Entorno web de desarrollo para el aprendizaje de paradigmas de programación

Juan Ramón Pérez Pérez, Mª del Puerto Paule Ruiz, Martín González Rodríguez, Ramón González Suárez
HCI-RG. Departamento de Informática,
Universidad de Oviedo
C/ Calvo Sotelo S/N 33007 – Oviedo - España
e-mail: {jrpp, paule}@pinon.ccu.uniovi.es, martín@lsi.uniovi.es, i9440259@petra.euitio.uniovi.es

Resumen

En este artículo, se propone un entorno orientado a la enseñanza de lenguajes de programación pensado para entornos de enseñanza virtuales. Se considera que la enseñanza de lenguajes de programación ha de tener una parte en la que el alumno realice prácticas de programación, esto plantea dificultades extra en entornos de enseñanza virtuales convencionales. Para superar estas dificultades, planteamos un entorno que recibe la denominación de DWE (Development Web Environment), en el que se integra el compilador del lenguaje en un entorno Web. Junto con este compilador se incluyen distintos módulos de apoyo para ayudar al alumno mientras realiza los programas, utilizando para ello la experiencia de todos los usuarios del grupo, facilitando la comunicación alumno – profesor y alumno – alumno. Por último, se propone un módulo que permite al profesor realizar el seguimiento de los alumnos, conocer sus puntos fuertes y débiles y orientar la enseñanza para superar estos puntos débiles.

1. Introducción

El aprendizaje de un paradigma de programación y su aplicación en un lenguaje concreto no es una tarea trivial. Son muchos los autores que abordan las dificultades cognitivas que plantea el aprendizaje de un nuevo paradigma o el cambio entre paradigmas; incluso cuando se supone que el paradigma conlleva procesos mentales más "intuitivos". Este es el caso del cambio desde la programación estructurada a un paradigma orientado a objetos [10].

La asimilación y puesta en práctica de los conceptos que conlleva un paradigma de programación requiere un importante esfuerzo para lograr su asimilación por parte de alumno; por otra parte, para una correcta comprensión es imprescindible combinar el trabajo de conceptos teóricos con la puesta en práctica de estos utilizando un lenguaje de programación concreto. Esto ayuda al alumno a consolidar todo aquello aprendido en teoría y a elevar su nivel de conocimiento. Si nos fijamos en la taxonomía del conocimiento de Bloom, es en la práctica donde el alumno maneja más objetivos de aplicación y análisis y donde puede descubrir errores en el ámbito de comprensión.

Este proceso, para que sea efectivo, requiere un seguimiento y un apoyo continuo por parte del profesor. Esta tarea de tutorización es casi imprescindible en las fases iniciales de aprendizaje. Sin esta tutorización, un alumno tendría dificultades para aplicar los conocimientos obtenidos en teoría. Estos problemas van desde la configuración de determinadas características del compilador, hasta la interpretación de los errores de compilación y que pasos hay que llevar a cabo para corregir un determinado tipo de error. Además de esto, el profesor suele proporcionar patrones de programación que permiten evitar errores y relacionar conceptos de teoría que el alumno no ha comprendido correctamente y se reflejan en errores en los programas.

Todo esto conlleva, una constante interacción entre alumno – profesor que después de un tiempo le proporcionará un conocimiento adecuado del lenguaje de programación y de la herramienta de compilación, lo que permitirá al alumno aplicar determinadas técnicas para evitar que se

produzcan errores y para solucionarlos si se producen. En entornos de enseñanza presencial, esta interacción se produce de forma más o menos fluida. Incluso habría que tener en cuenta el factor de la comunicación alumno – alumno.

Este modelo de enseñanza de la programación es difícilmente transferible a los entornos de enseñanza virtual convencionales. Por definición, en estos entornos el alumno sólo puede establecer cauces de comunicación alumno – profesor y alumno – alumno a través del propio entorno. Normalmente estos entornos disponen de herramientas más o menos específicas; pero que se basan en el correo electrónico, los grupos de discusión y chats; con estos medios el alumno tiene dificultades para transmitir los problemas de programación con los que se encuentra y, por tanto, también será más difícil para el profesor ayudarle a resolveros.

2. Herramientas y problemas para la enseñanza virtual

Actualmente existe una gran proliferación de entornos de enseñanza virtual que utilizan Internet como medio para transmitir los contenidos de una determinada asignatura, complementando la exposición teórica con ejercicios que suelen ser del tipo de preguntas de respuesta cerrada.

Existen en la actualidad varias herramientas que permiten la construcción y la impartición de este tipo de cursos [4]. En concreto, nuestra universidad esta utilizando WebCT [3] para el desarrollo de este tipo de cursos dentro de un proyecto denominado AulaNet [2] dentro un campus virtual integrado por las universidades del grupo G7.

Estos entornos permiten al alumno navegar, siguiendo ciertas pautas, a través de los temas teóricos, hacer evaluaciones del alumno y ofrecen herramientas que permiten la comunicación entre alumno y profesor o entre varios alumnos basadas en el correo electrónico, los chats o los grupos de discusión.

Estos entornos no son suficientes cuando se están realizando prácticas de lenguajes de programación. Al alumno le resulta difícil expresar las dificultades que tiene con un fragmento de código; muchas veces los alumnos se refieren al error; pero el profesor no tiene el contexto en el que se produce para poder dar una respuesta; con frecuencia los alumnos incluyen el propio fragmento de código en el mensaje, y el profesor, para poder responder, tiene que examinar el fragmento detenidamente o incluso compilarlo, lo cual no siempre es posible. Esto también provoca una pérdida de tiempo. Todo esto conlleva ambigüedades y falta de precisión que dificultan la solución de problemas.

En la actualidad hay herramientas que permiten la corrección automática de prácticas de programación [5,6] que permiten ahorrar tiempo al profesor a la hora de evaluar los conocimientos del alumno e incluso hay planteamientos [9] para que estas herramientas puedan dar información al alumno del tipo de errores que comete y que conceptos tiene que repasar para no cometerlos; pero no existen herramientas que permitan asesorar al alumno mientras está realizando las prácticas de programación y antes de que el alumno obtenga el resultado final de la evaluación.

Lo que planteamos en este artículo es un entorno de desarrollo basado en Internet para la enseñanza virtual del paradigma orientado a objetos utilizando el lenguaje Java, que denominamos DWE (Development Web Environment). Este entorno se realiza en relación con el proyecto IDEFIX (Integrated Development Environment based on Internet and eXtensible technologies) [1,7, 8].

3. Descripción del entorno

DWE utiliza una interfaz web, por tanto el alumno sólo necesita un navegador para acceder a la aplicación, y no deberá instalar nada en su equipo local. Esta característica proporciona dos importantes ventajas que son imprescindibles para integrarse en un entorno virtual de enseñanza en el que el mantenimiento y la configuración son siempre a distancia y, por tanto, costosa de realizar. La primera ventaja es que tendremos un mantenimiento centralizado de la aplicación, cuando se modifica cualquier componente de la aplicación sólo es necesario modificarlo en el servidor y no es necesario modificar nada en el ordenador del usuario final. Como segunda ventaja, el usuario final no tiene que realizar ningún proceso de instalación local del sistema, evitando posibles problemas de compatibilidad y / o configuración del ordenador del usuario final.

Para poder utilizar esta aplicación el alumno debe estar registrado previamente y debe introducir su nombre de usuario en el sistema y su contraseña. Esto permite a la aplicación identificar al usuario y monitorizar las operaciones que realiza sobre el sistema.

La interfaz de usuario, en el desarrollo actual de la aplicación, es una interfaz general para todos los usuarios pero podría personalizarse fácilmente para cada uno de los usuarios mediante un módulo de configuración.

DWE permite a los usuarios escribir, compilar y corregir errores de sus programas escritos en java. Para ello, como se puede ver en la figura 1, la interfaz de compilación del entorno dispone de una barra de botones desde donde se acceden a las principales funciones del gestor de ficheros con el que el cada usuario podrá introducir, desde su máquina local, nuevos ficheros fuente para compilar y, descargar otros de su directorio privado en el servidor, además de poder crear nuevos ficheros. El entorno también dispone de un área de edición de código.

Cuando el usuario compila un fichero, el sistema captura los errores que se producen, informa al usuario de los errores de su programa y guarda la información sobre los errores en una base de datos.

4. Implementación del entorno

DWE está basado en tecnologías Java del servidor: servlets y JSP. Como se dijo anteriormente, es un requisito básico el no tener que instalar nada en los ordenadores de los usuarios finales; además con tecnologías Java se puede utilizar tanto servidores con sistema operativo UNIX como servidores con sistema operativo Windows e incluso combinar los dos tipos, lo cual da una flexibilidad extra al sistema respecto a la infraestructura del sistema.

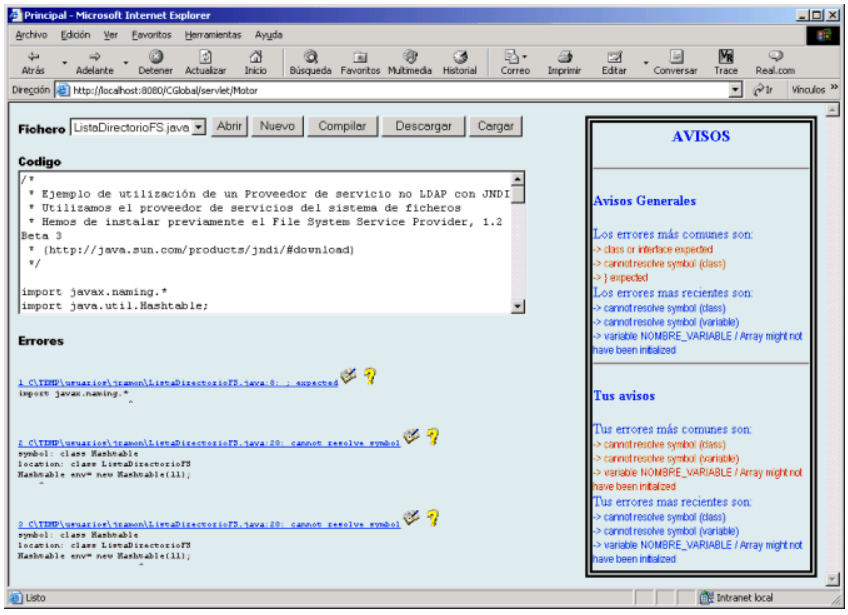

Figura 1. Interfaz del entorno DWE

Se trata de una combinación de páginas JSP, servlets y JavaBeans que separa la presentación de los demás módulos siguiendo el patrón *Modelo Vista Controlador*.

Un servlet se encarga de identificar y autenticar al usuario y crear una sesión que permite a la aplicación hacer un seguimiento de todas las acciones que lleva a cabo este usuario.

El servlet controlador se encarga de identificar el evento que genera el usuario al pulsar uno de los botones de opciones e invocar al JavaBean correspondiente que es que realiza la acción. Uno de estos JavaBeans se encargará de la gestión de la compilación y la captura de los errores. Además, existen clases asociadas a este JavaBean que almacenan estos errores en la base de datos utilizando la API JDBC de Java.

5. Arquitectura de DWE

DWE está formado por distintos módulos funcionales especializados (figura 2).

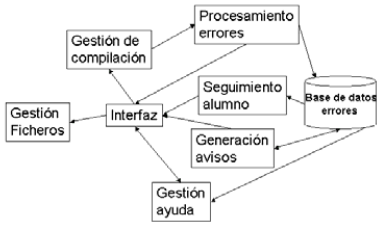

Figura 2. Módulos que constituyen DWE

Mód. Gestión de ficheros, se encarga de gestionar ficheros fuente y compilados, su organización jerárquica y posibilita el intercambio entre el servidor y el ordenador local del usuario.

Mód. Gestión de compilación, gestiona la llamada al compilador con las opciones adecuadas pasando los ficheros fuente para realizar la compilación.

Mód. Procesamiento de errores, recoge la salida del compilador, comprueba si hay errores y si los hay los almacena en la base de datos, los clasifica y actualiza los contadores correspondientes.

Gestión de ayuda, es el módulo encargado de proporcionar ayuda al usuario ante los errores. Esto módulo también se encarga de construir la ayuda con las aportaciones de profesores y alumnos.

Generación de avisos, este módulo se encarga de generar avisos para el usuario, a partir de la información almacenada en la base de datos, que le indica cuales son sus errores más frecuentes y los últimos que ha cometido.

Seguimiento del alumno, permite a los profesores de la asignatura hacer un seguimiento de los alumnos y de grupos de alumnos, que acciones han realizado los usuarios, cuantas compilaciones han hecho, los errores más frecuentes de un alumno o un grupo de alumnos, etc.

Módulo Interfaz, valida e identifica a los usuarios y les permite interactuar con los distintos módulos.

5.1. Análisis de los tipos de errores

En primer lugar, el módulo de gestión de errores debe clasificar los errores en su tipo correspondiente, cada tipo tiene una información asociada que puede ser distinta para distintos tipos. Supongamos que al compilar un fichero llamado HolaMundo.java javac genere el siguiente error:

```
HolaMundo.java:7:cannot    resolve
symbol
symbol  : class Usuario
location: class HolaMundo
Usuario user = new Usuario ();
^
```

Y ahora supongamos que otro usuario compila un fichero llamado Helloword.java y se genera el siguiente error:

```
Helloword.java:8:cannot    resolve
symbol
symbol  : class Admin
location: class Helloword
Admin ad = new    Admin ();
^
```

En este ejemplo se ve claramente que se trata del mismo tipo de error, tiene el mismo tipo de información con distintos valores. Para clasificar cada error concreto en un tipo, el módulo de procesamiento de errores se apoya en el conjunto

de palabras comunes a todos los errores de ese tipo, esto es lo que llamamos *patrón del error*. En este caso, por ejemplo, el patrón del error es, en la primera línea: "cannot resolve symbol" y en la segunda línea: "class". En la base de datos de la aplicación, se han definido patrones para distintos tipos de errores, de manera que ante cada error de compilación se realiza una búsqueda para descubrir que patrón cumple, cuando se encuentra una coincidencia se puede clasificar el error dentro de ese tipo y almacenarlo para su posterior uso en la generación de estadísticas y avisos que sirvan de ayuda al usuario.

Por otra parte, diferentes tipos de errores no sólo tienen un patrón diferente, sino que la información asociada que proporcionan sigue diferentes formatos. Se ha realizado una recopilación de los distintos tipos de errores de compilación java que se pueden producir con el fin de conocer y almacenar el formato de los mismos, para que el módulo de gestión de errores pueda extraer de forma automática los datos que se quieren almacenar en la base de datos.

5.2. Entorno colaborativo de ayuda a los errores

El módulo de gestión de ayuda, además de mostrar al usuario los errores de compilación que se produzcan, clasifica los errores, como se ha visto en el apartado anterior y almacena una serie de datos relacionados con los errores generados en la compilación de los ficheros. Se pretende que el sistema pueda proporcionar la información que tendría un usuario avanzado para solucionar el error y que en un entorno presencial podría proporcionar el profesor.

Así, cada tipo de error tendrá asociada información de posibles causas y acciones que debe tomar el alumno para corregirlo. Esta información puede ser creada inicialmente por el profesor; pero además el alumno puede añadir sus propios comentarios sobre el error: causas que producen el error, cómo puede solucionarse,... esta información completa y complementa a la creada inicialmente por el profesor. Cuando aparezca un error de un tipo determinado, el alumno podrá ver la información proporcionada por el profesor y la que él mismo a anotado anteriormente.

Queremos seguir desarrollando este módulo para que todos los usuarios pudieran ver las anotaciones de cualquier otro usuario y que hubiera un sistema para seleccionar los comentarios más "valiosos", los que a otros usuarios les han parecido más eficaces para comprender y solucionar el error. De esta forma estaríamos aprovechando la experiencia de determinados usuarios para los demás usuarios con lo que el aprendizaje debería ser más rápido.

5.3. Gestión de avisos y seguimiento

Se pretende que el sistema sea preventivo, es decir, que no sólo ayude a la corrección de errores en un programa ya realizado, sino que permita al usuario, mientras esté escribiendo el código, darse cuenta de cuales son los errores más frecuentes para no volver a cometerlos.

Para realizar esto, el sistema almacena en la base de datos contadores de los distintos tipos de errores y el momento en el que se generó el último, de esta forma puede obtener los errores que el usuario comete con mayor frecuencia y los últimos errores que ha cometido. Con esta información, el sistema muestra una serie de avisos personalizados mientras el usuario está escribiendo el código (figura 3).

Además, el sistema aprovecha la trayectoria de todo el grupo de usuarios que está utilizando el sistema, analiza el conjunto de todos errores del sistema y proporciona al usuario avisos sobre los errores más recientes y más frecuentes globalmente para todos los usuarios.

5.4. Base de datos de errores

La aplicación almacena toda la información sobre los errores en una base de datos relacional. Una de las tablas almacena los datos de cada uno de los errores de compilación que se producen. Los datos que se introducen en los campos de esta tabla se obtienen de la salida producida por el javac al compilar un fichero. En primer lugar, se clasifica el error en uno de los tipos mediante la búsqueda del patrón correspondiente y se separa en tokens siguiendo el formato correspondiente al tipo de error. Además, una vez clasificado el error, existen tablas en las que se contabilizan los errores de cada tipo por cada usuario y a nivel de todo el grupo y se guardan los errores más recientes también para estas dos categorías.

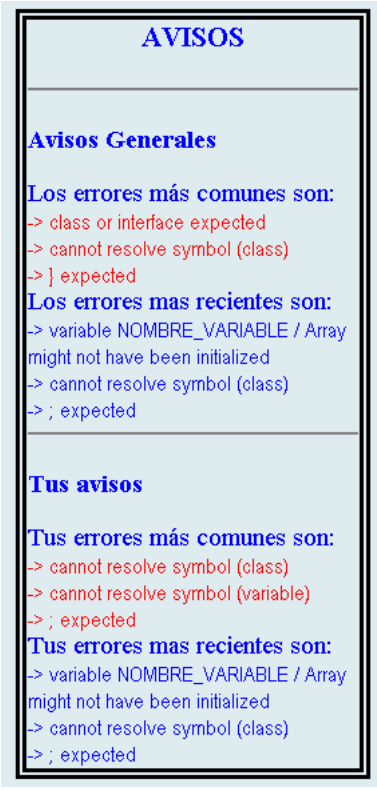

Figura 3. Avisos sobre los errores

Por otra parte para realizar el proceso de clasificación tenemos una tabla que contiene el patrón y el formato de cada tipo de error.

Por último, la base de datos tiene una tabla en la que recoge la información de ayuda que aportan los profesores y los alumnos sobre cada uno de los tipos de errores.

6. Conclusiones

Se ha diseñado e implementado un entorno para la enseñanza de lenguajes de programación que aporta varias novedades sobre otros entornos ya existentes:

- Compilador integrado en un entorno web, lo que evita la instalación y configuración por parte del usuario del entorno y permite la centralización del mantenimiento.
- Aportación de ayuda al usuario mientras está escribiendo código, no sólo cuando surgen errores. Mediante el módulo de gestión de avisos que obtiene métricas de frecuencia y últimos errores, tanto para el propio usuario como para el grupo en global. Esto ayuda a prevenir la aparición de nuevos errores.
- El usuario puede aprovechar la experiencia de otros en la solución de errores mediante el módulo de gestión de ayuda, que utiliza métodos de trabajo colaborativo para recoger información de todos los usuarios.
- El profesor puede hacer un seguimiento de los alumnos individualmente consultando los errores que comete el alumno. También puede obtener información de los errores que comete todo el grupo con mayor frecuencia y reforzar la explicación de los temas relacionados con los conceptos que están fallando, para evitar esos errores.

Actualmente se ha realizado una primera implementación del entorno y se han hecho pruebas de uso con un reducido grupo de alumnos. Se está planificando la implantación de este entorno para la realización de prácticas en asignaturas presenciales y con la experiencia adquirida implantarlo en un entorno de enseñanza virtual.

Referencias

[1] Integrated Development Environment based on Internet and eXtensible technologies: IDEFIX Proyect. http://www.di.uniovi.es/aplt/idefix , 2002.

[2] Página de inicio del proyecto Aulanet http://aulanet.uniovi.es/, 2002.

[3] Página de inicio de la herramienta WebCT. http://www.webct.com, 2002.

[4] Clear, T., Haataja, A., Meyer, J., Suhonen, J., and Varden, S. A. Dimensions of distance learning for computer science education. SIGCSE Bulletin 33, 2 (2001). ITiCSE 2000 Working Group Reports.

[5] Dawson-Howe, K.M. Automatic submission of programming assignments. SIGCSE Bulletin 27, 4 (1995) 51-53.

[6] Huizinga, Dorota M. Identifying Topics for Intructional Improvement Through On-line Tracking of Programming Assignments. SIGCSE Bulletin 33, 3 (Sep. 2001) 129-132. 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education.

[7] Labra Gayo, J.E., Morales Gil, J. M., Turrrado C., R. Plataforma de enseñanza de lenguajes de programación a través de Internet: Proyecto IDEFIX. Actas de JENUI 2002, 13-20.

[8] Labra Gayo, J.E. , Morales Gil, Jose M., Fernández Álvarez, A. M., and Sgastegui Chigne, H. A Generic e-Learning Multiparadigm Programming Language System: IDEFIX Project. ACM Technical Symposium on Computer Science Education (SIGCSE 2003), Reno, Nevada, USA, Feb. 2003.

[9] Marco Galindo, Ma Jesús; Prieto Blázquez, Joseph Necesidades específicas para la docencias de programación en un entorno virtual. Actas JENUI 2002, 5-12.

[10] Whitelaw, M. and Weckert, J. The Humannness of Object-Oriented Programming. I Cognitive Technology Conference. Hong Kong (1995).