Sistemas Informáticos
Curso 2008 - 2009

Un Tutor Inteligente para la Visualización de Métodos Algorítmicos y Estructuras de Datos

Pablo Fernández Poblaciones

Salvador Muñoz Sánchez

Antonio Murillo Melero

Dirigido por Rafael del Vado Vírseda

Departamento de Sistemas Informáticos y Computación

Facultad de Informática

Universidad Complutense de Madrid

Agradecimientos

Nos gustaría agradecer la ayuda y el apoyo que hemos recibido de distintas personas durante la realización de este proyecto:

- En primer lugar, a Rafael del Vado Vírseda, director de nuestro proyecto, por su inquebrantable paciencia, dedicación, buen humor y ayuda.

- A Juan Artalejo Hortalá, por todo el trabajo que le hemos dado con la parte de seguridad en el servidor gpd.

- A nuestros amigos y familiares por su apoyo.

Página de autorización

Se autoriza a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Madrid, 3 de Julio de 2009

Fdo. Pablo Fernández Poblaciones

Fdo. Salvador Muñoz Sánchez

Fdo. Antonio Murillo Melero

Índice

Resumen del proyecto

En castellano

Este proyecto representa el desarrollo de una herramienta educativa que unifica las ya existentes, las cuales han sido desarrolladas en años pasados en la asignatura de Sistemas Informáticos, Vedya y Vedya-Test, a través de una sola herramienta capaz de controlar la evolución pedagógica del alumno en la asignatura de “Estructura de Datos y de la Información”.

La herramienta consta de dos partes, Vedya-Alumno y Vedya-Profesor, las cuales permiten un correcto aprendizaje de dicha asignatura por parte del alumno y un control de la evolución de estos por parte del profesor.

En inglés

This project represents the development of an educational tool that unify the existents tools, developed years before in the “Computer Systems” subject, Vedya and Vedya-Test, in one tool that is able to control the teaching progress of the student in the subject “Data Structures and Information”.

The tool consist of two parts, Vedya-Alumno and Vedya-Profesor, which allows a correct learning of this subject by the student and a control of the progress of them by the professor.

Palabras Clave

- *Tutor*

Persona encargada de asegurar que la educación sea verdaderamente integral y personalizada y no quede reducida a un simple trasvase de conocimientos.

- *Inteligente*

Dícese de aquel que esta dotado de inteligencia. Sabio, perito, instruido.

- *Aprendizaje*

Es la adquisición de nuevos conocimientos, conductas, habilidades, valores o pensamientos, a partir de determinada información percibida.

- *Educación*

El proceso multidireccional mediante el cual se transmiten conocimientos, valores, costumbres y formas de actuar.

- *Conocimiento*

Es el conjunto organizado de datos e información que permiten resolver un determinado problema o tomar una decisión.

- *Motivación*

Son los estímulos que mueven a la persona a realizar determinadas acciones y persistir en ellas para su culminación.

- *Estructura de datos*

Es una colección de datos cuya organización se caracteriza por las funciones definidas utilizadas para almacenar y acceder a elementos individuales de datos. Las estructuras de datos pueden descomponerse en los elementos que la forman. La manera en que se colocan los elementos dentro de la estructura afectará la forma en que se realicen los accesos a cada elemento.

1. Nuevas tecnologías en educación

1.1. Objetivos de los Sistemas de Educación

Los objetivos de los sistemas de educación son los siguientes:

- Fomentar la participación del alumno durante todo el proceso de aprendizaje.
- Incrementar el flujo y la cantidad de información recibida.
- Incentivar el aprendizaje interactivo y activo.
- Modificar las estructuras de pensamiento del alumno.
- Ayudar a descubrir la naturaleza de las materias.

1.2. Panorama histórico

En el desarrollo de los Sistemas de Educación se consideran diversas etapas que a continuación iremos explicando.

En primer lugar, se distingue la etapa inicial de los sistemas llamados **CAI** (del inglés, *Computer Aided Instruction*)[5]. Los tradicionales programas de este tipo se centraban en representar la estructura de la materia a enseñar y transmitir dicha estructura siguiendo los métodos tradicionales de enseñanza. Estos programas pueden considerarse como los descendientes evolutivos de los libros de texto, ya que al igual que ellos, están organizados estáticamente de tal forma que contienen tanto el dominio de conocimiento como el conocimiento tutorial de los maestros, como expertos humanos.

En los libros se encuentras algunas ayudas, como por ejemplo: secciones, índices, capítulos, tablas y figuras; son las herramientas con las que se facilita la presentación de un tema, y los autores tienen conocimiento tanto del dominio de lo que hay que comunicar como en la escritura del propio libro.

Un libro puede propiciar varios niveles de lectura, proporcionar referencias cruzadas entre secciones y suministrar glosarios adecuados, siempre y cuando el autor lo haya indicado en el texto. Un libro no puede responder a preguntas inesperadas por el lector debido a su naturaleza estática. Tampoco puede modificarse “al vuelo” el conocimiento presentado, de tal forma que el mismo libro se adaptase a un lector específico. Las limitaciones propias del medio de impresión no permiten flexibilidad y dinamismo en el acceso al conocimiento contenido en él.

Volviendo a los CAI, los autores de programas hacen lo mismo: piensan con antelación en acciones educacionales, anticipan las circunstancias que

requieren decisiones y escriben el código apropiado que permita capturar tales decisiones. Por lo tanto, los programas CAI tradicionales se aprovechan de la experiencia tutorial de los maestros expertos y directamente reflejan esta habilidad en el comportamiento de los programas. Esta circunstancia hace potente el enfoque de estos sistemas, pero también es uno de sus principales defectos ya que pocos maestros pueden anticipar todos los errores de concepto que un estudiante puede llegar a adquirir. Además, es prácticamente imposible realizar programas de aplicaciones que contengan todas las decisiones imaginables. Aún cuando es verdad que los programas CAI, una vez que son desarrollados y probados pueden ser usados por un gran número de personas, también es cierto que son muy difíciles de modificar.

Las principales desventajas que presentan los programas CAI se pueden resumir de la siguiente manera:

- La calidad de los programas de aplicaciones está estrechamente relacionada, en muchos sistemas de tutorización dirigida, con la habilidad del autor para anticipar (e incluir en ellos) tantas respuestas del estudiante como sea posible y poder así especificar la trayectoria de tutorización que resulte más apropiada.

- Una estrategia de enseñanza, trasladada a líneas de código, no se ajusta a las necesidades específicas del estudiante.

- La autonomía que ofrecen los sistemas de tutorización más flexibles representan un obstáculo para aquellos estudiantes que no tienen la posibilidad de aprovechar esta autonomía.

- En los sistemas CAI no hay una definición explícita del objetivo de enseñanza, ni el sistema se comporta dinámicamente adaptando su funcionamiento a las características del usuario para alcanzar dicho objetivo.

- Los programas CAI, a menudo, son desventajosos en lo referente al costo, dado que desarrollar, y sobre todo mantener tales programas, implica la inversión de recursos de una magnitud considerable.

A continuación surgen los **Sistemas de Tutorización Inteligentes (STI)** [4,6,12], considerados los primeros sistemas basados en la aplicación de técnicas del campo de la Inteligencia Artificial en la educación. Estos sistemas seguían incidiendo en la utilización de métodos tradicionales de enseñanza (en los que el alumno seguía presentando una actitud más o menos pasiva). El alumno se enfrenta al proceso de aprendizaje mediante una guía de acciones previamente prefijadas que se corresponden con un modelo de autorización individual “uno a uno”. No obstante, es de destacar que en estos sistemas tomaban un papel relevante dos elementos que desde entonces están presentes en la mayoría de aplicaciones, estos elementos son:

- El **Modelo del Estudiante**. Se utiliza para representar lo que el sistema supone que el estudiante ha aprendido.

- El **Modelo Pedagógico**. Contiene el conocimiento referido a la forma de gestionar el propio proceso de aprendizaje.

El gran problema de estos sistemas es que estaban más centrados en el conocimiento que se quería transmitir que en el propio proceso de aprendizaje de dicho conocimiento. En cualquier caso, supusieron un avance considerable al establecer la necesidad de introducir el conocimiento y de modelar el comportamiento.

Con la intención de superar los problemas detectados en los STI surgieron los **Entornos de Aprendizaje Interactivo (EAI)** [7] y los **Micromundos** [7]. En este tipo de entornos, la inteligencia ya no está centralizada en un tutor, que actúa para garantizar que lo que aprende el alumno se corresponde con el modelo de un experto, en su lugar, se proporcionan diferentes herramientas que fomentan la investigación sin un riguroso control externo. En otras palabras, se realiza un planteamiento *constructivista* del aprendizaje frente a la formulación *conductista* de etapas anteriores. En estos sistemas predomina el uso de imágenes, vídeo y otras representaciones gráficas.

Más adelante, además del desarrollo de sistemas *mixtos* que incorporan diversos paradigmas, se produce una paulatina división del campo en diversas áreas de interés: aprendizaje colaborativo, modelado del usuario, interfaces adaptativos, aprendizaje a través de la Web, estandarización de contenidos y cursos, etc. Para poder integrar los diversos trabajos realizados en cada uno de estos campos, se plantea la necesidad de introducir mayor metodología y la de concretar formatos y estándares de control y gestión de este tipo de aplicaciones.

1.3. Métodos de Enseñanza/Aprendizaje

Para poder proporcionar un comportamiento inteligente en los sistemas interactivos de Enseñaza/Aprendizaje, el campo de la Inteligencia Artificial, se plantea el diseño de estos sistemas desde la perspectiva de ofrecer contribuciones a la solución de los problemas educacionales, referentes a las siguientes preguntas:

- ¿Cuál es la naturaleza del conocimiento?
- ¿Cómo debe ser aprendido?
- ¿Cuál es el papel de los tutores?
- ¿Deben instruir, tutorizar, guiar o entrenar a los estudiantes?
- ¿Cuáles son las medidas de efectividad?

Esta serie de cuestiones derivan de dos preguntas básicas, que son: *la naturaleza del conocimiento* y *el paradigma de aprendizaje*.

1.3.1 Naturaleza del conocimiento

En cuanto a la naturaleza del conocimiento se puede hablar de *constructivismo* en primer lugar. Se usa a favor del estilo de pedagogía centralizado en el estudiante ejemplificado en los EAIs. De acuerdo con este punto de vista, el conocimiento debe ser construido por el aprendiz paso a paso, de modo que en estos sistemas se ven como las herramientas ideales para poder potenciar esta construcción auto-guiada.

Siguiendo con las naturalezas podemos hablar de *situacionismo*. Comparte ciertos principios con el constructivismo pero pone un mayor énfasis en que el conocimiento construido no existe en la memoria sino que emerge de la interacción con el entorno.

Por último se puede hablar de *conexionismo*. Este punto de vista del conocimiento establece que el conocimiento está implícitamente representado en los pesos y enlaces entre un gran número de nodos modelados en redes neuronales.

1.3.2 Paradigma de aprendizaje

El diseño de sistemas educacionales refleja un enfoque bastante ecléctico en la naturaleza del aprendizaje. Muchos tipos diferentes de circunstancias y actividades pueden llevar al aprendizaje y muchos de ellos se han soportado, hasta cierto punto, dentro de estos sistemas. Así, se distinguen los siguientes paradigmas de aprendizaje:

- Aprendizaje basado en casos. La idea de este aprendizaje proviene del campo del razonamiento basado en casos en Inteligencia Artificial, es que los estudiantes aprenden situaciones (casos) si se les presentan en el punto preciso en que están interesados en conocer la información que la situación les transmite. Así, en lugar de aprender reglas abstractas para aplicar a situaciones, los estudiantes explotan las analogías encontradas en casos similares para sintetizar sus propias reglas de decisión.

- Aprendizaje orientado a fallos. Estos sistemas se basan en que la ocurrencia de fallos proporciona una oportunidad de aprender.

- Aprendizaje a través de la experimentación. Un escenario estándar es un entorno para la resolución de problemas donde los estudiantes realizan experimentos guiados por un sistema de interpretación.

- Aprendizaje basado en simulación. Se basan en el éxito de los simuladores de vuelo. Estos dispositivos proporcionan un ilimitado y muy barato tiempo de práctica para perfeccionar habilidades que serían muy costosa, lentas y peligrosas de adquirir usando un avión real.

- Aprendizaje a través del diálogo. Describen un entorno con el que los estudiantes intercambian argumentos durante un debate, mientras el sistema actúa como un árbitro, usando directrices de diálogo tomadas de la teoría de juegos para determinar la validez de los movimientos.

- Aprendizaje reflexivo. Se basa en la experiencia de la clarificación de ideas que puede derivarse de la discusión con compañeros de estudio, que tampoco tienen una comprensión absoluta de la materia bajo estudio, pero son capaces de formular preguntas que provocan una reestructuración del pensamiento.

- Aprendizaje visual. La visualización con frecuencia permite a los aprendices y profesionales obtener representaciones de un gran número de datos, para “generar intuiciones” y sugerir hipótesis para posteriores comprobaciones. En algunos casos se abren nuevos campos de estudio y se definen nuevos objetivos curriculares al tiempo que proporcionan nuevos métodos de aprendizaje.

- Aprendizaje en colaboración. Las tecnologías para colaboración basadas en computadores proporcionan nuevos métodos cooperativos de trabajo y aprendizaje.

2. Sistemas de Enseñanza / Aprendizaje

2.1. Estructura general de los sistemas de Enseñanza / Aprendizaje

Los sistemas de Enseñanza/Aprendizaje tienen la siguiente estructura modular:

- Modelo pedagógico.
- Modelo del alumno.
- Dominio de enseñanza.
- Aproximación a la adaptación.

2.1.1 Modelo pedagógico

Es una de las partes esenciales del sistema. En ella se materializa el conocimiento pedagógico estratégico sobre el proceso de instrucción. La representación de este conocimiento puede ser reutilizada en diferentes dominios.

Aun siendo una parte esencial del sistema, muchas aplicaciones tienen una funcionalidad muy limitada. La razón es la dificultad de recoger con precisión dicho conocimiento. El objetivo sería crear una especie de sistema experto que fuera capaz de resolver cualquier tipo de eventualidad en el proceso de aprendizaje (por ejemplo, determinar y representar todos los errores que podría cometer un alumno y las acciones que el tutor puede realizar para remediar dichos errores). Esa postura, adoptada en la mayoría de los STIs, se transformó paulatinamente en propuestas centradas en entornos interactivos, donde el control sobre el proceso de aprendizaje se relajaba notablemente y se ponía mayor énfasis en la libertad de indagación por parte del alumno. Esto hacía perder en cierto modo la confianza en la validez de estos sistemas como mecanismos para la *educación autónoma*, aunque fuera mínimo.

En el modelo pedagógico se distinguen **decisiones globales**, como la secuencia de etapas en la instrucción y **decisiones locales**, como por ejemplo decidir si se debe o no interrumpir a un estudiante en la realización de una tarea.

La rigidez o no del modelo pedagógico determinará el grado de control sobre las actividades realizadas. Así se puede distinguir: **control del sistema**, **control mixto** (estudiante y sistema), **descubrimiento guiado**, **asistencia**, etc.

2.1.2 Modelo del alumno

El modelo del usuario es una parte fundamental en los sistemas que personalizan las respuestas que proporcionan a sus usuarios. Un modelo de usuario es una representación explícita de las propiedades de un usuario específico. Se utiliza para razonar acerca de las necesidades, preferencias o comportamiento futuro del usuario, así como para diagnosticar fallos en su interacción con el sistema.

La construcción de este modelo es dinámica y en el caso de los sistemas para la educación debe cubrir todos los aspectos del alumno que puedan repercutir en su aprendizaje.

Aparte de la información menos volátil, como puedan ser los datos personales, datos académicos, experiencias previas, etc. el gran problema que se aborda en la definición de este modelo es determinar cuál es el estado del conocimiento del alumno con respecto al conocimiento del dominio que se está aprendiendo.

La determinación del estado del conocimiento del alumno es un **problema de diagnostico**. Se trata de establecer, a partir de las evidencias existentes (datos recogidos en la interacción; recordemos que cualquier interfaz es limitada), cuáles son los aciertos y los fallos existentes en dicho estado. Las opciones para tratar este problema son múltiples. Por ejemplo se pueden codificar de antemano errores típicos que simplifiquen la identificación del problema. Por otro lado, la opción más utilizada es la de establecer una correspondencia entre la red de nodos que representa el conocimiento aprendido del alumno y la del conocimiento experto del dominio. Este modelo se denomina **overlay** o **modelo superpuesto**. En este caso, el conocimiento del alumno se trata como un subconjunto del conocimiento del experto. Evidentemente, aunque este modelo se haya utilizado para aproximar la solución del problema, no refleja la mayoría de las situaciones de aprendizaje en las que el alumno puede tener conocimiento en cantidad y en calidad muy distinta a la del experto.

En general existen varios métodos para representar el conocimiento del alumno, algunos de los más importantes son:

- *Modelo overlay*. Aunque ya lo hemos comentado anteriormente, simplificando, los modelos overlay tratan el conocimiento del estudiante como un subconjunto del conocimiento de un experto. Es el método más utilizado.

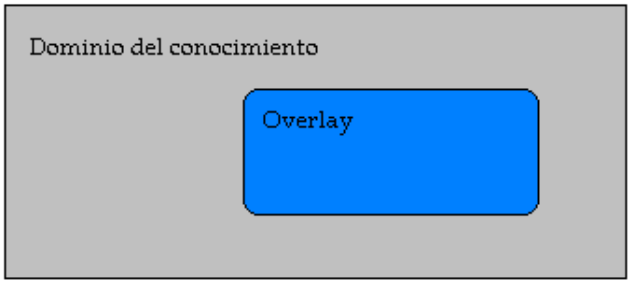


Figura 1. Modelo overlay

- *Modelo diferencial*. Este modelo es una modificación del modelo overlay. Divide el conocimiento del estudiante en dos categorías, que son: conocimiento que el estudiante debería conocer y conocimiento que el alumno no conoce.

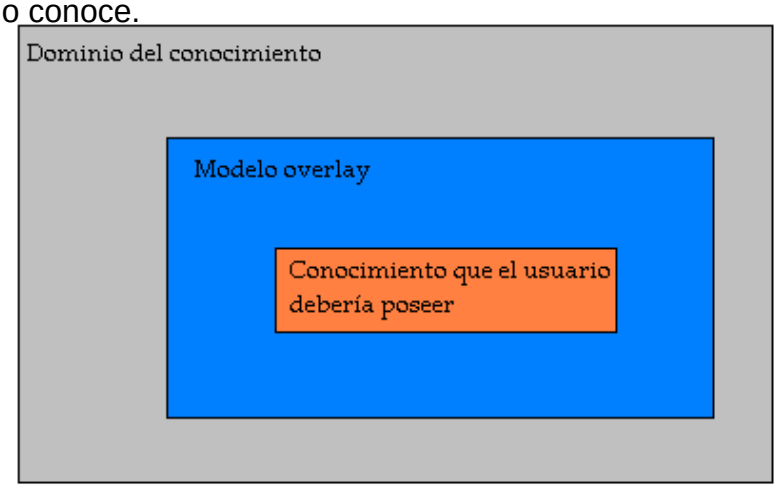


Figura 2. Modelo diferencial

- *Modelo de perturbación*. En esta representación se supone que el alumno posee conocimiento potencialmente diferente en cantidad y calidad respecto al de un experto. En el modelo de perturbación también puede representar el conocimiento y creencias del estudiante más allá del rango del modelo del experto.

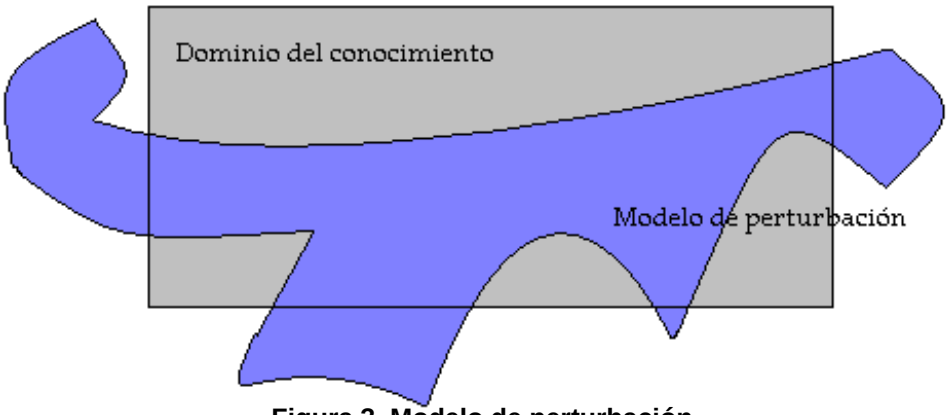


Figura 3. Modelo de perturbación

- *Modelo de estado vs. proceso*. Existen otros métodos para la representación del conocimiento del estudiante, los cuales utilizan deducciones para generar predicciones. Como ejemplo, se puede citar el modelo de estado vs. proceso, el cual puede ser visto como la capacidad de simular el proceso por el cual un usuario elige consultar material. Con este tipo de modelos debería ser posible predecir el material que el usuario consultará posteriormente.

2.1.3 Dominio de enseñanza

El conocimiento es objeto del aprendizaje. Actúa como una fuente de la información que se le presenta al alumno. Por ejemplo, generando explicaciones y respuestas al estudiante. También se utiliza para medir el conocimiento aprendido por el alumno. En este caso, hay que establecer un método de comparación entre ambos.

El conocimiento del dominio se puede organizar en una red de nodos en la que se representan conceptos, objetos, relaciones, hechos, reglas, condiciones, acciones, prerrequisitos, objetivos, etc. Este tipo de representación es el que utilizaban los sistemas **ASK** [10] (del inglés, preguntar) propuestos por Roger Schank. Los sistemas ASK, son sistemas hipermedia que simulan las situaciones que se presentan en el caso en que un determinado usuario tuviera una conversación con un experto (o con un grupo de ellos). En esta conversación, el usuario se hace una serie de preguntas que el sistema *contesta*. Sin embargo, por lo general, los participantes en una conversación real influyen en el desarrollo de la misma. Esto ocurre también en este tipo de sistemas, donde el usuario influye en el flujo de la conversación seleccionando ciertas preguntas, y el sistema influye con las respuestas que proporciona a dichas preguntas.

Este tipo de sistemas combinan numerosos tipos de material dependiendo del dominio en que se apliquen, de manera que se consiga simular mejor una conversación con el experto. En cualquier caso, el material siempre debe de estar estructurado de forma que se pueda elegir ir presentando al alumno diferentes opciones entre las que éste debe elegir. El tutor debe construir y gestionar la estructura de contenidos.

Los contenidos se presentan mediante una red de conceptos (nodos) prefijados por el tutor y que deben ser aprendidos por el alumno. Los arcos representan la transición de un concepto a otro en función de los conocimientos que el alumno va aprendiendo y sus propios intereses. Dichas transiciones se presentan en forma de preguntas o de opciones que el alumno va eligiendo. En cierto modo, este tipo de propuestas son una concreción de los llamados *mapas conceptuales*. Un *mapa conceptual* puede definirse como una herramienta de asociación, interrelación, discriminación, descripción y ejemplificación de contenidos, con un alto poder de visualización.

Independientemente de la representación que se elija para el modelado del dominio, un aspecto fundamental en este punto es el de plantear el dilema de cuánto conocimiento hay que representar. Existen diferentes posturas al respecto:

- **Introducir todo el conocimiento del dominio.** Se justifica esta opción en la medida en que se suponga que el sistema y los alumnos tenga que saber resolver el mismo tipo de situaciones.

- **Introducir una única guía de estudio** en el sistema, de manera que el resto de contenidos se le proporcionan al alumno por otro medio alternativo y el sistema irá guiando y aconsejando el estudio a lo largo de la materia.

- **Incrementar el conocimiento del dominio** aplicando técnicas de aprendizaje automático.

No obstante, además de la elección de una u otra opción, hay muchos otros problemas implicados. Por ejemplo, el hecho de que el conocimiento del dominio sea completo o no, no tiene nada que ver con que éste tenga una forma que sea legible para el estudiante.

Debe quedar claro que el conocimiento del dominio es algo dinámico que refleja decisiones, creencias, acciones, etc. realizadas en el dominio que se está aprendiendo, y desde luego no es una base de datos en la que se ha codificado información conocida.

2.1.4 Aproximación a la adaptación

La esencia de cualquier sistema interactivo de E/A es la de poder adaptarse dinámicamente a las necesidades del alumno. Estos sistemas llevan a cabo una adaptación basada en el modelo del usuario que contiene el estado del conocimiento del estudiante así como de sus preferencias y objetivos.

En este caso concreto de los STIs, el objetivo es usar el conocimiento del dominio proporcionado por un experto, el conocimiento que se tiene del alumno y el conocimiento del tutor para ofrecer una enseñanza personalizada y flexible.

Con la evolución de Internet y las posibilidades que ofrece, estos sistemas se han trasladado a Internet y constituyen los llamados Sistemas de Tutorización Inteligente en la Web.

Normalmente, todos estos sistemas se basan en una representación predefinida del conocimiento del dominio y del modelo de estudiante. A pesar de que este tipo de representación predefinida puede proporcionar cierta adaptación, ésta se restringe en su mayoría a un conjunto específico de reglas. Esto implica que la mayoría de situaciones que surgen a lo largo de una interacción con el estudiante deben conocerse de antemano. Como esto es lógicamente muy difícil de definir y de actualizar, los sistemas que modifican dinámicamente tanto los modelos del alumno como los de contenidos de acuerdo a las interacciones del alumno, pueden conseguir una mejor adaptación.

2.2. Tipos y filosofías de enseñanza

Una vez descrita la estructura general de los SIEAs es importante destacar que de las múltiples opciones que se tienen, la elección de una u otra depende del enfoque desde el punto de vista pedagógico que se quiera dar al sistema.

2.2.1 Aprendizaje y memoria según Roger Schank

Desde sus trabajos sobre la comprensión del lenguaje este autor es reconocido por sus numerosas aportaciones a la Inteligencia Artificial en general y, a la Inteligencia Artificial aplicada a la educación en concreto. Schank parte de los supuestos que ahora analizaremos para establecer un método de desarrollo de SIEAs.

Según Schank, la idea clave sobre la memoria está en recordar casos relacionados con la información que se está tratando y para ello nos basamos en la correspondencia en algún aspecto concreto entre la situación actual y la recordada. Para ello, establecemos una etiqueta que identifica dicho aspecto en nuestra memoria de casos. Recordemos que los casos son estructuras de conocimiento sobre situaciones experimentadas en el pasado. Mediante dichas etiquetas se organiza una red de experiencias o casos que se sitúan de forma que reflejan precisamente lo que más nos interesa sobre dichas situaciones. Este proceso es dinámico y la reestructuración de dicha red es continua. De esta forma, la red manifiesta la capacidad natural de aprender de nuevas experiencias.

Por tanto, la importancia de la memoria en este esquema es que refleja lo que nos interesa de una situación, su significado establece la organización correspondiente. Esta memoria se estructura en una gran red de nodos donde se producen generalizaciones que capturan los aspectos comunes encontrados. Si una generalización se refuta con un caso nuevo contradictorio se construye la explicación del fallo y se recuerda especialmente dicha información para evitar nuevas contradicciones en el futuro.

2.2.2 Estructuras de la memoria según Schank

El funcionamiento de la memoria depende de las estructuras mentales que sirven de base para su organización. Evidentemente, no se trata de recordar cada nueva situación, sino de establecer una organización que recoja los aspectos más relevantes.

Nuestra memoria contiene diferentes tipos de estructuras de organización que categorizan e interrelacionan otras estructuras de memoria. Estas estructuras de organización segmentan la memoria de forma que podamos localizar el elemento adecuado cuando sea necesario.

Las estructuras de organización nos ayudan a clasificar y localizar estructuras de niveles inferiores como hechos o casos. A su vez, recogen las generalizaciones que hacemos sobre dichos casos más específicos. La forma en que se organizan nos ayuda a hacer predicciones. Por ejemplo, sabemos que cuando vamos a realizar un examen y nos sentamos en nuestra silla luego nos entregarán los enunciados correspondientes.

2.2.3 MOPS y Scriptlets

Los **Scriptlets** [11] recogen lo que sabemos sobre cómo suelen ocurrir los hechos en situaciones típicas. Forman una pequeña parte de una escena o situación; algunos ejemplos son: mirar una carta, lavarse los dientes, aparcar un coche, etc.

Ya que los Scriptlets sólo constituyen una pequeña parte de la experiencia, necesitamos otro tipo de estructuras superiores denominadas **MOPS** [10], *paquetes de organización de memoria* (del inglés, *memory organization paquets*). Estas estructuras dividen las situaciones en escenas. Un MOP para un restaurante contendría escenas como “sentarse”, “pedir”,”pagar”, etc. Dichas escenas apuntan a su vez a los scriptlets que contienen el conocimiento sobre cómo comprendemos y nos desenvolvemos en dichas situaciones.

Esta división de la memoria en dos tipos de estructuras nos permite utilizar lo que hemos aprendido en una tarea a otra distinta. Por ejemplo, podemos “pagar un servicio en un restaurante”,”pagar un autobús”,”pagar un cine”, etc. Sobre dichos scriptlets los MOPS establecen los matices que distinguen el pago en cada una de las mencionadas situaciones.

Finalmente, resaltar que el motor de esta memoria de casos son las preguntas. Cuando nos enfrentamos a una situación nueva recordamos las relacionadas y nos preguntamos cuáles son las diferencias y los aspectos comunes. Cuando una predicción nos falla nos preguntamos qué causó el fallo y cómo podemos evitarlo en un futuro. Otras veces, cuando nos enfrentamos a un nuevo problema necesitamos elaborar un plan. Es entonces cuando nos preguntamos si ya conocemos un problema relacionado o podemos descomponerlo en problemas más sencillos. Aunque haya distintos tipos de preguntas, todas comparte una esencia común: juegan un papel esencial en el aprendizaje. Apuntan a huecos de nuestras estructuras de memoria que se intentan cubrir.

2.3. Uso y evaluación de los sistemas de educación

La clave para realizar una evaluación apropiada es el buen diseño y desarrollo de experimentos de forma que los factores individuales que se quieran testear puedan separarse fácilmente de otros factores que puedan interferir.

2.3.1 Evaluación de alumnos

La evaluación de los alumnos ha sido siempre una parte importante del sistema de Enseñanza/Aprendizaje. Por una parte, se necesita saber cuál es el conocimiento adquirido para actuar en consecuencia y, por otra, saber de manera objetiva el conocimiento adquirido por los propios alumnos.

Uno de los mecanismos de evaluación más extendidos, por su facilidad de corrección, es la realización de pruebas de evaluación mediante tests. La realización de estos tiene la ventaja de sistematizar la evaluación, por lo que ha sido ampliamente usada en aplicaciones de enseñanza asistida por ordenador y en sistemas de tutorización inteligentes.

En estos sistemas tradicionales, los tests tiene una serie de inconvenientes: las preguntas son siempre las mismas para todos los alumnos, su número es fijo, el tipo de preguntas es muy limitado, etc. Por otra parte, la evaluación que se obtiene tras la realización de un test suele estar basada en el número de respuestas acertadas y no suele considerar la variabilidad en la dificultad de las preguntas, ni otros factores como la probabilidad de acertar una pregunta al azar.

Un intento de solución a dichos problemas son los *test adaptativos*. Un ejemplo interesante es el sistema **SIETTE** [3] (*Sistema Inteligente de Evaluación mediante Test para TeleEducación*). Basándose en la cuantificación probabilística de los resultados de los tests y los posibles errores debidos al azar, junto con una medida de la dificultad de la pregunta, el sistema realiza una generación adaptativa de las preguntas que deberá ir respondiendo el alumno. Para ello, se mantiene un registro temporal de la evolución del alumno en el test, que se tiene en cuenta en el proceso de selección de la siguiente pregunta.

Aunque estas investigaciones sean interesantes, debemos recordar que la evaluación basada en la realización de un test condiciona muchas veces el propio proceso educativo, dado que se puede caer en “enseñar para el test”. Se cae asíen preguntar y centrar el proceso educativo en todo aquello que pueda ser preguntado objetivamente en un test.

2.3.2 Evaluación del sistema

El término de *evaluación empírica* (del inglés, *empirical evaluation*) de sistemas se refiere a determinar cómo se comportaría un determinado sistema en ciertos experimentos. Por lo general, la evaluación de los SIEAs debería ir encaminada a verificar que el proceso de aprendizaje del alumno se enriquece cuando se utiliza el sistema.

Por otra parte, los sistemas educativos adaptativos se evalúan comparando los resultados que se obtienen aplicando o no adaptación (en base al uso o no de modelos del alumno). Por ejemplo, se puede comparar las acciones previstas

para el alumno con las acciones que realmente se realizan. También se puede calcular el porcentaje de errores reconocidos.

Otra cuestión importante en la evaluación del sistema es considerar las características de la persona que se está evaluando. En estos casos los sistemas de educación suelen presentar distintos tipos de material dependiendo del estilo de aprendizaje del alumno.

Una evaluación completa sería aquella que comprendiera la validación tanto de los distintos componentes de un sistema, como por ejemplo, el uso de una determinada técnica de aprendizaje automático para el modelado del usuario, como la validación de todos los componentes en su conjunto. Un ejemplo de esta evaluación lo podemos encontrar en el sistema **ADVISOR** [1].

ADVISOR es una arquitectura de aprendizaje compuesta por dos agentes cuyo objetivo es el de centrar el razonamiento de un sistema de tutorización inteligente en un solo módulo. Uno de los agentes es responsable del aprendizaje de un modelo de cómo se comportan los alumnos usando el tutor en distintos contextos. El otro agente coge este modelo de comportamiento del alumno y un objetivo especificando el objetivo educacional que se persigue.

En este proyecto, primero se evaluaron (según los métodos propios de evaluación característicos de los sistemas de aprendizaje automático) los modelos que se construyen describiendo el comportamiento de los alumnos. Una vez validados los componentes que constituyen el sistema ADVISOR, como suele hacerse, se dividieron a los alumnos que participaron en el experimento en dos grupos, uno utilizaba el tutorial sin ADVISOR y el otro con él.

3. Sistemas de Tutorización Inteligente

3.1. Fundamentos

Los **Sistemas de Tutorización Inteligente** (abreviadamente, STIs) persiguen generalmente objetivos de aprendizaje bien definidos y comúnmente aceptados, como son el conocimiento factual y las habilidades procedurales que pueden medirse mediante tests estandarizados. Centrándose en el conocimiento a aprender, los diseñadores de estos sistemas con frecuencia empiezan por especificar este conocimiento tan precisamente como les sea posible. Para ello, adoptan una perspectiva de tipo objetivista (que establece que el mundo puede estructurarse de un modo completo y correcto en términos de entidades, propiedades y relaciones, y que el pensamiento racional consiste en la manipulación de símbolos abstractos vistos como representantes de la realidad). Para conseguir ésto, aplican las distintas técnicas de representación de conocimiento de la Inteligencia Artificial (como pueden ser: sistemas de producción, marcos, redes semánticas, lógica de predicados, etc.) Con este tipo de desarrollos se ha tratado de mostrar que, utilizando métodos tradicionales de aprendizaje, enseñanza y evaluación, se consigue mejorar significativamente la velocidad y la calidad del aprendizaje de los alumnos, y hasta cierto punto, obtener éxito en sus pretensiones.

Los STIs intentan capturar así un método de enseñanza y aprendizaje ejemplificado por una interacción humana de tutorización individualizada. Para los investigadores de la Inteligencia Artificial, este método de enseñanza ha sido de forma natural el primer objetivo del aprendizaje. Las versiones tipo ejercicio y práctica de la tutorización personalizada son formas de comunicación de conocimiento relativamente bien comprendidas. Este método es ampliamente aceptado tanto por la comunidad educacional como por nuestra cultura natural. Su popularidad se basa en buenas razones. La tutorización individualiza permite un aprendizaje consistente que conduce a mejores resultados que otros métodos. Sus desventajas frente a otros métodos analizados se atribuyen principalmente a la inadecuación de las técnicas de evaluación de los resultados de aprendizaje. No obstante, de los mencionados problemas de evaluación, es la tutorización individualizada la que sigue formando parte de los principios básicos del aprendizaje.

3.2. Componentes Básicos

La figura que se muestra más abajo esquematiza un STI genérico. Aparentemente, estos sistemas difieren poco de los sistemas CAI que les preceden. En general, ambos se caracterizan por una filosofía común que incluye un gran control del tutor y un formato de tarea de respuesta corta. En ambos sistemas los estudiantes aprenden trabajando en series de cuestiones relativamente breves y en ambos casos el sistema juega exclusivamente el papel de experto en la tarea, controlando la selección de tareas o problemas, mientras que el estudiante es el responsable de resolverlos. El sistema también juega el papel crítico, y en la mayoría de los STIs y los primeros CAI no reflejan diferencias en métodos de enseñanza ni filosofías de aprendizaje subyacentes sino logros de la ingeniería y la psicología que permiten al STI tutorizar de un modo "orientado al conocimiento". Al contrario que los primeros sistemas CAI, representan al menos parcialmente el conocimiento y razonamiento de un buen tutor humano individualizado y, por tanto, pueden adiestrar de una forma más detalla que aquellos.

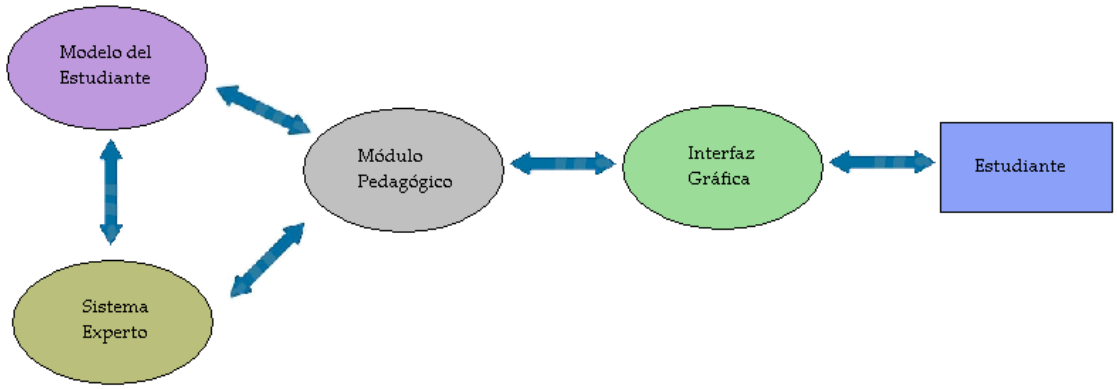


Figura 4. Componentes básicos de un STI

El núcleo de un STI es un sistema experto que incluye suficiente conocimiento sobre un área en particular para proporcionar respuestas ideales a preguntas, con el fin de corregir no sólo un resultado final sino cada pequeña etapa de razonamiento intermedia. Esto permite mostrar y modelar una forma correcta de resolver un problema. Con frecuencia, como un tutor humano, puede generar muchos caminos de respuestas diferentes.

Volviendo a la figura anterior, los componentes básicos de un STI son:

- *Sistema experto* o modelo del conocimiento del dominio. Contiene una representación del conocimiento específico del área de enseñanza en cuestión. El conocimiento del dominio recoge la experiencia operativa de resolver problemas en dicho dominio.

- *Modelo del estudiante*. En los STIs el conocimiento del estudiante se pone en relación con el conocimiento del dominio. Un primer mecanismo

muy sencillo e intuitivo es considerar que el conocimiento del alumno se va expandiendo hasta cubrir el conocimiento del experto. Para ello se establecen medidas de comparación diferentes, según sea el tipo de conocimiento, entre cada uno de los elementos de ambos modelos, el del estudiante y el del dominio.

- *Módulo pedagógico*. Contiene reglas similares a las del sistema experto que codifican la experiencia acerca de la propia tutorización, relativa, por ejemplo, a cuándo interrumpir a los estudiantes y qué tipo de información proporcionarles.

- *Interfaz con el alumno*. Es el módulo que sirve de intermediario entre el sistema y el estudiante. Es importante que el interfaz sea amigable y fácil de usar, para que no suponga un escollo añadido en el proceso de aprendizaje del alumno.

El modo en el que entran en acción cada uno de los elementos descritos es el siguiente. El STI, mediante la interfaz gráfica, transmite al alumno los conocimientos representados en el modelo de conocimientos del dominio, siguiendo las directrices pedagógicas especificadas en el módulo pedagógico. Según el alumno va interactuando con el sistema se va actualizando el modelo del estudiante, modificando el grado de conocimiento del alumno, módulos estudiados, etc.

Cada vez que el estudiante comete un error, el STI diagnostica el problema, actualizando el modelo del estudiante, y a continuación intenta remediarlo con un consejo muy detallado acerca de cómo el sistema experto habría operado en esa etapa. Este proceso se repite a cada paso de la evolución hacia la solución completa de un problema. Para que este esquema de funcionamiento sea factible se crea un marco de interacciones entre los distintos módulos. El control lo ejerce el módulo pedagógico en función del conocimiento estratégico diagnosticado y actúa conforme a una determinada estrategia de instrucción para alcanzar los objetivos declarados en el conocimiento del dominio.

En relación al proceso de diagnóstico de los errores cometidos por el alumno, éste constituye realmente el proceso de modelado del estudiante.

3.3. Entornos de Aprendizaje Interactivo

Estos sistemas surgieron como respuesta a las limitaciones del enfoque adoptado en muchos STIs en la etapa previa. Se trataba de superar algunos de los problemas detectados, como el control excesivo de la acción tutorial ejercida y la gran dependencia del conocimiento del dominio.

La teoría de aprendizaje utilizada era el *constructivismo*. El método de aprendizaje se denomina *basado en indagación*, también descrito como centrado en el estudiante o construccionista y basado en descubrimientos. El constructivismo pone énfasis en los proceso de estructuración activa del mundo y sostiene que existen múltiples significados o perspectivas para cualquier

evento o concepto, en lugar de existir un único significado correcto hacia el cual debe guiarse al estudiante.

Todo ello conduce a diseños de sistemas que difieren sustancialmente de los STIs. La inteligencia de los EAIs se distribuye entre un conjunto de herramientas en lugar de centralizarse en el tutor. Estas herramientas de computación con frecuencia incluyen videos interactivos u otras representaciones gráficas, y permiten a los estudiantes investigar y aprender libremente, sin un control externo. Esta libertad aporta asimismo beneficios prácticos ya que los EAIs no son tan intensivos en conocimiento como los STIs. Los EAIs proporcionan una representación explícita de los temas que el estudiante debe investigar pero que no necesariamente "conocen todas las respuestas correctas", ni deben incluir modelos de la cognición del estudiante ni deben tomar complejas decisiones pedagógicas. Por otra parte, estos sistemas proporcionan quizá herramientas demasiado potentes, que sobrevaloran la capacidad del alumno para descubrir ideas interesantes o juzgar qué tipo de conocimiento deben construir y, en muchos casos, pueden conducirle a naufragar pon un mar de cuestiones sin ningún interés.

Los micromundos son un tipo particular de EAIs que suponen una transición del tutor al concepto de "herramientas educacionales" y del método ejercicio-y-práctica basado en la indagación. También suponen un cambio en los objetivos del aprendizaje. En primer lugar, siguen considerando importante el aprendizaje de conocimiento específico del área, más precisamente, el aprendizaje de caracterización de patrones relacionales entre los objetos y propiedades que definen el mundo.

3.4. Sistemas adaptativos

La proliferación del uso de Internet en la educación impone algunos cambios lógicos, tanto en el modelo educativo como en los requerimientos que deben cubrir los sistemas que soportan los recursos en la Red.

A pesar de las ventajas evidentes que ofrece Internet, encontramos dos grandes dificultades. En primer lugar, cada alumno tiene unas necesidades especiales. En segundo lugar, el carácter estático de los sitios web educativos no permite cubrir de forma adecuada los requisitos cambiantes de los alumnos con necesidades, gustos y preferencias muy diversas.

La solución a este tipo de problemas son los **sistemas adaptativos**, un área de desarrollo que puede considerarse un caso de estudio dentro de un problema más genérico, la personalización del software.

Una verdadera adaptación sólo tiene lugar si el sistema considera que los gustos, experiencias y necesidades del alumno varían con el tiempo. Esto es especialmente significativo en los sistemas de educación, en los que la curva de aprendizaje condiciona fuertemente los cambios que deben producirse para mantener una buena adaptación. Para resolver este problema se introducen las técnicas de aprendizaje automático para el modelado del usuario.

3.4.1 Personalización del software

Los procesos que guían el aprendizaje y los nuevos modelos de enseñanza pueden beneficiarse del uso intensivo de los recursos ofrecidos por Internet. La variedad de información y servicios ofertados y, sobre todo, los canales de comunicación alternativos que pueden establecerse entre los distintos protagonistas, producirán cambios considerables en los modelos de enseñanza aplicados, especialmente en el modelo de enseñanza a distancia.

Frente a estas ventajas, se observa que la propia variedad y dispersión de las fuentes y servicios disponibles en la web educativa, unida a la naturaleza dispar del alumnado dificultan el aprovechamiento de este medio. Para paliar estos problemas y otros relacionados se aconseja el desarrollo de sistemas en la Web que faciliten un **acceso personalizado** a dichos recursos.

La personalización del software define un marco más general de problemas en el que la esencia la determina la **capacidad de identificar** los gustos, necesidades, preferencias, problemas, aptitudes, limitaciones, etc. relativos al usuario del software con el fin de **satisfacer dichas características.**

La **medida** del éxito de estos sistemas está en la **satisfacción del usuario**. Por ello, evidentemente, se trata realmente de aprender todas esas cuestiones sin que por ello el usuario se vea obligado a declararlas explícitamente ni a cambiar continuamente lo que declaró en un momento dado.

3.4.2 Modelado automático del usuario

Si el objetivo de los sistemas es adaptarse al usuario en función de su comportamiento, parece natural aplicar las técnicas de aprendizaje automático para recoger los datos referidos a dicho comportamiento, y así predecir su forma de actuar en el futuro. Sin embargo, para llevar a cabo este planteamiento hay que considerar una serie de problemas.

Los modelos del usuario guardan información sobre: intereses del usuario (a corto o a medio plazo), planes de los usuarios e intenciones a corto plazo, conocimiento del dominio, preferencias, habilidades, creencias (acerca del sistema, de la materia, etc.)

Según la manera en que se considera y se recoge esta información distinguimos los siguientes tipos de modelo de usuario: *Modelos explícitos* y *Modelos implícitos* o *automáticos* (predefinidos o construidos por el sistema respectivamente), *Modelado a corto y a largo plazo* (útil para una sesión o a lo largo de todas las interacciones).

Toda esta información puede cogerse, bien de forma explícita (mediante formularios de preguntas presentados a los usuarios) o bien de forma implícita, observando el comportamiento del usuario (qué opciones ha elegido, qué datos

ha enviado, etc.) Si para construir el modelo el usuario y el sistema deben colaborar, entonces estamos realizando un *modelado de usuario cooperativo*.

La representación de esta información depende de cómo se utilice en el sistema; lo más usual son los sistemas que utilizan modelos representados en forma de pares atributo – valor o reglas.

4. Herramientas de aprendizaje interactivo de estructuras de datos

4.1 Las herramientas Vedya y Vedya-Test

Llegados a este punto vamos a explicar las herramientas sobre las que construimos el Tutor Inteligente. Algunas de estas herramientas fueron desarrolladas en proyectos anteriores de Sistemas Informáticos.

4.1.1. Vedya

Vedya [8,9,13,14] es un entorno interactivo para el aprendizaje de estructuras de datos y esquemas algorítmicos. Cubre las estructuras de datos más comunes: pilas, colas, árboles binarios de búsqueda, árboles AVL, colas de prioridad y tablas. También proporciona otros tipos de datos abstractos como un “consultorio médico”. Con respecto a los esquemas algorítmicos, cubre los principales métodos de resolución: divide y vencerás, método voraz, programación dinámica, vuelta atrás, y ramificación y poda. Todas las estructuras de datos y métodos algorítmicos enseñados en las asignaturas afines se hallan así integrados en el mismo entorno. Vedya permite trabajar sobre distintas estructuras de datos y con varias secuencias de operaciones sobre la misma estructura mediante un sistema multiventana y multipestaña.

Actualmente existen dos versiones de la herramienta propiamente dicha. La primera contiene todas las estructuras de datos y esquemas algorítmicos mencionados anteriormente mientras que la segunda versión ofrece por el momento un subconjunto en un entorno visual más atractivo.

De cara al proyecto de Sistemas Informáticos hemos decido usar la segunda versión, ya que a parte del entorno más visual y atractivo, todas las ampliaciones de estructuras que se encuentran en la primera versión se irán incluyendo, en un futuro, en la segunda versión.

Con respecto a la adaptación de Vedya al Tutor Inteligente, decidimos que sería mucho mejor que el Tutor contuviera a Vedya en vez de al revés, ya que el Tutor Inteligente, a parte de realizar un proceso de tutorización, es un herramienta de integración de los anteriores proyectos de Sistemas Informáticos con toda la nueva funcionalidad que en el Tema 5 explicaremos detalladamente tanto para Vedya Alumno como Vedya Profesor.

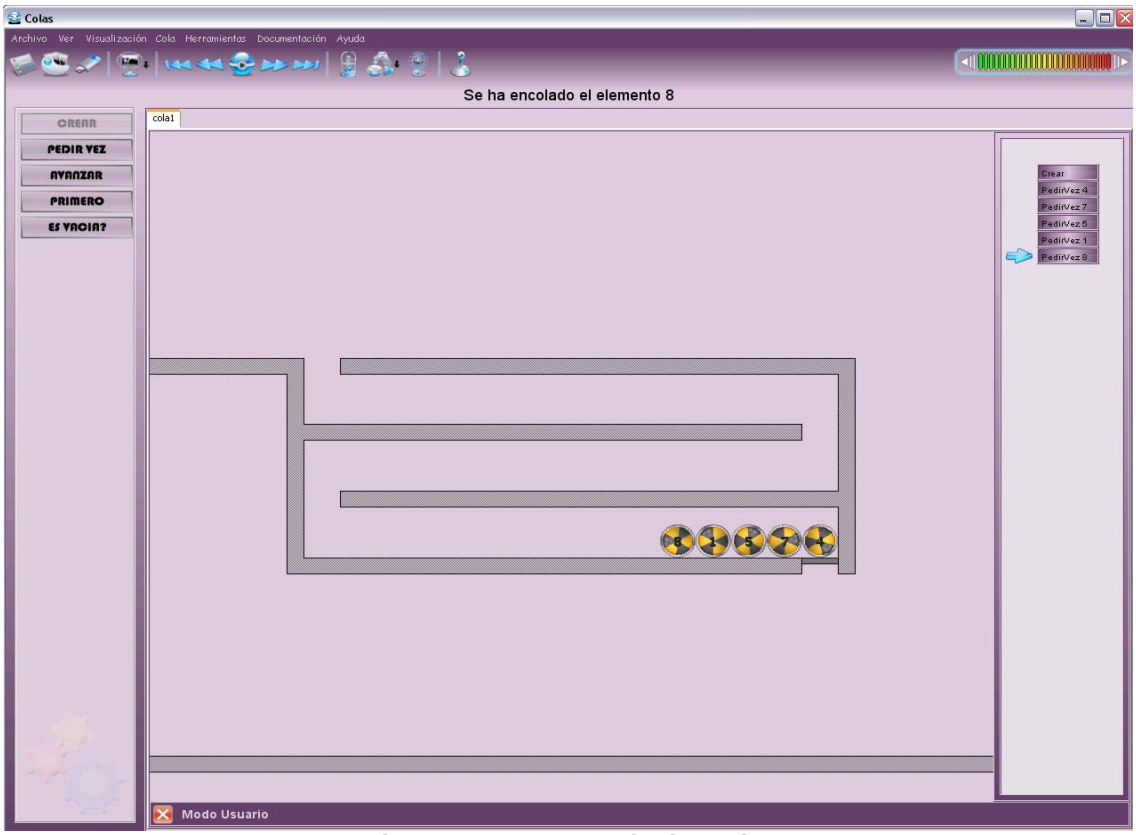


Figura 5. Ventana Cola de Vedya

4.1.1.1. Resultados obtenidos por la herramienta Vedya

Una de las principales motivaciones que nos ha llevado a construir nuestro STI sobre el sistema Vedya de visualización interactiva de estructuras de datos, ha sido el alto nivel de buenos resultados que se han obtenido en los cursos académicos pasados mediante su aplicación en las clases del segundo cuatrimestre. Como ejemplo concreto, de un total de 122 estudiantes matriculados en la asignatura, en la tabla que se muestra a continuación se ha tabulado el porcentaje de aciertos que tuvieron los alumnos en cada uno de los tests realizados. Se puede observar en esta tabla como, a partir del tercer test, el número de alumnos que realizaron este tipo de prueba disminuyó de manera considerable, manteniéndose en una cantidad reducida en comparación con los alumnos que accedían al Campus con regularidad. Sin embargo, también es posible observar que el número de respuestas acertadas por los alumnos que continuaron con el método de aprendizaje basado en la utilización del sistema Vedya, es muy alto. A pesar de que el número de estudiantes que participaban oscilaba entre un 30% y un 40%, aquellos que continuaban con el método de aprendizaje interactivo sí que lograban obtener resultados bastante buenos.

	Stacks 1	Stacks 2	Queues	Sequences	BST	AVL	RB	Heaps
Students	65	61	57	31	35	38	32	39
Answers	76.4%	82.5%	77.8%	65.6%	82.2%	84.9%	80.2%	86.3%

	2002/03	2003/04	2004/05	2005/06	2006/07	2007/08
Not attended	57.6%	45.3%	42.3%	64.7%	50.8%	40.2%
Passed	15.3%	22.2%	20.2%	18.2%	30.1%	42.6%
Failed	27.1%	32.5%	37.5%	17.1%	18.9%	17.2%

Figura 6. Tabla superior, respuestas de los alumnos a los test y porcentaje de respuestas correctas. Tabla inferior, porcentajes de los estudiantes no presentados, aprobados y suspensos.

En la segunda tabla observamos que, desde la aplicación del nuevo sistema de aprendizaje, el porcentaje de alumnos que realizaron el examen subió de manera elevada. Según estos datos, la herramienta no reduce en gran medida el número de estudiantes que suspenden el examen, pero sí consigue que muchos estudiantes que antes no se presentaban a la asignatura, se presenten ahora y además aprueben. Creemos pues, que con el Tutor, aplicado a la herramienta Vedya, se le puede dar al alumno una forma de continuar con su aprendizaje mucho más motivadora y participativa, de manera que no se sienta desmotivado y siempre vea cómo seguir avanzando en su conocimiento de la asignatura. Así pues, el objetivo de aplicar el Tutor a la herramienta Vedya no es tanto el de conseguir que alumnos que se presentaban y suspendían aprueben ahora, sino conseguir que los alumnos continúen estudiando la asignatura, y se presenten al examen con posibilidades reales de aprobar. Nuestro propósito es, por tanto, conseguir que si el alumno pierde (por la razón que sea) tiempo de clase y se queda rezagado con respecto al resto del grupo, siempre pueda usar el Tutor para poder recuperar ese tiempo perdido y seguir con la programación de la asignatura con el fin de poder equipararse al resto de la clase. En otro caso, el alumno que se queda rezagado tendría que estudiar la asignatura por su cuenta, no siguiendo así un programa diseñado, con lo que puede terminar no adquiriendo correctamente los conocimientos necesarios, y de manera que le llevase un gran esfuerzo su aprendizaje, lo que puede provocar que no lo consiga hacer correctamente. Esto genera sin duda una gran desmotivación en el alumno, que ve como le dedica un gran esfuerzo a una tarea para la que no ve resultados positivos.

Nuestras expectativas sobre el Tutor pasan por mejorar todavía más estos buenos resultados ya obtenidos, puesto que con el Tutor se espera que no decaiga el número de estudiantes que dejan de realizar los test. La idea es, por tanto, la de intentar que los alumnos que comienzan a usar el Tutor desde el principio, sigan el proceso hasta su finalización. Así, perderíamos menos alumnos que dejan de usar la herramienta y que pueden presentarse al examen con garantías. Si el porcentaje de alumnos que realizan todo el proceso aumenta, pensamos que el porcentaje de alumnos que podrían dejar la asignatura sería menor, consiguiéndose así mejores resultados.

4.1.2. Vedya – Test

Dentro del proyecto de Sistemas Informáticos citado en el apartado anterior se desarrolló una herramienta que se puede ejecutar de manera independiente. Dicha herramienta permite a los profesores crear, modificar o borrar preguntas de una base de datos, y crear tests a partir de ellas. Los estudiantes visualizan los tests, los resuelven y pueden consultar las soluciones correctas. Las preguntas están agrupadas por temas en la base de datos pero se pueden mezclar preguntas sobre distintas estructuras de datos en el mismo test.

Tal y como estaba desarrollada la herramienta, tanto alumnos como profesores tenían acceso a la totalidad de la funcionalidad, cosa que, pensándolo detenidamente no es lo más adecuado, ya que si el alumno puede consultar la base de datos de las preguntas puede resolver los test de manera fraudulenta y llegados a ese punto la herramienta deja de tener el propósito para el que fue creada.

Es por este motivo, que decidimos “crear” una nueva versión de Vedya – Test [9] adaptada al alumno, de forma que, el alumno sólo puede realizar los test existentes y comprobar un vez realizado el test su resultado. Así pues, no puede realizarse ninguna otra funcionalidad.

Figura 7. Vedya – Test

4.2. El sistema Maude bajo el entorno Eclipse

Maude [2] es un lenguaje de programación para especificaciones formales mediante el uso de términos algebraicos. Se trata de un lenguaje interpretado que permite la verificación de propiedades y transformaciones sobre modelos y que permite ejecutar la especificación como si fuera un prototipo.

Maude soporta de una manera sistemática y eficiente la lógica de reescritura. Esto le permite ser un lenguaje extremadamente potente y ampliable, a la vez que lo hace capaz de soportar un álgebra de operaciones de composición de módulos extensible. Algunas de sus aplicaciones más interesantes son las de metalenguaje, en las cuales Maude es usado para crear entornos ejecutables para distintas lógicas, demostraciones de teoremas, lenguajes y modelos de computación.

Con Maude se puede modelar casi todo, cualquier cosa que se pueda escribir, hablar o describir mediante el lenguaje humano, se puede expresar con instrucciones Maude. Puede llegar a ser extremadamente abstracto. Su diseño permite tanta flexibilidad que la sintaxis puede parecer en principio poco comprensible. Sin embargo, y a pesar de sus muchas ventajas, Maude no deja de ser un lenguaje declarativo, y como tal también tiene sus inconvenientes. Con lenguaje declarativo nos referimos a todos aquellos lenguajes de programación que basan su forma de funcionar en el pensamiento humano,

esto es, en las Matemáticas y la Lógica, en lugar de en el comportamiento del computador. De ahí que Maude se base principalmente en el álgebra y en la especificación de ecuaciones. Pues bien, como los demás lenguajes de este tipo, son tremendamente elegantes y claros a la hora de realizar cualquier especificación, además de soportar técnicas de desarrollo muy avanzadas y de existir aplicaciones variadas y de gran interés.

Las especificaciones creadas se pueden ejecutar en Eclipse mediante “plugins” especiales desarrollados en el Departamento de Sistemas Informáticos y Computación de la UPV y en el Departamento de Lenguajes y Ciencias de la Computación de la UMA. Este entorno, facilita la utilización a los alumnos al integrar el editor de texto con los comandos de ejecución del sistema. A la izquierda se presentan los proyectos desarrollados; en la parte central se encuentra el editor, sobre el cual está el panel de ejecución del sistema; y en la parte inferior la consola donde se muestra el resultado de la ejecución. En la parte derecha se pueden abrir otras ventanas que permiten definir diferentes opciones del sistema y hacer depuración.

En Maude el elemento básico de una especificación es un “módulo”. El lenguaje permite la definición de módulos funcionales utilizados para definir los tipos de datos; módulos de sistema utilizados para definir sistemas de reescritura; y módulos orientados a objetos, que permiten utilizar una sintaxis de clases, objetos y mensajes.

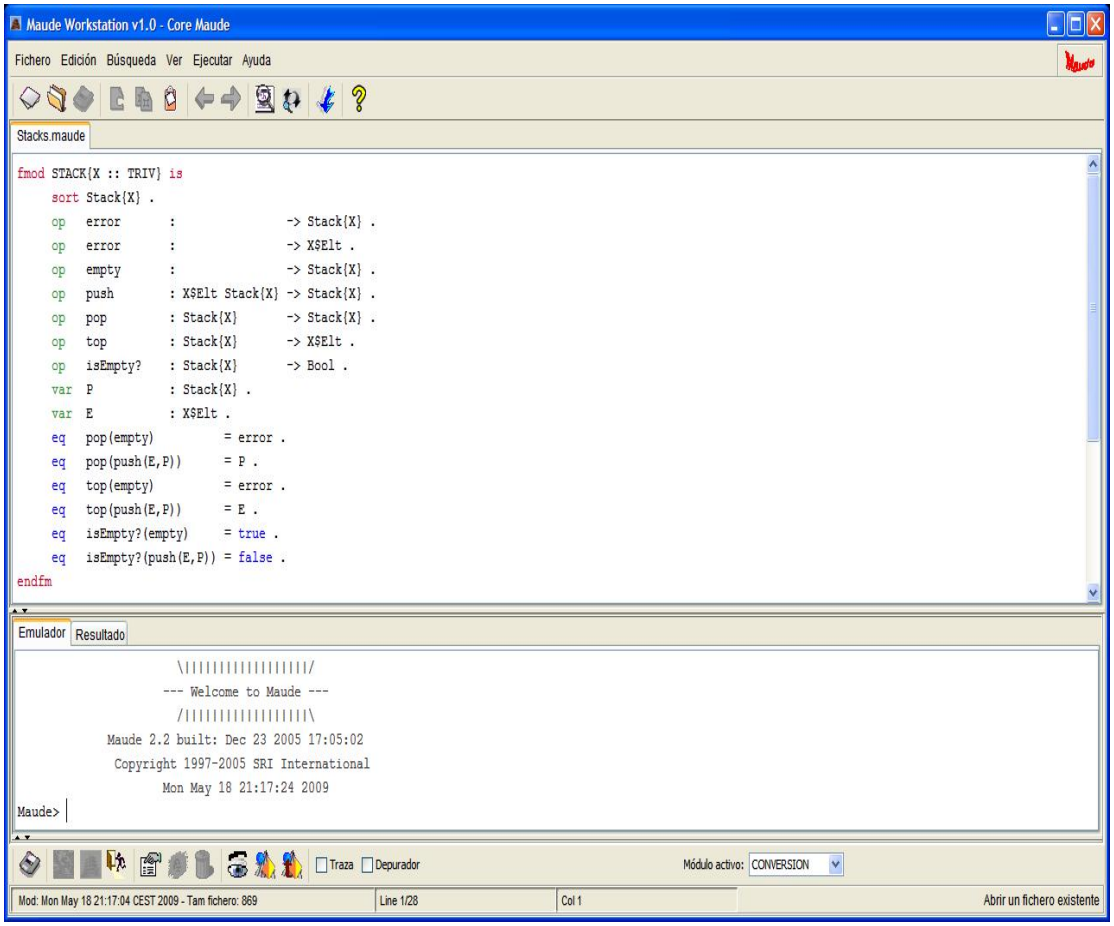


Figura 8. Maude bajo Eclipse

4.2.1 Maude Workstation

Tambiés es posible poder utilizar toda la funcionalidad de Maude a través de este applet que no necesita de Eclipse para funcionar.

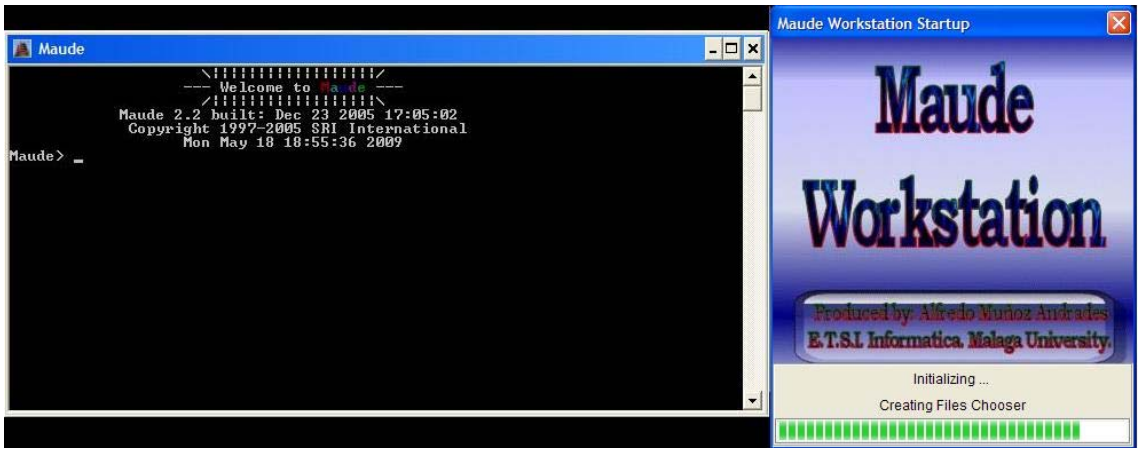


Figura 9. Maude Workstation

4.3. El Campus Virtual de la UCM

El Campus Virtual UCM (CV-UCM), accesible desde su página Web www.https://www.ucm.es/campusvirtual/CVUCM/index.php, extiende los servicios y funciones del campus universitario por medio de las tecnologías de la información y la comunicación.

El CV-UCM es un conjunto de espacios y herramientas en Internet que sirven de apoyo al aprendizaje, la enseñanza, la investigación y la gestión docente, y están permanentemente a disposición de todos los miembros de la comunidad universitaria.

En el CV-UCM pueden participar todos los profesores, personal de administración y servicios (PAS) y alumnos de la Complutense que lo soliciten. Es accesible desde cualquier ordenador con conexión a Internet que disponga de un navegador Web y de unos requisitos mínimos. También pueden participar en el CV-UCM, profesores, investigadores y alumnos que no pertenecen a la Complutense pero que colaboren con algún profesor de la UCM, lo que permite un mayor aprovechamiento de la participación del mismo personal que, de manera presencial, puede hacer online.

Para organizar el CV-UCM se utiliza una herramienta informática de gestión de cursos en la Web. La herramienta actualmente seleccionada por la UCM para estas funciones es WebCT (Web Course Tools). Esta herramienta incluye las funciones necesarias para crear y mantener, en el CV-UCM, asignaturas, seminarios de trabajo o investigación y otros espacios académico-administrativos:

- Gestión de alumnos y grupos de trabajo.
- Comunicación (foros, correo, charla, anuncios, agenda).
- Organización de contenidos.
- Envío, recepción y evaluación de prácticas, trabajos, exámenes.

El Campus Virtual lleva varios años implantado en la UCM con unos resultados muy satisfactorios en cuanto al uso por parte del alumnado como el personal docente.

Debido a este éxito todo los años se realizan unas jornadas sobre el Campus Virtual donde los profesores de la UCM muestran las numerosas experiencias y formas de trabajo cotidiano que han permitido sacar un gran rendimiento a los recursos disponibles en el Campus Virtual.

Figura 10. Campus Virtual de la UCM

5. Un Sistema de Tutorización Inteligente para el aprendizaje de estructuras de datos

5.1. Motivación, objetivos y diseño

5.1.1. Motivación

Como alumnos que han tenido que cursar y aprobar las asignaturas de *Estructuras de Datos y de la Información* (EDI) y *Metodología y Tecnología de la Programación* sabemos cuales son los principales problemas que se encuentra un alumno ante el estudio de estas asignaturas.

Cuando nosotros las cursamos justo otros compañeros hicieron las herramientas Vedya y Vedya – Test. La verdad es que nos fueron muy útiles, sobre todo para la parte de EDI, aunque presentaban algunas deficiencias en cuanto a su uso. Por ejemplo:

- La monotonía, cuestión que llevaba a dejar de usar las herramientas
- Desconocimiento del aprovechamiento de las herramientas
- No saber el nivel de la clase en la que te encuentras. Esto suponía no saber si tus fallos eran una cuestión personal o bien si se debían a un fallo global en el aprendizaje de la asignatura por parte de la mayoría de los alumnos.
- Debido a estas deficiencias nos planteamos crear un sistema que fuera capaz de solventar estos problemas, así como crear una herramienta que permita a futuros alumnos entender, aprender y, por consiguiente, aprobar las asignaturas con un esfuerzo adecuado.

5.1.2. Objetivos

El objetivo del proyecto ha consistido en desarrollar un Tutor Inteligente que se pueda integrar en la herramienta Vedya y que sea capaz de personalizar el proceso de enseñanza, permitiendo orientar de forma individualizada al alumno durante todo el proceso de aprendizaje de las principales estructuras de datos, diagnosticando sus principales deficiencias y ofreciéndole soluciones concretas a sus problemas.

Asimismo, se pretende que el Tutor Inteligente sea capaz de determinar los objetivos pedagógicos de cada una de las sesiones de uso de la herramienta, así como el protocolo a utilizar para que la sesión brinde los mejores resultados posibles.

Se ha pretendido, pues, incorporar un Tutor Inteligente a la herramienta Vedya que permita controlar en todo momento el proceso de aprendizaje del alumno, sin que resulte monótono o provoque desmotivación.

5.1.3. Diseño

En cuanto al diseño analizaremos cada uno de los puntos importantes y pasos que hemos seguido para diseñar la herramienta.

5.1.3.1. Casos de Uso

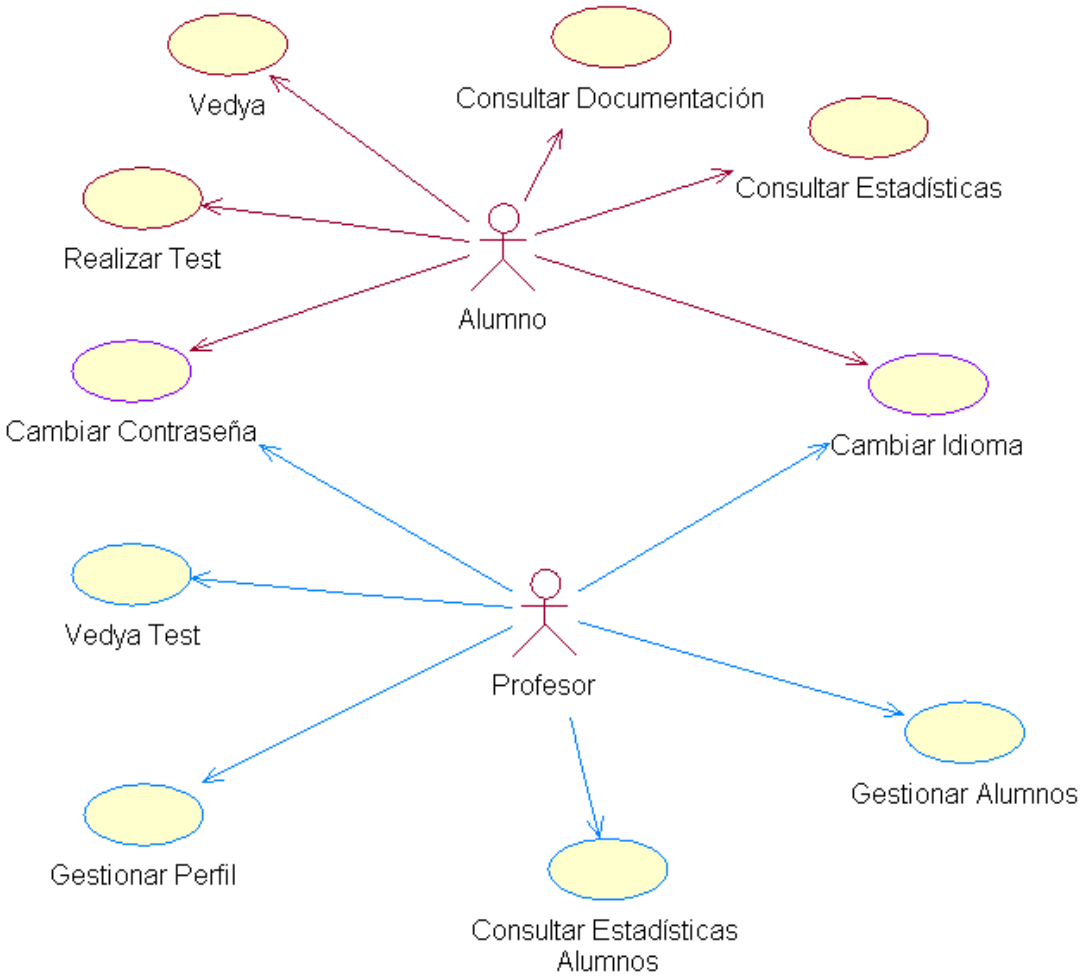


Figura 12. Casos de Uso

Una vez analizado el diagrama, observamos que tanto el Alumno como el Profesor realizan una serie de casos claramente diferenciados. Debido a ello, decidimos que el proyecto esté constituido por dos aplicaciones diferenciadas.

Estas son, Vedya Alumno y Vedya Profesor, que explicaremos con detalle a continuación.

5.1.3.2 Diagrama de clases

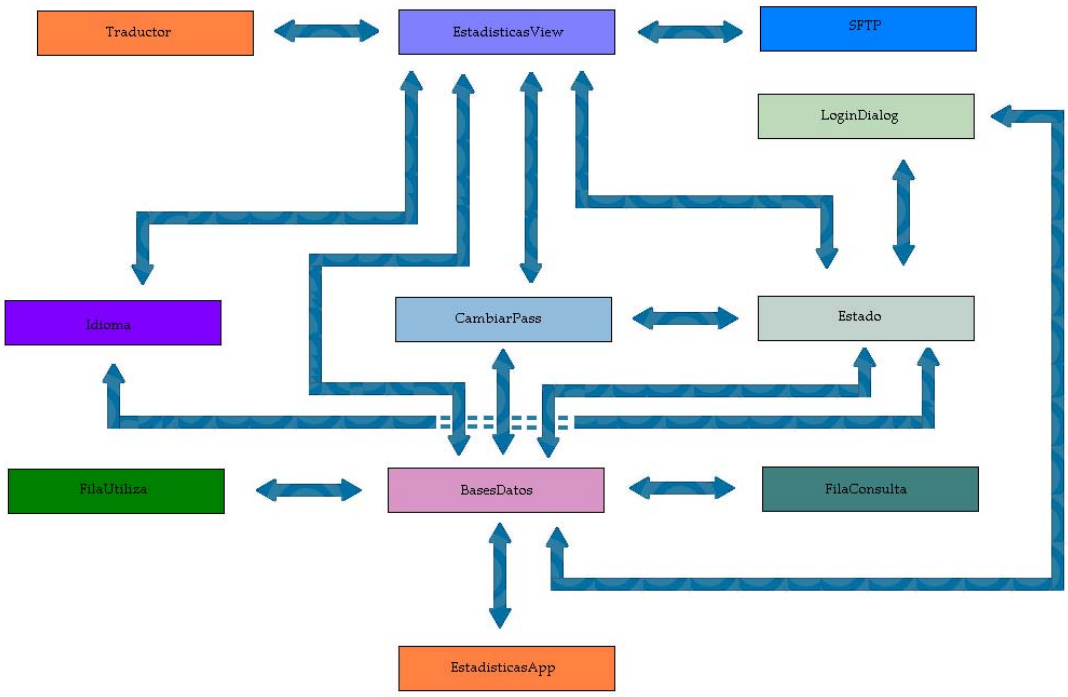


Figura 13. Diagrama de clases

5.1.3.3 Distribución de las clases en Vedya Alumno

Vamos a hacer referencia a las clases más importantes a la hora de implementar Vedya Alumno. Dado que tenemos un conjunto grande de herramientas y módulos, la implementación de Vedya Alumno se ha dividido en diversos paquetes. Dichos paquetes son ***ALUMNO***, ***BBDD***, ***Mensajes***, ***Gráficas***, ***VEDYA*** y ***VEDYA Test***.

- Paquete ***ALUMNO***

  Dado que hemos incorporado una opción de cambio de idioma en nuestras aplicaciones **Vedya Alumno** y **Vedya Profesor**, la clase **Idioma** representa el conjunto de métodos y atributos utilizados para tal efecto, y que permiten que el sistema sea modular y sea posible además la incorporación de más idiomas en un futuro.

  Dicha clase es apoyada en su tarea por la clase **Traductor**, cuyo fin es traducir todas las palabras que componen la interfaz así como sus mensajes, además de traducir las palabras generadas por las gráficas.

- Paquete ***BBDD***

  Que contiene las clases ***BasesDatos***, ***FilaUtiliza*** y ***FilaConsulta***, implementa el conjunto de métodos utilizados para tratar con la Base de datos y realizar peticiones a ésta.

- Paquete ***Gráficas***

  Es utilizado para todo el conjunto de representación en el módulo de gráficas, el cual recoge datos a través de peticiones a la base de datos y las clases antes mencionadas ***FilaUtiliza*** y ***FilaConsulta***.

- Paquete ***Mensajes***

  Se encarga de mostrar al usuario todo el conjunto de mensajes informativos, así como los mensajes mostrados en la barra de estado inferior de la aplicación, en el lenguaje elegido por el usuario.

Finalmente, los paquetes ***VEDYA*** y ***VEDYA TEST*** son el conjunto de clases que se encargan del funcionamiento de los módulos Vedya y Vedya-Test respectivamente.

Cabe destacar que tanto los paquetes ***VEDYA*** como ***VEDYA – TEST*** sufrieron una profunda remodelación y simplificación con respecto a las aplicaciones originales. Por ejemplo, en Vedya – Test, se ha limitado las opciones que se pueden hacer ya que el alumno no puede crear test, ni modificar la base de preguntas almacenadas, etc.

5.1.3.4 Distribución de las clases en Vedya Profesor

En cuanto a la distribución en paquetes de la aplicación Vedya Profesor presenta un aspecto bastante similar a la del alumno para poder permitir ahora una ampliación muy sencilla e intuitiva de ambas aplicaciones.

Los paquetes son ***PROFESOR***, ***BBDD***, ***Mensajes***, ***Gráficas*** y ***VEDYA Test***.

A simple vista cabe destacar que el profesor no necesita tener acceso a la herramienta Vedya, como ya comentamos en el diagrama de casos de uso, motivo por el cual en Vedya Profesor no esta integrada la herramienta Vedya. Si en un futuro se deseará incluir por el motivo que fuera, los cambios se podrían hacer en muy poco tiempo.

- Paquete ***PROFESOR***

  Realiza la misma función que el paquete ***ALUMNO*** en Vedya Alumno, ya que es la parte común a ambas aplicaciones.

- Paquete ***BBDD***

  El profesor puede realizar una gran cantidad de consultas y modificaciones en la Base de Datos haciendo que el paquete de BBDD de Alumno sea una simplificación de éste.

- Paquete ***Mensajes***

  Presenta el mismo comportamiento que en ***Vedya Alumno*** salvo que los mensajes son más variados debido a la distinta funcionalidad entre ambas aplicaciones.

- Paquete ***Gráficas***

  Ocurre algo similar al paquete ***BBDD***, es decir, paquete ***Gráficas*** de ***Vedya Alumno*** es una simplificación del paquete en Profesor, ya que el profesor puede consultar una mayor cantidad de información para la correcta supervisión del aprendizaje de los alumnos.

- Paquete ***Vedya – Test***

  Realiza la misma función que en ***Vedya Alumno*** salvo que para el profesor la herramienta ***Vedya – Test*** conserva totalmente la funcionalidad original.

5.2 Implementación

Vamos a comentar en este apartado nuestra elección concreta a la hora de diseñar la herramienta, así como a comentar brevemente el comportamiento de las clases más importantes que la componen.

5.2.1 Herramientas utilizadas para el desarrollo

A la hora de enfrentarnos al desarrollo de las herramientas Vedya Alumno y Vedya Profesor, hemos tomado una serie de decisiones de diseño que nos gustaría que se viesen reflejadas en este apartado.

Dado que nuestra herramienta es un compendio de varias herramientas ya existentes, consideramos que la mejor opción era desarrollar una nueva aplicación que englobara a todas estas, para lo cual lo más lógico era el desarrollo en Java de esta aplicación, ya que además nos permitía hacer nuestra herramienta multiplataforma.

Para lo cual, se ha optado por desarrollar estas aplicaciones bajo el IDE NetBeans, ya que nos ha permitido hacer una interfaz visualmente atractiva sin un esfuerzo excesivo.

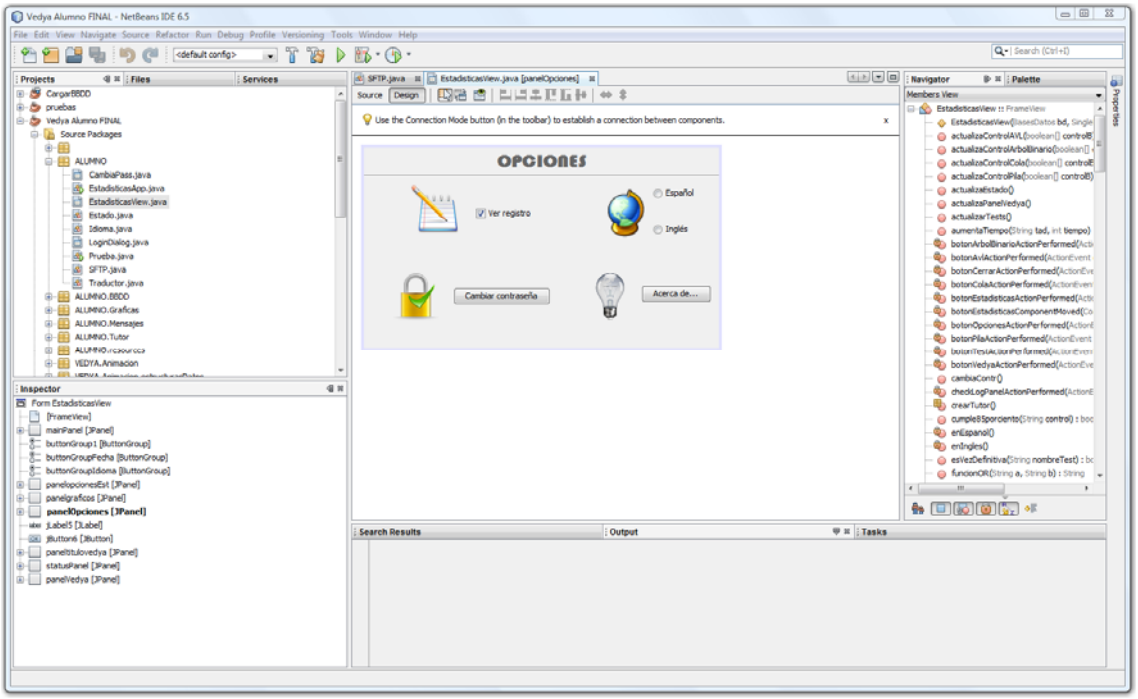


Figura 14. NetBeans

Dado que teníamos un conjunto grande de datos que procesar por parte tanto del alumno como del profesor, se desarrolló una base de datos que operase con las aplicaciones Vedya Alumno y Vedya Profesor.

Como dicha base de datos no tenía una complicación excesiva ni se requería una gran capacidad de proceso, se ha utilizado MySQL como implementación específica de dicha base de datos, ya que nos ha sido más que suficiente su potencia, a la vez que es una herramienta *Open-Source* y dispone de buenas aplicaciones para el monitorizado de tablas y servidores SQL.

A la hora de guardar los test, hemos considerado como lo más conveniente, dado que el profesor puede no tener creados todos los test antes del comienzo de la utilización de las herramientas, que la creación y el uso de los test debía ser dinámica, en el sentido de que el profesor podría ser capaz de crear un nuevo test durante el uso de las herramientas, y el alumno debiera ser capaz de poder realizarlo. Por ello, lo más indicado que hemos considerado es guardar dichos test en el SFTP privado que tiene cada profesor de la facultad, y que nuestra herramienta accede tanto para guardar los test como para comprobar cuales son los últimos test realizados por el profesor.

Para realizar correctamente dichas conexiones, se ha utilizado una librería java con licencia GPL llamada ***sshtools*** que permite realizar conexiones SSH de una manera sencilla y poder realizar el correcto sincronizado de test desde el servidor hasta el usuario local.

Para el diseño de nuestra interfaz gráfica, hemos utilizado el conjunto de iconos de KDE, también bajo licencia GPL, así como para la realización de las estadísticas, hemos contado con la librería jFreeChart, que permite de una forma cómoda la elaboración de gráficas en Java, y que nos ha sido muy útil en el módulo estadísticas de ambas aplicaciones Vedya Alumno y Vedya Profesor, que más adelante serán explicadas. Dicha librería es *Open-Source* y se encuentra disponible bajo licencia LGPL.

Creemos que es importante que nuestra herramienta se haya desarrollado siempre bajo herramientas Open-Source, ya que permiten que los alumnos venideros de *Sistemas Informáticos,* puedan continuar el desarrollo de dicha herramienta sin problemas de adaptación de programas específicos o costosos, o problemas de compatibilidades con herramientas propietarias tales como JBuilder, como ha sido en nuestro caso.

5.2.2 Diseño de la base de datos

Vamos a comentar brevemente las decisiones de diseño que hemos tomado a la hora de hacer el esquema ERR de la base de datos de los programas Vedya-Alumno y Vedya-Profesor.

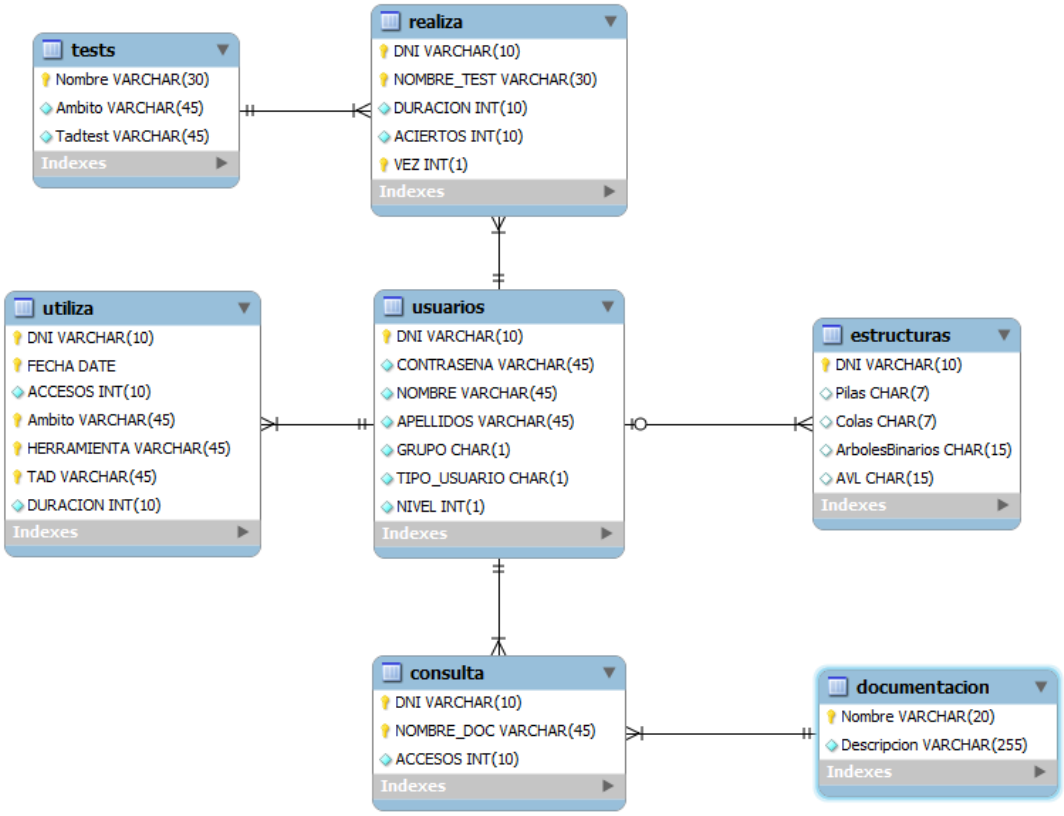


Figura 15. Diagrama ERR de la bbdd de Vedya Alumno y Vedya Profesor

Primeramente, tuvimos que analizar cuál era la información que iba a ser relevante en el diseño, tanto para el alumno como para representar y guardar toda la información que éste genera. Por ello, definimos una primera tabla "usuarios" en la que insertamos como campos su DNI, su CONTRASEÑA, NOMBRE, APELLIDOS, GRUPO, TIPO_USUARIO y NIVEL.

De esta información, es interesante recalcar que TIPO_USUARIO nos va a definir, mediante un carácter, si es un profesor o es un alumno. Esta decisión de diseño nos posibilita que si en un futuro se desea añadir otro tipo de usuario se pueda añadir sin más que asignarle un carácter en la base de datos y unos permisos sobre las distintas tablas.

El campo NIVEL es el que usaremos para definir cuál es el estado "teórico" del alumno, y dado que nuestro tutor es un tutor secuencial, debido que sigue el mismo temario y el mismo orden que la asignatura que se desea aprender, este campo estará definido por un número.

En cuanto a la realización de los test, hemos considerado pertinente crear una tabla con la información de estos, tanto su nombre, como su ámbito y el TAD al que se refieren, y una tabla intermedia que indique qué usuario ha realizado qué test, reflejando en esta última la DURACIÓN, la VEZ, y los ACIERTOS.

La tabla "utiliza" refiere a toda la información que hemos considerado útil cuando el usuario consulta el módulo teórico de VEDYA. Esta tabla está

apoyada por la tabla “estructuras”, que guarda fielmente las herramientas utilizadas para cada módulo teórico por parte del alumno.

Finalmente, las tablas “documentación” y “consulta” contienen información acerca de la documentación de VEDYA, y su utilización por parte del usuario respectivamente.

5.3. Vedya Alumno

5.3.1 Introducción

Vedya Alumno representa el módulo mediante el cual los alumnos son monitorizados y se registra las puntuaciones obtenidas en su evaluación mediante el correspondiente módulo Vedya – Test.

En Vedya Alumno hemos querido representar un conjunto de herramientas integradas para que el usuario no tenga que preocuparse de abrir distintas aplicaciones, siendo la propia herramienta la que ahora monitoriza la utilización de la documentación, el uso de los correspondientes tests y su nota respecto a la nota global de la clase, y muestra al alumno un “grafo teórico” en el que el usuario ve avanzar su conocimiento en la asignatura a través del desbloqueo de módulos teóricos consecutivos.

5.3.2 Descripción general de la herramienta

Vedya Alumno es un compendio de varios módulos que interaccionan entre sí para ofrecer al usuario un sistema de aprendizaje completo.

Los módulos son los siguientes:

- Módulo de autentificación
- Módulo de teoría
- Módulo Vedya – Test
- Módulo de Estadísticas
- Módulo de opciones

5.3.2.1 Módulo de autentificación

El módulo de autentificación es el módulo correspondiente de Vedya Alumno que se encarga del correcto acceso individualizado a través de un formulario en el que se le pide su clave y su contraseña para poder acceder al sistema.

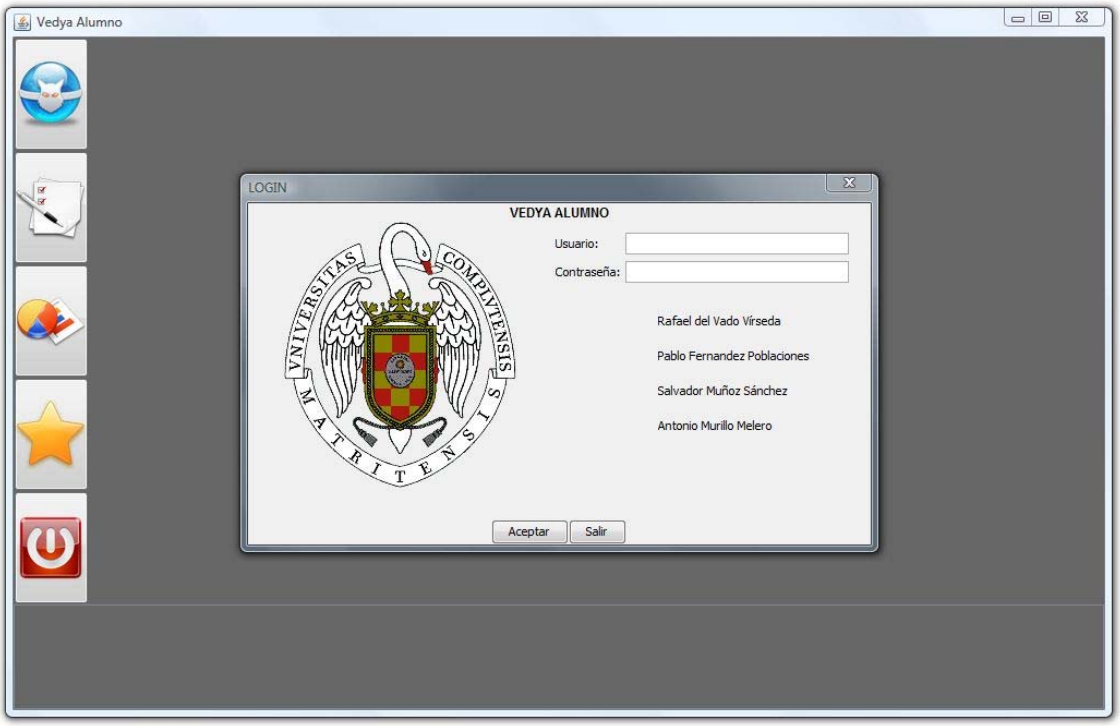


Figura 16. Acceso a la herramienta Vedya Alumno

Una vez introducidos los datos de acceso correctamente, el sistema carga sus preferencias, tales como último nivel guardado, y test que puede realizar para poder ascender al nivel teórico siguiente.

5.3.2.2 Módulo de teoría

El módulo de teoría es el módulo en el que el alumno va a aprender conocimientos a través de los diversos temas de la asignatura “Estructuras de Datos y de la Información”.

Este módulo se corresponde con la herramienta Vedya, aunque hemos adaptado su apariencia a un uso más intuitivo prescindiendo de animaciones superfluas, así como a un gasto innecesario de memoria con la opción de poder configurar colores y evitando, principal problema de la herramienta Vedya tal cual se conocía, el uso excesivo de repintados completos de toda la ventana.

Se ha representado el contenido teórico del segundo cuatrimestre mediante un grafo en el que sus nodos representan cada uno de los temas tratados en dicho cuatrimestre, y su evolución secuencial en el conocimiento de esta teoría.

Es por ello por lo que el alumno, al comenzar, solo encuentra disponible el primer de todos los módulos, el correspondiente a *Pilas*, como vemos en la siguiente ilustración:

Figura 17. Módulo teórico de Vedya Alumno

Hemos considerado importante que el alumno sea monitorizado en el uso de estas herramientas teóricas, ya que consideramos imprescindible que el alumno debe familiarizarse con la mayor parte de las operaciones posibles para cada una de las estructuras de datos propuestas.

Para ello, Vedya Alumno monitoriza el uso de estas operaciones, así como las posibles visualizaciones de dichas estructuras. Se ha considerado imprescindible que para que el usuario pueda avanzar al siguiente nivel o estructura de datos, el alumno debe utilizar el 85% de las distintas opciones disponibles en el módulo teórico.

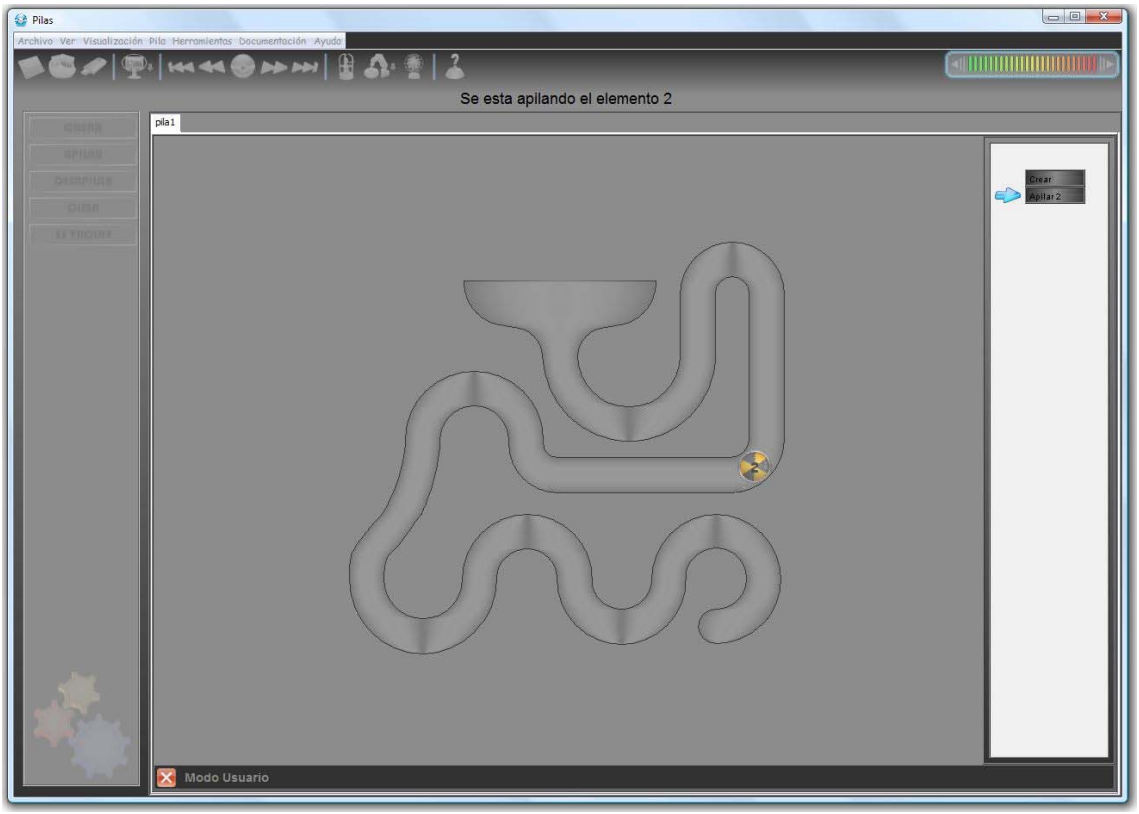


Figura 18 Ventana Pila de Vedya

Una vez que el alumno cierra dicha aplicación, estos datos de uso son guardados en la base de datos correspondiente a su información, y mediante la lectura de estos datos, la aplicación Vedya Alumno permitirá que el alumno pueda evaluar su nivel mediante la realización del test correspondiente al módulo estudiado.

Una vez que se dan estas dos condiciones el nivel umbral del 85% de uso y la realización del test correspondiente de dicho módulo teórico, se desbloquea para dicho alumno el siguiente módulo teórico.

Figura 19 Desbloqueo y secuenciación de la teoría en Vedya Alumno

El alumno por tanto, ve como su aprendizaje es representado mediante un grafo secuencial en el que para el total aprendizaje de la asignatura, ha tenido que aprender secuencialmente todos los módulos existentes en esta asignatura.

Esta fue una decisión de diseño ajustando el contenido del aprendizaje por parte del alumno al mismo temario y en el mismo orden en que se ve en la asignatura de Estructuras de Datos y de la Información.

5.3.2.3 Módulo de Vedya –Test

El módulo Vedya –Test representa el módulo principal mediante el cual se tiene referencia del nivel teórico del alumno. Es mediante dicho módulo donde el alumno refleja su correcto aprendizaje de cada uno de los puntos teóricos de la asignatura.

Una vez el usuario ha introducido los datos de acceso, la herramienta Vedya Alumno recupera automáticamente los test que el profesor ha colocado en su servidor propio SFTP de la Facultad, y en base a estos, se actualizan en el directorio local del ordenador del alumno.

Como hemos comentado anteriormente, Vedya –Test representa la realización de un pequeño examen teórico de la estructura de datos correspondiente, y por ello, solo es activado dicho test cuando el alumno sobrepasa el umbral de aprendizaje que escogió inicialmente.

En caso de que no se den estas condiciones, la herramienta muestra un mensaje al usuario informando de este requisito.

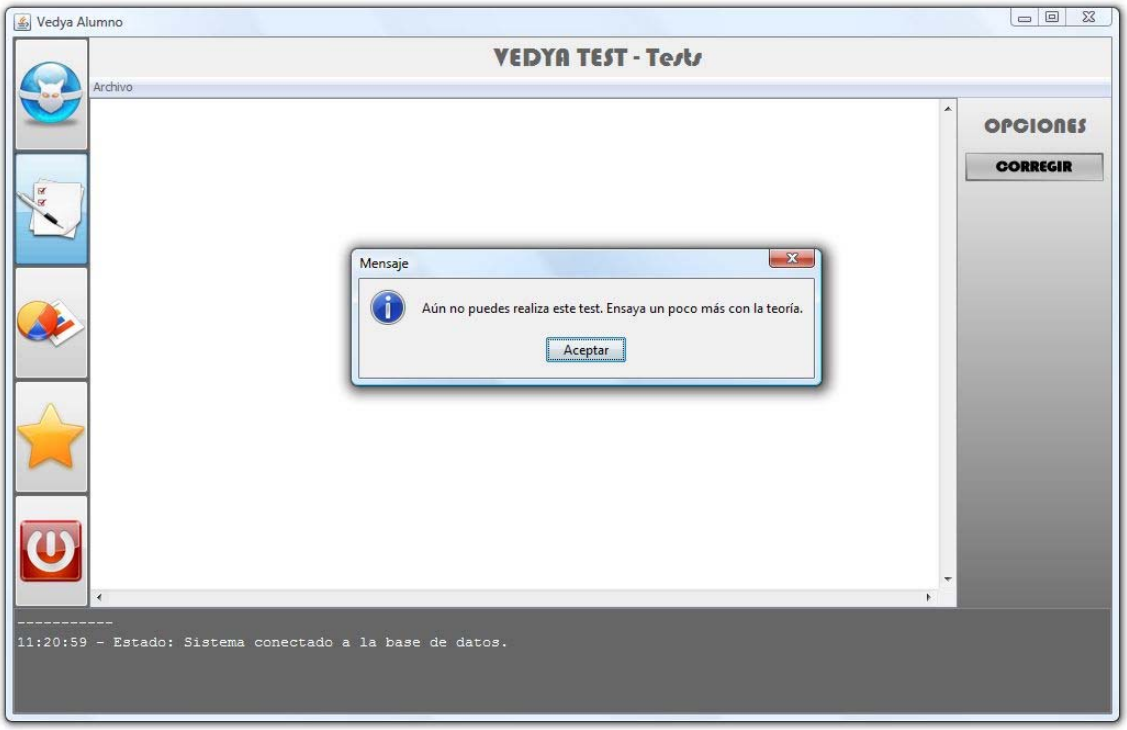


Figura 20. Información emitida por el programa para la autorización del alumno

Por el contrario, si dicho alumno ha utilizado la mayor parte del módulo teórico de Vedya, se le permite la realización de dicho test.

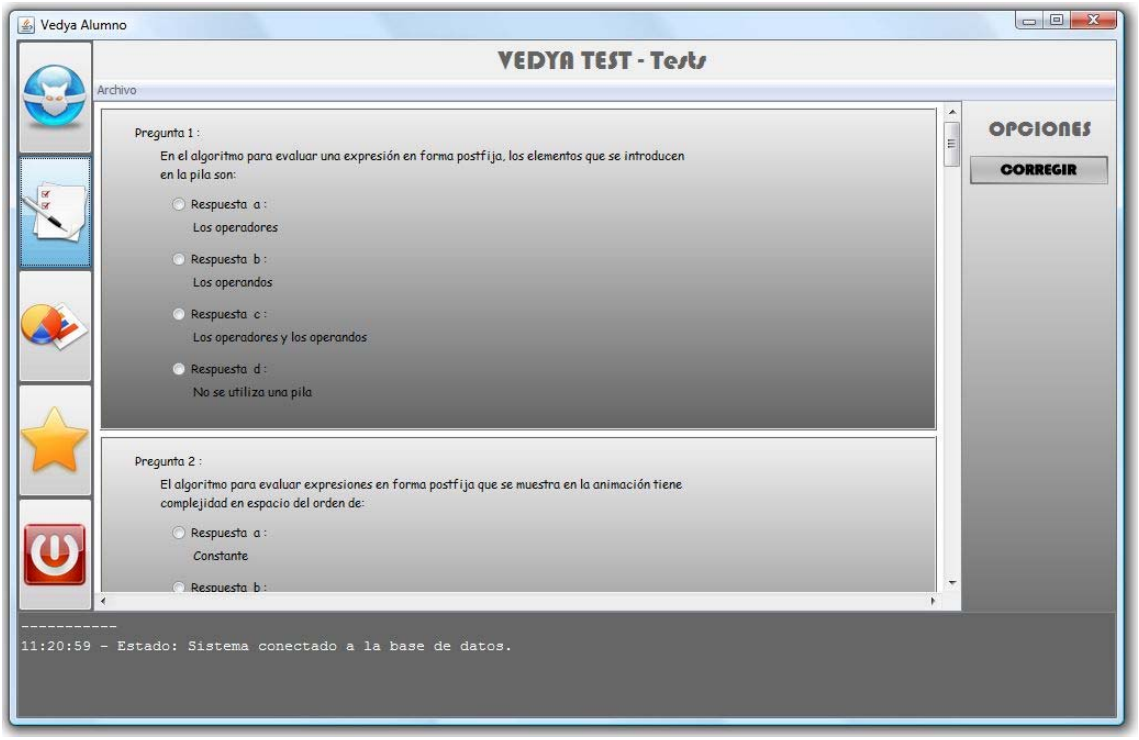


Figura 21. Módulo de Vedya – Test en Vedya Alumno

Se ha considerado conveniente que el alumno tenga dos posibilidades a la hora de realizar un test. La primera vez, el alumno responde a dichas preguntas, se le informa de su puntuación final, así como de las respuestas correctas e incorrectas en dicho test, y es a través de una segunda realización del test mediante la que la herramienta guarda la puntuación final del alumno, que podrá ver después en el módulo de estadísticas. Así mismo, el profesor podrá ver la evolución de las puntuaciones de dichos test para todos y cada uno de los alumnos.

Una vez realizado este segundo test, e independientemente de su puntuación, el siguiente módulo teórico es desbloqueado para que el alumno avance en su aprendizaje de la asignatura.

5.3.2.4 Módulo de Estadísticas

En el módulo de estadísticas, al igual que en Vedya Profesor, se ha querido tener una representación visual, clara y eficiente del nivel teórico obtenido por el alumno a través de la herramienta Vedya – Test.

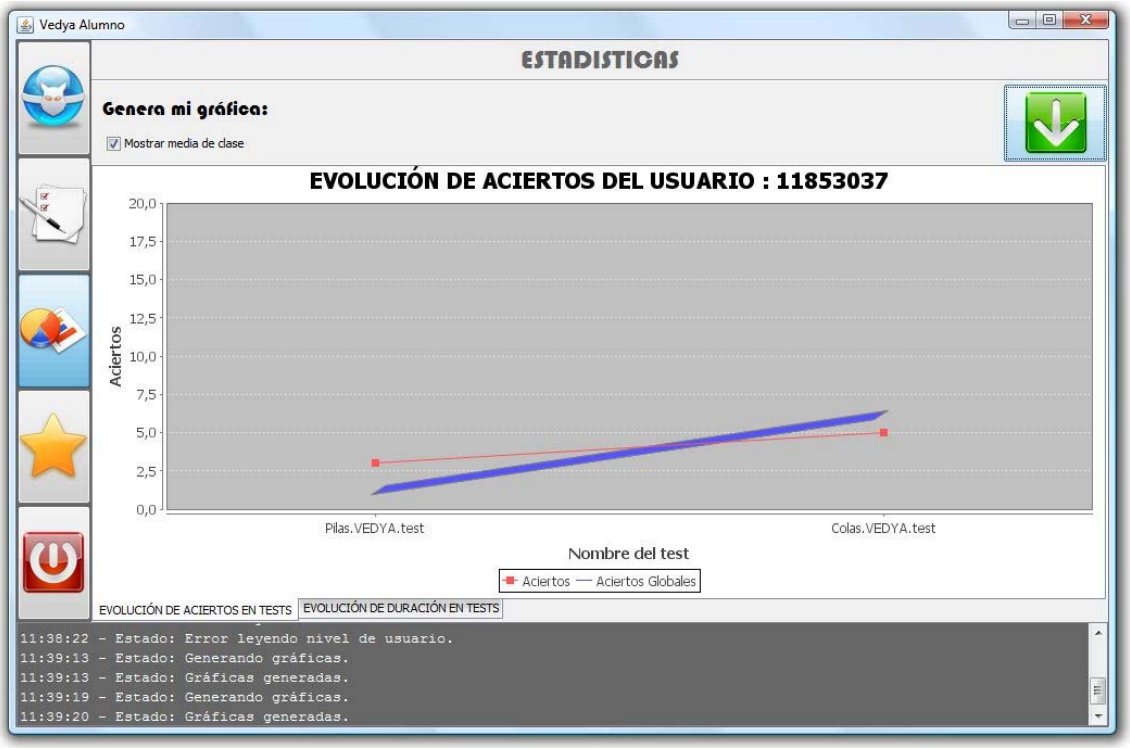


Figura 22. Estadísticas en Vedya Alumno

El alumno es informado, a través de dicho módulo, de su evolución en los test realizados, tanto de su duración, como de su puntuación.

Además se ha incorporado una opción en dicho panel para que el usuario pueda comparar su puntuación con la media de los alumnos de su clase, información que consideramos de gran interés pedagógico y que motiva al alumno al estudio de la asignatura al mismo tiempo que puede ayudar al alumno a ir a tutorías personalizadas con el profesor al comprobar, por ejemplo, que la mayoría de las veces está por debajo de la media.

Esta situación que, habitualmente puede hacer que el alumno decida abandonar la asignatura, queda ahora registrada como un problema de aprendizaje de ese alumno, de forma que sea el propio profesor de la asignatura el que pueda detectarlo y actuar en consecuencia.

Así como en la aplicación Vedya Profesor hemos considerado relevante un conjunto más amplio de información a la hora de representarla, en Vedya Alumno hemos considerado suficiente mostrar al usuario información sobre sus resultados en los test y su duración.

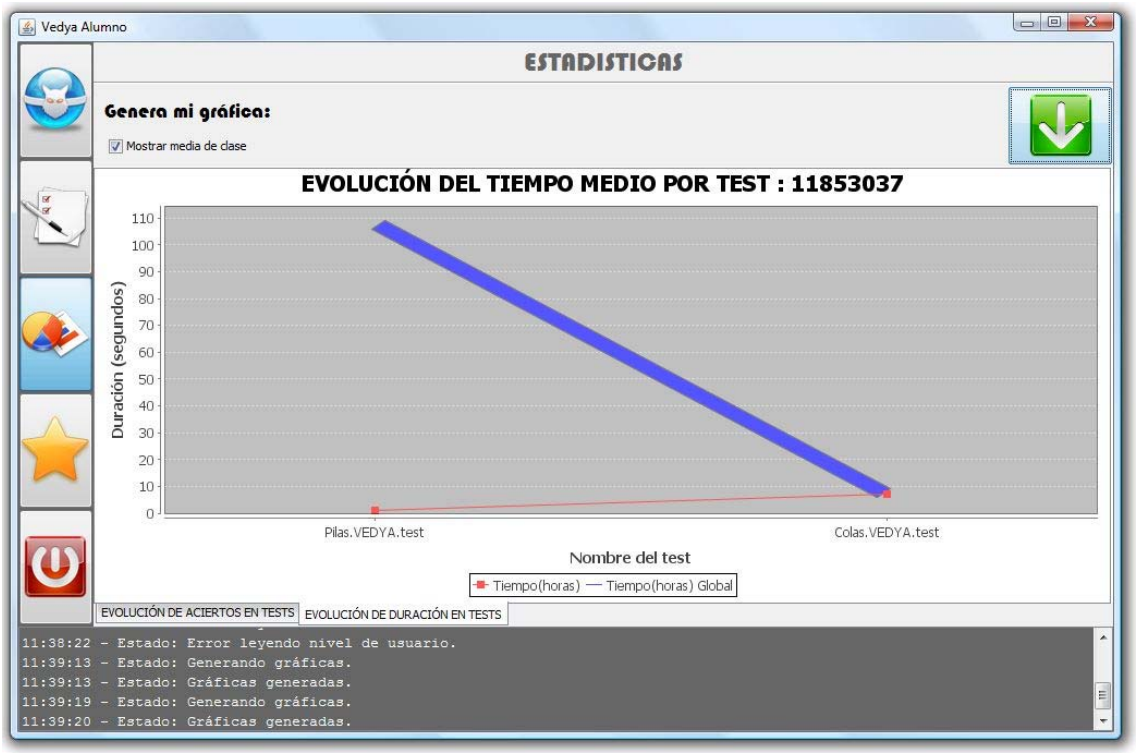


Figura 23. Estadísticas en Vedya Alumno

5.3.2.5 Módulo de opciones

El módulo de opciones representa un conjunto de posibles acciones que el usuario puede elegir a la hora de personalizar la aplicación.

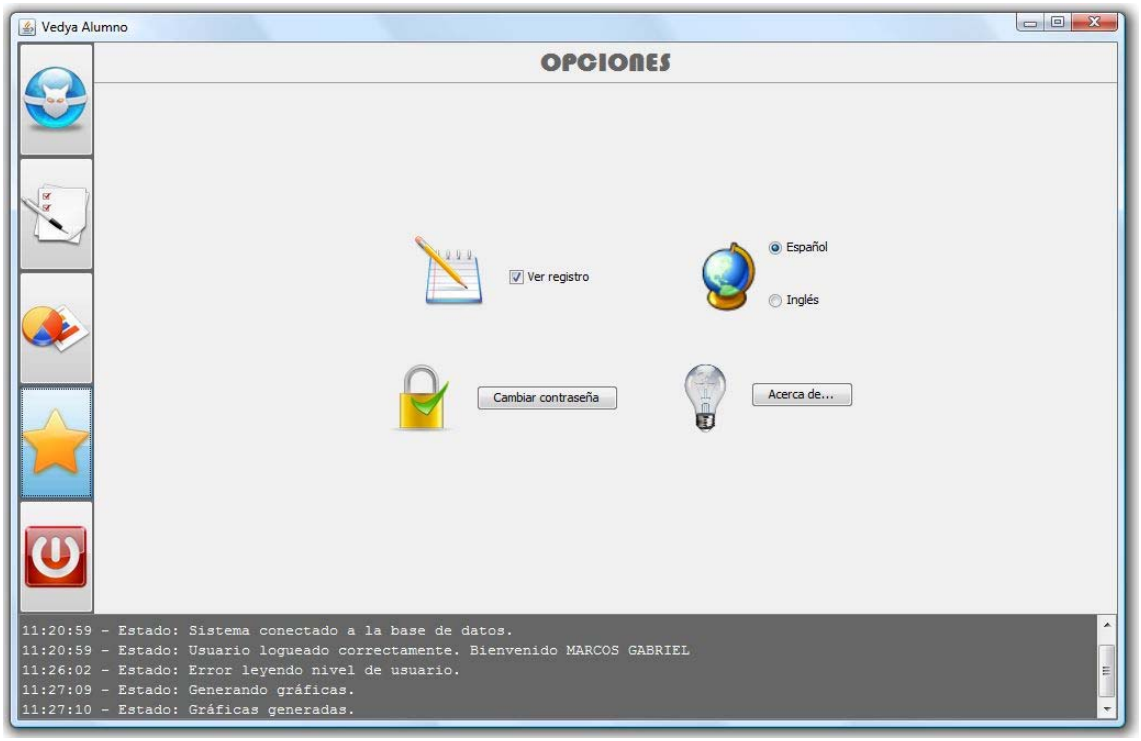


Figura 24. Panel de opciones en Vedya Alumno

Mediante la primera opción “Ver Registro”, el usuario puedo activar o desactivar la barra inferior de la aplicación que muestra información completa del transcurso de la ejecución de la aplicación, pero que en monitores pequeños puede ser contraproducente su visualización, por lo que puede desactivarse sin más que desmarcar dicho botón.

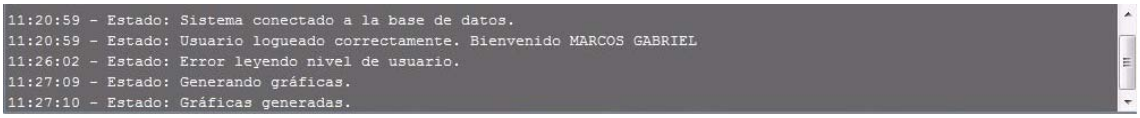


Figura 25. Barra de estado en Vedya Alumno

Así mismo, la aplicación ha sido traducida en gran medida al inglés, como es el caso tanto de la barra de estado como la mayor parte de las palabras de dicha aplicación.

La preferencia seleccionada se almacena localmente en un archivo denominado “Idioma” en el directorio donde es se está ejecutando Vedya Alumno.

Mediante la opción de “Cambio de contraseña”, el usuario puede cambiar su contraseña a través de un formulario mostrado para tal efecto.

Finalmente, hemos incorporado un botón “Acerca de” en el que se muestran los “créditos” de nuestro programa, y en el que reflejamos tanto nuestro trabajo como el de todos los alumnos que nos han precedido en el desarrollo de las herramientas que nosotros hemos continuado.

5.4. Vedya Profesor

5.4.1 Introducción

Como se ha visto en la parte de los casos de uso, queda claramente definido que como actores de nuestro escenario de autorización inteligente, tenemos al Profesor y al Alumno y, si bien es cierto que hay algunos casos que comparten características comunes, lo más adecuado de cara a la implementación ha sido separar el Tutor en dos programas independientes.

Vedya Profesor no incluye ninguna herramienta de tutorización inteligente. Es un programa que ofrece la funcionalidad requerida para el Profesor y que se apoya sobre la misma base de datos que utilizan los Alumnos.

5.4.2 Descripción general de la herramienta

En primer lugar vamos a introducir cada una de las partes que componen Vedya Profesor y que se irán desarrollando en los siguientes apartados. Está estructurado de la siguiente manera:

- Módulo de autentificación
- Módulo de estadísticas
- Módulo Vedya – Test
- Módulo de opciones

Todos estos módulos se encuentran integrados bajo la misma aplicación, consiguiendo así uno de nuestros principales objetivos, esto era, desarrollar una herramienta plenamente funcionalque integre las dos herramientas desarrolladas en proyectos anteriores de Sistemas Informáticos (Vedya y Vedya – Test).

5.4.2.1 Módulo de autenticación

Visualmente, este módulo está compuesto por una ventana de usuario y contraseña como se muestra a continuación.

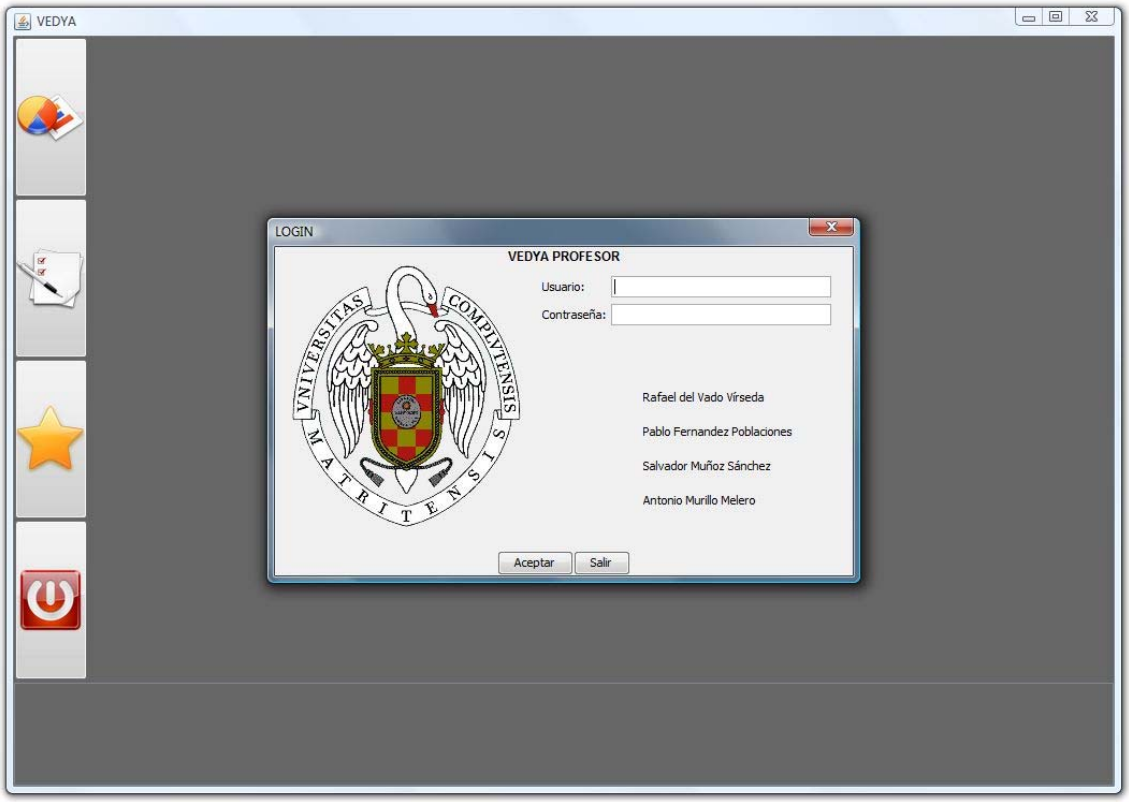


Figura 26. Ventana de Login en Vedya Profesor

Como diagrama de secuencia tendríamos el siguiente:

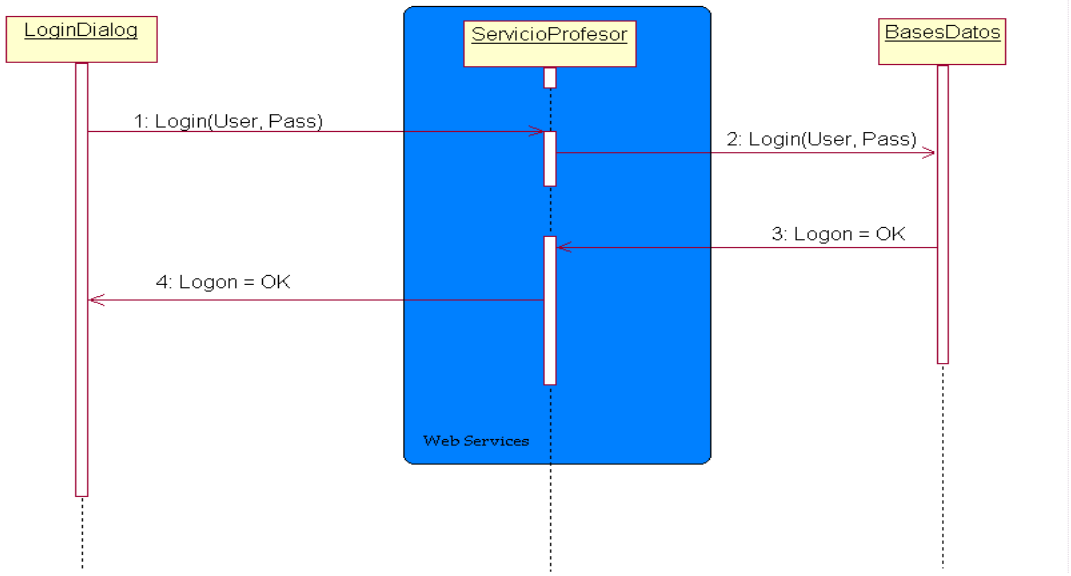


Figura 27. Diagrama de secuencia Login en Vedya Profesor

5.4.2.2 Módulo de estadísticas

Este es uno de los módulos más importante de la aplicación, ya que es el encargado de elaborar todas las gráficas de interés pedidas por el profesor del proyecto.

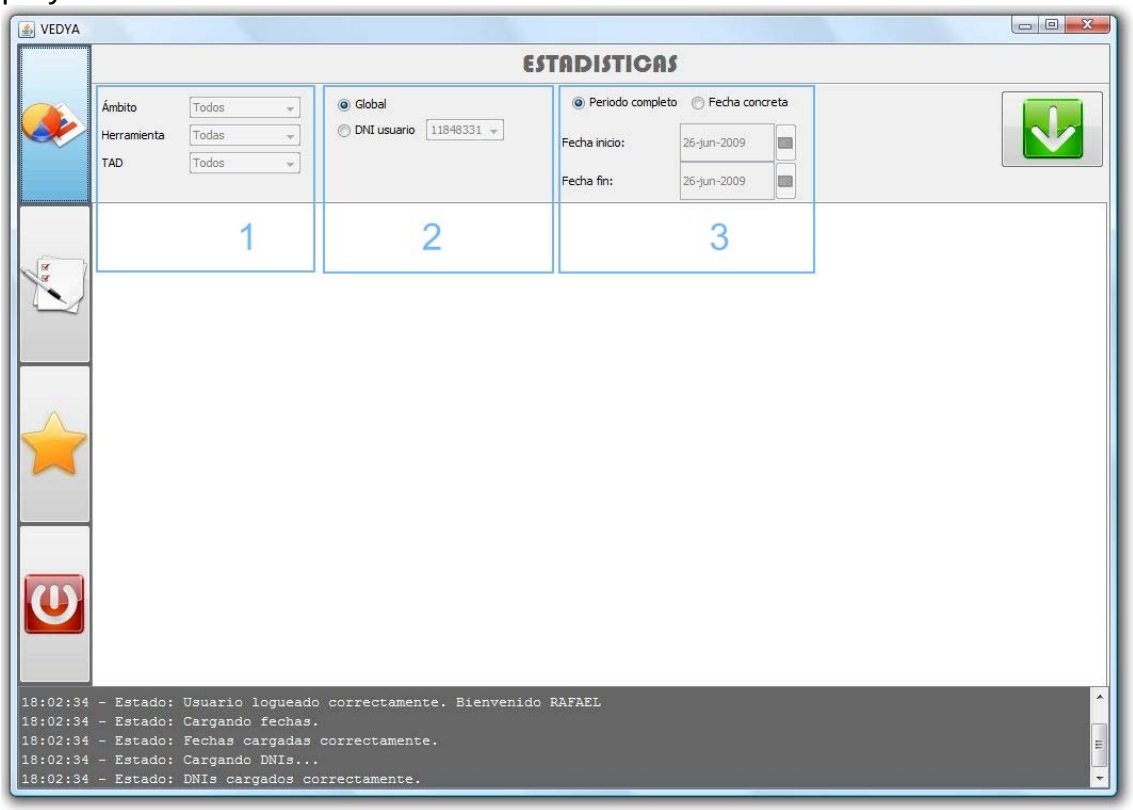


Figura 28. Módulo de estadísticas en Vedya Profesor

Está compuesto a su vez por tres partes que nos permiten configurar las gráficas que queremos generar

Parte 1

Esta parte está compuesta por tres menús desplegables con las siguientes opciones:

- **Ámbito**. Se refiere a si las gráficas mostradas corresponden a la parte de especificación de estructuras de datos, a la de implementación, a la de comportamiento (por ejemplo, a cómo se realiza una rotación doble en un AVL), o bien si engloba a todas ellas.

- **Herramienta**. Se refiere a si las gráficas mostradas corresponden al uso de la herramienta Vedya, a Vedya – Test, a Maude, a Eclipse, o bien si engloba a todas ellas.

- **TAD**. Se refiere a si las gráficas mostradas se corresponden a una estructura de datos en concreto como pilas, colas, listas, árboles, tablas, secuencias, o bien a todas ellas.

Parte 2

Esta parte nos permite saber si las gráficas las vamos a tomar referentes a todos los alumnos del profesor o si bien queremos centrarnos en uno solo en particular. Para poder seleccionar un alumno debemos elegirlo mediante su DNI.

Parte 3

Es la parte de selección de fechas. Con ella podemos generar las gráficas de todo el curso académico o bien ajustarnos a las fechas que queramos.

Una vez elegida la configuración de las gráficas, el profesor obtiene en Vedya Profesor lo siguiente:

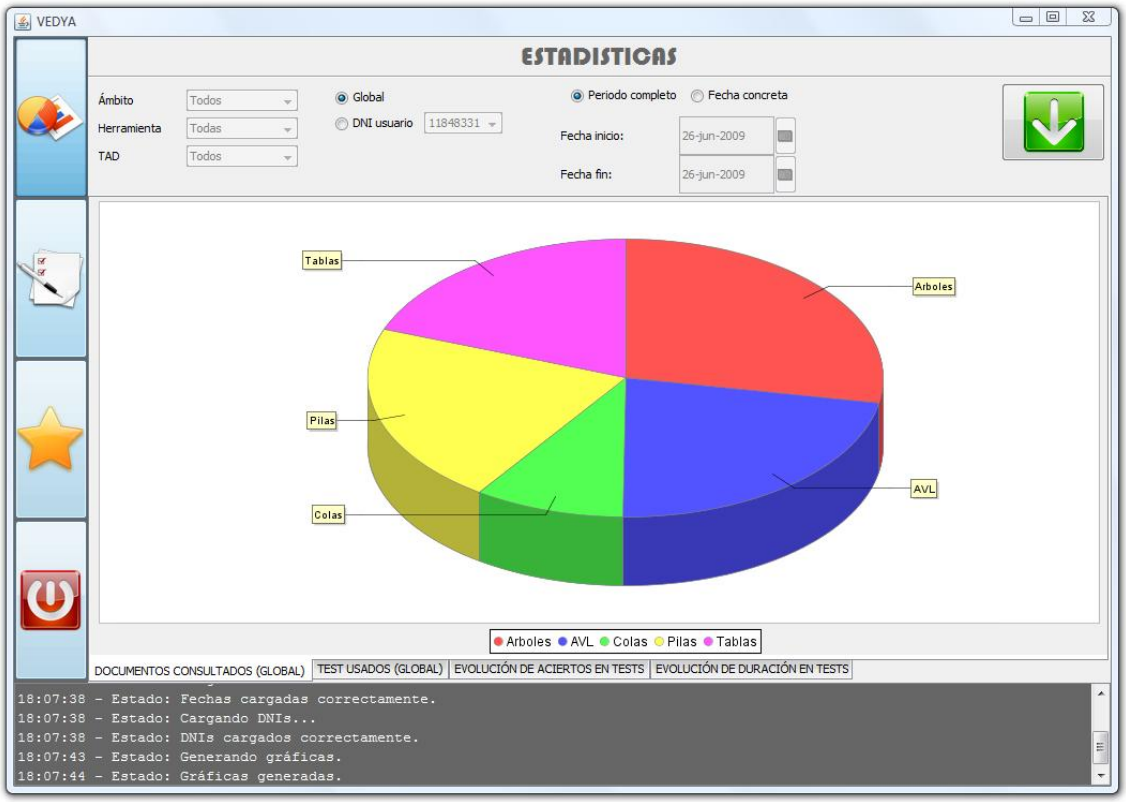


Figura 29. Ejemplo gráfica en Vedya Profesor

5.4.2.3 Módulo de Vedya – Test

Este módulo integra la herramienta ya existente de Vedya – Test

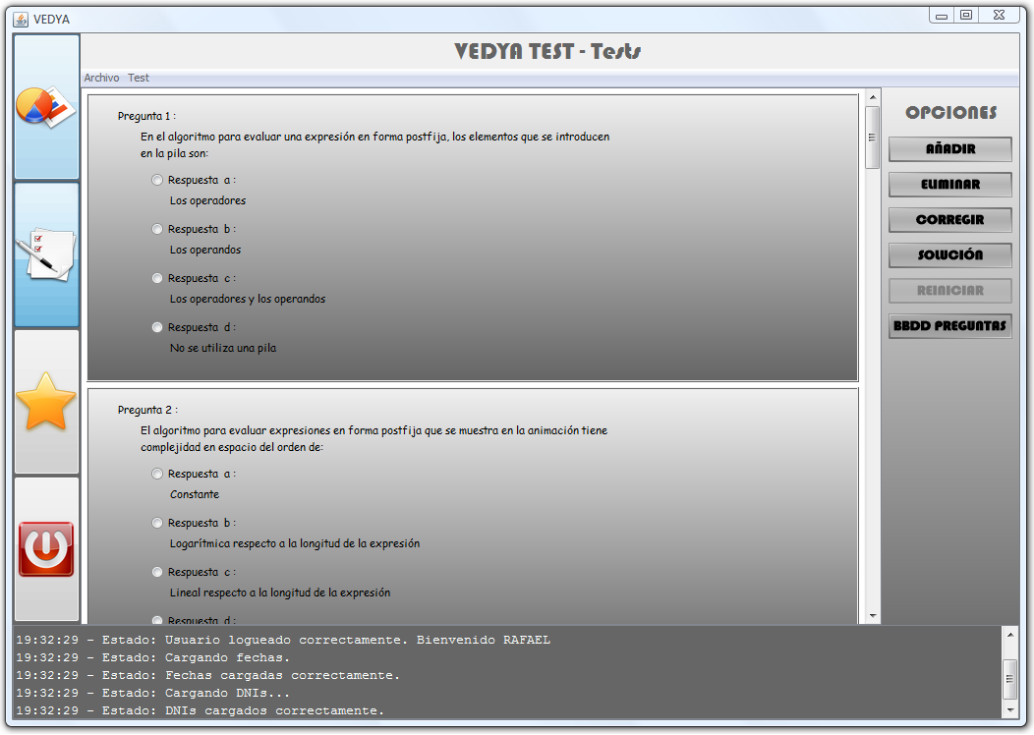


Figura 30. Panel de Vedya – Test en Vedya Profesor

5.3.2.4 Módulo de opciones

Está compuesto a su vez por cuatro partes que nos permiten una amplia variedad de opciones.

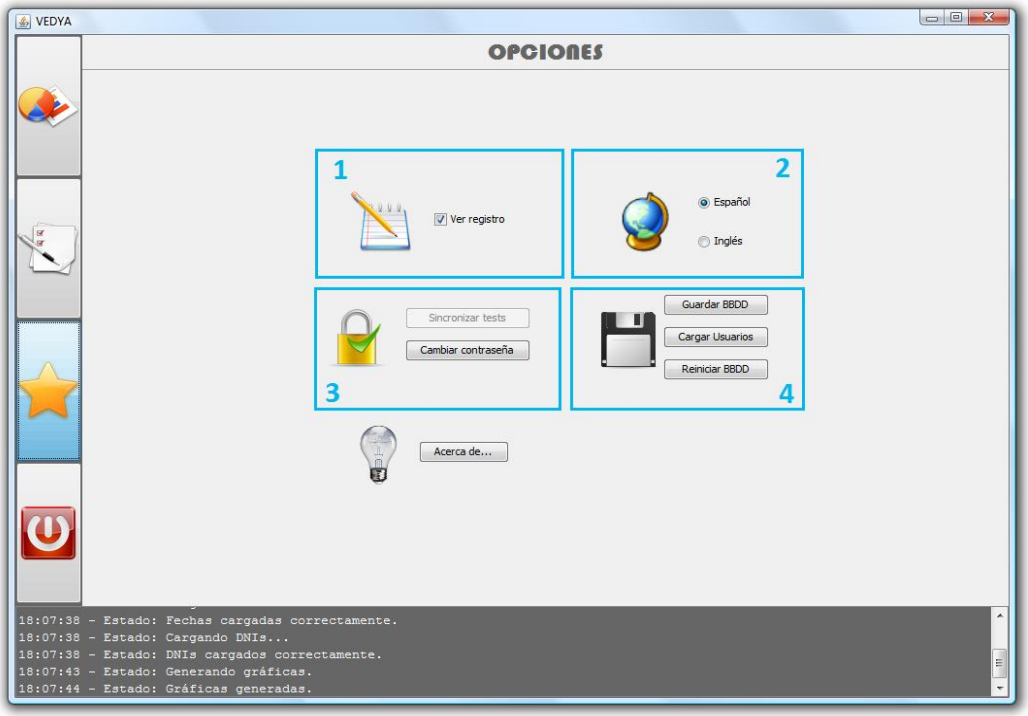


Figura 31. Panel de opciones en Vedya Profesor

Parte 1

Nos permite elegir si queremos habilitar en pantalla la barra de estado para poder llevar un seguimiento de nuestra sesión. Si por el contrario no la necesitamos la podemos deshabilitar.

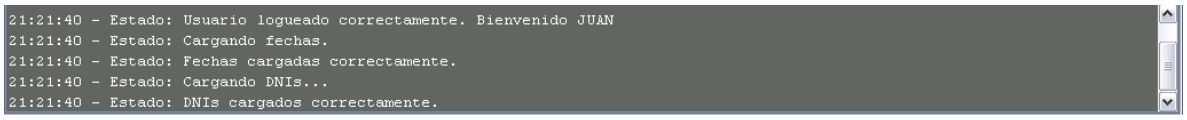


Figura 32. Barra de estado en Vedya Profesor

Parte 2

Es la parte correspondiente al idioma de la aplicación, pudiendose elegir si la queremos en español o bien en inglés.

Parte 3

Es la parte referente a la gestión de la contraseña del profesor, por si desea cambiarla.

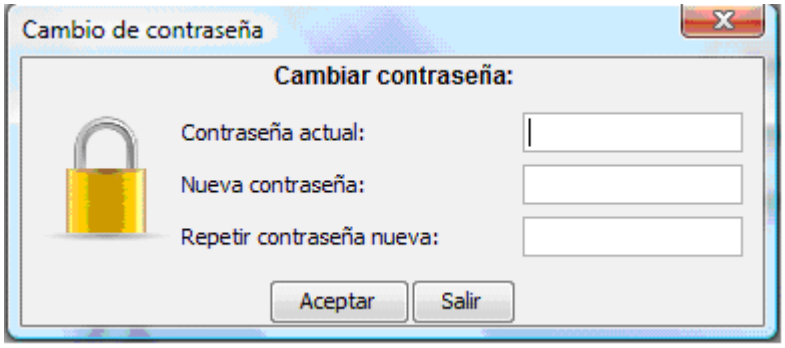


Figura 33. Panel de cambio de contraseña en Vedya Profesor

Opción para sincronizar los test

Debido a que en la parte de Vedya – Test no hemos necesitado desarrollar ninguna funcionalidad especial, decidimos crear una opción para que el profesor pueda actualizar los test existentes en el servidor, añadiendo los creados en el módulo de Vedya – Test, de forma queestos test nuevos estén disponibles para todos los alumnos instantáneamente a través de la herramienta Vedya Alumno, y de manera totalmente transparente para los alumnos.

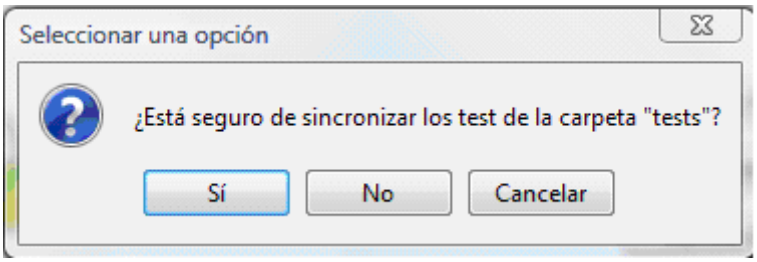


Figura 34. Panel de sincronización de tests en Vedya Profesor

5.5. Despliegue del Proyecto

Para el despliegue del Proyecto, se ha usado el servidor instalado en la Facultad de Informática de la Universidad Complutense de Madrid. De esta manera, se podrá garantizar la seguridad de los datos de los alumnos, ya que todos sus datos estarán almacenados en el servidor, y no serán visibles ni modificables por ningún personal no docente. Los usuarios deberán descargar y ejecutar en sus propios equipos la aplicación destinada al aprendizaje, y el tutor, de manera transparente, les guiará a través de la aplicación, desplegando las diferentes opciones y emitiendo datos a la BBDD.

El sistema estará compuesto por 5 componentes fundamentales:

- **La aplicación Alumno**: Software que podrá ser descargado para usarse por los alumnos. Esta aplicación engloba los sistemas Vedya y Vedya-Test y les añade la funcionalidad de las estadísticas personales y todo el sistema interactivo de control de progreso del alumno y desbloqueo de nuevas funcionalidades en el programa. Esta aplicación se conectará al Servicio Web ServicioAlumno. Para realizar la conexión, usará un encriptado SSL, de manera que las transmisiones entre el servicio y el software distribuido queden aseguradas.

- **La aplicación Profesor:** Software que será manejado sólo por el profesor. Esta aplicación tiene capacidad para subir test a la Base de Datos, modificar test existentes, consultar estadísticas globales de las clases y estadísticas particulares de cada alumno. Podrá cargar nuevas clases con alumnos, borrar las existentes y reiniciar la BBDD siempre que lo desee.

- **El servicio Alumno**: Es un servicio Web encargado de recibir todas las peticiones de los programas clientes de los alumnos. El servicio ofrece una serie de utilidades al programa Aplicación Alumno. Por cada petición, lanza un hilo en el Servidor para responder a las necesidades del alumno. En nuestro proyecto está configurado para conectarse a una Base de Datos local, pudiendo ser configurado para conectarse a una base de datos remota. Este servicio solo tiene permisos de escritura y de modificación de determinadas tablas.

- **El servicio Profesor**: Servicio que comparte muchas de las funcionalidades de servicioAlumno, pero con la aplicación del profesor. Tiene mayores privilegios sobre las BBDD que servicioAlumno, ya que puede modificar todas las tablas, reiniciarlas, añadir tablas, etc. Está pensado para atender a las peticiones que se realizan desde la aplicaciónProfesor.

- **Bases de Datos**: Ubicadas en el servidor, es el lugar donde se almacenarán los datos de manera que se garantice la privacidad de los mismos. En éste caso, se han almacenado en el mismo sistema en el

que se encuentra el Servidor Web donde están desplegados los Servicios Web.

- **SFTP:** Ubicado en un servidor aparte, usaremos esta versión segura de FTP para transmitir datos de mayor peso. Un ejemplo de estos datos serían los test que el programa debe descargar para que el alumno pueda hacer. Esos test pueden ser modificados por el profesor, y el programa descargaría las nuevas versiones automáticamente desde el SFTP. Así mismo, las bases de datos también deben tener información sobre qué archivos se encuentran en el SFTP, ya que son los que darán al programa la ruta del archivo a descargar.

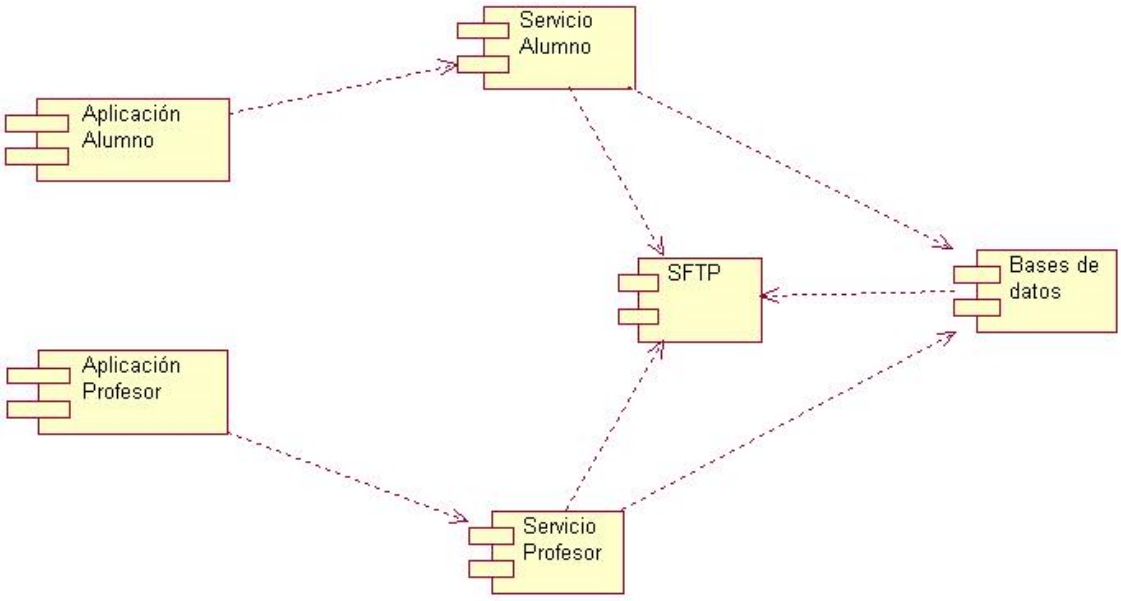


Figura 35. Despliegue del proyecto

Si observamos el diagrama superior, se diferencian claramente la capa de aplicación, la capa de comunicación y presentación de datos, y las bases de datos y sistemas de almacenamiento.

5.5.1. Justificación de la elección de Servicios Web

Hemos elegido la tecnología de los Servicios Web para la implementación de nuestro proyecto porque las características del mismo creemos que se ajustan adecuadamente a nuestras necesidades, ya que podemos aprovechar muchos de los puntos fuertes de ésta tecnología.

Las ventajas en la utilización de esta tecnología son las siguentes:

- *La base de Datos no es visible directamente al usuario*. Esto hace que los alumnos no puedan ver directamente las Bases de Datos, y sólo sean visibles a ellos las funciones para las que están previstos los servicios. Así conseguimos que los alumnos no puedan acceder ni modificar sus propios datos fuera de los límites para los que está diseñado el servicio.

- *Fácil realojo de las BBDD*. Al no conectarse directamente el software de Usuario a las BBDD, sino los servicios Web, es mucho menos costoso el hecho de cambiar de servidor los datos de los usuarios, ya que sólo necesitaríamos modificar el servicio. No haría falta así redistribuir el software entre los usuarios.

- *Interoperable con otros sistemas*. En caso de cambio de las aplicaciones de Alumno y Profesor, se podría seguir usando las funcionalidades de los servicios, aun en caso de implementarlos en otro lenguaje distinto, o ejecutarlos en otro tipo de Sistema Operativo. La comunicación con los servicios es independiente de la plataforma y de lenguaje. Luego permiten una gran flexibilidad a la hora de la implementación.

- *La interfaz con el usuario se realiza en un Servidor Web*. Esto permite que el usuario sólo vea la parte del servidor Web, no pudiendo acceder realmente al propio servidor. El servidor Web no suele contener datos de tanta importancia como el servidor real. Éste hace peticiones al servidor real. De esta manera se consigue que el usuario nunca pueda llegar realmente a los datos, si no es a través de las propios interfaces que ofrece la aplicación.

- *Permite cambios en el propio servidor*. Al conectarse mediante el puerto 80 (puerto http), la comunicación con los servicios, independientemente de las reglas de filtrado que usen, se puede seguir realizando de la misma manera ya que este puerto suele permanecer abierto para la comunicación Web.

Los puntos más débiles de esta tecnología son:

- *La comunicación vía texto es lenta*. Los servicios Web usan una comunicación mediante XML. Esto hace que la comunicación mediante caracteres sea mucho menos eficiente que una comunicación mediante

código byte. Debido a esta desventaja de la tecnología, hemos minimizado la conexión con el servidor, de manera que la comunicación sea lo más fluida posible. La Aplicación Alumno no debería tener demasiado retardo, ya que los datos a enviar no son muchos. La Aplicación Profesor sí podría tener más problemas a la hora de comunicarse, en especial al principio de un curso, cuando debe enviar los datos de los alumnos. Pero este caso solo se va a dar una vez al año, y creemos que merece la pena las ventajas obtenidas a cambio de que el profesor, en una ejecución al año, vaya algo más lento.

- *Los Servicios Web no ofrecen gran seguridad*. Para solucionar esto hemos usado una conexión SSL, de ésta manera conseguimos que los mensajes queden encriptados y no se pueda modificar el contenido de los mensajes para obtener mejores resultados.

5.5.2. Otras alternativas tecnológicas

5.5.2.1. Comunicación mediante RMI

RMI que proviene de las siglas en inglés de *Remote Method Invocation*, presenta una dependencia del lenguaje Java, por lo que descartamos esta comunicación. A pesar de ser una tecnología mucho más eficiente, no queríamos cerrar el paso a nuevas implementaciones en nuestro proyecto.

Utilizar este tipo de tecnología de conexión directa hacía muy poco portable el proyecto, ya que no se podían cambiar las Bases de Datos del servidor sin tener que rehacer todo el programa de nuevo.

Presentaba también un problema de seguridad, ya que habría que realizar la conexión mediante un túnel SSH, dando permisos innecesarios a los usuarios en el servidor en el que se desplegaran las bases de datos.

En nuestro caso, tratándose de usurarios que están estudiando Informática, y siendo el servidor de la Facultad, no creímos que fuera lo más adecuado que tuvieran permisos, ya que sería generar puntos vulnerables para el servidor.

5.5.2.2. Comunicación puerto a puerto sin una capa superior

Pensamos que trabajar con Sockets directamente podría ser una solución muy válida, ya que los propios Servicios Web trabajan con Sockets indirectamente.

Los Sockets ofrecen, además, grandes ventajas como la capacidad del servidor para emitir señales en un determinado momento, lo que nuestra aplicación ahora mismo no requiere.

Los servicios no permiten que el propio servicio haga peticiones al usuario, simplemente se limita a recibir peticiones y a darles una solución.

No elegimos esta tecnología porque los Servicios Web nos ofrecían un rango de acciones suficientes para lo que nuestra aplicación necesitaba, y a cambio de esas funciones que no nos daban, nos proporcionaba una seguridad y una flexibilidad mucho mayores que la comunicación puerto a puerto. Aun así, es una opción muy buena, ya que es mucho más eficiente que los servicios y permite transmitir datos en un formato distinto que no sea texto, lo cual podría ser necesario si la aplicación se ve modificada para enviar gran cantidad de datos.

5.5.3. Dificultades encontradas en el despliegue

Las principales dificultades encontradas a la ahora de realizar el despliegue de las aplicaciones Vedya Profesor y Vedya Alumno han sido las siguientes:

- **Desconocimiento de la tecnología**. Hemos tenido que aprender a usar esta tecnología de comunicación sin tener ninguna noción de ella. Esto nos hizo tener que estudiar todas sus características para ver que se adecuaba correctamente a nuestros fines. Y una vez elegida, adaptar nuestro diseño del programa a esta tecnología para poder usarla.

- **Envío de sólo texto**. Tal y como hemos explicado, los servicios Web se comunican mediante formato texto. Los datos de nuestra aplicación existían en una base de datos, y los datos que nos pedían las aplicaciones que íbamos a integrar dentro de nuestro sistema no estaban en formato texto. Así pues, tuvimos que codificar nuestros datos a texto tanto en el cliente como en el servidor para adaptarnos a la tecnología elegida. Ante este problema nos planteamos 2 opciones: Crear un sistema de reglas de conversión de nuestras clases a texto para que fuera ejecutado por un elemento externo a nuestra aplicación, o convertir nuestras propias clases a texto para la transmisión. Elegimos la segunda opción porque creíamos que crear un sistema de reglas para nuestras clases, hacía que las posibles futuras implementaciones del servicio tuvieran que usar nuestros propias clases de comunicación. De esta manera, creemos que si en un futuro se decide cambiar la implementación, se verá cómo se ha tratado la conversión y podrán hacer su propia conversión sin tener en cuenta nuestro diseño.

- **Conexión con un servidor seguro**. Debido a las características de la tecnología, esta parte fue muy sencilla, ya que la conexión al servidor Web se hizo de manera poco costosa, y la tecnología viene preparada para usar un servicio de encriptamiento de manera bastante intuitiva.

- **No control del servidor**. Aunque el trabajo de los técnicos ha sido muy bueno, el no poder controlar el servidor y depender de ellos para cada prueba a realizar hace que la fase de pruebas en el servidor sea más lenta. Así, después de cada prueba fallida debíamos contactar con los

propios técnicos para poder hacer la siguiente, lo cual hacía que, en muchas ocasiones, se parara la progresión del proyecto a la espera de nuevos resultados.

5.5.4. Operaciones de un servicio: Servicio Alumno

A continuación mostramos las operaciones que permite hacer el Servicio Alumno. Para el Servicio Profesor la estructura y las operaciones son similares, motivo por el cual solo mostramos un servicio.

getResultSetDatosUtilizaAmbitoNombres

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
where		java.lang.String	

getResultSetDatosUtilizaAmbitoAccesos

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
where		java.lang.String	

cuantosGetEvolucionTiempoServ

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
where		java.lang.String	

getResultSetEvolucionUsuarioNombres

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
where		java.lang.String	

getResultSetEvolucionUsuarioAciertos

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
where		java.lang.String	

cuantosGetEvolucionUsuarioServ

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
where		java.lang.String	

recuperaGrupo

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	

recuperaContrasena

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	

recuperaNivel

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	

recuperaVeces

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	
nombreTest		java.lang.String	

getControlPilas

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	

getControlColas

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	

getControlAVL

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	

getControlArbolBinario

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	

setControlPilas

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	
control		java.lang.String	

setControlColas

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	
control		java.lang.String	

setControlArbolesBinarios

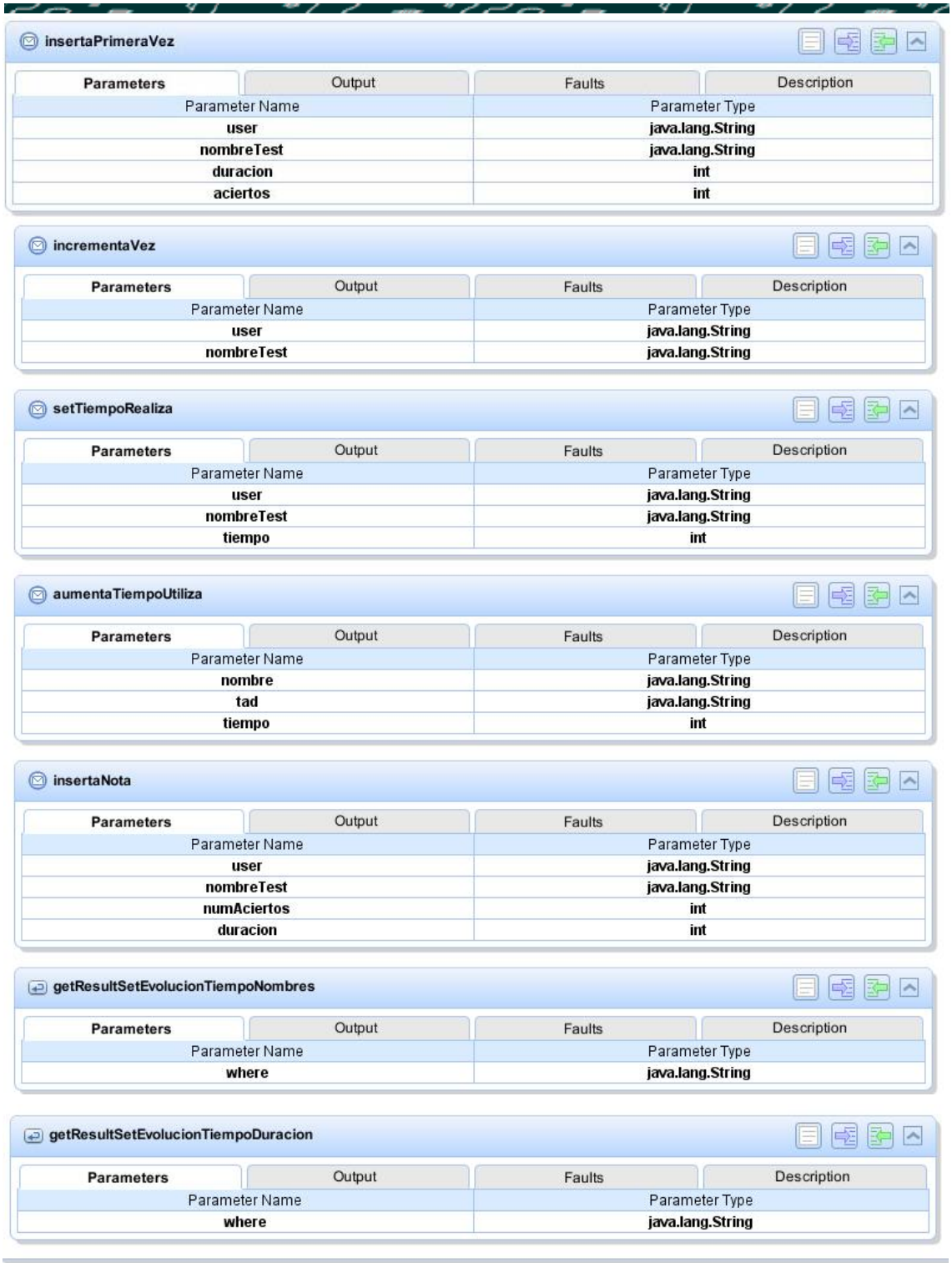
Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	
control		java.lang.String	

aumentaDeNivel

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	

setControlAVL

Parameters	Output	Faults	Description
Parameter Name		Parameter Type	
user		java.lang.String	
control		java.lang.String	



insertaPrimeraVez

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
user	java.lang.String
nombreTest	java.lang.String
duracion	int
aciertos	int

incrementaVez

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
user	java.lang.String
nombreTest	java.lang.String

setTiempoRealiza

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
user	java.lang.String
nombreTest	java.lang.String
tiempo	int

aumentaTiempoUtiliza

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
nombre	java.lang.String
tad	java.lang.String
tiempo	int

insertaNota

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
user	java.lang.String
nombreTest	java.lang.String
numAciertos	int
duracion	int

getResultSetEvolucionTiempoNombres

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
where	java.lang.String

getResultSetEvolucionTiempoDuracion

Parameters	Output	Faults	Description

Parameter Name	Parameter Type
where	java.lang.String

Quality Of Service

- ☐ Optimize Transfer Of Binary Data (MTOM)
- ☐ Reliable Message Delivery
- ▣ Secure Service

5.6. Logros y Limitaciones

5.6.1. Logros

Se ha desarrollado completamente el Módulo del Estudiante, recogiendo todo el comportamiento observable que hemos creído necesario para guiar adecuadamente al estudiante y poder mejorar su aprendizaje y el adecuado uso que hace de la Herramienta Vedya, Vedya-Test, y de los apuntes de la asignatura.

Se ha contemplado gran parte del conocimiento de la asignatura (todo lo implementado en la Herramienta Vedya y los test hechos) de manera que se tenga un esquema general de cómo debe ser el aprendizaje y los módulos que quedan implementados. Asimismo, se ha hecho un esquema genérico de manera que no resulte excesivamente costoso añadir nuevos módulos de temario al Tutor y poder monitorizar el aprendizaje de estos, así como poder aplicar las mismas técnicas que a los ya implementados.

Se ha establecido una completa comunicación entre el Estudiante, y el Sistema, y entre el Profesor y el Sistema. De esta manera conseguimos que el aprendizaje pueda ser controlado por el Sistema y revisado por el Profesor. El propio alumno puede comprobar si su aprendizaje está siendo correcto comparando sus resultados con los de una estimación del resto de la clase, para poder comprobar así si su nivel de conocimiento es el adecuado.

El Tutor Inteligente está listo para ser usado en las primeras etapas del temario, estando a la espera de nuevos contenidos para recoger datos sobre estos y poder controlar toda la asignatura.

Se ha conseguido un sistema dinámico de manera que el alumno no sienta que no avanza en el estudio, con ello se consigue que el aprendizaje sea más dinámico y que el alumno no se sienta desmotivado ante la cantidad de temario.

El Tutor es capaz de detectar las partes no exploradas del temario por parte del alumno y de avisarle de sus carencias. De esta manera, cada sesión el alumno deberá potenciar sus capacidades en un área en la que todavía tiene deficiencias.

5.6.2. Limitaciones

Creemos que las futuras versiones del Tutor podrían incluir:

- *Mayor adaptación a la herramienta Vedya*. El funcionamiento del Tutor se restringe al aprendizaje de Estructuras de Datos. Este funcionamiento podría ser ampliado a más estructuras de datos (ampliando a su vez la herramienta Vedya para que pudiera trabajar con un mayor número de estructuras). También debería poder trabajar con el

apartado de Métodos Algorítmicos. Actualmente el tutor no recoge información del trabajo y resultados del alumno en este campo.

- *Expansión del tutor para poder trabajar con otros sistemas de aprendizaje*. Actualmente el Tutor está preparado para trabajar, específicamente, con la herramienta Vedya. Sería muy interesante, de cara a una posible ampliación, generalizar su otro sistema de comunicaciones para que pueda ser conectado a cualquier sistema de aprendizaje que se desee utilizar (especificando protocolos de conexión que deberán cumplir los sistemas a monitorizar). Habría que replantearse algunas estructuras usadas y determinados conceptos, pero el resultado sería un Tutor mucho más flexible, e incluso podría llegar a comercializarse si los resultados resultan ser suficientemente satisfactorios.

- *Mayor número de opciones*. El sistema es limitado en cuanto al número de idiomas en los que puede trabajar (ahora mismo sólo hay 2 implementados, español e inglés), y carece de opciones de visualización. Creemos que este apartado se puede mejorar incluyendo opciones de visualización más personalizables (colores, tamaños, etc.) y aumentando el número de idiomas en los que se trabaja. Es especialmente importante esta expansión si se lleva a cabo la pretensión de generalizar el programa, ya que en este momento el programa no necesita trabajar en más idiomas que los actualmente implementados, pero en caso de que pudiera trabajar de manera genérica, debería incluirse más idiomas para poder usar el tutor en el mismo idioma que los programas que va a monitorizar.

- *Los test pertenecen a un temario específico*. Se podría intentar generar test que fueran algo más específicos para cada alumno. A pesar de que existen varios test para cada etapa del aprendizaje, creemos que sería bueno diferenciar a nivel de pregunta, cada test, para poder enfatizar las partes en las que el alumno necesita un mayor refuerzo.

- *Alterar el orden de las preguntas en los test*. Puesto que actualmente se permite hacer dos veces el mismo test para pasar al siguiente nivel, los alumnos tienen la opción de falsear los resultados en el segundo. Si bien es cierto que para evitar este problema decidimos que aunque no se aprobara el test la segunda vez se permitiera el desbloqueo del siguiente tema de teoría, esta solución quedaría mejor si se realizara una alteración en el orden de las preguntas la segunda vez que se resuelve el test.

- *El sistema nunca propone ejercicios de repaso*. Creemos que para un buen aprendizaje, sería de gran utilidad que el sistema propusiese test de repaso del temario cada cierto tiempo. Se deja en manos del alumno la decisión de repasar o no una vez adquirido el conocimiento para pasar de etapa. No obstante, sí se tiene en cuenta en el sistema si un

alumno repasa, y el profesor posee información de qué alumnos hacen test de repaso.

5.7. Expansión del Tutor Inteligente

5.7.1. Motivación

Una de las carencias críticas más usuales de este tipo de Sistemas de Tutorización Inteligente es su falta de genericidad. El Tutor está pensado para trabajar con la herramienta Vedya, y no puede trabajar con ninguna otra aplicación de aprendizaje que no sea ésta. Debido a ello, creamos este apartado, para explicar cómo habría que cambiar la herramienta para poder hacerla genérica, y los requerimientos que necesitaría cualquier herramienta para poder interactuar con un tutor.

5.7.2. Requerimientos

Primeramente, la herramienta que trabaje con el Tutor deberá seguir un índice (lo cual, dado que estamos trabajando con herramientas pedagógicas, parece una característica que la mayoría de las herramientas de aprendizaje deberían poseer). Junto con el índice, el creador de la herramienta deberá especificar la secuencia de logros y posibilidades que se van abriendo, en forma de grafo dirigido. Gráficamente:

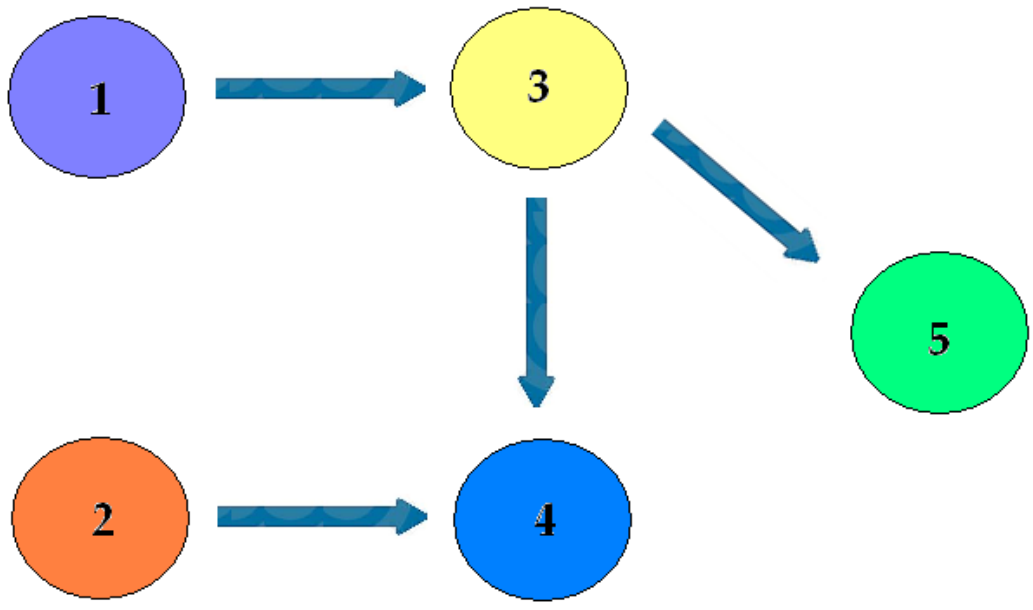


Figura 36. Ejemplo de grafo de estados

En cada uno de los nodos se deberá especificar la puntuación necesaria en los test para considerarlo como apto, y el grado de exploración de las opciones de cada nodo (si aplicable).

Según el grafo mostrado, una vez superado el nivel 1, se nos desbloqueará el nivel 3. Una vez superado éste, tendremos acceso a los niveles 4 y 5 de nuestro sistema de estudios.

- El sistema de Bases de Datos, salvo leves modificaciones (ausencia de número identificativo o características propias de cada centro académico), es un sistema válido para la expansión.

- La herramienta del profesor será la misma, ya que los datos de acceso son similares, y a él no le afectaran los cambios.

- Para la comunicación de los datos que necesite el Tutor (ver primer requisito de este documento), recomendamos que se haga mediante un archivo cifrado. Ejecutando el programa en el mismo sistema que el Tutor, el programa deberá generar un archivo cifrado (recomendados XML cifrado por la estructura de los datos que se vayan a entregar). El Tutor monitorizará cambios en ese archivo y emitirá la información entregada por la herramienta a las Bases de Datos.

- Es necesario conocer a priori, en cada nivel, que grado de exploración se necesita. En el caso de Vedya, el grado de exploración se calcula mediante el tiempo usado en leer las transparencias, el número de botones pulsados, y el número de simulaciones realizadas por el alumno. Es necesario conocer el grado de exploración que se requiere en cada nivel, y los parámetros que se van a dar. El usuario deberá especificar qué elementos se van a tener en cuenta en un primer archivo de configuración general (el cuál debería ser una aplicación de la herramienta del profesor). Finalmente, la herramienta que fuera a usar el Tutor, debería enviar los datos de exploración según lo indicado para ese curso concreto por el profesor.

5.7.3. Trabajo a realizar de cara a la expansión

El primer paso a realizar es modificar la herramienta del profesor para que sea posible incluir posible parámetros gracias a los cuales queden completamente definidos los objetivos de cada uno de los apartados en los que se va a dividir el temario. Habrá que añadir esa funcionalidad a los Servicios Web. Otra opción es la de cambiar manualmente las bases de datos, pero se perdería parte de la generalidad del proyecto si optamos por cambiar las bases de datos a mano, ya que habría que hacerlo para cada aplicación que quisiéramos tutorizar, y ese creemos que no es el objetivo de esta expansión.

Una vez tenemos completamente definidas las tablas y los objetivos a realizar, pasamos a trabajar con la parte del Tutor que se ejecutará a la vez que el programa que queremos tutorizar. Este programa debe estar en contacto con la Aplicación Cliente, para recibir los datos de cada sesión. Se recomienda que no se dejen todos los datos de una sesión sin enviar hasta el final de la sesión, ya

que generaría mucho tráfico y tendría implicaciones en la seguridad. La idea propuesta es que se envíen archivos XML cifrados por parte del programa cliente al cambiar determinadas ventanas. Recoger dichos archivos con nuestro Tutor, desencriptarlos en función de las claves, mandar la información y destruir los archivos. Este sistema es fácilmente implementable por cualquier programa, pero a cambio es muy débil desde el punto de vista de la seguridad.

Debido a ello, creemos que la mejor opción es la conexión vía puertos a nuestro propio sistema para enviar la información y hacer que el tutor esté a la escucha en uno de esos puertos. El problema de este tipo de implementaciones es que es necesaria una modificación muy importante del programa del cliente. A cambio se puede conseguir la comunicación por un canal seguro, ganando así la aplicación en robustez.

El Servicio Web del alumno no debería ser modificado para esta expansión, ya que es suficientemente genérico para soportar este tipo de expansiones.

Un esquema general de cómo quedaría el sistema completo podría ser:

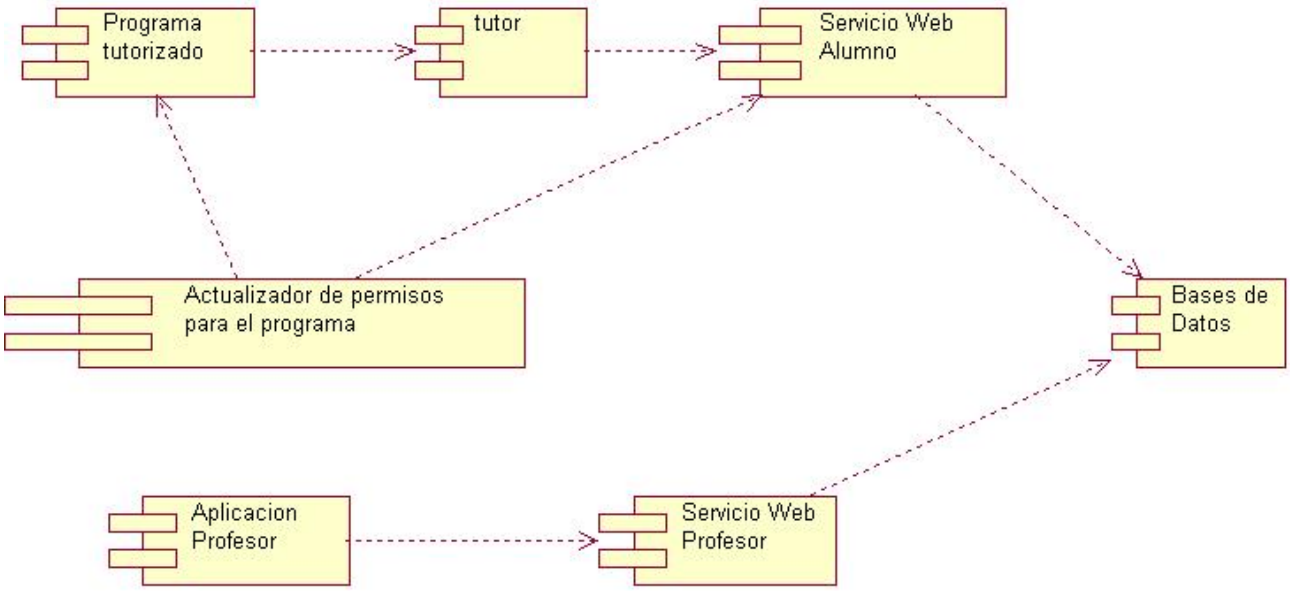


Figura 37. Despliegue del proyecto en extensión

El Módulo de Actualizador de permisos para el programa se encarga de indicarle al sistema el cual vamos a tutorizar, que elementos deben estar activos en cada momento, y qué mensajes debe mostrar a los usuarios en función de los resultados obtenidos. Para ello es necesaria una comunicación con las bases de datos a través del servicio con el fin de pedir datos de cuales son las acciones que debe realizar.

El punto crítico del sistema es la comunicación entre el programa y nuestros componentes, tanto el Tutor como el actualizador de permisos. Dependiendo de la seguridad y de los requisitos de dicha comunicación, habrá que emplear diferentes mecanismos de conexión.

Valoración personal

Durante la carrera lo normal es que todo se desarrolle desde cero prácticamente. En todas las asignaturas de prácticas siempre te enfrentas al problema de cómo crear lo que quieren que crees.

Ahora bien, el desarrollo de este proyecto nos ha permitido ir un paso más allá, esto es, aparte de enfrentarnos al problema de crear el Tutor nos hemos tenido que enfrentar con un par de problemas que creemos haber solventado correctamente.

Estos dos problemas fueron:

- *Código heredado*

  Ser los cuartos que continúan con el desarrollo de un proyecto de mayor envergadura es un problema añadido a la hora de desarrollar, puesto que hay muchas decisiones de diseño que fueron tomadas por nuestros predecesores y que nos condicionó nuestro desarrollo en su misma dirección. Algunas de ellas nosotros mismos hubiéramos decidido enfocarlas desde otro punto de vista. Este problema lo hemos solventando satisfactoriamente con la integración en nuestro proyecto de Vedya y Vedya – Test.

- *Desconocimiento de tecnologías*

  Nos hemos enfrentado a tecnologías que no conocíamos, como son los Web Services y toda la parte de seguridad en conexiones con servidores ya que en la carrera no se estudian.

  Con las bases asentadas durante nuestra formación como alumnos hemos sido capaces de entender, aprender y desarrollar basándonos en estas tecnologías con un muy buen resultado.

Creemos que este proyecto nos ha hecho madurar de cara al mundo laboral que se nos abre en este momento. Estamos muy contentos con el desarrollo de este proyecto, el trabajo en equipo realizado y que a nuestro director del proyecto le haya gustado el resultado final.

También nos llena de orgullo saber que tenemos dos publicaciones internacionales debidas a este proyecto en el CSEDU 2009 [15] y en el ICCS 2009 [16].

Índice de figuras

Herramientas en la Web

- Vedya

  http://www.fdi.ucm.es/profesor/rdelvado/Vedya.zip

- Vedya-Test

  http://www.fdi.ucm.es/profesor/rdelvado/vedya-test.zip

- Maude

  http://www.fdi.ucm.es/profesor/rdelvado/plugins-eclipse.zip

- Eclipse

  http://www.eclipse.org/

- Maude workstation

  http://www.lcc.uma.es/~duran/MaudeWorkstation/

Bibliografía

[1] J. E. Beck, B. P. Woolf y C. R. Beal, *ADVISOR; A machine learning architecture for improving collaboration: the DEGREE approach*. International Journal of Artificial Intelligence in Education, págs.221 – 241,2000.

[2] M. Clavel*. All About Maude - A High-Performance Logical Framework*. LNCS, vol. 4350. Springer, Heidelberg (2007).

[3] R. Conejo, E. Millán, J. L. Pérez de la Cryz y M. Trella. *An empirical approach to on-line learning in siette*. Procedings of the International Conference on Intelligent Tutoring Systems (ITS 2001), págs. 604-614, 2001.

[4] R. Freedman. *What is an Intelligent Tutoring System?* Journal of Artificial Intelligence in Education 11(3): 15-16, 2000.

[5] J. González y E. Glaudioso. *Sistemas Interactivos de Enseñanza / Aprendizaje*. Sanz y Torres, 2003.

[6] D. Kopec y R. B. Thompon. *Artificial intelligence and intelligence tutoring systems: knowledge-based systems for learning and teaching*. Ellis Horwood series in Artificial Intelligence, 1992.

[7] D. McArthur, M.W. Lewis y M. Bishay. *The roles of artificial intelligence in education: current progress and future prospects*. Technical Report DRU-472-NSF, RAND, 1993.

[8] I. Pita y R. del Vado. *Estudio de una metodología de aprendizaje interactivo para la asignatura de Estructura de Datos a través del Campus Virtual*. Actas de las *IV Jornadas Campus Virtual UCM*, 2007.

[9] A. I. Saiz, P. Soler y M Cayeiro. *Aprendizaje Interactivo de Estructuras de Datos y Métodos Algorítmicos*. Sistemas Informáticos, Facultad de Informática de la UCM. Curso 2006 – 2007.

[10] R. C. Schank y C. Cleary. *Engines for education*. Lawrence Erlbaum Associates, 1995

[11] R. C. Schank. *Dynamyc Memory: A theory of learnig in computers and people*. Cambridge University Press, 1982.

[12] J. Self. *Theoretical Foundations for Intelligent Tutoring Systems*. Journal of Artificial Intelligence in Education 1(4): 3-14, 1990.

[13] C. Segura, I. Pita, R. del Vado, A. I. Saiz y P. Soler. *Interactive Learning of Data Structures and Algorithmic Schemes.* Actas de la *International Conference on Computational Science* (ICCS 2008). Springer LNCS 5101, págs. 800 – 809, 2008.

[14] C. Segura, I. Pita, R. del Vado, A. I. Saiz y P. Soler. *Un entorno educativo integrado para la visualización de métodos algorítmicos y estructuras de datos: de la especificación algebraica a la implementación*. Actas de las XIV Jornadas de Enseñanza Universitaria de la Informática (JENUI 2008).

[15] R. del Vado, P. Fernández, S. Muñoz y A. Murillo. *An Innovative Educational Environment For The Interactive Learning Of Data Structures*. CSEDU2009, 2009.

[16] R. del Vado, P. Fernández, S. Muñoz y A. Murillo. *An Intelligent Tutoring System for Interactive Learning of Data Structures*. ICCS2009, 2009.

[17] A.Tannenbaum. *Redes de Computadoras*. Prentice-Hall, 4ª Edición. 2004

Apéndice: Artículos publicados

A continuación incluimos los dos artículos que mandamos al CSEDU 2009 y al el ICCS 2009 y que ambos fueron publicados.

AN INNOVATIVE EDUCATIONAL ENVIRONMENT FOR THE INTERACTIVE LEARNING OF DATA STRUCTURES

From Algebraic Specification to Implementation

Rafael del Vado, Pablo Fernández, Salvador Muñoz, and Antonio Murillo
Dpto. de Sistemas Informáticos y Programación, Universidad Complutense de Madrid, Spain

Keywords: Virtual learning environments, virtual universities and classrooms, learning and teaching methodologies, computer supported education, interactive learning, data structures, algorithmic schemes, algebraic specification.

Abstract: The high level of abstraction necessary to teach "data structures" and "algorithmic schemes" have been more than a hindrance to students. In order to make a proper approach to this issue, we have developed and implemented, during the last years, at the Computer Science Department of the Complutense University of Madrid, an innovative interactive learning system according to the new guidelines of the *European Higher Education Area*. In this paper, we present the new main contributions to this system. In the first place, we describe the tool called *Vedya* for the visualization of data structures and algorithmic schemes. In the second place, the *Maude* system to execute the algebraic specifications of abstract data types using *Eclipse*, by which it is possible to study from the more abstract level of a software specification up to its specific implementation in *Java*, thereby allowing the students a self-learning process.

1 MOTIVATION

The study of "data structures" and "algorithmic schemes" constitute one of the essential aspects of the academic formation of every engineer in Computer Science. Nevertheless, the high level of abstraction necessary to teach these topics occasionally hinders its understanding to students. In order to make a proper approach to this issue, we have developed and implemented, during the last years, at the Computer Science Department of the Complutense University of Madrid, an innovative interactive learning system according to the new guidelines of the *European Higher Education Area* and the teaching model focused on the student.

In this paper, we present the two main contributions to this system. On the one hand, the *Vedya* tool (Segura et al., 2008), a visualization tool by means of which it is possible to provide the student with a complete learning system of both the main data structures and the more relevant algorithmic schemes. On the other hand, the *Maude* system (Clavel and et al., 2006) for the execution of "algebraic specifications" of abstract data types using the language of formal specification provided by this system.

Thanks to the development environment *Eclipse* (`http://www.eclipse.org/`), we have obtained a fully complete system that is useful for the students as well as the professors, that allows to go from the most abstract level of data structures, provided by its algebraic specification in *Maude*, until its specific implementation in a modern programming language as happens with *Java*. All this learning process can be guided and overseen in a completely autonomous way by using the *Vedya* tool, through which it is possible to make enquiries about the documentation related to each of the algebraic specifications, to distinguish between the behavior of the structure and its different implementations through the use of different views or to browse information regarding the cost of the different implementations that have been proposed.

2 THE VEDYA TOOL

Vedya is an integrated interactive environment for learning data structures and algorithmic schemes. It covers the most common data structures: Stacks, queues, binary search trees, AVL trees, priority

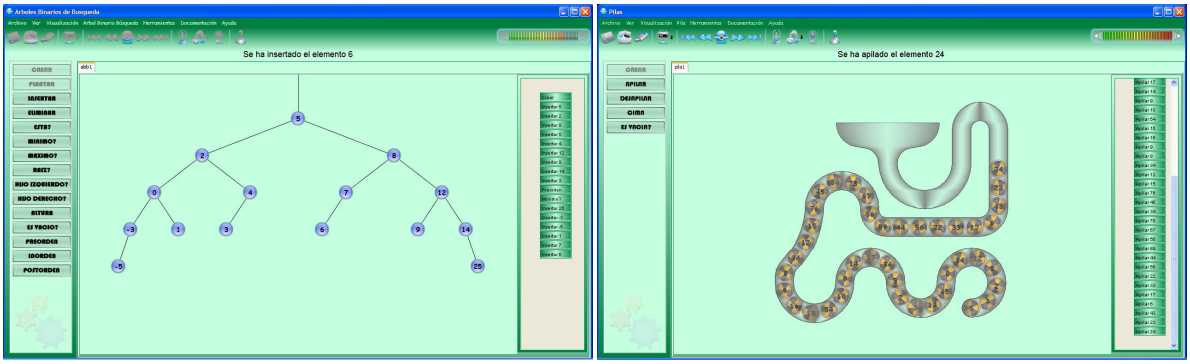


Figure 1: Data structures and algorithmic schemes in the *Vedya* tool.

queues, and sorted and hash tables. Moreover, it also provides other different types of abstract data types, like one for an implementation of a "doctor's office". Concerning the algorithmic schemes, it covers the most common resolution methods (Brassard and Bratley, 1996; Cormen et al., 2001; Neapolitan and Naimpour, 2003): divide and conquer, dynamic programming, backtracking, and branch and bound. All data structures and algorithmic schemes taught in the related study courses are thereby integrated in the same environment: *Vedya* allows the execution of different data structures and several sequences of operations on the same structures at the same time making use of a multi-windows and multi-frame system.

Currently, there are two versions of the *Vedya* tool. The first version contains all the data structures and algorithmic schemes mentioned above while the new one offers a subset of them in a more attractive visual environment. This last version can be found at `http://www.fdi.ucm.es/profesor/rdelvado/`.

There are several options to use this tool. The main one is the interactive execution, but it is also possible to create simulations that are automatically executed, to visualize tutorials and to solve tests within the same environment. It also integrates a set of animations that show how data structures are used to solve certain problems. Figure 1 shows an example of the main windows for data structures (stacks and binary trees). The central panel is used to represent the structure. In the case of linear structures and binary trees, drawing facilities are offered to allow the expansion or contraction of the data structure or to move it over the screen to see the hidden parts. On the left, there is a list of the actions that can be executed. Partial non-allowed actions are disabled. The right panel shows the visualization of the actions that have been already executed. Next, the user may continue executing actions, go up on the sequence of actions to see previous states or she/he may use the stimulation facilities (standard buttons to execute, stop, move forward and move backwards at the top of the screen) to restart the sequence from the beginning. Notice that just above the central panel the result of the last action is shown.

There are two types of views: The one of data structure behavior to intuitively comprehend its operation, and one or several implementation views, either static or dynamic. On Figure 1 we show the specific behavior view of a stack. Representations of the static implementation based on an array and dynamic implementation based on pointers can be also shown.

Furthermore, the environment provides documentation about algebraic specification, the implementation code and the cost of each implementation. On the top of the screen, there is a menu that facilitates managing the system. We can create a new data structure, open an existing one or save the state of the editing one. We can also execute the operations on the data structure, use the simulation facilities and change the execution speed of the animations.

The main window for the execution of algorithmic schemes looks similar. We have implemented algorithmic schemes based on divide and conquer of binary search and quicksort; algorithmic schemes that solve backtracking problems (in its fractional and non-fractional version) based on dynamic programming; branch and bound, as well as the Dijkstra algorithm to obtain minimum path in a graph.

As has been previously mentioned, *Vedya* is complemented with tutorials on data types (stacks, queues, binary search trees, red-black trees, priority queues, and 2-3-4 trees) and animations of algorithms that show the use of a data structure to solve a problem (evaluation of an expression in postfix form, the transformation of an infix expression to a postfix one, breath-first tree transversal, checking of palin-

```
fmod STACK{X :: TRIV} is
  sort Stack{X} .
  op error    : -> Stack{X} .
  op error    : -> X$Elt .
  op empty    : -> Stack{X} .
  op push     : X$Elt Stack{X} -> Stack{X} .
  op pop      : Stack{X} -> Stack{X} .
  op top      : Stack{X} -> X$Elt .
  op isEmpty? : Stack{X} -> Bool .
  var P : Stack{X} .
  var E : X$Elt .
  eq pop(empty) = error .
  eq pop(push(E,P)) = P .
  eq top(empty) = error .
  eq top(push(E,P)) = E .
  eq isEmpty?(empty) = true .
  eq isEmpty?(push(E,P)) = false .
endfm
```

```
fmod QUEUE{X :: TRIV} is
  sort Queue{X} .
  op error    : -> Queue{X} .
  op error    : -> X$Elt .
  op empty    : -> Queue{X} .
  op enqueue  : Queue{X} X$Elt -> Queue{X} .
  op dequeue  : Queue{X} -> Queue{X} .
  op first    : Queue{X} -> X$Elt .
  op isEmpty? : Queue{X} -> Bool .
  var C : Queue{X} .
  var E : X$Elt .
  eq  dequeue(empty) = error .
  ceq dequeue(enqueue(C,E)) = empty   if isEmpty?(C) .
  ceq dequeue(enqueue(C,E)) = enqueue(dequeue(C),E)
                                      if not isEmpty?(C) .
  eq  first(empty) = error .
  ceq first(enqueue(C,E)) = E if isEmpty?(C) .
  ceq first(enqueue(C,E)) = first(C)
                                      if not isEmpty?(C) .
  eq  isEmpty?(empty) = true .
  eq  isEmpty?(enqueue(C,E)) = false
endfm
```

Figure 2: Algebraic specifications of stacks and queues in *Maude*.

dromes). Moreover, there are animations on graphs: To obtain the minimum spanning tree using the Prim and Kruskal algorithms and to compute minimum paths using the Dijsktra algorithm.

Finally, *Vedya* offers the *Vedya-Test* tool to solve tests. This tool can be independently executed and allows teachers to create, modify or delete questions in a database, and to create tests from the database of questions. The student visualizes the tests, solves them and obtains the correct solutions. Questions are grouped by subject-matter on the database, but it is possible to mix questions about different data structures in the same test. The last version of this tool can be also found at `http://www.fdi.ucm.es/profesor/rdelvado/`.

3 EXECUTION OF ALGEBRAIC SPECIFICATIONS IN MAUDE

For the execution of algebraic specifications, the language *Maude* (Clavel and et al., 2006) based on *rewriting logic* has been used. *Maude* is a high-level language and high-performance system supporting both equational and rewriting computation for a wide range of applications. *Maude* and its formal tool environment can be used in three mutually reinforcing ways: as a declarative programming language, as an executable formal specification language, and as a formal verification system. Moreover, (Clavel and et al., 2006) describes the equational specification of the data structures included in the *Vedya* tool now in *Maude* syntax (stacks, queues, lists, binary and search trees, AVL and 2-3-4 trees). The language is available for *Linux* and *MacOS* at `http://maude.cs.uiuc.edu`, and there are also extensions for its execution in *Windows* at `http://moment.dsic.upv.es`.

The specifications can be executed in *Eclipse* (`http://www.eclipse.org/`) by means of special "plugins" developed in the Department of Information Systems and Computation of the Technical University of Valencia (DISC-UPV) and in the Computational Languages and Sciences Department of the University of Málaga (DLCC-UMA). This environment facilitates the student its usage by integrating the text editor with the execution commands of the system. On the left, there appear the developed projects; the central part shows the editor and the execution panel of the system is on it; on the inferior part, the control panel that shows the result of the action. On the right part, the user can open other windows that allow the definition of different system options and depuration.

The basic element of a specification in *Maude* is a "module". The language allows defining the functional modules used to define the data types; system modules used to define rewriting systems and modules focused on objects that allow the usage of syntax of classes, objects and messages. The functional modules for stacks and queues are showed with more detail in Figure 2.

The language allows importing other modules,

defining several data types, defining operations on the types and equations that define the behavior of those operations. The modules can be customized, using "theories" to such end in order to define the parameters and "views" to relate the formal parameter to the real parameter. The system has predefined the abstract data types most commonly used, as well as the most common theories and views:

```
view Int from TRIV to INT is
  sort Elt to Int .
endv

fmod STACK-INTEGERS is
  including STACK{vInt} .
endfm

fmod QUEUE-INTEGERS is
  protecting QUEUE{Int} .
endfm
```

As can be observed, the syntax is similar to the one used in several texts of algebraic specifications of data types (Weiss, 1998).

In order to execute the specification, the student enters the text in the editor; then, she/he executes the *Maude* system using the existing buttons in the console and enters the module. The system detects existing syntax errors and shows them on the console. Once the module shows no more errors, the student may reduce terms by using the equations of the module. To such end, the student may use the commands chart placed at the top of the screen or she/he may directly write the command in the editor and enter it into the system. For example, in order to obtain the first of a queue, we can reduce the term:

```
red first(enqueue(enqueue(empty, 5),4)).
```

This term must be reduced over the module of the queues using the integer number theory `INT`. In our example, this module is named: `QUEUE-INTEGERS`.

The possibility of reducing terms, in an automatic way, allows the students to carry out an initial test of their specifications by detecting many of the errors committed when defining the operations using equations.

Another greater advantage of executing the specifications is that the student comprehends the difference between the parameterized module and the instantiated module by being able to reduce terms on different modules. For example, a new module could be named `QUEUE-CHARACTERS` on which terms of type

```
red first(enqueue(enqueue(empty, 'a'),'c')).
```

can be easily reduced.

During the last academic courses, we have done practical classes on the data structures included in *Vedya*, which can be found at `http://www.fdi.ucm.es/profesor/rdelvado/`.

Other examples of data types, such as "medieval queues" or a "doctor's office" were also proposed (Weiss, 1998). In all of them, the aim was to define parameterized or instantiated data type with different theories. The practical classes are complemented with different terms that the student must reduce over some type of instantiated modules to prove the specification, as well as proposals to make little changes in some actions or erroneous definitions to detect them.

Taking into consideration that students from the second year were involved, just a few of the language facilities have been used. In superior courses where students have more knowledge on the subject, a richer language can be used (Clavel and et al., 2006) (e.g., many-sorted equational specifications, order-sorted equational specifications, equational attributes, and membership equational logic specifications).

4 FROM SPECIFICATION TO IMPLEMENTATION

The *Vedya* tool turns into a pedagogical instrument of high practical interest since it attempts to address the whole self-learning process of the main data structures, from the algebraic specification in *Maude* until the possible implementations in *Java*, within such a powerful and integrated environment as the one that has been described in the previous section by means of the *Eclipse* system.

The students have their first contact with the data structures that they are going to study by means of the usage of *Vedya*. For example, if their learning of data structures is focused on binary search trees or linear data structures, they will start learning the corresponding section of the tool, where they will be able to experiment, freely and on their own, each one of the actions offered by these structures (see Figure 1). In order to strengthen and evaluate this intuitive knowledge, the student has, in addition, the possibility of using the *Vedya-Test* tool.

Once the student has a clear idea of the informal behavior of the data structure, she/he may start working on the *Eclipse* system. The first step would be: to formally capture that intuitive knowledge she/he has obtained through the usage of *Vedya* in a specific algebraic specification written in *Maude* syntax. In order to facilitate this difficult step in the student's self-learning, she/he may use, interactively, the documentation that is included in the manual of the *Vedya* tool.

Once the specification (see Figure 2) is entered into the *Eclipse* system, the student can now go on executing little tests using *Maude*, in order to check whether it coincides with the intuitive and informal notion of data structure from which he initially departed in *Vedya*. Such experience would allow the student to reach the high level of abstraction that is necessary in computer supported education for each formal specification of a software component, always based on the intuitive and experimental knowledge.

Once the algebraic specification of the data structure is obtained, the next step would be to develop an implementation in an object-oriented programming language such as *Java*, by means of the facilities provided by the programming environment in *Eclipse*. This time, the student may use the algebraic specification that she/he has built, as if dealing with an authentic "instructions manual". The main advantage of our methodology is that the specification behaves now as a prototype of the data structures to be implemented, in a way that the student is able to find out the exact behavior for all those moments of doubt that may appear during the design process, even before she/he is able to compile the program. In order to be able to guide, in a more specific way, the step of specification to implementation, the student may make use again of the *Vedya* tool. This time, the student may access to the part that would correspond with the implementation of data structure that she/he is studying from the options menu (see Figure 1). From there, she/he may try different implementation possibilities based on arrays or pointers.

Once the student is familiar with the different implementations of the structure, she/he is finally ready to properly decide on a suitable representation in the *Java* language. The possibility of having understood and previously evaluated the different implementations by means of *Vedya* allows the student the possibility to acquire a clear knowledge of the *algorithmic cost* of the chosen implementation in *Java* for each specific operation of the data structure, so that this would also be a decisive criterion at the moment of designing its own implementations. In this part, the "*algorithmic schemes*" part of the *Vedya* tool plays an important role, since it allows the student to acquire a good programming methodology.

5 EVALUATION

In order to obtain a detailed evaluation of the usage of *Vedya* and *Maude* in our integrated system, we have proposed several tests related to the behavior, specification, implementation and application of the main data structures offered by the tool. We also collect students' opinion using *Vedya* in the "Data Structures" academic subject at the second year, and in the "Programming Methodology and Technology" subject at the third year, respectively.

The vast majority of our engineering and computer science students have taken an introductory programming course in the first academic year, typically in *Pascal*. Although the learning of the main algorithmic schemes and programming techniques is not a prerequisite to the subject of "Data Structures", many students choose to take it either prior to, or concurrent with, "Programming Methodology and Technology". As a result, although a pseudocode programming language is the assumed language for "Data Structures", many students have enough knowledge about *C++* or *Java* programming languages through the integrated programming laboratories of parallel academic courses and subjects.

Taking into account this profile, skills and background of our engineering and computer science students, we have proposed 8 tests in the *Virtual Campus* of the Complutense University of Madrid (`http://www.ucm.es/campusvirtual/CVUCM/`). The number of engineering students registered in the *Virtual Campus* was just over 320 distributed in three groups (130 in group A, 59 in group B, and 131 in group C). Figure 3 shows the number of the students who answered each of the tests in the corresponding group.

We observe that, from the second test on, the number of students becomes stable in a number lightly low to the number of students who access regularly to the *Virtual Campus*. These numbers, though seemingly high, are only between 23 % (75 students of 320) and 37 % (118 of 320) of registered students, which shows the high rate of students giving up in this topic from the beginning.

Figure 3 also shows the percentage of correct answers in the three groups: In general, it is high, which demonstrates the interest of the students who have taken part. In group B, the percentage is slightly higher than groups A and C; since 85 % of the students who have decided to complete the tests across the *Virtual Campus* of group B are not "new" students of this academic subject.

Figure 3 shows the percentage of students that did not attend the final exam, those who passed, and those who failed during the last six years. We observe that in the last academic courses, in which we have applied the *Vedya* tool, we have reduced by 14% the percentage of students giving up the course with respect to the previous course, and at the same time, we have increased by 12% the percentage of students

	Stacks 1	Stacks 2	Queues	Sequences	BST	AVL	RB	Heaps
Group A (130)	76.4%	82.5%	77.8%	65.6%	82.2%	84.9%	–	86.3%
Group B (59)	78.9%	83.6%	85.0%	63.6%	86.2%	87.7%	90.9%	90.2%
Group C (131)	76.2%	79.8%	73.5%	69.0%	83.5%	–	68.9%	86.8%

	2002/03	2003/04	2004/05	2005/06	2006/07	2007/08
Not attended	57.6%	45.3%	42.3%	64.7%	50.8%	40.2%
Passed	15.3%	22.2%	20.2%	18.2%	30.1%	42.6%
Failed	27.1%	32.5%	37.5%	17.1%	18.9%	17.2%

Figure 3: Students answering the tests and percentage of correct answers.

that passed the exam. The percentage of students that failed the exam increased by 2% due to the rise of students attending the exam. Comparing with previous courses (2003 to 2004) the percentage of students that passed has increased between 8% (with respect to the course 2003/04) and 15% (with respect to the course 2002/03).

6 CONCLUSIONS

In this paper, we have described the usage of an innovative educational environment for the interactive learning of data structure and algorithmic schemes by means of the visualization tool called *Vedya* and the specification language *Maude* with its programming environment in the *Eclipse* system.

In the last years, many papers on visualization of data structures and algorithms have been written. For example, a tool with a similar style is presented in (Chen and Sobh, 2001). Nevertheless, there is a lack in many of them of a graphic user interface of data structures and algorithms or they can only be executed in a few operative systems. In this sense, *Vedya* is something more than a simple tool for the execution of data structure as has been shown in Section 2 and (Segura et al., 2008).

The application of *Vedya* and *Maude* in a complete system as *Eclipse* allows the students the possibility of acquiring the capacity of implementing, correctly and properly, a data structure according to its formal algebraic specification, using in their design, the proper algorithmic schemes. As a consequence, it is possible to provide the students with a complete and professional methodology of software development that is very useful in the current teaching of Computer Science.

During the academic courses 2006/07 and 2007/08, we have carried out a detailed study in classroom on the application of this innovative educational environment by the *Virtual Campus* of the Complutense University of Madrid, in the "Data Structure" and "Programming Methodology and Technology" courses corresponding to the second and third academic courses of Computer Science.

As future work, we plan to design an *Intelligent Tutoring System* in order to guide the interactive *self-learning* process of data structures from the algebraic specification to the real implementation.

ACKNOWLEDGEMENTS

This work has been partially supported by the Spanish National Projects FAST-STAMP (TIN2008-06622-C03-01), MERIT-FORMS (TIN2005-09027-C03-03) and PROMESAS-CAM (S-0505/TIC/0407).

REFERENCES

Brassard, G. and Bratley, P. (1996). *Fundamentals of algorithms*. Prentice Hall.

Chen, T. and Sobh, T. (2001). A tool for data structure visualization and user-defined algorithm animation. In *Frontiers in Education Conference*.

Clavel, M. and et al. (2006). All about maude. A high performance logical framework. In *How to Specify, Program and Verify Systems in Rewriting Logic*, LNCS. Springer.

Cormen, T., Leiserson, C., Rivest, R., and Stein, C. (2001). *Introduction to Algorithms*. The MIT Press.

Neapolitan, R. and Naimpour, K. (2003). *Foundations of algorithms using C++ pseudocode*. Jones and Bartlett.

Segura, C., Pita, I., del Vado, R., Saiz, A. I., and Soler, P. (2008). Interactive Learning of Data structures and algorithmic schemes. In *ICCS*, volume 5101 of *LNCS*, pages 800–809. Springer.

Weiss, M. (1998). *Data Structures and Problem Solving Using Java*. Addison-Wesley.

An Intelligent Tutoring System for Interactive Learning of Data Structures*

Rafael del Vado Vírseda, Pablo Fernández, Salvador Muñoz, and Antonio Murillo

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid, Spain
rdelvado@sip.ucm.es,
{pablo.fdez.p,salva.ms,murillo925}@gmail.com

Abstract. The high level of abstraction necessary to teach *data structures* and *algorithmic schemes* has been more than a hindrance to students. In order to make a proper approach to this issue, we have developed and implemented during the last years, at the Computer Science Department of the Complutense University of Madrid, an innovative *intelligent tutoring system* for the interactive learning of data structures according to the new guidelines of the *European Higher Education Area*. In this paper, we present the main contributions to the design of this intelligent tutoring system. In the first place, we describe the tool called *Vedya* for the visualization of data structures and algorithmic schemes. In the second place, the *Maude* system to execute the algebraic specifications of abstract data types using the *Eclipse* system, by which it is possible to study from the more abstract level of a software specification up to its specific implementation in *Java*, thereby allowing the students a self-learning process. Finally, we describe the *Vedya Professor* module, designed to allow teachers to monitor the whole educational process of the students.

1 Introduction

The study of *data structures* and *algorithmic schemes* constitutes one of the essential aspects of the academic formation of every student in Computational Science. Nevertheless, the high level of abstraction necessary to teach these topics occasionally hinders its understanding to students. In order to make a proper approach to this issue, we have developed and implemented during the last years, at the Computer Science Department of the Complutense University of Madrid, an innovative interactive and visual learning framework according to the new guidelines of the *European Higher Education Area* and the teaching model focused on the student.

Our innovative approach is based on an *Intelligent Tutoring System* [8] (shortly, ITS), a computer system that provides direct customized instruction

* This work has been partially supported by the Spanish National Projects FAST-STAMP (TIN2008-06622-C03-01), MERIT-FORMS (TIN2005-09027-C03-03) and PROMESAS-CAM (S-0505/TIC/0407).

G. Allen et al. (Eds.): ICCS 2009, Part II, LNCS 5545, pp. 53–62, 2009.

and feedback to our students, a personal training assistant, and a range of tutoring techniques according to the student's response without the intervention of human beings. Thus, our ITS implements the underlying theory of *Abstract Data Types* [5] by doing and enable students to practice their skills by carrying out tasks within highly interactive learning environments. Based on these learner tools, the ITS tailors instructional strategies, providing explanations, hints, examples, demonstrations, and practice problems on data structures and algorithmic schemes [1,4,7]. The evaluation of our research on that systems indicates that students taught by our ITS generally learn faster and translate the learning into improved performance better that classroom-trained participants.

Despite the concept of ITS has been pursued for more that thirty years by researches in education, psychology, and artificial intelligence, few systems are in practical use today for the interactive learning in Computational Science. In order to remedy this lack, this paper describes the design of an ITS which guides the interactive learning of data structures from the *algebraic specification* to the real implementation [5]. The main components of this system are, on the one hand, the *Vedya tool* [9], a visualization tool by means of which, it is possible to provide the students with a complete learning system of both, the main data structures and the more relevant algorithmic schemes. On the other hand, the *Maude system* [3] for the execution of algebraic specifications of abstract data types using the language of formal specification provided by this system. And third, thanks to the development environment of *Eclipse*, we have obtained a fully complete system that is useful for the students as well as the professors, that allows to go from the most abstract level of data structures, provided by its algebraic specification in *Maude*, until its specific implementation in a modern programming language as happens with *Java*. All this learning process can be guided and overseen in a completely autonomous way by using the ITS presented in this paper, through which it is possible to make enquiries about the documentation related to each of the algebraic specifications, to distinguish between the behavior of the data structure and its different implementations through the use of different views or to browse information regarding the cost of the different implementations that have been proposed.

2 The Vedya Tool

Vedya is an integrated interactive environment for learning data structures and algorithmic schemes presented for the first time in [9]. It covers the most common data structures: Stacks, queues, binary search trees, AVL trees, priority queues, and sorted and hash tables. Moreover, it also provides other different types of abstract data types, like one for an implementation of a "doctor's office". Concerning the algorithmic schemes, it covers the most common resolution methods [1,4,7]: Divide and conquer, dynamic programming, backtracking, and branch and bound. All data structures and algorithmic schemes taught in the related study courses are thereby integrated in the same environment.

Currently, there are two versions of the *Vedya* tool. The first version contains all the data structures and algorithmic schemes mentioned above while the new

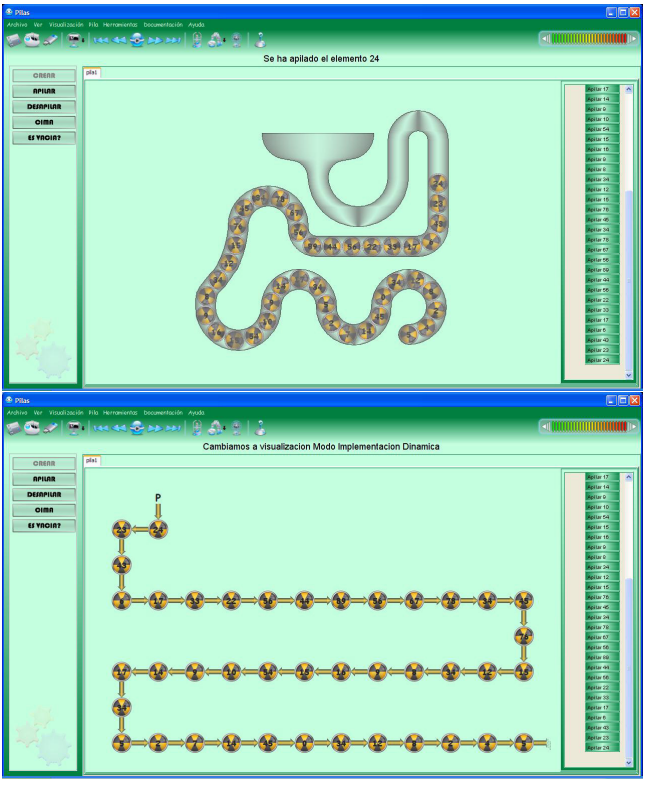


Fig. 1. Data structures in the Vedya tool

one offers a subset of them in a more attractive visual environment. This last version can be found at `http://www.fdi.ucm.es/profesor/rdelvado/Vedya.zip`.

There are several options to use this tool. The main one is the interactive execution, but it is also possible to create simulations that are automatically executed, to visualize tutorials and to solve tests within the same environment. It also integrates a set of animations that show how data structures are used to solve certain problems. For instance, Fig. 1 shows an example of the main windows for stacks. The central panel is used to represent the structure. On the left, there is a list of the actions that can be executed. Partial non-allowed actions are disabled. The right panel shows the visualization of the actions that have been already executed. There are two types of views: The one of data structure behavior to intuitively comprehend its operation, and one or several implementation views, either static or dynamic. On Fig. 1 we show the specific behavior view of a stack and a dynamic implementation based on pointers. The environment also provides documentation about algebraic specification, the implementation code and the cost of each implementation. Moreover, the current version of *Vedya* offers the *Vedya-Test* tool to solve tests (see Fig. 2). This tool can be independently executed and

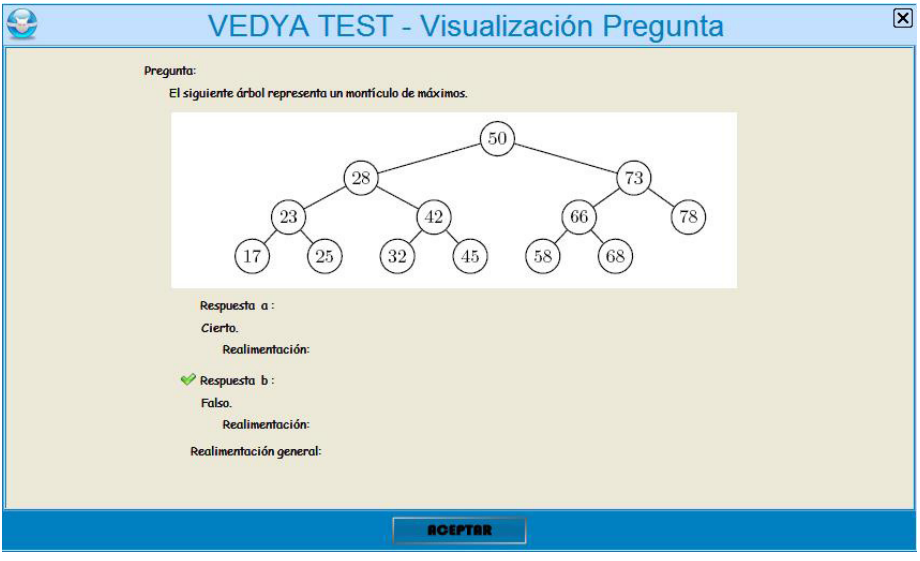


Fig. 2. The Vedya-Test tool for the student evaluation

allows teachers to create, modify or delete questions in a database. The student visualizes the tests, solves them and obtains the correct solutions. Questions are grouped by subject-matter on the database, but it is possible to mix questions about different data structures in the same test. The last version of this tool can be also found at `http://www.fdi.ucm.es/profesor/rdelvado/vedya-test.zip`.

3 Execution of Algebraic Specifications in Maude

For the execution of algebraic specifications in our ITS, we use the language *Maude* [3] based on *rewriting logic*. *Maude* is a high-level language and high-performance system supporting both equational and rewriting computation for a wide range of applications. *Maude* and its formal tool environment can be used in three mutually reinforcing ways: as a declarative programming language, as an executable formal specification language, and as a formal verification system. Moreover, [6] describes the equational specification of the data structures included in the *Vedya* tool now in *Maude* syntax (stacks, queues, lists, binary and search trees, AVL and 2-3-4 trees). The language is available for *Linux* and *Mac-OS* at `http://maude.cs.uiuc.edu`, and there are also extensions for its execution in *Windows* at `http://moment.dsic.upv.es`.

The algebraic specifications can be efficiently executed in the *Eclipse* system (`http://www.eclipse.org/`) by means of special "plugins" (which can be downloaded from `http://www.fdi.ucm.es/profesor/rdelvado/plugins-eclipse.zip`) developed in the Department of Information Systems and Computation of the Technical University of Valencia (DISC-UPV) and in the Computational Languages and Sciences Department of the University of Málaga (DLCC-UMA).

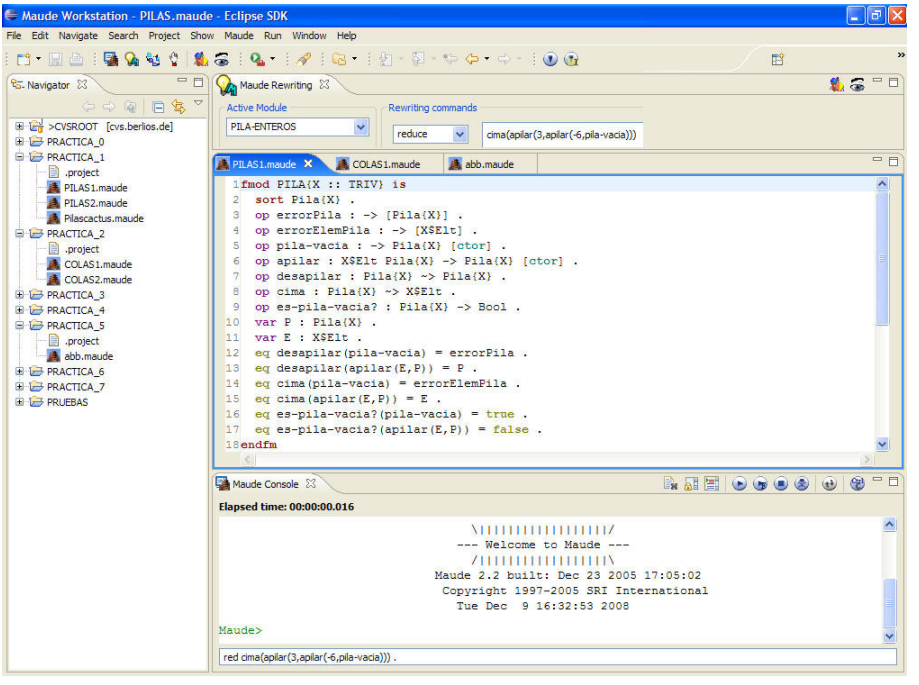


Fig. 3. Integration of Maude in Eclipse for the execution of algebraic specifications

This environment, as shown in Fig. 3, facilitates the student its usage by integrating the text editor with the execution commands of the system. On the left, there appear the developed projects; the central part shows the editor and the execution panel of the system is on it; on the inferior part, the control panel that shows the result of the action.

The basic element of a specification in *Maude* is a "module". The language allows defining the functional modules used to define data types. For example, the functional module for stacks used in Fig. 3 is showed with more detail in Fig. 4. The modules can be customized, using "theories" to such end in order to define the parameters and "views" to relate the formal parameter to the real parameter. The system has predefined the abstract data types most commonly used, as well as the most common theories and views:

```
view Int from TRIV to INT is
  sort Elt to Int .
endv
```

```
fmod STACK-INTEGERS is
  including STACK{vInt} .
endfm
```

In order to execute the specification, the student enters the text given in Fig. 4 in the editor of *Eclipse* (see Fig. 3); then, she/he executes the *Maude* system using

```
fmod STACK{X :: TRIV} is
     sort Stack{X} .
     op   error       :                -> Stack{X} .
     op   error       :                -> X$Elt .
     op   empty       :                -> Stack{X} .
     op   push        : X$Elt Stack{X} -> Stack{X} .
     op   pop         : Stack{X}       -> Stack{X} .
     op   top         : Stack{X}       -> X$Elt .
     op   isEmpty?    : Stack{X}       -> Bool .
     var  P           : Stack{X} .
     var  E           : X$Elt .
     eq   pop(empty)          = error .
     eq   pop(push(E,P))      = P .
     eq   top(empty)          = error .
     eq   top(push(E,P))      = E .
     eq   isEmpty?(empty)     = true .
     eq   isEmpty?(push(E,P)) = false .
endfm
```

Fig. 4. Algebraic specification of parametric stacks in Maude syntax

the existing buttons in the *Maude Console* of *Eclipse* and enters the module. The system detects existing syntax errors and shows them on the *Maude Console*. Once the module shows no more errors, the student may reduce terms by using the equations of the module. To such end, the student may use the commands chart placed at the top of the screen or she/he may directly write the command in the editor and enter it into the system. For example, in order to obtain the top of a stack, we can reduce the term: `red top(push(push(empty,5),4)).` This term must be reduced over the module of the stacks using the integer number theory `INT`. In our example, this module is named: `STACK-INTEGERS`.

The possibility of reducing terms, in an automatic way, allows the students to carry out an initial test of their specifications by detecting many of the errors committed when defining the operations using equations. Another greater advantage of executing the specifications is that the student comprehends the difference between the parameterized module and the instantiated module by being able to reduce terms on different modules. For example, a new module could be named `STACK-CHARACTERS` on which terms of type `red top(push(push(empty, 'a'),'c')).` can be easily reduced. Other examples of abstract data types, such as a "doctor's office" can be proposed [5]. In all of them, the aim was to define parameterized or instantiated data type with different theories. The practical classes are complemented with different terms that the student must reduce over some type of instantiated modules to prove the specification, as well as proposals to make little changes in some actions or erroneous definitions to detect them (see `http://www.fdi.ucm.es/profesor/rdelvado/codigo-maude.zip`).

Taking into consideration that students from the second year were involved, just a few of the language facilities have been used. In superior courses where students have more knowledge on the subject, a richer language can be used [3]

(e.g., many-sorted equational specifications, order-sorted equational specifications, equational attributes, and membership equational logic specifications).

4 An Intelligent Tutoring System for Data Structures

An *Intelligent Tutoring System* (shortly, ITS) for the *Vedya* tool turns into a pedagogical instrument of high practical interest since it attempts to address the whole self-learning process of the main data structures, from the algebraic specification written in *Maude* until the real implementation written in *Java*, within such a powerful and integrated environment as the *Eclipse* system described in the previous section.

The students have their first contact with the data structures that they are going to study by means of the usage of the ITS on *Vedya*. In order to control the student's self-learning process correctly, an online database has been built in on this tool. This means that now, the user has to be logged before using the *Vedya* tool, in order to oversee he/she evolution properly. For this purpose, the additional module called *Vedya Professor* (which can be obtained from `http://gpd.sip.ucm.es/rafav/`), has been designed to take full advantage of this feature. This tool allows teachers to monitoring the current progress of their students as a whole (see Fig. 5), according to the information stored in the database (tests realized, time spent on each test or most consulted documents

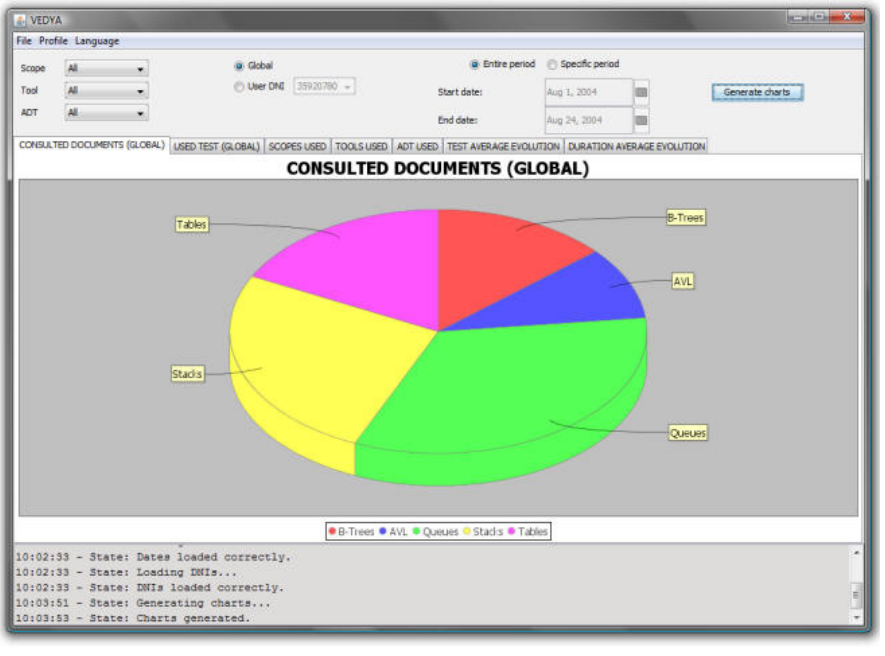


Fig. 5. Vedya Professor Module for the Vedya tool

on the help of *Vedya*). Moreover, this tool also allows seeing detailed information of each specific student, by selecting their identification number.

For example, if their learning of data structures is focused on linear data structures or binary search trees, the ITS would suggest that the student should start their learning process in the corresponding section of the tool, where they will be able to experiment, freely and on their own, each one of the actions offered by these structures (see Fig. 1). In order to strengthen and evaluate this intuitive knowledge, the student has, in addition, the useful possibility of using the *Vedya-Test* tool (see Fig. 2).

Once the student has a clear idea of the informal behavior of the data structure, the ITS may continue working on the *Eclipse* system. The first step would be to formally capture the intuitive knowledge that the student has obtained through the usage of *Vedya* in a specific algebraic specification written in *Maude* syntax. In order to facilitate this difficult step in the student's self-learning, the student may use, interactively, the documentation that is included in the manual of the *Vedya* tool.

Once the algebraic specification in *Maude* syntax (see Fig. 4) is entered into the *Eclipse* system (see Fig. 3), the student can now go on executing little tests using the *Maude Console*, in order to check whether it coincides with the intuitive and informal notion of data structure from which he/she initially departed in *Vedya*. Such experience would allow the student to reach the high level of abstraction that is necessary in computer supported education for each formal specification of a software component, always based on the intuitive and experimental knowledge.

Once the algebraic specification of the data structure is obtained, the next step performed by the ITS would be to develop an implementation in an object-oriented programming language such as *Java*, by means of the facilities provided by the programming environment in *Eclipse*. This time, the student may use the algebraic specification that she/he has built, as if dealing with an authentic "instructions manual". The main advantage of our methodology is that the specification behaves now as a prototype of the data structures to be implemented, in a way that the student is able to find out the exact behavior for all those moments of doubt that may appear during the design process, even before the student is able to compile their programs. In order to be able to guide, in a more specific way, the step from specification to implementation, the student may make use again of the *Vedya* tool. This time, the student may access to the part that would correspond with the implementation of the data structure that she/he is studying from the options menu (see Fig. 1). From there, he/she may try different implementation possibilities based on arrays or pointers.

Once the student is familiar with the different implementations of the structure, she/he is finally ready to properly decide on a suitable representation in the *Java* language. The possibility of having understood and previously evaluated the different implementations by means of the ITS allows the student the possibility to acquire a clear knowledge of the *algorithmic cost* of the chosen implementation in *Java* for each specific operation of the data structure, so that

this would also be a decisive criterion at the moment of designing its own implementations. In this part, the "*algorithmic schemes*" part of the *Vedya* tool plays an important role, since it allows the student to acquire a good programming methodology.

5 Evaluation

In order to obtain a detailed evaluation of the usage of the ITS on *Vedya* and *Maude* in our integrated *Eclipse* system, we have proposed several tests (see `http://www.fdi.ucm.es/profesor/rdelvado/Tests.zip`) related to the behavior, specification, implementation and application of the main data structures offered by the tool in the "Data Structures" academic subject at the second year, and in the "Programming Methodology and Technology" subject at the third year.

Taking into account this profile of our engineering and Computer Science students, we have proposed 8 tests in the *Virtual Campus* of the Complutense University of Madrid (`http://www.ucm.es/campusvirtual/CVUCM/`). The number of engineering students registered in this *Virtual Campus* was just over 122. Fig. 6 shows the number of the students who answered each of the tests. We observe that, from the third test on, the number of students becomes stable in a number lightly low to the number of students who access regularly to the *Virtual Campus*. These numbers, though seemingly high, are only between 30% and 40% of registered students, which shows the high rate of students giving up in this topic from the beginning. Fig. 6 also shows the percentage of correct answers: In general, it is high, which demonstrates the interest of the students who have taken part. Fig. 7 shows the percentage of students that did not attend the final exam, those who passed, and those who failed during the last six years. We observe that in the last academic courses, in which we have applied the ITS on *Vedya* tool, we have reduced by 10% the percentage of students giving up the course with respect to the previous course, and at the same time, we have increased by 12% the percentage of students that passed the exam. The percentage of students that failed the exam decreased by 2%. Comparing with previous

	Stacks 1	**Stacks 2**	**Queues**	**Sequences**	**BST**	**AVL**	**RB**	**Heaps**
Students	65	61	57	31	35	38	32	39
Answers	76.4%	82.5%	77.8%	65.6%	82.2%	84.9%	80.2%	86.3%

Fig. 6. Students answering the tests and percentage of correct answers

	2002/03	**2003/04**	**2004/05**	**2005/06**	**2006/07**	**2007/08**
Not attended	57.6%	45.3%	42.3%	64.7%	50.8%	40.2%
Passed	15.3%	22.2%	20.2%	18.2%	30.1%	42.6%
Failed	27.1%	32.5%	37.5%	17.1%	18.9%	17.2%

Fig. 7. Comparison of academic results with previous courses

courses (2003 to 2004) the percentage of students that passed has increased between 20% (with respect to the course 2003/04) and 25% (with respect to the course 2002/03).

6 Conclusions and Future Work

In the last years, many papers on visualization of data structures have been written (see, e.g., [2]). Nevertheless, there is a lack in many of them of a tutoring system which guides the interactive learning of data structures from the algebraic specification to the real implementation by means of appropriate user tools.

In this paper, we have described the design and usage of an innovative educational environment for the interactive learning of data structure by means of an *Intelligent Tutoring System* [8]. This system can be efficiently applied on the visualization tool *Vedya* [9] and the specification language *Maude* [3] with its programming environment in the *Eclipse* system, allowing the students the possibility of acquiring the capacity of implementing, correctly and properly, a data structure according to its formal algebraic specification, using in their design, the proper algorithmic schemes. As a consequence, it is possible to provide the students with a complete and professional methodology of software development that is very useful in the current teaching of Computer Science.

As future work, we plan to integrate, as part of the development of our intelligent tutoring system, an interface of the current "*Vedya-Maude*" system in *Eclipse*, in order to control and guide students along their self-learning process in a more autonomous way.

References

1. Brassard, G., Bratley, P.: Fundamentals of algorithms. Prentice Hall, Englewood Cliffs (1996)
2. Chen, T., Sobh, T.: A tool for data structure visualization and user-defined algorithm animation. In: Frontiers in Education Conference (2001)
3. Clavel, M.: All About Maude - A High-Performance Logical Framework. LNCS, vol. 4350. Springer, Heidelberg (2007)
4. Cormen, T., Leiserson, C., Rivest, R.: Introduction to Algorithms. The MIT Press, Cambridge (2001)
5. Horowitz, E., Sahni, S., Mehta., D.: Fundamentals of Data Structures in *C++*. W.H. Freeman & Co., New York (1995)
6. Martí, N., Palomino, M., Verdejo, A.: A tutorial on specifying data structures in Maude. Elsevier ENTCS 137(1), 105–132 (2005)
7. Neapolitan, R., Naimpour, K.: Foundations of algorithms using C++ pseudocode. Jones and Bartlett (2003)
8. Psotka, J., Mutter, S.A.: Intelligent Tutoring Systems: Lessons Learned. Lawrence Erlbaum Associates, Mahwah (1988)
9. Segura, C., Pita, I., del Vado, R., Saiz, A., Soler, P.: Interactive Learning of Data Structures and Algorithmic Schemes. In: Bubak, M., van Albada, G.D., Dongarra, J., Sloot, P.M.A. (eds.) ICCS 2008, Part I. LNCS, vol. 5101, pp. 800–809. Springer, Heidelberg (2008)