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 287-294

Enfoque de diseño de herramientas informáticas educativas para metodologías activas de aprendizaje

J.J. Castro-Schez[1] J. Albusac[2] D. Vallejo[1] M.A. Redondo[1]
Departamento de Tecnologías y Sistemas de Información
[1] Escuela Superior de Informática, [2] Escuela de Ingeniería Minera e Industrial de Almadén
Universidad de Castilla-La Mancha
[1] Paseo de la Universidad, 4; [2] Plaza de Manuel de Meca, 1
13170 Ciudad Real, 13400 Almadén
JoseJesus.Castro@uclm.es, JavierAlonso.Albusac@uclm.es,
David.Vallejo@uclm.es, Miguel.Redondo@uclm.es

Resumen

El paradigma educativo que plantea el Espacio Europeo de Educación Superior (EEES) ha traído consigo una renovación de las metodologías educativas en las universidades españolas. Existen muchas iniciativas en marcha relacionadas con esta renovación y su adaptación al EEES, pero se echan de menos propuestas o análisis sobre cómo deberían de ser las herramientas software educativas (más allá de los Sistemas para la Gestión de Aprendizaje LMSs), cada vez más importantes en los procesos formativos de las universidades, que se usen en este nuevo contexto.

En este trabajo se destaca la importancia del software educativo en las metodologías educativas activas y se presenta una propuesta sobre cómo creemos que debería diseñarse y desarrollarse dicho software, además de analizar cómo se debería integrar en el proceso formativo, considerando los nuevos roles que juegan profesor y estudiante en estas metodologías. Finalmente, se mencionan algunos ejemplos indicando cómo estas herramientas se pueden integrar en sistemas de gestión de aprendizaje reales como es el caso particular de CopperCore o DotLRN.

Summary

The new education paradigm devised by the European Space for Higher Education, known as the Bologna Process, has transformed the teaching methodologies in Spanish Universities. In fact, there are many ongoing initiatives related to this transformation and the adaptation to the Bologne Process. However, there is still a lack of proposals or analyses that discuss how the software educational tools should be, tools that are becoming more and more important in today's University.

In this work, we stress the importance of educational software within the context of active educational methodologies and present a proposal that addresses the design and development of this educational software (beyond the *Learning management system* LMSs). Furthermore, we discuss how such tools can be integrated into the formative process, taking into consideration the new roles that both lecturer and students play. Finally, we enunciate some examples highlighting how these software tools can be integrated into real learning management systems, such as CopperCore or DotRLN.

Palabras clave

Herramientas software educativas, Aprendizaje activo, TIC en el proceso aprendizaje-enseñanza

1. Motivación

El Espacio Europeo de Educación Superior (EEES) ha traído consigo en la mayoría de universidades españolas un cambio de paradigma educativo [3]. Actualmente, se fomenta un aprendizaje más activo por parte del estudiante, lo cual implica un cambio en el sujeto preponderante de la acción [7]. De este modo, el estudiante es el protagonista de su propio aprendizaje, es decir, en el nuevo proceso de aprendizaje el estudiante es quien aprende y el profesor es un guía u orientador del trabajo del estudiante. En otras palabras, el profesor es un estratega que planifica actividades para motivar el aprendizaje del estudiante y establece las condiciones para que dicho aprendizaje tenga éxito. El objetivo final será formar profesionales con competencias (que

integran conocimientos, habilidades, actitudes, valores) que les capaciten para responder a los retos de la Sociedad del Conocimiento.

Para que la transición a este modelo se realice con ciertas garantías de éxito, son necesarios varios cambios. Algunos son meramente administrativos, otros de carácter docente y otros de carácter instrumental que se reflejan en el software educativo que se necesita para materializar el proceso formativo.

En los últimos años se han desarrollado y desarrollan muchas jornadas y talleres para facilitar la adaptación de asignaturas o titulaciones al EEES, en los que se pueden encontrar una gran cantidad de trabajos sobre experiencias de adaptación real de asignaturas o titulaciones en multitud de estudios de distinta naturaleza (Derecho, Informática, Química,...) al EEES. Uno de los objetivos generales de este tipo de actividades consiste en revisar la práctica docente en este modelo educativo, haciendo especial hincapié en cómo mejorar el proceso de aprendizaje, las metodologías pedagógicas o los proyectos educativos.

Esta problemática está vinculada con dos líneas de trabajo previamente introducidas: la docencia y la administración, claves para garantizar el éxito en el proceso de adaptación al EEES. Sin embargo, otros aspectos clave, como el software educativo empleado y su papel en el nuevo paradigma, no están recibiendo toda la atención que realmente requieren y esto se hace especialmente patente en el caso de la enseñanza de la Informática.

Las herramientas software educativas han ido adquiriendo cada vez un mayor peso en los procesos formativos de las universidades [4], tanto para su uso en el proceso de enseñanza-aprendizaje por parte del profesor, con el objetivo de hacer su trabajo más entretenido y provechoso, como por parte del estudiante para facilitar el aprendizaje autónomo. El proceso de adaptación al EEES no hace sino potenciar esta realidad.

En este artículo se destaca la importancia del software educativo en el nuevo paradigma y se discute una propuesta de diseño y desarrollo del mismo teniendo en cuenta el nuevo paradigma, considerando los nuevos roles que juegan profesor y estudiante en este nuevo contexto, y planteando cómo se podrían integrar estos aspectos en el proceso de aprendizaje.

El resto del artículo continúa con una sección en la que se analizan de las implicaciones que tiene el nuevo paradigma educativo, y cuál es el papel que juegan profesor y estudiantes en este nuevo paradigma. Seguidamente se presentará una propuesta de diseño y desarrollo del software educativo en consonancia con lo expuesto anteriormente. En la sección siguiente se apuntan algunos casos de estudio que suponen ejemplos de herramientas diseñadas teniendo la propuesta formulada. Finalmente, se extraen algunas conclusiones y se plantearán vías de trabajo futuro.

2. Aprendizaje Activo en el EEES

En la actualidad, el relativamente nuevo paradigma de educación, derivado del proceso de Bolonia, tiene como objetivo fundamental la transición de un modelo basado en la enseñanza magistral a un modelo basado en el aprendizaje activo por parte del estudiante. En este contexto, se pretenden que la adquisición de competencias, habilidades y destrezas complementen el aprendizaje de conocimiento, al mismo tiempo que el estudiante basa gran parte de su aprendizaje en la búsqueda de información de manera autónoma y en la realización de actividades que fomenten el aprendizaje por descubrimiento, sin depender totalmente de la información suministrada. No obstante, y aunque el estudiante pasa a tener un rol fundamental, el profesor sigue jugando un papel dinamizador y estratégico.

En el nuevo paradigma de enseñanza, el profesor ha de evolucionar hacia el papel de guía o tutor, adaptando la enseñanza a las necesidades del estudiante y en coordinación con el resto de profesores. Así pues, se pretende reducir la carga de las sesiones magistrales e invertir más tiempo en la realización de actividades que fomente e incentiven la participación de los estudiantes. De este modo, el aprendizaje no se debe contrastar únicamente con la evaluación de los contenidos, sino que también ha de contemplar las competencias y habilidades adquiridas por los estudiantes [4].

Una reflexión importante que hay que hacerse después de analizar el nuevo contexto que establece el EEES y que tiene una implicación directa sobre cómo deben ser las herramientas informáticas educativas, es la existencia de dos tipos de usuarios, el estudiante y el profesor. Cada

uno de ellos con sus propios requisitos y ambos con la misma importancia a la hora de construir una herramienta útil.

A continuación se analizará el rol que juega cada uno en este nuevo paradigma, resaltando aquellas características que tienen una implicación directa sobre cómo deben ser las aplicaciones informáticas educativas que se usen en este marco.

2.1. Rol del Profesor

Las herramientas de software educativo deben ser un medio que facilite el alcance de los objetivos planteados en el nuevo paradigma educativo. Desde el punto de vista del profesor, estas herramientas deberían ofrecer la posibilidad de planificar y soportar actividades orientadas a la adquisición de competencias. Asociada a cada una de estas actividades, las herramientas deben permitir al profesor la inclusión de contenidos de diversa naturaleza.

Por otro lado, es fundamental que la herramienta facilite al profesor el seguimiento personalizado de cada alumno y esto implica conocer el grado de participación de cada uno de ellos, si realmente están aprendiendo de forma adecuada (en función de los resultados obtenidos en las pruebas planteadas) o detectar las principales necesidades en caso contrario. En este proceso es fundamental una comunicación apropiada entre profesor y alumno. Las herramientas educativas deben proporcionar mecanismos de comunicación asíncronos y síncronos. No sólo la comunicación entre profesor-alumno, sino también entre profesores de diferentes asignaturas. En el nuevo paradigma educativo se fomenta la planificación de actividades interdisciplinares y para ello es fundamental una correcta comunicación entre los profesores.

Finalmente, el seguimiento de grupos numerosos de estudiantes se convierte en una tarea complicada en la mayoría de las ocasiones; sobre todo cuando el número de profesores implicados es reducido. La herramientas deberían disponer de un componente de aprendizaje que tuviera en cuenta experiencias anteriores para ayudar al alumno y guiarlo en el proceso de aprendizaje. De esta forma se elimina carga de trabajo al profesor en tareas automatizables, y éste puede invertir más tiempo en aquellas tareas en donde su colaboración es fundamental.

2.2. Rol del estudiante

En cuanto al estudiante, ya se ha mencionado anteriormente que éste debe adoptar un enfoque activo y ser partícipe de su propio proceso de aprendizaje con la ayuda/guía del profesor. Para ello, las herramientas educativas deben facilitar al estudiante la posibilidad de realizar actividades muy variadas (y no sólo registrar los resultados) que le ayuden en el aprendizaje autónomo. Es decir, las nuevas herramientas deben permitir al estudiante realizar las tareas propuestas por el profesor, y también proponer nuevos problemas, introducir las soluciones a estos y recibir una retroalimentación inmediata por parte del sistema (autoevaluación). Lógicamente, la respuesta rápida agilizará el proceso de aprendizaje.

La comunicación no debe ser exclusiva entre profesor-estudiante, sino que se deben proporcionar los medios adecuados para que se produzca el intercambio de experiencias entre estudiantes. El aprendizaje cooperativo afianza en mayor medida los conocimientos adquiridos, ya que el estudiante es un participante directo y asume cierta responsabilidad en el proceso de aprendizaje.

2.3. Software educativo en EEES

La transición hacia un modelo orientado al aprendizaje de los estudiantes ha de estar soportado por software educativo que fomente y estimule su utilización con el objetivo final de mejorar el proceso de aprendizaje. En este contexto, el desarrollo de herramientas software que se encuadren en los niveles altos de la taxonomía de Bloom [1] es imprescindible, es decir, herramientas software que fomenten la evaluación y la comparación de las distintas soluciones ideadas por los estudiantes.

El uso de herramientas software que, de manera automática o semi-automática, permitan la corrección y evaluación de soluciones por parte de los alumnos es sin duda un paso fundamental hacia el autoaprendizaje. Además, este tipo de herramientas software se pueden orientar hacia el concepto de interactividad, es decir, proporcionando a los estudiantes mecanismos para comparar las distintas soluciones existentes para un mismo problema. El nivel de automatismo supone una complejidad añadida al desarrollo de este tipo de herramientas software pero, sin embargo, permite aprovechar de manera eficiente

tanto los recursos del profesor y como los de los estudiantes.

3. Software educativo para el aprendizaje activo: una propuesta metodológica

Como parte de nuestra propuesta, lo que primero sintetizamos son las características generales que pensamos deben tener dichas aplicaciones en el contexto que establece el EEES para ejercer correctamente cada uno de los roles, posteriormente, se apuntan los componentes fundamentales que requieren para soportarlas y, finalmente, la metodología para llevar a cabo el diseño de las herramientas.

3.1. Características

Para establecer las características que creemos deben poseer las herramientas informáticas educativas que se usen en el EEES, además de analizar lo que demanda cada rol, también se han analizado los servicios que ofrecen una serie de herramientas que existen en la actualidad para la enseñanza y el aprendizaje:

1. *Disponibilidad Física*: Debería poder usarse en cualquier momento, es decir el tiempo en el que la herramienta esté disponible para trabajar con ella debe ser las 24 horas del día.
2. *Disponibilidad Geográfica*: Debería poder usarse en cualquier dispositivo, para facilitar su uso en clase, en laboratorio, en el despacho, en casa o allá donde el estudiante aprenda (bibliotecas, salas de estudio,…).
3. *Fácil de Mantener*: Debería ser fácil de actualizar o reparar. Puesto que se pretende que estas aplicaciones se puedan usar en presentaciones de clase, laboratorios y en los propios ordenadores de estudiantes y profesor, un cambio en las aplicación debe ser fácilmente trasladable a cada uno de ellos.
4. *Control de accesos y uso:* Debería permitir ver el trabajo que cada estudiante realiza con la aplicación, y permitir la monitorización del progreso en el aprendizaje.
5. *Interfaz amigable, intuitiva y potente*: Debería ser fácil de interactuar con ella (por parte de ambos tipos de usuarios). Aunque con esto no estamos diciendo que no pueda requerir un pequeño entrenamiento inicial. Por otra parte la información generada se debe de mostrar de la manera más adecuada dependiendo de cómo se vaya a usar. Y por último, y no menos importante, debe de ser lo suficientemente potente para establecer una comunicación sin ambigüedad.
6. *Correcta*: Debería proporcionar una información precisa y libre de errores. Esta es una característica común a cualquier software.
7. *Completa*: Debería proporcionar toda la información necesaria para que se comprendan sus salidas (p.e. las soluciones a ejercicios o las correcciones realizadas,...).
8. *Eficiente*: Debería proporcionar la información requerida de manera rápida.
9. *Potencie la autonomía*: Debería disponer de medios para en función de los resultados obtenidos por usos previos oriente al estudiante en su proceso de aprendizaje.
10. *Fácil de integrar*: Debería poder integrarse en sistemas de gestión de aprendizaje reales mediante la utilización de los estándares que se manejan para los mismos (particularmente, IMS-LD [6]).
11. *Disponibilidad de un sistema de comunicación:* Debería contar con un mecanismo que permita comunicarnos con la aplicación sin ambigüedad. La información dada como entrada debe tener un único significado sin cabida a interpretaciones erróneas.
12. *Disponibilidad de un sistema para el intercambio de experiencias, información, conocimiento*: Debería proporcionar un mecanismo que permita el intercambio entre estudiantes de experiencias de aprendizaje o para transmitir conocimiento o información por parte del profesor al estudiante.
13. *Plataforma Libre*: Debería poder ejecutarse en cualquier plataforma y Sistema Operativo.

3.2. Componentes fundamentales

En esta sección proponemos unas ideas que, a nuestro entender, deberían considerarse en el diseño y desarrollo de aplicaciones informáticas educativas con las características antes expuestas. La principal funcionalidad de estas herramientas es que, permitan por una parte a los estudiantes un aprendizaje autónomo permitiéndoles especificar sus propios problemas, introducir soluciones alternativas a los mismos y recibir una autoevaluación y feedback por parte del sistema

que los guíe en su proceso de aprendizaje, y por otra a los profesores les de soporte para ayudar a aprender y comprobar si el estudiante aprende bien o no.

Arquitectura: Para satisfacer los requisitos 1, 2, 3, 13 y parte del 4, se ha tomado la decisión de recomendar aplicaciones centralizadas usando una arquitectura cliente/servidor web. En la que toda la carga de trabajo recaerá cobre el servidor, haciendo que la aplicación en el lado del cliente sea muy ligera, únicamente necesitará un navegador web conectado a internet. Esto hace que no se necesite instalaciones locales. Por otra parte, al ser una aplicación centralizada se puede controlar fácilmente los accesos y usos de la herramienta, además se puede incorporar con facilidad repositorios centralizados de materiales docentes.

Comunicación e intercambio de información: Para satisfacer los requisitos 11 y 12, proponemos el uso de un ***Lenguaje Formal*** que nos permitirá comunicarnos con la herramienta de manera precisa. Este lenguaje nos debe permitir especificar el problema para el que el estudiante desea obtener una resolución correcta. Además, debe permitir al estudiante especificar la solución que él propone para el mismo. Obviamente, este lenguaje es dependiente del dominio o más formalmente podemos decir que es un Lenguaje Específico de Dominio (DSL) y debe ser diseñado después de un estudio o adquisición de conocimiento del dominio. El diseño de un DSL no es una tarea sencilla y puede que incluso no esté justificado. Por otra parte, este lenguaje nos va a permitir intercambiar experiencias, conocimiento o información gracias al uso de archivos escritos en este lenguaje.

Procesador del Lenguaje: Para procesar el Lenguaje Formal anterior, adquirir la información necesaria para resolver el problema y evaluar la solución propuesta por el estudiante necesitamos un procesador de dicho lenguaje. Este procesador debe de validar la cadena de entrada en sus aspectos léxicos (vocabulario correcto), sintácticos (estructuras correctas) y semánticas (entradas con sentido). Además, debe recopilar aquella información en la entrada necesaria para poder obtener soluciones y comprobar soluciones.

Resolvedor de Problemas: Para satisfacer los requisitos 6, 7 y 8 se propone el diseño de algoritmos que permitan resolver los problemas del dominio en el que se usa la aplicación, así como para explicar o justificar cómo es alcanzada la solución. Para esto último, deberá proporcionar la salida correspondiente con toda la información necesaria para que el estudiante comprenda y pueda analizar la solución generada por el sistema. Empleará la información recopilada por el Procesador del Lenguaje para invocar a aquellos algoritmos con capacidad para obtener las soluciones a los problemas dados como entrada, y para comprobar la bondad de las soluciones aportadas por el estudiante con respecto a la obtenida por el sistema.

Instructor de Aprendizaje: Para satisfacer el requisito 4 (en lo que respecta a monitorización del trabajo realizado por el estudiante) y 9, se propone el diseño de módulos instructores de aprendizaje cuya misión va ser dar feedback al estudiante, destacando lo que ha aprendido o aquello en lo que ha fallado, en base, al resultado de la comparación realizada por el Resolvedor de Problemas. Esta información será empleada para guiar al estudiante en su proceso de aprendizaje. Este módulo o componente debe tener unos modelos o caminos de aprendizaje que establezcan por donde deben transitar los estudiantes en el proceso de aprendizaje.

Presentador de la información generada: Para satisfacer el requisito 5, se propone el diseño de presentadores o formateadotes de información, cuya misión va a ser la de crear la vista más apropiada en función de la salida obtenida por el sistema. Dado que uno de los propósitos de la técnica es que el estudiante pueda experimentar con la salida, el presentador podría generar animaciones o dotar al estudiante de instrumentos de simulación. De este modo, se facilitaría la comprensión de los conceptos de una manera más eficiente y rápida. Este módulo trabajará con la información que le proporciona el Instructor (parte de la cual, a su vez fue pasada por el Resolvedor de Problemas).

Integrador de información: Para satisfacer el requisito 10, se propone el diseño de un módulo que convierta la información a manejar siguiendo lo establecido en IMS-LD [6], generando y planificando actividades particularizadas para cada estudiante que puedan ser recogidas y gestionadas en LMSs que lo soportan como es el caso de CopperCore[1] o DotLRN[2]. No se menciona

[1] http://coppercore.org
[2] http://openacs.org/projects/dotlrn/

Moodle ya que no tiene versión estable de soporte a IMS-LD, es decir, lo que se conoce como "*player*" de IMS-LD.

3.3. Metodología de diseño

En esta sección se explican los pasos que se tendrían que dar para diseñar aplicaciones como la que se propone en este trabajo.

Primer Paso: **Estudio de la viabilidad de la solución empleando DSL** [10]. Hay que tener en consideración dos factores a la hora de afrontar la decisión de crear una aplicación basada en las ideas presentadas en este trabajo:

- ¿Son resolubles los ejercicios de manera algorítmica?, es decir existen o se pueden diseñar e implementar algoritmos para solucionar en un tiempo razonable el ejercicio dado como entrada. Si no es posible, no se pueden aplicar las ideas sugeridas a este problema.
- ¿Es formalizable la entrada?, es decir, hay que determinar si se puede formalizar la escritura de ejercicios y soluciones en el problema que se está abordando. Si no se puede, la propuesta no es válida para el campo.

Segundo Paso: **Creación del DSL**. En el desarrollo de un DSL pueden distinguirse 2 fases:

Fase de Análisis. En esta fase hay que identificar el dominio y obtener el conocimiento necesario de las fuentes disponibles. Como producto de esta fase hay que obtener una terminología específica del dominio junto con la semántica expresada de manera más o menos abstracta.

Fase de Diseño. En esta fase se diseña el DSL, para ello habrá que definir su vocabulario (aspecto léxico), establecer las reglas que permiten construir sentencias válidas para el lenguaje usando elementos del vocabulario (aspecto sintáctico) y establecer el significado de las frases y aquellas comprobaciones que establecen la validez de las frases con estructura correcta (aspecto semántico). Como producto de esta fase habría que obtener una descripción, empleando una notación formal, del lenguaje, por ejemplo usando un metalenguaje como Extended Backus-Naur Form (EBNF); además debería obtenerse una tabla de elementos del vocabulario (o tokens) y una definición de la semántica del lenguaje. El diseño debe de hacerse teniendo en cuenta una serie de propiedades deseables en el lenguaje diseñado:

- Claridad, simplicidad y unidad de conceptos. El lenguaje debe de suministrar a su usuario un conjunto de conceptos claros, simples y unificados.
- Sintaxis y semántica bien definidas. Con unas reglas claras se asegura la no ambigüedad y completitud del lenguaje.
- Consistencia con las notaciones usuales. El lenguaje debe poseer elementos que signifiquen lo que deben significar en el campo.
- Legibilidad. El lenguaje debe de ser fácilmente leído y entendido.
- Costo de uso reducido. El lenguaje debe de ser fácil de aprender y usar para potenciar su uso. Es una propiedad muy relacionada con la anterior.

Para facilitar esta fase, el diseño se puede realizar en base a un lenguaje existente en el campo.

Tercer Paso: **Diseño e implementación del Procesador del DSL**. El diseño del procesador se debe hacer como es usual en fases, distinguiendo entre el front-end y back-end, para facilitar de este modo la modularidad. En el front-end se analizará el texto de entrada, comprobando su validez (léxica, sintáctica y semántica) y recopilando la información necesaria de la entrada para poder resolver los ejercicios así como las soluciones que son aportadas por los estudiantes. En la fase de back-end es donde se van a invocar los algoritmos que resuelven los ejercicios y comprueban las soluciones con la información recopilada en la fase anterior. La implementación del mismo se va a poder realizar empleando herramientas de generación automática, como la pareja ampliamente conocida Lex y Yacc [9] o la herramienta ANTRL[3] (ANother Tool for Language Recognition) [11] que serán empleadas para obtener el Scanner y el Parser con acciones semánticas del lenguaje diseñado, otras alternativas son usar herramientas que faciliten la creación, edición y mantenimiento de programas escritos en un DSL [12] o incluso la propuesta que ofrece Microsoft para el desarrollo de DSL denominada DSL Tools [2].

[3] http://www.antlr.org

Cuarto Paso: **Diseño e implementación de los algoritmos**. Tres fases pueden ser identificadas en este paso:

Fase de identificación. En esta fase hay que determinar con claridad los ejercicios que se pretende que solucione la herramienta.

Fase de resolución de los ejercicios. Se tendrá que establecer el método que nos lleva a la solución de cada tipo de ejercicio, es decir la sucesión de pasos finitos que hay que dar y en que orden para alcanzar la solución.

Fase de implementación en un lenguaje de implementación.

Quinto Paso: **Diseño del Instructor de Aprendizaje**. Para poder diseñar el Instructor hay que tener claro de que situación se parte (es decir, que es lo que el estudiante sabe), que es lo que se quiere conseguir (es decir, establecer los objetivos de aprendizaje), como se ordenan y se relacionan esos objetivos (es decir, la secuenciación del proceso de aprendizaje), y por último se debe establecer como evaluar el cambio que se produce. Para poder implementar este módulo, se necesita un mecanismo para almacenar la evolución del estudiante durante el proceso de aprendizaje y método de aprendizaje a aplicar siguiendo un diseño de aprendizaje [5][8]. Otra alternativa podría ser emplear una solución basada en máquinas de estado finito para modelar la secuenciación de objetivos en el proceso de aprendizaje con cambio de estados cuando ocurran cambios en el nivel del estudiante.

Sexto Paso: **Diseño del Presentador de información**. El diseño de este módulo implica determinar que información se va a generar, establecer cual va a ser el uso de cada una de ellas y en función de ello elegir el formato de presentación más adecuado, generando animaciones o dotando a la herramienta de mecanismos que permitan al estudiante interactuar por medio de la simulación con las soluciones obtenidas.

Séptimo Paso: **Integrar la aplicación en el sistema de gestión de aprendizaje usado**. En este paso, básicamente, se abordará el diseño de un motor de ejecución de diseños instruccionales con soporte para IMS-LD.

4. Casos de estudio

En esta sección se muestran algunas herramientas que han sido diseñadas e implementadas siguiendo la propuesta y que se aplican a la enseñanza universitaria de distintas materias fundamentales de la Informática.

Proletool[4] está destinada a asistir a los estudiantes en el proceso de aprendizaje de la materia Procesadores de Lenguajes (PL), así como a servir de ayuda a los profesores en sus explicaciones o incluso para delegar en ella trabajo (corrección de ejercicios mecánicos). Esta herramienta se emplea para facilitar la enseñanza y aprendizaje de las técnicas de análisis léxico, sintáctico (técnicas LL1, SLR1, LR1, LALR1) y la relación entre ambas. La versión en desarrollo posee todos los componentes.

Selfa-Pro[5] es una aplicación para facilitar la enseñanza-aprendizaje de la materia Teoría de Autómatas y Lenguajes Formales. El objetivo de la misma es completar los materiales que existen en la actualidad en la materia, con el fin de ayudar a entender la relación existente entre autómatas y gramáticas. Esta herramienta resuelve ejercicios sobre autómatas finitos, expresiones regulares, gramáticas regulares, autómatas con pila y gramáticas libres de contexto. Además se pueden realizar simulaciones sobre los autómatas finitos y los autómatas con pila. La herramienta permite que los estudiantes se entrenen en todos los contenidos de la materia, a excepción de las máquinas de Turing, mejorando la calidad tanto en el aprendizaje como en la enseñanza, ya que el profesor puede emplearla para explicar la mayoría de conceptos y algoritmos de la asignatura. La versión actual de la herramienta no permite especificar soluciones a los ejercicios y el Revolvedor de Problemas por tanto no corrige ejercicios, únicamente resuelve. Como consecuencia tampoco posee Instructor de Aprendizaje.

COALA[6] es un entorno que soporta el aprendizaje práctico de la Programación, utiliza técnica de lógica difusa para valorar los algoritmos y programas construidos y recomienda, de forma personalizada, problemas específicos para cubrir carencias observadas o para seguir

[4] http://portal.esi.uclm.es/proletool/
[5] http://portal.esi.uclm.es/selfa/
[6] http://chico.esi.uclm.es/coala/

profundizando en el aprendizaje. COALA incorpora un módulo específico para integrarse en base a IMS-LD con CopperCore (utilizando EJB/WebServices) y con DotLRN (utilizando WebServices).

Y por último destacar la herramienta *Electronics*[7] diseñada para facilitar la enseñanza y aprendizaje de la resolución de circuitos electrónicos por los métodos de malla, empleando las ecuaciones de la ley de mallas de Kirchhoff. Esta herramienta está en su fase inicial de desarrollo, existe un prototipo disponible en aplicación de escritorio que implementa todos los módulos, a excepción del *Instructor de Aprendizaje*, obviamente (al no estar disponible vía web) tampoco está integrada con ningún sistema de gestión de aprendizaje.

5. Conclusiones

En este artículo se ha destacado la importancia del software educativo las metodologías educativas activas que se aplican en el contexto que establece EEES. Se ha presentado una propuesta sobre cómo creemos que debería diseñarse y desarrollarse dicho software, y cómo debería integrarse en el proceso formativo. Este software permite a los estudiantes especificar sus propios ejercicios, introducir soluciones alternativas a los mismos y recibir una autoevaluación por parte del sistema que los guíe en su proceso de aprendizaje. Su aplicabilidad a un caso particular depende de las características del dominio.

Los dominios en los que son aplicables son aquellos en los que existen o se pueden diseñar e implementar algoritmos para solucionar en un tiempo razonable el ejercicio dado como entrada. Si no es posible, no se pueden aplicar las ideas sugeridas a este problema. Además estos deben permitir formalizar la escritura de ejercicios y soluciones en el problema que se está abordando por medio de un lenguaje formal. Si no se puede realizar tal formalización, entonces la propuesta no es válida para ese caso.

Referencias

[1] Bloom, B. S., *Taxonomy of educational objectives, handbook 1: Cognitive domain*. New York: Longmans Green, 1956.

[2] Cook, S., Jones, G., Kent, S., Wills, A., *Domain-specific development with visual studio DSL tools*, Addison-Wesley Prof., 2007.

[3] Crosier, D., Purser, L., Smidt, H., Trends V: *Universities shaping the European higher education area*, European University Association Brussels, 2007.

[4] Esteve, F., *Bolonia y las TIC: de la docencia 1.0 al aprendizaje 2.0*, La cuestión universitaria, 5(2009):59-68.

[5] IMS Global Learning Consortium, *IMS Guidelines for Developing Accesible Learning Applications: Version 0.6 White Paper*, 2001.

[6] IMS Global Learning Consortium. Information Model, Best Practice and Implementation Guide, XML Binding, Schemas. Final Specification, 2003

[7] Joint Quality Initiative, *Shared Dublin descriptors for the Bachelor's, Master's and Doctoral awards*, http://www.jointquality.nl/content/descriptors/CompletesetDublinDescriptors.doc, 2004.

[8] Koper, R., Tattersall, C., *Learning design: A handbook on modelling and delivering networked education and training*, Springer-Verlak New York, 2005.

[9] Levine, J.R., Mason, T., Brown, D., *Lex & Yacc*, O'Reilly Media, 1992.

[10] Mernik, M., Heering, J., Sloane, A.M., *When and how to develop domain-specific languages*, ACM Computing Surveys, 37(4):316-344, 2005.

[11] Parr, T., *The definitive ANTLR reference*, The Pragmatic Bookshelf, 2007.

[12] Wächter, B., *The Bologna Process: development and prospects*, European Journal of Education, 39(3):265-273, 2004.

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