las operaciones realizadas pueden ser almacenados en variables sobre las que después se podrá volver a operar.

- Existe una única operación imprimir que actuará de una forma u otra dependiendo de lo que se le pase como argumento (un autómata o gramática).

La sintaxis en notación EBNF del lenguaje propuesto se muestra en la Tabla 1.

El símbolo `ID` corresponde al lexema identificador cuyo patrón de construcción es el siguiente: `ID=[A-Za-z][A-Za-z0-9]*.`

En este lenguaje existen unas restricciones que no pueden ser expresadas por medio de la sintaxis y que el procesador del lenguaje deberá detectar si no se cumplen, informando al usuario. Estas restricciones son:

- Toda gramática y/o autómata debe tener un identificador único.
- En la declaración de elementos de una gramática no se permite la asignación de dos tipos diferentes (terminal y nonterminal) a un mismo elemento.
- El símbolo inicial de una gramática ha de estar declarado anteriormente como un elemento nonterminal.
- La parte izquierda de una producción debe haber sido declarada como un elemento nonterminal.
- En la definición de un autómata los elementos que se definan como estados (states) no pueden definirse también como elementos del alfabeto (alphabet) y a la inversa.
- El elemento declarado como inicial debe haber sido previamente definido como un estado.
- Los elementos declarados como final deben haber sido declarados como estados.
- En las transiciones el primer identificador debe haber sido declarado como un estado, el segundo como un elemento del alfabeto y el resto de identificadores deben ser elementos del conjunto de estados.
- Las operaciones sobre autómatas (OPA) y sobre gramáticas (OPG) deben realizarse sobre autómatas o gramáticas previamente declarados o resultados de operaciones sobre estos.

```
PROGR   ::=  (DECL|OPNS){DECL|OPNS}
DECL    ::=  GRAM|AUTM
GRAM    ::=  grammar ID BGRAM
BGRAM   ::=  '{'DS SINI DP'}'
DS      ::=  T NT | NT T
T       ::=  terminal {ID','} ID';'
NT      ::=  nonterminal {ID','} ID';'
SINI    ::=  si ID';'
DP      ::=  productions '{'PROD{PROD}'}'
PROD    ::=  ID ':=' PDCHA;
PDCHA   ::=  ID{ID}{'|'ID{ID}}('|'$|$)
AUTM    ::=  automaton ID BAUTM
BAUTM   ::=  '{' EST ALF INI FIN TRANS '}'
EST     ::=  states {ID','} ID';'
ALF     ::=  alphabet {ID','} ID';'
INI     ::=  initial ID';'
FIN     ::=  final {ID','}ID';'
TRANS   ::=  transition '{'TRA{TRA}'}'
TRA     ::=  ID','ID '=' DEST
DEST    ::=  ID';'|'(' ID {','ID}')'';'
OPNS    ::=  OPG|OPA|PRIM
OPG     ::=  OG1|OGN|sim_null '(' ID ')' ';'
OG      ::=  ID '=' OPG1 '('ID')'';'
OPG1    ::=  inaccesibles|nodevterm|inutils|null|unitary|clean|fnc|fng
OGN     ::=  cyk '('ID','ID{ID}')'';'
OPA     ::=  OA1|OA2|OAN|equals'('ID','ID ')'';'
OA1     ::=  ID '=' OPA1 '('ID')'';'
OPA1    ::=  passND|passAFD|passAFDM|complement|inverse
OA2     ::=  ID '=' OPA2 '('ID','ID')'';'
OPA2    ::=  Intersection | union
OAN     ::=  recognize '('ID','ID{ID}')'';'
PRIM    ::=  print '('ID')'';'
```

Tabla 1. Notación EBNF de la sintaxis del lenguaje propuesto.

Como ejemplo de uso de SELFA, se muestra un ejercicio propuesto a los alumnos. El objetivo es, dado un Autómata Finito No Determinista con lambda transiciones, obtener un AFD mínimo equivalente y ejecutar este AFD sobre una palabra en la entrada. Para terminar, se propone al alumno que obtenga, sin usar SELFA, el AFD mínimo, aplicando los algoritmos vistos en teoría, y compare su resultado con el resultado de SELFA, usando la instrucción equals(A1,A2).

```
automaton prueba{
          states q0,q1,q2,q3,q4,q5;
          alphabet a,b;
          initial q0;
          final q5;
          transition{
                    q0,$=q1;
                    q0,$=q3;
                    q1,a=q1;
                    q1,b=q2;
                    q2,$=q5;
                    q3,a=q4;
                    q4,b=q4;
                    q4,$=q5;
                    q5,$=q0;   }
}
print(prueba);
pruebaAFDM=passAFDM(prueba);
print(pruebaAFDM);
recognize(pruebaAFDM, a b b a b b);
```

El resultado de esta ejecución provoca que SELFA muestre los resultados de los Autómatas intermedios que ha generado antes de obtener el AFD mínimo. Respecto al reconocimiento de la palabra, SELFA muestra la secuencia de transiciones aplicadas al ejecutar el autómata sobre la palabra en la entrada.

Usando este ejercicio, el alumno debería resolver el mismo problema y comprobar si lo ha hecho correctamente, usando la instrucción *equals (A, pruebaANDM)*, donde A será el AFD mínimo obtenido por él.

3.3. Tratamiento de la entrada de ejercicios

Para comprobar si el texto de entrada tiene una sintaxis correcta, verificar si se cumplen las restricciones antes mencionadas y extraer la información relevante de los ejercicios para poder solucionarlos es necesario que SELFA tenga un procesador de dicho lenguaje.

El procesador del lenguaje se ha diseñado por etapas y por fases. Primero, se ha prestado atención a la etapa de análisis, diseñando el analizador léxico, después el sintáctico y, por último, el analizador semántico. La integración de estos tres analizadores ha permitido construir el procesador buscado con las siguientes características:

- Portable. Se ha implementado en Java con lo que es multiplataforma.
- Extensible. Se ha diseñado de manera que se puedan añadir características nuevas si se desea, por ejemplo, nuevos algoritmos que solucionen más ejercicios.
- Integro. El procesador funciona como debe, obteniendo soluciones cuando la entrada es correcta e informando del mayor número de errores posible cuando la entrada no es correcta (posee un mecanismo de recuperación de error).

Una vez desarrollada la etapa de análisis, nos enfrentamos a la etapa de síntesis. En esta etapa se han implementado una serie de algoritmos sobre gramáticas: Eliminación de símbolos y producciones que deriven en la cadena vacía, eliminación de símbolos y producciones que no deriven en cadena de terminales, eliminación de símbolos inaccesibles y eliminación de producciones unitarias, así como pasar una gramática libre de contexto a Forma Normal de Chomsky (FNC) y a Forma Normal de Greibach (FNG) y algoritmo de Cocke-Younger-Kasami, (CYK) para decidir si una palabra pertenece al lenguaje definido por una gramática.

Respecto a las operaciones sobre autómatas, se implementa los algoritmos para pasar un autómata finito no determinístico con transiciones nulas a un autómata finito no determinista, pasar un autómata finito a un autómata finito determinista, pasar un autómata finito a un autómata determinístico mínimo, así como realizar la unión, la intersección, el inverso, el complemento autómatas. Comprobar si una cadena es reconocida por un autómata y comprobar si dos autómatas son iguales.

Por último, se ha integrado el procesador de lenguaje con los algoritmos que resuelven los ejercicios y la parte de la interfaz. La etapa de análisis del procesador de lenguaje obtiene, de las entradas correctas, la información relevante de cada ejercicio y se la comunica a la etapa de síntesis que se encarga de resolver los ejercicios mediante la llamada a los algoritmos que realizan las operaciones pertinentes. La integración del procesador de lenguaje con los algoritmos ha sido relativamente sencilla ya que ambas partes usaban un lenguaje común que era Java.

3.4. La salida de la herramienta

Una vez determinado como se introducen los ejercicios a la herramienta por medio de un lenguaje, como se analizan las entradas, o mejor aún los ejercicios, por medio de un procesador de dicho lenguaje y como se resuelven estos mediante la invocación de los algoritmos correspondientes, queda determinar cómo se van a presentar las soluciones al usuario de manera que se permita navegar por ellas y puedan ser mostradas en cualquier ordenador.

El formato interno en el que se generan las soluciones ha sido XML por ser un lenguaje fácil de generar y de utilizar por aplicaciones externas. Además, gracias a la tecnología XSL, estas soluciones en XML pueden ser formateadas pudiendo devolver la solución en formato HTML. Esto será de mucha utilidad al integrar esta parte con la de la interfaz externa de la herramienta.

3.5. Arquitectura de la herramienta

Uno de los requisitos importantes de la herramienta es que su instalación y actualización debía ser lo más sencilla posible, siendo lo deseable, incluso, que no necesite instalación. De este modo, se pretendía facilitar su uso en cualquier ordenador independientemente de su ubicación física. Además, otro requisito esencial era que debía registrar el uso que los usuarios, principalmente los alumnos, hacían de ella. Para cumplir estos objetivos se ha tomado la decisión de crear una aplicación centralizada usando una arquitectura cliente/servidor Web.

La arquitectura elegida para la herramienta ha sido una arquitectura multicapa Web, en la que cada capa ofrece sus servicios a la capa inmediatamente superior y recibe los servicios de la capa inferior. De este modo, se ha reducido el acoplamiento y la complejidad de la herramienta. La arquitectura Web tiene las siguientes ventajas:

- Toda la carga de trabajo cae sobre el servidor, que es en la que se ejecutará SELFA, haciendo la aplicación en el lado del usuario (el cliente) una aplicación muy ligera.
- No necesita instalación local, sólo un navegador conectado a Internet.
- Al ser una aplicación centralizada, se permite registrar el uso de la herramienta mediante una base de datos situada en el servidor.

De este modo, las actualizaciones de SELFA son inmediatas ya que, al estar la aplicación centralizada, basta con actualizar la parte del servidor para que los cambios sean visibles en todos los sitios donde se use.

La parte Web de la herramienta ha sido desarrollada usando el lenguaje de script PHP que tiene las características de ejecutarse en el servidor, ser mutiplataforma y soportar gran cantidad de usuarios. Además, la base de datos en la que se registrará todo el uso de la herramienta será MySQL. Este tipo de base de datos es muy potente, es gratuita, tiene una productividad elevada, un bajo consumo y un tiempo de respuesta bajo. Las soluciones son devueltas en formato HTML.

3.6. Capacidades incorporadas a SELFA

La herramienta SELFA, a diferencia de otras herramientas, permite:

- Gestión y administración de usuarios (alta, baja, modificación de datos).
- Administrar las opciones de configuración de la herramienta, qué se puede hacer, cuándo y por quién.
- Publicar noticias relacionadas con la herramienta, así como su envío a través del correo electrónico a los usuarios registrados.
- Registrar las estadísticas de uso referidas a la clase de ejercicios que ha realizado cada alumno y generar informes en forma tabular y gráfica.
- Gestionar un repositorio de ejercicios escritos en el lenguaje de entrada de SELFA.

4. Resultados Observados

En la etapa de evaluación de la herramienta los resultados han sido muy satisfactorios. SELFA está en funcionamiento desde el segundo cuatrimestre del Curso 05/06, momento en el que comenzaron las clases de la asignatura de TALF de la Ingeniería Técnica en Informática de Gestión. Los resultados de uso, de valoración por parte de los usuarios de este grupo y de utilidad pueden ser consultados en la Tabla 4.

Los datos muestran no sólo que los usuarios han utilizado y utilizan la herramienta, sino que también la consideran útil y fácil de usar.

En cuanto a la utilidad de la herramienta, tras un análisis de las calificaciones obtenidas en las

convocatorias de Junio y Septiembre por los alumnos que han usado la herramienta (25, ver Tabla 4), se ha comprobado que 24 de ellos han aprobado la asignatura, lo que supone el 75% del total de aprobados. Se puede concluir que SELFA ha ayudado en el aprendizaje de la asignatura de TALF. De los 18 aprobados en Junio, 14 habían usado SELFA. De los 10 aprobados de Septiembre, 10 habían usado SELFA y es por esta razón por lo que se pretende que siga en funcionamiento en los próximos cursos.

Variable	Curso 05/06
Alumnos registrados	45
Alumnos que la usaron	25
Archivos sometidos	1196
Ejercicios realizados	1449
Ejercicios de Gramáticas	
Elim. Produc. No Dev. Term.	134
Símbolos Inaccesibles	40 (2,76%)
Producciones Cadena Vacía	63 (4,35%)
Eliminación Cadena Vacía	60 (4,14%)
Eliminación Unitarias	104 (7,18%)
Forma Normal Chomsky	55 (3,8%)
Forma Normal Greibach	45 (3,11%)
Algoritmo CYK	3 (0,21%)
Ejercicios Autómatas	
Pasar a AFND	6 (0,41%)
Pasar a AFD	228 (15,73%)
Pasar a AFDM	144 (9,94%)
Complemento	44 (3,04%)
Igualdad	69 (4,76%)
Intersección	121 (8,35%)
Inverso	63 (4,35%)
Reconocimiento	232 (16,01%)
Unión	38 (2,62%)
Puntuaciones	
Media en Facilidad de Uso	4,389 sobre 5
Media en Valoración Total	4,417 sobre 5

Tabla 2. Resumen de las estadísticas de uso y valoración.

5. Conclusión

En este trabajo se ha mostrado el trabajo realizado en la ESI (UCLM) para mejorar la calidad docente de la asignatura de TALF.

Se ha presentado una herramienta cuyo diseño está basado en un procesador de lenguaje que se ejecuta en una arquitectura cliente/servidor que permite su uso a través de Internet, principal ventaja de *SELFA* frente a otras herramientas diseñadas con el mismo propósito. El principal objetivo de la herramienta es permitir la propuesta de ejercicios sobre diferentes temas de la asignatura y la resolución de los mismos.

Esta herramienta puede ser utilizada tanto por el profesor para su uso en las clases magistrales como por los alumnos para su uso en el estudio de los conceptos de la materia.

Respecto a trabajos futuros, se pretende ampliar esta herramienta para conseguir cubrir toda la materia de la asignatura, añadiendo expresiones regulares, autómatas a pila y máquinas de Turing.

Agradecimientos

A la ESI, a la UCLM y a los alumnos del curso 05/06 por sus sugerencias y apoyo a este proyecto.

Referencias

[1] http://www.aneca.es/modal_eval/docs/libroblanco_jun05_informática.pdf

[2] http://europa.eu.int/comm/education/programmes/Sócrates/ecos_en.html

[3] Diez, J.L., Diaz J. JavaPars. http://paginaspersonales.deusto.es/josuka/jparser/parser.html

[4] Hopcroft, J.E., Motwani, R., Ullman, J.D. *Introduction to automata theory, languages and computation*. Addison Wesley, 2001.

[5] Jodar, J.F., Revelles, J. SEFALAS, http://lsi.ugr.es/~pl/software.php

[6] Kelley, D. *Automata and Formal Languages: An Introduction*. Prentice Hall, 1998.

[7] Lewis, H.R., Papadimitriou, C.H. *Elements of theory of computation*. Prentice-Hall, 1997.

[8] Martin, J.C. *Introduction to languages and the theory of computation*. McGraw-Hill, 2003.

[9] Santos, P.A., Castro-Schez, J.J. Una herramienta para la enseñanza y aprendizaje de la asignatura Procesadores de Lenguajes. *XII Jornadas de la Enseñanza Universitaria de la Informática*, 499-506, Bilbao, 2006.

[10] Santos, P.A., Castro-Schez, J.J. PROLETOOL, http://oreto.inf-cr.uclm.es/proletool/

[11] Hortolano, J., Castro-Schez, J.J., Castillo, E., SELFA, http://apps.oreto.inf-cr.uclm.es/SELFA/

[12] Rodger, S., Finley, T. JFLAP, http://www.jflap.org

[13] Proyecto SEPa!