Actas de las XX JENUI. Oviedo, 9-11 de julio 2014
ISBN: 978-84-697-0774-6
Páginas: 371-378

Aprendizaje de la programación guiado por los errores de compilación

Carlos Fernandez-Medina
Departamento de Informática
Universidad de Oviedo
Oviedo
carlosfernandezmedina@gmail.com

Juan Ramón Perez-Perez
Departamento de Informática
Universidad de Oviedo
Oviedo
jrpp@uniovi.es

Mª del Puerto Paule-Ruiz
Departamento de Informática
Universidad de Oviedo
España
paule@uniovi.es

Víctor Alvarez-Garcia
Department of Computer Science
K.U. Leuven, Belgium
Belgica
Victor.Alvarez@cs.kuleuven.be

Resumen

Los errores de programación y en concreto los de tiempo de compilación siempre han sido un área de interés en el campo de la enseñanza de la programación. Esto lo demuestran los distintos artículos que aparecen periódicamente en congresos de docencia de la informática, tanto nacionales como internacionales. Estos estudios se basan en la relación entre errores de programación y carencias conceptuales o malas prácticas de programación, y orientar a los docentes en las áreas y conceptos de programación que se tendrían que reforzar en la enseñanza de la programación. Sin embargo, estos estudios se realizan siempre fuera del proceso de aprendizaje de los estudiantes, con lo cual sólo se extraen conclusiones generales para posteriores cursos y sin poder proporcionar una realimentación real e individualizada a los alumnos.

Hemos creado un sistema, denominado COLMENA que se integra con el entorno de desarrollo integrado para recopilar información sobre el análisis estático, errores de compilación y warnings, que los estudiantes generan durante un ejercicio de programación. Además, esta información sobre errores se visualiza a los usuarios clasificada por familias de errores, sesiones de prácticas y alumnos. Permitiendo al profesor realizar un seguimiento completamente personalizado, para un grupo o un alumno concreto. Por otra parte, facilita la documentación de los distintos errores, estableciendo una relación de estos con los conceptos de programación y buenas prácticas relacionadas. Además, permite a los alumnos darse cuenta de los errores que cometen con más frecuencia y reforzar el conocimiento de estos.

A través de esta herramienta, por tanto, se consigue un flujo de intercambio de información que puede ayudar a profesores y alumnos en la mejora del proceso enseñanza-aprendizaje, y así perfeccionar explícitamente detalles sobre pautas de programación o conceptos relativos a la misma que antes no se apreciaban o conocer la evolución de los errores de un grupo a lo largo de una asignatura.

Abstract

Studies on errors in programming activities have always been considered relevant among scientific communities. Thus, different approaches are explored every year in national and international conferences about programming teaching and learning. All of these studies deal with the relationship between programming errors and bad habits in programming, in order to generate guidance for teachers in concepts where their students fail. However, these kinds of studies are conducted outside the students' natural programming process, which prevents teachers from receiving a feedback report in real-time.

We have created an eclipse plug-in, called COLMENA, whose purpose is to recover compilation errors and warnings generated by the students during programming lessons. Moreover, this information is displayed in a specific portal where students can retrieve information about the errors, their solutions and different reports about individual and collective information, such as specific groups or sessions inside a subject. Students have the possibility to learn about the solution to their most common errors in the system or their top-10 error list.

Through COLMENA, we aim to assist teachers and students with programming tips and ideas that generate less errors and improve the teacher-learning process. This kind of information, previously discarded, allows us to gain a new perspective about the problems that users have in a specific practical lesson or in a whole subject.

Palabras clave

Programación, Java, COLMENA, Entorno de Desarrollo Integrado, Lenguaje de programación, Asistencia en la programación, Minería de Datos.

1. Introducción

En los últimos años, el aprendizaje de la programación ha evolucionado, pasando de ser una disciplina muy teórica y con niveles bajos de práctica, a tener un amplio espectro de disciplinas donde se aplican diferentes conocimientos prácticos. Hoy en día, en las facultades y escuelas se desarrollan proyectos software de distintos tamaños, de manera individual, teniendo el estudiante a su disposición un ordenador propio, cosas que, unos pocos años atrás sería mucho más difícil de considerar.

Es por esto, que la enseñanza de la programación adquiere también mayor protagonismo en las clases y sesiones prácticas, pudiendo aplicar prácticamente los conceptos que se ven en el apartado teórico en proyectos realistas. Una evolución similar ha ocurrido con las herramientas disponibles para desarrollar estos proyectos. Los alumnos tienen a su disposición entornos de desarrollo cada vez más completos, que facilitan el uso de los lenguajes de programación, liberando al estudiante de tareas adicionales, como la utilización de distintas herramientas, la definición de rutas para librerías adicionales, etc. que pueden ocasionar que la atención se distraiga del tema o puntos claves de la sesión.

A pesar de disponer de una mayor asistencia hacia los alumnos, este tipo de herramientas no alcanzan todavía, desde nuestro punto de vista, el nivel necesario de implicación en el proceso de desarrollo como para apoyar activamente a los alumnos en el aprendizaje conceptual de un lenguaje. En ocasiones, facilitan al usuario algunos mensajes o señales de ayuda, que el usuario debe interpretar para corregir el error. Nuestra experiencia como docentes nos ha brindado casos en los que los alumnos, tras cometer errores sencillos, no analizan estos mensajes, y para tratar de solucionarlo deciden cambiar, arbitrariamente, partes de su código fuente. Esta actitud no cambia hasta que el profesor (o un compañero) le insiste a detenerse, analizar este tipo de mensajes para conocer su error y, por tanto, modificar la parte correspondiente, momento en que el alumno se da cuenta de la simplicidad de su error.

Desde el punto de vista del profesor, en determinadas circunstancias de limitación de tiempo o gran número de alumnos, es difícil tener un conocimiento real y continuo de sus alumno y su comportamiento a la hora de programar. Por norma general, se explican casos sobre los errores más comunes y conocidos, de manera genérica, pero sin tener una constancia si el grupo concreto al que se dirige responde a este patrón de comportamiento. Los docentes disponen de su experiencia para saber cuáles serán los problemas típicos de los alumnos durante la sesión práctica, pero es prácticamente imposible que tenga una certeza sobre los conceptos concretos en los que puede fallar cada alumno individualmente. Disponer de una herramienta que le ayude a conocer qué errores están ocurriendo en un momento concreto puede suponerle una ayuda sustanciosa en el proceso de tutorizar a sus alumnos.

A través del proyecto COLMENA se pretende proporcionar una doble asistencia, complementando tanto a profesores como a estudiantes en el proceso de enseñanza aprendizaje. Así el profesor podrá conocer en tiempo real el estado de los alumnos en las asignaturas que imparte y con ello los fallos que está cometiendo dicho alumno o incluso el grupo completo. Esta nueva información le proporciona una perspectiva con un conocimiento profundo de los errores que se generan. Para los alumnos, COLMENA proporciona un conocimiento extenso y en tiempo real sobre los errores que se producen. Esta información, antes inexistente e inaccesible, puede generar mejoras sustanciales en los malos hábitos de programación de los estudiantes.

Este artículo se estructura de la siguiente forma: en la sección 2 se hará un estudio preliminar de las investigaciones y desarrollos en líneas similares a la del presente trabajo. La sección 3 describe el modelo propuesto y la implementación realizada para cada una de las capas descritas. En la sección 4 se describe toda la componente visual del COLMENA, en forma de portal web, siendo descrito un caso de uso de la misma en el apartado 5. Se finaliza con las conclusiones del presente trabajo y las futuras líneas de estudio se detallan en el apartado 6.

2. Los errores como guía en el aprendizaje de la programación

Son varios los autores que coinciden en la importancia en la detección y tratamiento de los errores de programación y también en el escaso tratamiento en los libros de texto para el aprendizaje de la programación. El profesor Barr [1] ya planteaba un estudio basado en los grandes tipos de errores que cometen los alumnos principiantes mediante encuestas. El profesor Luján-Mora [2] plantea en JENUI la importancia de la detección y eliminación de errores. La profesora Pillay [3] realiza una clasificación de errores independiente del lenguaje realizando una distribución para un grupo de programadores principiantes. En todos estos estudios, la recogida de datos se realiza ad hoc y mediante métodos manuales para un grupo de estudio determinado y se realizan generalizaciones para aplicarlas a otros grupos.

La captura de los errores descritos anteriormente no se debe llevar a cabo al final del proceso de realización de un programa, sino que, para que sea más efectivo debería ser una monitorización a lo largo del proceso de programación. En este sentido varios autores [4, 5] plantean el uso de herramientas durante el proceso de programación y cómo esto puede influir en los resultados del aprendizaje de la programación.

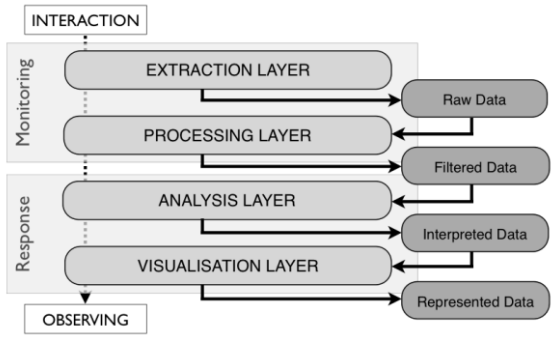

Figura 1: Modelo de capas de COLMENA

En nuestro grupo de investigación llevamos varios años desarrollando un sistema que permita la monitorización de los errores de programación [6, 7]. El sistema está pensado para trabajar con lenguaje Java, actualmente uno de los más utilizados tanto en la enseñanza de la programación. En artículos anteriores [6] se describió el modelo para la captura, almacenamiento y análisis de errores. En este artículo, planteamos la forma de uso del entorno para que el profesor pueda realizar un seguimiento continuo de los errores concretos cometidos por sus estudiantes, clasificarlos en diferentes familias y descubrir la frecuencia real con la que se cometen para así tomar medidas correctivas. Esto permitirá detectar los errores más importantes para el grupo o para un alumno concreto y extraer los conceptos relacionados para poder reforzar su aprendizaje.

3. Cómo se capturan, procesan y visualizan los errores

El proyecto COLMENA se apoya en un modelo teórico basado en capas. Este modelo, planteado por el grupo de investigación en trabajos anteriores [6], es una evolución de otros trabajos del grupo en otras líneas relacionadas con la adaptación y la evaluación de contenidos [8].

COLMENA se concibe como un modelo de 4 capas, que cubren desde la interacción del usuario, hasta la observación del usuario sobre la máquina. Tal y como ilustra la Figura 1, el usuario pasa de ser un sujeto que no es consciente de su proceso de aprendizaje, a ser un sujeto que a través del sistema se da cuenta de las acciones que ha realizado en el aprendizaje y cuáles han sido sus carencias; para ello el sistema monitoriza sus acciones y le proporciona un feedback visual. Las cuatro capas del modelo cubren esta interacción, conectándose entre sí de tal manera que la salida de una de las capas constituye la entrada de la siguiente.

3.1. Capa de Extracción (Extraction layer)

Esta capa es la encargada de capturar las acciones que los usuarios están desarrollando en un momento concreto del proceso de aprendizaje de la programación. En nuestro caso particular, el proceso de escritura, compilación y prueba de las prácticas de programación y los errores y warnings que se generan en él. Para que esta extracción sea posible, es necesario recuperar información del entorno de desarrollo integrado sobre los errores generados por el mismo. Estos mensajes se producirán cada vez que el usuario modifica o compila una clase, momento en que el entorno devuelve esta información.

Para llevar a cabo esta tarea se ha desarrollado un plug-in para el entorno de desarrollo integrado Eclipse. La selección de este entorno no ha sido trivial, sino que se ha elegido considerando que es un entorno común en muchas universidades, así como por su gran capacidad de extensión y ampliación (en forma de plug-ins). El plug-in COLMENA desempeña la función principal de conectar con el núcleo (core) de Eclipse, para obtener los mensajes de la "vista de problemas" (Figura 2), donde se exponen todos los warnings y errores producidos en una compilación determinada.

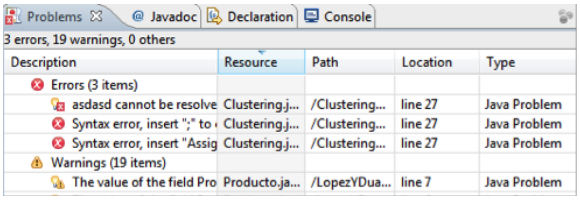

Figura 2: Vista de problemas del Eclipse

3.2. Capa de Procesado (Processing layer)

Esta capa es la responsable del filtrado y procesado de los mensajes recolectados en la capa de extracción (Figura 3). El plug-in en esta capa se ocupa de analizar las redundancias de errores sobre una sección de código donde el usuario no está focalizado. Este proceso permite eliminar redundancias de errores, lo cual proporciona una mejor perspectiva de la cantidad real de errores cometidos, evitando así el posible ruido que pudiera generar un error que el estudiante no está atendiendo y no se preocupa por corregir.

Además de eliminar estas redundancias, en esta capa se realiza una clasificación previa que organiza los errores y warnings obtenidos en familias. Estas familias, dependen de los elementos de código a los

que se refiere el error, tales como: métodos, tipos, constructores, etc. Esta clasificación se basa en estudios previos y taxonomías internas del entorno de desarrollo [9].

- Internal: Relativas al funcionamiento interno de Java, como las asignaciones de variables, creaciones de objetos, etc.
- Type: Errores relacionados con tipos
- Method: Errores relacionados con métodos
- Constructor: Errores en los constructores
- Javadoc: Errores producidos en el Javadoc
- Import: Errores relacionados con los paquetes o clases importadas
- Syntax: Errores de sintaxis del lenguaje
- Field: Errores relacionados con variables y campos

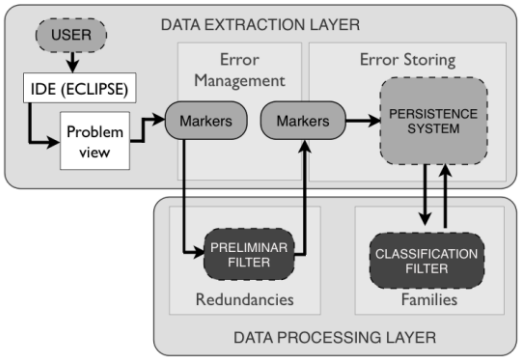

Figura 3: Operaciones de las capas de extracción y procesado

3.3. Capa de Análisis (Analysis layer)

La capa de análisis busca determinar diferentes enlaces entre los errores y warnings previamente clasificados y las asignaturas, cursos y correspondencias existentes entre los diferentes estudiantes, profesores y miembros partícipes del sistema. Así mismo es en esta capa donde suceden las operaciones de agrupación de la información en diferentes formatos, tales como agrupaciones por fecha, cursos, tipos de error, generando muchas interpretaciones diferentes de los mismos datos.

Otra operación que se realiza en esta capa es la caracterización de un error, tarea que consiste en complementar la información recogida sobre un error (mensaje, tipo, fecha) y enriquecerla con datos que facilitarán su comprensión, tales como explicaciones sobre sus causas, referencias a capítulos de libros donde se expliquen sus causas, o incluso ejemplos de código fuente donde se ilustre el buen o mal uso del código fuente para evitar o producir ese error.

Tras el análisis de la información, se realiza la persistencia de los errores a un servidor en un formato pre-configurado como ficheros texto o sobre una base de datos. Esta operación se ejecuta de forma automática cada vez que un usuario guarda (tras haber modificado) o compila el proyecto.

3.4. Capa de Visualización (Visualisation layer)

La capa de visualización juega un papel fundamental, ya que será la encargada de mostrar los resultados de las capas anteriores de una manera amigable, intuitiva, y complementaria al aprendizaje a los usuarios del sistema.

El profesor dispondrá de toda la información relativa a sus asignaturas, como estudiantes de la misma, errores producidos, resúmenes de errores en sesiones, etc. El alumno también dispondrá de una vista con un panel de control y seguimiento de su propia actividad en las asignaturas en las cuales ha participado, con un desglose controlado de sus errores y actividad en comparación con su grupo. Además, desde los dos tipos de usuario se dispondrá del catálogo de errores y warnings donde podrán conocer (o aportar en el caso de docentes) información complementaria al error.

A nivel de implementación, se ha desarrollado una aplicación web que, bajo autenticación de usuario, muestra la información adecuada al rol del usuario. Esta plataforma COLMENA supone el mayor nivel de abstracción sobre el modelo y constituye la herramienta para los usuarios finales, que podrán consultar la información a través de sus diferentes vistas. Algunas de ellas están disponibles para el docente y el alumno, y otras, solamente para los docentes, ya que aporta información global sobre el grupo que no resulta útil para el alumno. A continuación describiremos, en los siguientes apartados, las distintas vistas con más detalle.

4. Visualización de los errores en las prácticas

4.1. Vista de asignatura/sesión (orientada a profesores)

La vista de asignatura o sesión permite conocer al profesor los datos generales sobre la asignatura y cada una de las sesiones prácticas que la componen (Figura 4). Ya sea la asignatura completa o una única sesión, la información es mostrada de manera similar, en forma de gráficos y tablas fáciles de interpretar en un vistazo rápido. Los gráficos y tablas disponibles en esta vista son los siguientes:

- Gráfico de barras con las frecuencias de familias: En este diagrama el docente puede ver para

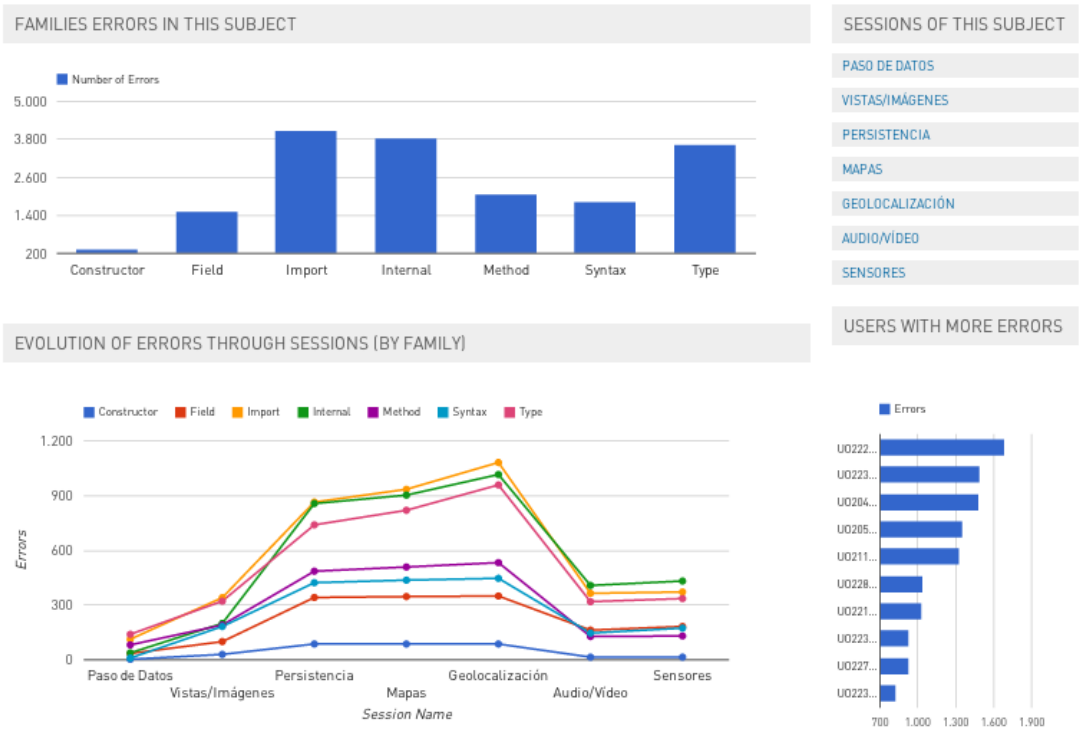

Figura 4: Vista de asignatura/sesión, incluyendo comparación de los errores según su familia y por sesión de prácticas. En el panel de la derecha también aparece el ranking de usuarios con más errores.

qué familias de error se aparecieron mayor cantidad de errores. Con esta información el profesor puede inferir qué tipos de error son los más críticos en la asignatura, pudiendo aplicar diferentes soluciones para tratar de reducir la frecuencia de errores de la familia en cuestión.

- Gráfico de líneas con la evolución de errores en la asignatura: En este gráfico, el profesor puede conocer a simple vista que tipos y cantidad de errores han generado sus alumnos para cada sesión. Esta información es difícil de conocer para el docente, donde solamente disponía de su intuición para pensar en qué aspectos se podría haber fallado en cada sesión,
- Tablas de usuarios de la asignatura: En esta tabla se pone a disposición del docente la información específica para cada alumno de los errores que ha cometido para cada una de las familias. De esta forma, puede ver de manera individual la cantidad global de errores que ha generado un estudiante en el proceso de aprendizaje.
- Tabla de frecuencias de error: En esta tabla se dan a conocer, para cada tipo concreto de error el porcentaje que abarca sobre el total generado y la frecuencia absoluta de los mismos en la asignatura. Interpretando estos datos el docente puede detectar qué errores han sido generados más veces con el objeto de tratar de reducirlos.
- Ranking comparativo de usuarios con más errores: permite conocer qué estudiantes generan más errores en la sesión o la asignatura, enlazando directamente con su perfil.

Además de ver esta información en forma global para la asignatura o individual para una sesión concreta, el profesor puede seleccionar una serie de sesiones sobre las que se calcularán los datos considerando las sesiones seleccionadas. Esta funcionalidad resulta muy útil en los casos en que un docente realiza un conjunto de prácticas en la que se ponen en juego objetivos de aprendizaje similares, pudiendo así hacer una comparativa y análisis de solamente estas sesiones.

4.2. Vista de perfil de usuario (orientada al alumno o al seguimiento individual por parte del profesor)

En esta vista se dispone toda la información relativa a un usuario individual. El objetivo fundamental de esta vista es que los docentes conozcan de manera directa lo que su estudiante hecho en su asignatura y que los estudiantes conozcan su evolución y los errores que han generado, teniendo la media del grupo como referencia comparativa. En esta vista se disponen los siguientes gráficos:

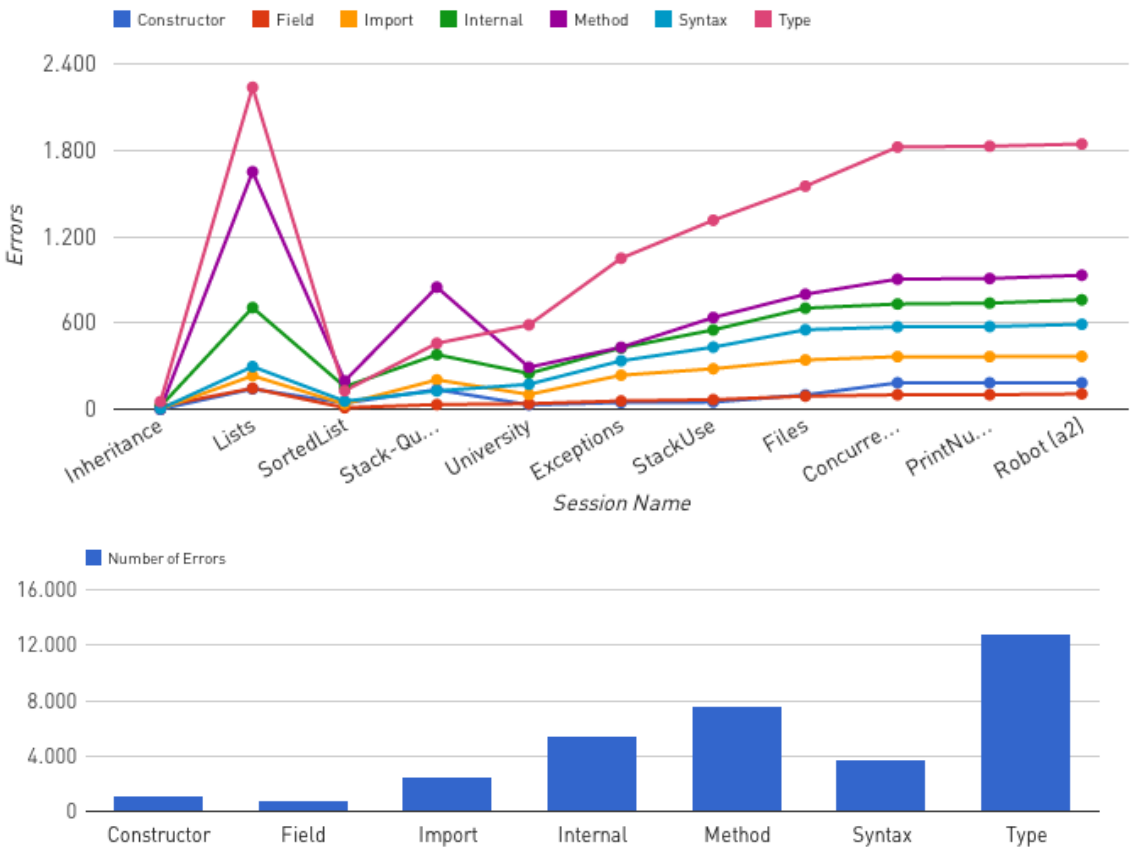

Figura 5: Gráficas de errores por sesión de prácticas y clasificados por familias para el caso de la asignatura de metodología de la programación.

- Gráfico de líneas de evolución del estudiante: Se muestra un gráfico con la evolución de los errores de una familia concreta a lo largo de las sesiones, en comparación con la media del curso. Esta información muestra una referencia de la situación del estudiante respecto a sus compañeros en la asignatura
- Gráfico de sectores resumen de sesión: para cada una de las sesiones en las que ha participado el estudiante, se muestra un gráfico con el porcentaje de errores cometidos para cada familia así como un ranking con el top 5 de errores cometidos por el estudiante en comparación con el ranking del top 5 del grupo. A través de este tipo de paneles el estudiante puede conocer nuevamente su situación o semejanza respecto al grupo global.
- Tabla con el ranking general con los diez errores más frecuentes del estudiante.
- Tabla con la cantidad general de errores por familia: la cantidad total de errores que ese estudiante ha generado para cada una de las familias de error consideradas.

4.3. Vista de errores

Además de las vistas generales de estudiantes y asignatura/sesión, es posible consultar los errores que los usuarios han cometido hasta el momento, en forma de *catálogo de errores*. Actualmente, dadas las herramientas que los alumnos y profesores tienen a su disposición, esta información no se aprovecha de ninguna manera para el aprendizaje, ni por parte del profesor en el seguimiento, ni por parte de los alumnos que no profundizan sobre qué tipos de errores existen, su significado y las causas de los mismos. A través de esta vista es posible conocer los errores que son más frecuentes en cuanto a repetición, número de estudiantes distintos que lo han generado o las asignaturas donde ha aparecido.

La herramienta permite establecer la relación entre cada error, la familia a la que pertenece, los conceptos que el alumno tiene que aprender para evitar este error. Además, donde es posible, se especifican los temas, ejemplos y ejercicios para que el alumno pueda reforzar el aprendizaje y realizar prácticas para evitar estos errores en el futuro.

Para realizar esta parte de explicación e ilustración de los distintos errores nos basamos en estructuras en portales de internet que están teniendo relativo éxito, cuya finalidad es aprender un lenguaje específico o proponer soluciones a una problemática o ejemplos de uso de funciones, relacionadas con el error o en un contexto más amplio. En el caso de COLMENA, son los profesores de la asignatura, los que con su experiencia académica y docente van introduciendo información sobre el error.

5. Caso de estudio

El modelo presentado anteriormente junto con la implementación descrita se ha aplicado en un contexto real de enseñanza del lenguaje de programación Java. El plugin se ha instalado para los alumnos de primer curso del grado de Ingeniería Informática del Software en la Universidad de Oviedo. Concretamente, se ha estudiado el colectivo de alumnos que participaron en la asignatura de Metodología de Programación, se basa en el paradigma de la orientación a objetos y se trabajan aspectos fundamentales de la programación como la comprensión de la encapsulación, el trabajo con colecciones, la persistencia, la genericidad o la herencia. Esta asignatura, fundamental para los alumnos en el desarrollo de sus competencias de programación, se imparte en el segundo semestre.

La monitorización se ha realizado a lo largo de 10 semanas (se han excluido las primeras sesiones), y han participado hasta 113 alumnos diferentes en diferentes grupos de prácticas. Un detalle importante a considerar es que las dos últimas semanas de trabajo comprenden la última sesión.

La gráfica de sesiones del panel de la asignatura (Figura 5) refleja la evolución de la cantidad de errores de cada familia. Vemos que en las primeras sesiones se observa un gran crecimiento de los errores, especialmente los relacionados con la declaración y utilización de tipos y clases, variables y métodos. Concretamente esto ocurre en una sesión relacionada con colecciones (List), donde se trabaja en profundidad el uso de variables y de llamadas a métodos (movimiento de los índices, avance y retroceso a través de la colección, etc). Analizando el resto de sesiones, se observa un aumento progresivo de los errores, acorde al aumento de complejidad de las prácticas.

Dejando a parte la evolución de la cantidad de errores y estudiando el volumen de errores de las familias se detectan muchos más errores, en cualquiera de las sesiones, de la familia “Type”, relacionada con tipos de datos. Esta diferencia es significativa ya que los errores de esta familia están en general relacionados con tipificación de variables, instanciación de objetos, etc. De acuerdo al tipo de asignatura que es “metodología de programación”, es aceptable que la mayor cantidad de errores se produzca en esta familia. La segunda familia en la que más cantidad de errores aparecen, es la relacionada con los métodos, seguida de los errores relacionados con asignación de variables (*Internal*).

La herramienta también proporciona información sobre los errores concretos, detectados en la asignatura y por cada alumno, a través de la vista del catálogo de errores. Algunos de los errores capturados no tienen mucha semántica y son debidos fundamentalmente a erratas a la hora de teclear y a que el alumno no sigue el orden correcto a la hora de definir y luego usar clases y métodos. Sin embargo, hay otros errores como: *Unhandled exception type %*, *The method % of type % must override or implement a supertype method* o *The type % must implement the inherited abstract method %* que pueden ayudar al profesor a darse cuenta de determinadas carencias de los alumnos en el manejo de conceptos clave con la gestión de excepciones, la redefinición de métodos en la herencia o la implementación de métodos abstractos al heredar de una clase, respectivamente. Este tipo de errores que tienen una incidencia relativa, pueden ser objeto de un seguimiento continuo para reforzar determinados conceptos de la programación.

Como se puede observar a partir del caso de estudio descrito, el entorno Colmena proporciona información útil tanto para el profesor en el seguimiento de grupos y alumnos concretos, como para el trabajo autónomo de cada alumno con las prácticas de programación.

6. Conclusiones y trabajo futuro

En el presente trabajo se propone utilizar los errores de programación como guía en la detección de lagunas conceptuales y dificultades a la hora de programar por parte de los alumnos. Se ha desarrollado un sistema software capaz de extraer, procesar, capturar y visualizar los errores que los alumnos cometen en tiempo de compilación, compuesta por un *plugin* integrado en el entorno de desarrollo Eclipse y un entorno web para visualización de los datos capturados. Para ello se disponen de distintos paneles gráficos que permiten al docente extraer conclusiones sobre dónde y en relación a qué se están produciendo los distintos errores que se clasifican en familias. Este entorno también está disponible para los alumnos de tal forma que sean conscientes de cuáles son los errores que cometen con más frecuencia, así como consultar información en relación a cada error.

Este modelo y su implementación han sido utilizados en una asignatura de primer curso en el Grado de Ingeniería Informática del Software en la Universidad de Oviedo. En esta asignatura se han detectado errores frecuentes relacionadas con la tipología de las

prácticas y se le proporciona al docente un punto de vista diferente, permitiéndole conocer y monitorizar en detalle las sesiones de la asignatura. Gracias a COLMENA, tanto profesores como alumnos disponen de una mayor información a su alcance para conocer en qué fallan y porqué.

El *plug-in* se encuentra actualmente disponible, como software libre para ser utilizado en cualquier universidad. Está disponible a través de la web del proyecto (http://www.pulso.uniovi.es/colmena/). Adicionalmente se proporciona un usuario de prueba (invitado/colmena) para el portal COLMENA, para que pueda ser objeto de estudio.

El trabajo desarrollado posee diferentes líneas en las que se pretende seguir trabajando:

- Estandarización: Un punto clave es estandarizar los tipos de errores, ampliando y especializando el grupo de familias a un espectro mayor y más jerarquizado que el empleado actualmente, que consiste en la taxonomía empleada por Eclipse para clasificar internamente los errores. Utilizar una especificación el formato de escritura de la información capturada.
- Extensión y automatización del *plug-in*: Integrar un mayor número de variables en la capa de extracción podría generar un mayor conocimiento del estado de los alumnos. Conocer datos como el número de líneas generadas, la relación de compilaciones exitosas o el número de variables puede refinar la interpretación de los errores generados sobre el código fuente.
- Ampliación de asignaturas y grupos sobre los que integrar la herramienta: Conseguir una mayor diversidad de grupos aporta el conocimiento necesario como para interpretar los patrones de error característicos de cada tipo de asignatura, y qué temas generan más errores de un tipo específico.
- Integración de feedback: Conseguir que la plataforma COLMENA, a nivel tecnológico, se integre con los usuarios de una manera más transparente y sin necesidad de depender de que el usuario acceda a consultar la información. Integrar parte de la plataforma con el entorno de desarrollo Eclipse puede ser una buena solución para tener constancia de los datos adicionales que COLMENA proporciona en el mismo momento que se está desarrollando el software.

7. Agradecimientos

Queremos agradecer la colaboración de los docentes y alumnos de la asignatura de Metodología de la Programación, sin los cuales este trabajo no habría sido posible.

Referencias

[1] Barr, M., Holden, S., Phillips, D. and Greening, T. 1999. An Exploration of Novice Programming Errors in an Object-oriented Environment. *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education* (New York, NY, USA, 1999), 42–46.

[2] Luján-Mora, S. Un enfoque para la enseñanza de la depuración de errores en las asignaturas de programación. *Actas de las IX Jornadas de Enseñanza universitaria de la Informática (Jenui 2003)* (Cádiz, España), 473-480.

[3] Pillay, N. 2009. A Study of Object-oriented Design Errors Made by Novice Programmers. *Proceedings of the 2009 Annual Conference of the Southern African Computer Lecturers' Association* (New York, NY, USA, 2009), 101–104.

[4] Spacco, J., Fossati, D., Stamper, J. and Rivers, K. 2013. Towards Improving Programming Habits to Create Better Computer Science Course Outcomes. *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education* (New York, NY, USA, 2013), 243–248.

[5] Matsuzawa, Y., Okada, K. and Sakai, S. 2013. Programming Process Visualizer: A Proposal of the Tool for Students to Observe Their Programming Process. *Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education* (New York, NY, USA, 2013), 46–51.

[6] Fernandez-Medina, C., Pérez-Pérez, J.R., Álvarez-García, V.M. and Paule-Ruiz, M. del P. 2013. Assistance in computer programming learning using educational data mining and learning analytics. *Proceedings of the 18th ACM conference on Innovation and technology in computer science education* (New York, NY, USA, 2013), 237–242.

[7] Fernández-Medina, C., Pérez-Pérez, J.R., Paule-Ruiz, M.P. and Álvarez García, V.M. 2011. COLMENA : Collaborative knowledge and user classification environment based on programming experience. *Proceedings of the VIII Multidisciplinary Symposium on Design and Evaluation of Digital Content for Education* (Ciudad Real, Spain, Jul. 2011), 50–58.

[8] Álvarez García, V.M., del Puerto Paule Ruiz, M. and Pérez Pérez, J.R. 2010. Voice interactive classroom, a service-oriented software architecture for speech-enabled learning. *Journal of Network and Computer Applications*. 33, 5 (Sep. 2010), 603–610.

[9] Ben-Ari, M.M. 2007. *Compile and Runtime Errors in Java*.