Capítulo 10. Clasificación de usuarios basada en la detección de errores

En este capítulo abordamos la realización de un estudio inédito hasta el momento, en el que analizamos errores de usuarios teniendo en cuenta su nivel de experiencia. Para ello utilizamos los avisos recogidos por de los proyectos desarrollados por los alumnos de distintos cursos de la titulación de Ingeniería Técnica en Informática tanto de Gestión como de Sistemas de Escuela Universitaria de Ingeniería Técnica en Informática[9] y alumnos del segundo ciclo de Ingeniería en Informática de la Escuela Politécnica Superior[10] de la Universidad de Oviedo. El objetivo del estudio es obtener datos que permitirán construir las bases para un modelo de usuario basado en el análisis de errores y realizar una clasificación de usuarios. De esta forma, el sistema podría permitir una adaptación a cada usuario dependiendo de su perfil de experiencia [González 2002][González 2004].

10.1 Precedentes en este tipo de estudios

Como se describía en el 3.6 *Gestores de prácticas avanzados* (Capítulo 3) existen actualmente gestores de prácticas que permiten recoger automáticamente proyectos realizados por estudiantes y someterlos a distintos test: compilarlos, realizar pruebas de ejecución, comprobar la coherencia de la documentación, etc. Utilizando estas facilidades algunos autores han realizado estudios en la línea en la que planteamos el nuestro.

Por ejemplo Huizinga [Huizinga 2001] realizan un estudio sobre los problemas encontrados en una práctica de una asignatura de programación. Define una serie de objetivos de cada proyecto relacionados jerárquicamente con los objetivos de la materia a enseñar y realiza una evaluación del cumplimiento de estos objetivos. Para ello utiliza una herramienta de entrega automática de prácticas. La herramienta compila y ejecuta una serie de test predefinidos y genera un informe de registro recogiendo la información sobre los errores de compilación y ejecución. Elabora una serie de datos estadísticos sobre el comportamiento del conjunto de los alumnos al presentar una práctica. Huizinga plantea una asociación, basada en la intuición, entre cada dato estadístico y objetivo de aprendizaje. Por ejemplo: número medio de reenvíos por estudiante puede significar una especificación confusa del proyecto, objetivos conceptuales del proyecto poco dominados; porcentaje de envíos finales con errores de compilación refleja deficiencias en la sintaxis del lenguaje;

[9] La Web de la Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo, donde se puede encontrar el plan de estudios completo y el programa de las asignaturas citadas es: http://www.euitio.uniovi.es

[10] La Web de la Escuela Politécnica Superior, donde se puede encontrar el plan de estudios de las asignaturas de segundo ciclo: http://www.epsig.uniovi.es/

porcentaje de envíos finales con errores de ejecución puede reflejar una incapacidad para manipular las estructuras de datos.

Satratzemi [Satratzemi 2001] con ayuda de su herramienta AnimPascal (ver apartado RC.3.2.2) estudia como los alumnos van resolviendo los errores que surgen en una práctica planteada y pensada de antemano por los profesores. AnimPascal permite guardar las acciones de los estudiantes. Se consideran estas acciones como pasos en el proceso de solución del problema. Haciendo un seguimiento del camino recorrido por el estudiante para resolver cada problema, los profesores pueden descubrir "lagunas" de conocimiento y problemas de aprendizaje de sus estudiantes. Estudiando estos problemas de aprendizaje podrán ayudar a cada alumno particular.

La pretensión de nuestro trabajo es, como en los anteriores, estudiar los errores con los que se encuentran los estudiantes cuando realizan un proyecto software. Sin embargo, nuestro trabajo tiene varias diferencias con los descritos anteriormente:

- En nuestro trabajo no nos limitaremos a una práctica de una única asignatura, sino que trabajaremos con cuatro asignaturas más el proyecto fin de carrera nos interesa no sólo detectar errores puntuales en una práctica sino descubrir tendencias y la evolución de los errores según la tendencia.
- Utilizaremos nuestro sistema SICODE para detectar errores en el código fuente. El sistema, además del compilador, utiliza herramientas de análisis estático. Esto permitirá realizar un análisis más profundo que sólo con el compilador.
- En este experimento no estudiamos una evolución de los errores en el desarrollo de un proyecto sino que analizamos siempre proyectos ya terminados.
- Para descubrir problemas de aprendizaje nos basamos exclusivamente en errores en el código fuente, no tenemos en cuenta cuestiones relacionadas con la entrega, ni con la ejecución del proyecto.
- Los proyectos analizados son todos relativamente grandes. No se hace un análisis detallado sobre posibles errores lógicos en una práctica concreta; sino que pretendemos realizar un análisis estático de todos los errores que pueden cometer los desarrolladores.

10.2 Descripción del trabajo

El trabajo realizado se ha dividido en las siguientes etapas:

1. Recopilación de proyectos de distintos niveles (cursos) a partir de los proyectos entregados por los alumnos en cada asignatura.
2. Análisis de los mismos utilizando nuestro sistema SICODE y más concretamente el Sistema de Análisis de Errores en Programas (PBA) descrito en el Capítulo 7 *Sistema de análisis de errores de programas: PBA*.
3. Comparación de los datos de los proyectos en distintas etapas de aprendizaje de la programación, en concreto los tres primeros cursos de la titulación de ingeniería técnica en informática y en cuarto del segundo ciclo de Ingeniería Informática.
4. Caracterización de los errores en relación a los niveles de experiencia de los desarrolladores.

10.3 Elección de las muestras

Para realizar el estudio hemos utilizado los proyectos correspondientes a cuatro asignaturas: tres de la Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo y una del segundo ciclo de Ingeniería en Informática de la Universidad de Oviedo. Además del proyecto fin de carrera.

Las asignaturas seleccionadas son: Metodología de la Programación, Estructura de Datos y de la Información, Bases de Datos y Procesadores de Lenguaje. En todas se trabaja sobre conceptos del diseño y la programación orientada a objeto aplicados a diferentes aspectos de la programación.

La situación en el plan de estudios de las asignaturas es la descrita en la Tabla 11.

Tabla 11. Grupos de proyectos que se utilizarán en el experimento y descripción de las asignaturas a las que pertenecen

Asignatura	Curso	Duración	Tipo	Año de las muestras
Metodología de la programación	1º	2º cuatrimestre	Troncal	Curso 2002-2003
Estructura de datos y de la información	2º	Anual	Troncal	Curso 2003-2004
				Curso 2004-2005
Bases de Datos	3º	Anual	Troncal	Curso 2004-2005
Procesadores de Lenguaje	4º	Anual	Troncal	Curso 2004-2005
Proyecto Fin de Carrera	5º		Obliga	Distintos años.

10.3.1 Descripción de las prácticas de las asignaturas

En las tres primeras asignaturas: Metodología de la Programación, Estructuras de Datos y de la Información y Bases de Datos se realizan prácticas en las que los alumnos utilizan, de forma obligatoria, el lenguaje de programación Java para implementar el problema pedido; además de realizar el diseño, expresarlo mediante diagramas UML y realizar una documentación adjunta a la práctica. Para programar en lenguaje Java los alumnos utilizan un entorno de desarrollo integrado: el entorno JBuilder de Borland (ver descripción en el apartado 3.3.2 JBuilder).

Las prácticas están divididas en las tres asignaturas en dos módulos, con la diferencia que:

- Prácticas en Metodología de la Programación (primer curso). El tiempo de realización de cada módulo es la mitad del cuatrimestre y los dos módulos son independientes.
- Prácticas en Estructura de Datos y de la Información (segundo curso). El desarrollo de cada uno de los módulos ocupa un cuatrimestre completo y el segundo módulo es continuación del primero, suele conllevar cambios y ampliaciones; pero se reutiliza gran parte del código del primero.
- Prácticas en Bases de Datos (tercer curso). Como en los casos anteriores también se desarrollan dos módulos; pero el primero se desarrolla exclusivamente con SQL así que sólo consideramos el segundo módulo en el que se accede a la base de datos desde Java.

En la asignatura de Procesadores de Lenguaje (cuarto curso) los alumnos eligen el lenguaje de implementación entre C++ y Java y el entorno de desarrollo también es a elección del alumno, los alumnos que trabajan en Java suelen utilizar los entornos de desarrollo JBuilder

o Eclipse. Para nuestro estudio se han seleccionado exclusivamente las prácticas implementadas en Java.

Los alumnos realizan las prácticas de forma individual; excepto en el módulo de Bases de Datos. En clase de prácticas se trabaja sobre la comprensión del problema y sus requisitos y un diseño preliminar del mismo. Cada alumno trabaja personalmente en el diseño detallado y la implementación, buscando la mejor forma de realizar esta implementación y resolviendo los problemas que vayan surgiendo. El profesor resuelve dudas concretas en esta parte y orienta en el diseño detallado del proyecto.

10.3.2 Evaluación de las prácticas en las asignaturas

Para los proyectos de todas las asignaturas los criterios de evaluación incluyen el correcto funcionamiento de la aplicación cumpliendo los requisitos establecidos; pero también otras facetas como:

- Realización del diseño de la práctica utilizando diagramas UML, fundamentalmente de clases y de interacción, que ilustren las facetas más representativas del problema.
- Documentación del código mediante comentarios y utilización de Javadoc para la documentación de clases.
- Realización de una documentación complementaria adecuada.
- Escritura de código que permita un buen mantenimiento; para lo cual se establecen normas de escritura de código.

10.3.3 Entorno de desarrollo utilizado por los estudiantes

En todos los módulos, excepto en Procesadores de Lenguajes, los alumnos han utilizado como herramienta de desarrollo Borland JBuilder, en el caso de los cuatro primeros grupos se trataba de JBuilder 7.0 que utilizaba como compilador la versión 1.3.1 de la plataforma Java Software Develpment Kit (JSDK) de Sun Microsystems. En el caso de los dos últimos grupos que pertenecen al curso 2004-2005 han utilizado la versión JBuilder 9.0 que utiliza como compilador el de la plataforma JSDK 1.4.1_02 de Sun Microsystems. En la asignatura de procesadores de lenguaje se permite la libre elección del entorno de desarrollo por parte del alumno y un porcentaje alto elige el entorno Eclipse.

10.3.4 Descripción de los proyectos utilizados en el trabajo

Los proyectos evaluados son exclusivamente proyectos completos que, se supone, cumplen con la funcionalidad pedida en el problema. Por tanto, el número varía debido al número de alumnos matriculados en la asignatura ese año y al número de alumnos que entregan su proyecto. Se procesan todos los proyectos entregados independientemente de la evaluación posterior por parte del profesor; es decir se incluyen también los proyectos de los alumnos suspensos.

En la Tabla 12 se pueden ver los proyectos procesados para cada uno de los subgrupos. Se puede apreciar la mayor complejidad de las prácticas de Estructura de Datos y de la Información en relación al número de clases por proyecto y al tamaño de cada una de las clases.

Tabla 12. Descripción de los grupos de proyectos seleccionados para el experimento

Asignatura	Curso	Núm. proyectos	Num. Fich. *.java	Media fich/proy	Tamaño fich .java (bytes)	Media tam/fich (bytes)
Metodología de la programación	1º	316	4.473	14,16	14.029.045	3.136,38
Estructura de datos y de la información	2º	337	8.319	24,69	45.192.259	5.432,41
Bases de Datos	3º	22	492	22,36	3.427.435	6.966,33
Procesadores de Lenguaje	4º	5	398	79,60	1.692.506	4.252,53
Proyectos Fin de Carrera		9	695	77,22	5.509.856	7.927,85

10.4 Proceso previo de los datos

Los proyectos utilizados para la realización de este trabajo son prácticas realizadas por los estudiantes de las asignaturas enunciadas anteriormente. Los proyectos fueron realizados en los laboratorios de prácticas con la presencia del profesor de la asignatura y otro tiempo de trabajo individual por parte del estudiante. No se impuso ningún requisito extra a los alumnos, en relación con este estudio, simplemente los ya establecidos en las especificaciones de prácticas. Debido en parte a que los alumnos han utilizado herramientas externas a nuestro sistema como entornos integrados de desarrollo (en concreto JBuilder de Borland), fue necesario un proceso previo para adaptar los proyectos al tratamiento que queríamos realizar con nuestro sistema.

JBuilder genera dos tipos de archivos que afectan a nuestro sistema:

- Genera una serie de archivos de copias de seguridad *.java~<num>~
- Tenemos los .class generados en un directorio aparte de los .java

Para adaptar los proyectos a nuestro sistema debemos realizar una serie de operaciones que no son necesarias si hubiésemos utilizado SICODE para el desarrollo:

- Quitar el atributo sólo lectura, debido a que hay proyectos que proceden de un CD y por tanto los directorios impiden escribir archivos resultado de la compilación necesarios para el proceso que realizamos.
- Eliminar todos los archivos que cumplen el patrón "*.java~*"
- Eliminar todos los archivos que cumplen el patrón "*.class"
- Verificar que los nombres de los directorios no sean muy largos para no hacer paths largos.
- Comprobar que la extensión de los archivos es .java en minúsculas ya que sino el parser de Ant no los encuentra.

Por otra parte, también se han eliminado determinados errores que aparecían en el análisis; pero que venían provocados por determinadas configuraciones de las herramientas y por tanto no son reales:

- En los entornos integrados se pueden declarar paquetes externos comunes que luego los alumnos no incluyen a la hora de hacer la entrega del proyecto.
- En mpmod1 y mpmod2 existe un archivo que se proporcionó a los alumnos que tenía código que nunca se llegaba a ejecutar; esto aparecía como un aviso en la versión 1.3 del compilador javac (la que utilizaron los alumnos), con lo cual no ocasionaba problemas; sin embargo, en la versión 1.4 de javac (la que utilizamos en el análisis) esto es notificado como un error.

10.5 Descripción del proceso al que se someten los proyectos

Una vez recopilados los proyectos de las distintas asignaturas y realizadas las operaciones previas utilizamos el prototipo *Sistema de análisis de errores de programas: PBA* descrito en el Capítulo 7 para realizar un análisis exhaustivo del código fuente escrito por los estudiantes y obtener así que errores suelen cometer.

10.5.1 Herramientas utilizadas y configuración de cada herramienta en el análisis

En este apartado enumeramos cada una de las herramientas que utiliza el prototipo *Sistema de análisis de errores de programas: PBA* para realizar el análisis. Estas herramientas se han descrito en el Capítulo 7. Algunas de ellas permiten configurar los tipos de errores que detectan, esto ha permitido especificar los errores concretos que queremos buscar con estas herramientas para potenciar su efectividad.

Javac

Utilizamos el compilador estándar de Java javac que está incluido en la plataforma sdk versión 1.4.2 de Sun Microsystems. La configuración es la que tiene por defecto ya que no se pasa ninguna opción de configuración.

Antic

La única opción que se le pasa al hacer la llamada es '-java' que indica que va a procesar un programa escrito en lenguaje Java, en vez de C que es la opción por defecto.

Findbugs

Herramienta findbugs versión 0.7.3.

Opciones de configuración que se utilizan: no se comprueban los errores de concurrencia entre hilos.

Jlint

Opciones que se utilizan al llamar a este programa desde la unidad de compilación realizada para el experimento: no se comprueban los errores de sincronización entre hilos.

PMD

Herramienta PMD versión 1.6. Esta herramienta permite definir reglas para verificar la corrección de los programas analizados. Se ha configurado esta herramienta para que detecte los errores recogidos en la Tabla 13 y la Tabla 14.

Tabla 13. Tipos de errores que hemos activado en PMD para su detección (1)

Sentencias vacias	Simplicidad del código	Convenciones de código y de nombres
NonCaseLabelInSwitchStatement	DontImportJavaLang	ForLoopsMustUseBracesRule
EmptyWhileStmt	DuplicateImports	IfElseStmtsMustUseBracesRule
EmptyFinalizer	SimplifyBooleanExpressions	WhileLoopsMustUseBracesRule
EmptySwitchStatements	UnnecessaryReturn	IfStmtsMustUseBraces
EmptyFinallyBlock	ForLoopShouldBeWhileLoop	SwitchStmtsShouldHaveDefault
EmptyStaticInitializer	SimplifyBooleanReturnsRule	VariableNamingConventionsRule

Sentencias vacias	Simplicidad del código	Convenciones de código y de nombres
EmptyTryBlock	UnconditionalIfStatement	ClassNamingConventionsRule
EmptyStatementNotInLoop	UnnecessaryConversionTemporaryRule	MethodNamingConventions
EmptyIfStmt	StringInstantiation	AbstractNamingRule
EmptySynchronizedBlock	StringToString	MethodWithSameNameAsEnclosingClass
	ImportFromSamePackage	AvoidDollarSigns

Tabla 14. Tipos de errores que hemos activado en PMD para su detección (2)

Código no utilizado	Mejoras de diseño	Errores potenciales
UnusedLocalVariable	UseSingletonRule	ExplicitCallToFinalize
UnusedImports	LooseCouplingRule	JumbledIncrementer
UnusedFormalParameter	SwitchDensity	OverrideBothEqualsAndHashcodeRule
UnusedPrivateField	NonStaticInitializer	DoubleCheckedLockingRule
	FinalFieldCouldBeStatic	AvoidReassigningParametersRule
	ProperCloneImplementationRule	CloseConnectionRule
	ConstructorCallsOverridableMethodRule	BooleanInstantiation
	FinalizeShouldBeProtected	FinalizeOnlyCallsSuperFinalize
	AvoidDuplicateLiterals	FinalizeDoesNotCallSuperFinalize
	DefaultLabelNotLastInSwitchStmt	AvoidCatchingThrowable
	SignatureDeclareThrowsException	SuspiciousHashcodeMethodName
	ExceptionTypeChecking (AvoidInstanceofChecksInCatchClause)	FinalizeOverloaded
		ReturnFromFinallyBlock

La forma de indicar al *Sistema de análisis de errores de programas: PBA* las herramientas que queremos utilizar, la configuración de cada una de ellas y el orden de ejecución se realiza a través de lo que llamamos *unidad de compilación*. En el Capítulo 7 se describe el diseño de estos archivos, así como la forma de uso.

10.5.2 Tipos de problemas buscados

La combinación de las herramientas enumeradas en el apartado anterior permiten a nuestro sistema detectar gran cantidad de errores, proporcionando una efectividad mucho mayor que el análisis realizado por el compilador del lenguaje.

A partir de los errores que detectan cada una de las herramientas con la configuración planteada hemos realizado una clasificación en una serie de tipos de error que permiten orientar el análisis.

En la Tabla 15 se incluyen los tipos de errores junto con su descripción que resumen el ámbito de errores buscados en este trabajo.

Tabla 15. Descripción de los distintos tipos de problemas buscados

Tipo error	Descripción
Aviso	Aviso de posible error en una aplicación.
Aviso de código ineficiente	Aviso sobre la posible ineficiencia de zonas del código.
Aviso de código innecesario	Aviso sobre zonas del código que son innecesarias o redundantes.
Aviso de código no usado	Aviso sobre zonas de código no usadas en el programa.
Aviso de comparación	Aviso de posibles problemas a la hora de llevar acabo una comparación.
Aviso de diseño	Aviso de un posible fallo a la hora de diseñar las aplicaciones, puede ser interesante pensar en una Refactorización del código.
Aviso de dominios	Aviso de posible fallo en el dominio de una variable, en el rango de una matriz, en la valor de retorno de una función, etc.
Aviso de estilo	Aviso de una violación de los convenios de estilo de codificación del Lenguaje. Estas normas pueden ayudar a evitar errores en algunas ocasiones.
Aviso de estructura de control	Aviso de un posible error en una estructura de control de flujo del programa.
Aviso de excepciones	Aviso de zonas de código que pueden ser propensas a lanzar excepciones, o bien de zonas donde se está haciendo un mal tratamiento de estas.
Aviso de finalize()	Aviso sobre posibles problemas en el uso del método finalize().
Aviso de inicialización	Aviso sobre un posible problema en las inicializaciones o en los constructores.
Aviso de JUnit	Aviso sobre un posible problema en el uso de la herramienta de prueba JUnit.
Aviso de literales	Aviso sobre un posible error en la escritura de un literal.
Aviso de nombre	Aviso sobre un problema potencial por no seguir las normas estándares de estilo para los nombres o por una mala elección de los nombres de campos, métodos o clases que pueden llevar a confusiones.
Aviso de ocultación	Aviso provocado por una posible ocultación de campos o métodos en una clase.
Aviso de operadores	Aviso sobre posibles errores en el uso de los operadores.
Aviso de sincronización	Aviso sobre posibles errores en la sincronización en aplicaciones multihilo.
Aviso de vulnerabilidad	Aviso sobre posibles vulnerabilidades de las clases ante código mal intencionado.
Error de compilación	Error encontrado por el compilador al analizar el código fuente.

10.6 Resultados del estudio

10.6.1 Información incluida en las tablas de resultados

En los siguientes apartados mostramos los resultados obtenidos tras el tratamiento de los proyectos correspondientes a diferentes cursos de la titulación de Ingeniería Informática (en la Universidad de Oviedo, la titulación se encuentra dividida entre primer y segundo ciclo).

Tabla 16. Tabla de avisos de incumplimiento de convenciones descartados en el análisis.

Código error	Descripción del error	Tipo de error
13019	Avoid using "if...else" statements without curly braces	Convenciones de código
13017	Avoid using "if" statements without curly braces	Convenciones de código
13071	Method name does not begin with a lower case character.	Convenciones de nombres
13020	Avoid using "for" statements without curly braces	Convenciones de código
13089	Method names should not contain underscores	Convenciones de nombres
13088	Variables should start with a lowercase character	Convenciones de nombres
13087	Variables that are not final should not contain underscores.	Convenciones de nombres
13091	Class names should not contain underscores	Convenciones de nombres
13073	Abstract classes should be named AbstractXXX"	Convenciones de nombres

Aunque la configuración de las herramientas incluida en la *unidad de compilación* utilizada para este trabajo detecta y recoge avisos correspondientes al incumplimiento de convenios de nombres y de código, estos convenios se han eliminado a la hora de elaborar las tablas (ver Tabla 16), ya que lo que pretendemos estudiar son avisos que indiquen la presencia de errores reales.

Los resultados del estudio se muestran en forma de tabla y aparecen ordenados en orden decreciente por el número absolutos de avisos que han aparecido para cada grupo de proyectos. En cada tabla de datos se incluye la siguiente información para cada uno de los avisos:

- Código del error. Es el código asignado al aviso correspondiente. Es un identificador único para cada aviso asignado por el Sistema de Análisis de Errores en Programas e independiente de la herramienta utilizada.
- Descripción del error. Descripción del aviso detectado. Se incluye en inglés basándonos en la descripción original que hace cada herramienta del error.
- Número de errores. Número total de avisos de este tipo en el grupo de proyectos analizado.
- Núm. proyectos con errores. Número de proyectos que han generado este tipo de aviso.
- Media de errores por proyecto. Es el número medio de avisos de un tipo determinado en los proyectos que contienen este tipo de error.

$$Media\_errores\_por\_proyecto = \frac{Número\_errores}{Núm\_proyectos\_con\_errores}$$

- Porcentaje de proyectos con errores. Es el porcentaje de proyectos que contienen este tipo de avisos respecto al total de proyectos analizados para este grupo.

$$Porcentaje\_proy\_errores = \frac{Núm\_proyectos\_con\_errores}{Total\_proyec} \cdot 100$$

- Total proyectos. Es el total de proyectos analizados para este grupo.
- En las gráficas el valor de la frecuencia de error para cada uno de los errores, se refiere a:

$$Frecuencia\_error = \frac{Número\_errores}{Total\_proyec}$$

10.6.2 Resultados del análisis de los proyectos correspondientes a primer curso

En este apartado describimos los resultados del análisis de los proyectos correspondientes a la asignatura de primer curso: Metodología de la programación. En primer lugar hacemos una mínima descripción de la asignatura y las prácticas planteadas en ella. Incluimos una tabla resumen de los errores más frecuentes con la información descrita en el apartado 10.6.1 y una gráfica con la misma información generada por el prototipo PBA, por último hacemos un análisis de los resultados para este curso.

Descripción de la asignatura: Metodología de la programación

- Es una asignatura donde se profundiza en los conceptos de orientación a objetos y técnicas de programación de algoritmos: recursividad y algoritmos de ordenación.
- Situación dentro del plan de estudios. Esta asignatura se imparte en primer curso de la carrera en su segundo cuatrimestre.
- Asignaturas de programación precedentes. Los alumnos que realizan esta asignatura han visto los fundamentos de la programación orientada a objeto en otra asignatura

del primer cuatrimestre denominada Introducción a la Programación. En esta asignatura también se realizan prácticas utilizando Java como lenguaje de implementación y el entorno JBuilder con lo que los alumnos ya están familiarizados con las funciones básicas del entorno y los fundamentos del lenguaje.

Prácticas que se proponen en la asignatura para su resolución

- Módulo 1 de Metodología de la programación (identificador: mpmod1). Consiste en la simulación de un juego de rol muy sencillo en el que se crea un personaje con determinadas características, unos escenarios con objetos o adversarios y una forma de luchar con los adversarios y recorrer los escenarios. En este proyecto el alumno debe crear estructuras dinámicas secuenciales, esencialmente listas y llevar a cabo una interacción con el usuario para decidir las acciones del personaje. No existe interacción con archivos.
- Módulo 2 de Metodología de la programación (identificador: mpmod2). Gestión de una biblioteca con autores y libros. Estos se organizan en memoria mediante listas dinámicas y la información se lee desde archivos almacenados en disco.

Tabla 17. Avisos más frecuentes en la asignatura de primer curso: Metodología de la Programación

Código error	Descripcion	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13044	Avoid calls to overridable methods during construction	2456	260	9,45	82,28%
12001	Component in this class shadows one in base class.	1873	118	15,87	37,34%
13039	Avoid unnecessary comparisons in boolean expressions	1484	183	8,11	57,91%
13080	The same String literal appears several times in this file.	1213	268	4,53	84,81%
13084	Avoid unused local variables.	1181	275	4,29	87,03%
12002	Local variable shadows component of class.	1033	176	5,87	55,70%
12014	Compare strings as object references.	769	119	6,46	37,66%
13040	Switch statements should have a default label	520	191	2,72	60,44%
13000	Avoid empty catch blocks	416	237	1,76	75,00%
11004	Comparison of String objects using "==" or "!="	296	117	2,53	37,03%
13016	An empty statement (semicolon) not part of a loop	289	229	1,26	72,47%
	Número total de proyectos	316			

Análisis de los avisos detectados con mayor frecuencia en los proyectos de este curso

Aparecen cuatro tipos de avisos que se dan en más del 75% de los proyectos:

- ***Avoid calls to overridable methods during construction***. Este aviso denota un proceso de construcción del objeto con muchos pasos y que por tanto, requiere la llamada a métodos auxiliares; sin embargo, estos métodos auxiliares no son exclusivos de la clase que pretendemos instanciar sino que redefinen método que ya existen en la superclase. Esto conlleva un error conceptual en el modelo de constructor que pretende hacer demasiadas cosas. Además, esto podría provocar errores difíciles de encontrar al invocar a métodos sin la seguridad de que todos los datos miembro de los objetos están correctamente inicializados.
- ***The same String literal appears several times in this file***. Este aviso simplemente afecta a la claridad del código ya que se podría haber creado una

constante y utilizarla en los distintos puntos de código; sin embargo, esto unido a otras cuestiones puede hacer más complicado la modificación y el mantenimiento del código fuente.

- ***Avoid unused local variables***. El aviso indica que después de las sucesivas modificaciones del código fuente han quedado variables locales que no se utilizan. Esto fundamentalmente afecta a la comprensión del código fuente ya que el desarrollador que lee el código va a analizar cada una de las variables que se declaran que sin embargo no van a ser utilizadas posteriormente.

Figura 66. Avisos más frecuentes para el primer curso

- ***Avoid empty catch blocks***. Este aviso denota que no hay código que maneje la excepción capturada para avisar al usuario de un problema irresoluble de forma automática o hacer una recuperación del error. Esto fundamentalmente es debido a que Java obliga a capturar las posibles excepciones y por eso el alumno utiliza la cláusula *catch*; pero no sabe lo que puede hacer con ella una vez capturada. Esto puede provocar un "retorno silencioso" es decir que salte una excepción y como es capturada en el catch, continúe la ejecución como si nada hubiese pasado y más tarde empiecen a ocurrir errores en la ejecución. Con lo cual el usuario que ejecuta la aplicación no se da cuenta que ha ocurrido un problema, ya que la excepción es capturada; pero hay problemas de funcionamiento cuya causa no está clara.

Otro aviso que no aparece en tantos proyectos; pero que es significativo ya que denota problemas en el modelo mental del alumno es: "Comparison of String objects using "==" or "!=""". Aquí se intentan comparar cadenas que en Java son objetos como si fuesen tipos primitivos, con el operador ==, con lo que en realidad se están comparando las referencias, en vez de utilizar el método equals.

10.6.3 Resultados del análisis de los proyectos correspondientes a segundo curso

En este apartado describimos los resultados del análisis de los proyectos correspondientes a la asignatura de primer curso: Estructura de datos y de la información. En primer lugar hacemos una mínima descripción de la asignatura y las prácticas planteadas en ella. Incluimos una tabla resumen de los errores más frecuentes con la información descrita en el apartado 10.6.1 y una gráfica con la misma información generada por el prototipo PBA, por último hacemos un análisis de los resultados para este curso.

Descripción de la asignatura: Estructura de Datos y de la Información

- Se estudian las estructuras de datos básicas: listas y sus variantes, árboles, tablas hash, grafos.
- Situación dentro del plan de estudios. Esta asignatura se imparte en el segundo curso de la carrera y tiene una duración anual.
- Asignaturas de programación precedentes. Previamente han visto como asignaturas de programación las citadas anteriormente de Introducción a la Programación, Metodología de la Programación y Comunicación Persona Máquina (segundo cuatrimestre de primer curso). En esta última asignatura también se realizan prácticas que conllevan la implementación de prototipos aunque haciendo hincapié en la interfaz de usuario. Sin embargo, también se utiliza el mismo entorno de desarrollo con lo que sirve a los alumnos para adquirir una mayor experiencia con él.

Tabla 18. Avisos más frecuentes para la asignatura de segundo: Estructuras de Datos y de la Información

Código error	Descripcion	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13046	This final field could be made static	4161	224	18,58	66,47%
11021	Unread field: should this field be static?	3792	211	17,97	62,61%
13080	The same String literal appears several times in this file.	2249	287	7,84	85,16%
13084	Avoid unused local variables.	1993	292	6,83	86,65%
13000	Avoid empty catch blocks	1764	207	8,52	61,42%
12002	Local variable shadows component of class.	1684	262	6,43	77,74%
13078	A method should'nt have Exception in throws declaration.	1358	98	13,86	29,08%
13042	Avoid reassigning parameters.	1342	267	5,03	79,23%
13040	Switch statements should have a default label	1279	254	5,04	75,37%
13039	Avoid unnecessary comparisons in boolean expressions	1121	135	8,30	40,06%
10013	May be wrong assumption about ELSE branch association	1020	166	6,14	49,26%

337

Prácticas que se proponen en la asignatura para su resolución

- Módulo 1 de Estructura de Datos y de la Información (identificador: edi3mod1). Aplicación para una empresa de autocares que tiene líneas regulares que comunican distintas poblaciones. La aplicación debe permitir solucionar a la empresa todas las necesidades de información pre-venta de los billetes a los viajeros que preguntan sobre itinerarios, horarios, tipo de autocar, servicios que se ofrecen durante el viaje y precio del billete. Toda la información sobre itinerarios, horarios, tipo de autocar y servicios estará almacenada en un archivo del tipo BML un lenguaje creado a

partir de XML. La aplicación deberá permitir cargar el archivo sobre estructuras de datos en memoria para gestionar esta información. Así mismo, debe permitir volver a guardar la información de memoria en un archivo con la misma estructura. Los alumnos deben desarrollar a partir de los paquetes básicos la lectura de los archivos XML, no utilizarán ningún paquete específico para este propósito. El usuario dispondrá de un menú para elegir las acciones que quiere realizar.

- Módulo 2 de Estructura de Datos y de la Información (identificador: edi3mod2). Se trata de ampliar edi3mod1 con las siguientes características: un Arbol Binario de Búsqueda que permita almacenar y buscar poblaciones, este almacenamiento se realiza en un formato diferente al resto de los datos y para su lectura se emplea introspección mediante la cual se crean los objetos adecuados para el almacenamiento y el tratamiento de la información en memoria, un TAD Matriz Dinámica Genérica que permita almacenar horarios y desarrollar una opción que permita mostrar las posibles alternativas para viajar entre dos poblaciones pudiendo hacer transbordos.

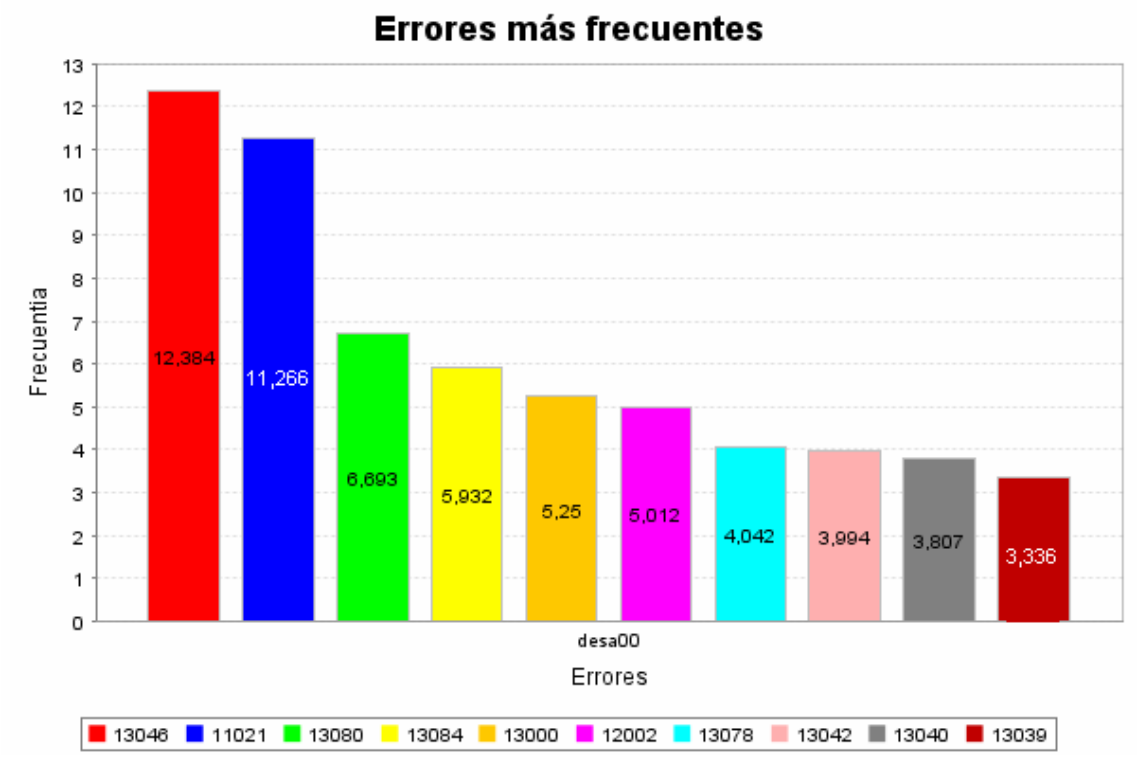

Figura 67. Avisos más frecuentes para el grupo de proyectos de segundo curso

- Módulo 1 de Estructura de Datos y de la Información (identificador: edi4mod1). Gestor de datos de una mini biblioteca en la cual sólo habrá libros. Toda la información del gestor estará almacenada en un archivo del tipo DML (un lenguaje creado a partir de XML). La aplicación deberá permitir cargar el archivo sobre estructuras de datos en memoria y gestionar esta información. Así mismo, debe permitir volver a guardar la información de memoria en un archivo con la misma estructura. La carga del archivo DML se realizará en la siguiente estructura de datos, un nodo con dos referencias: Lista de libros, es una lista simplemente encadenada y ordenada en la que tenemos el código del libro y el título del mismo. Índice, es un árbol de búsqueda que contiene los códigos de los libros así como una

referencia a un nodo de la lista de libros. Las acciones sobre el gestor se realizan mediante una interfaz en línea de comandos.

- Módulo 2 de Estructura de Datos y de la Información (identificador: edi4mod2). Se trata de ampliar edi4mod1 con las siguientes características: generalizar el gestor para que pueda leer archivos con distinto tipo de datos, se pide a los alumnos que realicen esto mediante introspección; realizar joins entre distintas tablas.

Análisis de los avisos detectados con mayor frecuencia en los proyectos de este curso

Existen cinco tipos de avisos que están muy extendidos (aparecen en más del 75% de los proyectos):

- ***The same String literal appears several times in this file***. Este aviso ya aparecía entre los más extendidos en los proyectos de primer curso y puede provocar problemas en el mantenimiento de las aplicaciones.
- ***Avoid unused local variables***. Este aviso también aparecía en el apartado anterior y provoca problemas en la lectura y por tanto en la comprensión del código.
- ***Local variable shadows component of class***. Este problema también dificulta la comprensión del código ya que al coincidir el nombre de la variable local y del componente hay que tener claro en primer lugar que hay dos elementos que se llaman igual y además, hay que saber a cuál nos estamos refiriendo en cada punto del código.
- ***Avoid reassigning parameters***. La reasignación de parámetros puede provocar errores debido a los efectos laterales y además dificulta la lectura del código.
- ***Switch statements should have a default label***. Java no obliga a que todas las sentencias switch tengan una etiqueta default sin embargo en estas condiciones puede ocurrir que la condición de la sentencia tome un valor inesperado y por tanto el flujo de ejecución no entre por ninguna de las etiquetas previstas provocando un mal funcionamiento cuya causa es difícil de determinar ya que ocurre lejos de esta.

10.6.4 Resultados del análisis de los proyectos correspondientes a tercer curso

En este apartado describimos los resultados del análisis de los proyectos correspondientes a la asignatura de primer curso: Bases de datos. En primer lugar hacemos una mínima descripción de la asignatura y las prácticas planteadas en ella. Incluimos una tabla resumen de los errores más frecuentes con la información descrita en el apartado 10.6.1 y una gráfica con la misma información generada por el prototipo PBA, por último hacemos un análisis de los resultados para este curso.

Descripción de la asignatura: Bases de Datos

- El objetivo de la asignatura es que el alumno sepa diseñar y crear el esquema de una base de datos y conozca distintas herramientas de acceso como la utilización de SQL como lenguaje de modificación y consulta; así como el acceso desde lenguajes de programación convencionales. En el caso del módulo que analizamos se trata de desarrollar un pequeño proyecto en el que se diseñe y cree una base de datos adecuada al problema, programación del servidor mediante procedimientos

almacenados en la base de datos, programación del cliente mediante lenguaje Java y un controlador que permita el acceso a la base de datos.

- Situación dentro del plan de estudios. Esta asignatura se estudia en tercero y tiene una duración anual.

Tabla 19. Avisos más frecuentes para la asignatura de tercer curso: Bases de Datos

Código error	Descripcion	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13078	A method shouldn't have Exception in throws declaration.	269	15	17,93	68,18%
13084	Avoid unused local variables.	113	13	8,69	59,09%
11023	This field is never read. Consider removing it from the class.	113	13	8,69	59,09%
13080	The same String literal appears several times in this file.	111	17	6,53	77,27%
13044	Avoid calls to overridable methods during construction	97	16	6,06	72,73%
13061	Avoid unused imports	58	13	4,46	59,09%
13059	Avoid duplicate imports	38	12	3,17	54,55%
11011	Redundant comparison of a reference value to null.	25	2	12,50	9,09%
13000	Avoid empty catch blocks	21	5	4,20	22,73%
13039	Avoid unnecessary comparisons in boolean expressions	15	5	3,00	22,73%
13083	Avoid unused private fields.	15	5	3,00	22,73%

Número de proyectos	22

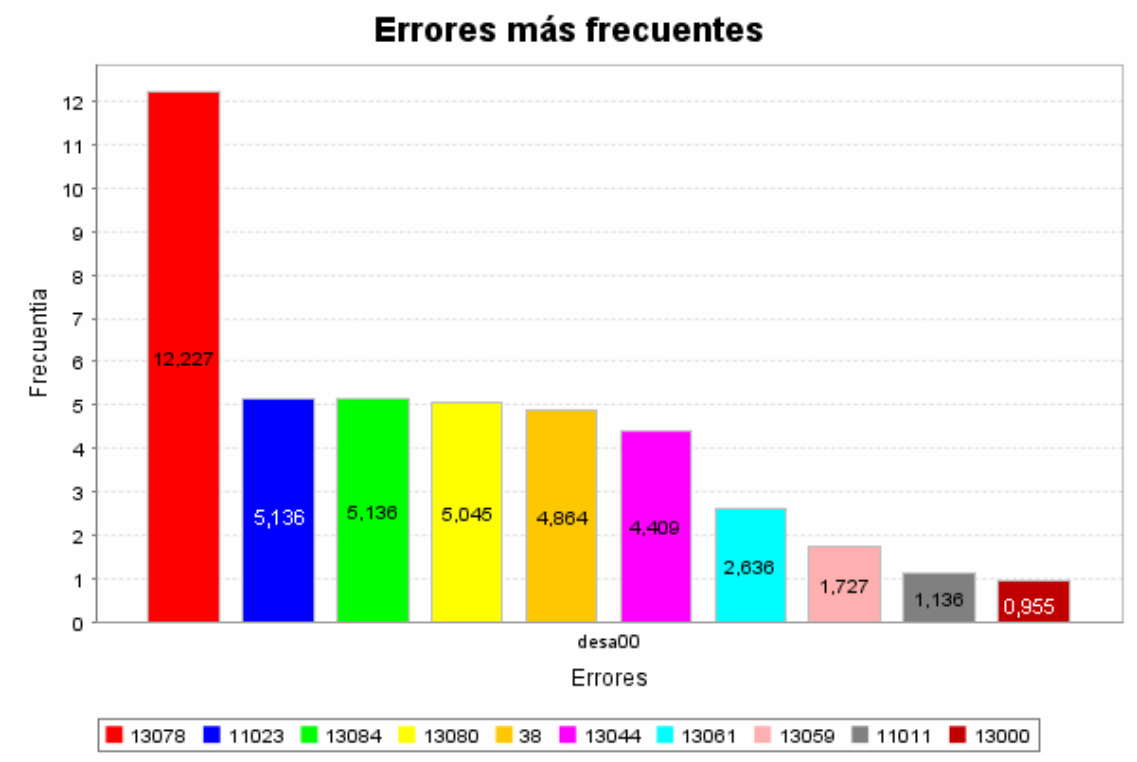

Figura 68. Avisos más frecuentes para el grupo de proyectos de tercer curso

- Asignaturas de programación precedentes. Estructuras de Datos y de la Información (segundo curso), la parte de las estructuras de datos en memoria secundaria, como archivos indexados, árboles B, etc. Introducción a la Programación, Metodología de la Programación y Tecnología de la Programación (primer y segundo curso), en cuanto a conocimientos generales de desarrollo de

proyectos de software, el modelo orientado a objetos y lenguajes de programación, y programación en general.

Prácticas que se proponen en la asignatura para su resolución

- Bases de datos. Se trata de desarrollar proyecto amplio de gestión de un campeonato de Formula 1. Con un módulo en el servidor basado en procesos almacenados en la base de datos, en este caso Oracle, programados en PL-SQL y un módulo en el cliente que hace interfaz con el usuario y realiza varios tipos de consultas y modificaciones sobre la base de datos, este módulo programado con lenguaje Java.

Análisis de los avisos detectados con mayor frecuencia en los proyectos de este curso

Sólo hay un aviso que aparece en más del 75% de los proyectos: ***The same String literal appears several times in this file***. Este es un error que aparece reiteradamente en todos los cursos, ya que al no tener gran importancia no se trata sobre él en clase y por tanto no se corrige.

Por otra parte, el error que aparece mayor número de veces aunque en menos proyectos es: ***A method shouldn't have Exception in throws declaration***. Este también es un problema de claridad de lectura del código, ya que el programador que vaya a utilizar un método no sabrá exactamente que excepción puede lanzar. También denota un problema conceptual ya que normalmente es debido a que el programador lanza excepciones de distintos tipos en un método, lo cual indica que seguramente deberíamos de tener varios métodos para hacer ese trabajo o bien que el programador no tiene claro que excepciones va a lanzar el método y deja la declaración más genérica posible.

10.6.5 Resultados del análisis de los proyectos correspondientes a cuarto curso

En este apartado describimos los resultados del análisis de los proyectos correspondientes a la asignatura de primer curso: Procesadores de lenguaje. En primer lugar hacemos una mínima descripción de la asignatura y las prácticas planteadas en ella. Incluimos una tabla resumen de los errores más frecuentes con la información descrita en el apartado 10.6.1 y una gráfica con la misma información generada por el prototipo PBA, por último hacemos un análisis de los resultados para este curso.

Descripción de la asignatura: Procesadores de Lenguaje

- El objetivo principal de esta asignatura es enseñar al alumno los problemas y técnicas que se plantean en la construcción de procesadores, traductores, compiladores e intérpretes de los distintos lenguajes de programación. El estudio de estas técnicas permite al alumno una visión más amplia de los lenguajes de programación, habitualmente estudiados desde el punto de vista del programador y no del constructor de compiladores o intérpretes.

- Situación dentro del plan de estudios. Esta asignatura se imparte en cuarto curso del segundo ciclo de Ingeniería en Informática y tiene una duración anual.

- Asignaturas de programación precedentes. Los alumnos cuando llegan a esta asignatura tienen un conocimiento amplio del lenguaje de programación empleado y de las técnicas de programación y depuración.

Tabla 20. Avisos más frecuentes para la asignatura de cuarto curso: Procesadores de Lenguaje

Código error	Descripcion	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13084	Avoid unused local variables.	64	3	21,33	60,00%
13080	The same String literal appears several times in this file.	50	4	12,50	80,00%
13081	Avoid instantiating String objects; this is usually unnecessary.	47	2	23,50	40,00%
13078	A method shouldn't have Exception in throws declaration.	34	3	11,33	60,00%
11003	Innecesary calls to methods.	33	2	16,50	40,00%
13039	Avoid unnecessary comparisons in boolean expressions	30	1	30,00	20,00%
12005	Value of referenced variable may be NULL.	23	1	23,00	20,00%
12001	Component in this class shadows one in base class.	21	3	7,00	60,00%
13000	Avoid empty catch blocks	20	2	10,00	40,00%
12014	Compare strings as object references.	18	2	9,00	40,00%

Número de proyectos	5

Prácticas que se proponen en la asignatura para su resolución

- Procesadores de Lenguaje. El proyecto consiste en desarrollar un compilador de un lenguaje sencillo que compile a un lenguaje intermedio. Este compilador se realiza con la ayuda de varias herramientas que permiten generar determinadas partes del compilador: analizador léxico y analizador sintáctico ascendente.

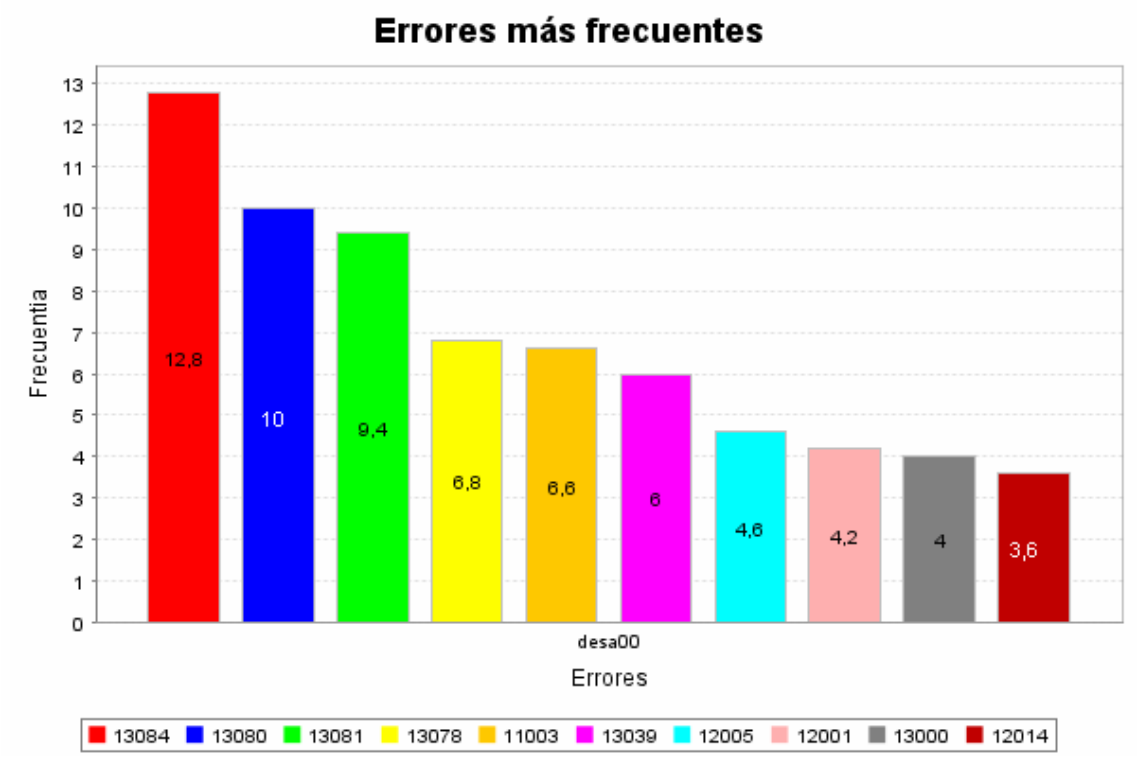

Figura 69. Avisos más frecuentes para el grupo de proyectos de la asignatura de cuarto curso.

Análisis de los avisos detectados con mayor frecuencia en los proyectos de este curso

Sólo hay un avisos que está presente en más del 75% de los proyectos: ***The same String literal appears several times in this file***. De nuevo nos encontramos con este problema que se hereda desde primer curso.

10.6.6 Resultados del análisis de los proyectos correspondientes al proyecto Fin de Carrera

En este apartado describimos los resultados del análisis de los proyectos correspondientes al proyecto fin de carrera. En primer lugar hacemos una mínima descripción de la asignatura y las prácticas planteadas en ella. Incluimos una tabla resumen de los errores más frecuentes con la información descrita en el apartado 10.6.1 y una gráfica con la misma información generada por el prototipo PBA, por último hacemos un análisis de los resultados para este curso.

Descripción del Proyecto Fin de Carrera

- Objetivo. Esta materia no consiste en una asignatura como tal, con clases teóricas y prácticas; sino que el estudiante tiene que realizar un proyecto autorizado e individual en el que aplique los conocimientos adquiridos en la carrera. Cada alumno tendrá su propio proyecto diferente al resto y se pretende que realice el ciclo completo en el proceso de desarrollo del software: análisis, diseño, implementación, pruebas, implantación.
- Situación dentro del plan de estudios. Este proyecto como su nombre indica se realiza al final de la carrera, se suele comenzar en el último cuatrimestre cuando el alumno ha visto la mayoría de las asignaturas de la carrera.
- Asignaturas de programación precedentes. El alumno ya ha visto todas las asignaturas relacionadas con la programación. Aunque debido a la diversidad de los proyectos puede tener que emplear técnicas o herramientas que no ha visto durante la carrera.

Objetivos que se proponen en los proyectos fin de carrera para su resolución

- Los trabajos que se proponen tienen una tipología muy variada; pero tienen en común que son proyectos reales o que podrían ser reales, y por tanto tienen un tamaño medio o grande donde el alumno tiene que trabajar desde la especificación de requisitos hasta el proceso de implantación en un entorno real.
- El lenguaje y entorno de desarrollo varían con el proyecto.

Para el estudio se han seleccionado proyectos con las siguientes características:

- La mayoría son aplicaciones Web con una interfaz de usuario realizada en HTML, JSP. Aunque una de ellas utiliza interfaz swing para realizar una aplicación de escritorio.
- Están realizados con el lenguaje Java, el entorno de desarrollo más utilizado es JBuilder, aunque alguno de ellos utiliza Eclipse o NetBeans.

Tabla 21. Avisos más frecuentes para el Proyecto Fin de Carrera

Código error	Descripcion	Número errores	Núm. proyectos con errores	Media errores por proyecto	Porcentaje de proy errores
13080	The same String literal appears several times in this file.	354	4	88,50	44,44%
13061	Avoid unused imports	220	1	220,00	11,11%
13000	Avoid empty catch blocks	128	2	64,00	22,22%
13084	Avoid unused local variables.	119	4	29,75	44,44%
13081	Avoid instantiating String objects; this is usually unnecessary.	116	4	29,00	44,44%
13044	Avoid calls to overridable methods during construction	110	2	55,00	22,22%
13011	Avoid returning from a finally block	98	3	32,67	33,33%
13039	Avoid unnecessary comparisons in boolean expressions	88	3	29,33	33,33%
13078	A method shouldn't have Exception in throws declaration.	65	1	65,00	11,11%
13042	Avoid reassigning parameters.	54	3	18,00	33,33%

Número total de proyectos	9

Análisis de los avisos detectados con mayor frecuencia en los proyectos de este curso

Ningún aviso alcanza ni el 50% de los proyectos, lo cual indica que los errores están muy repartidos entre los distintos proyectos.

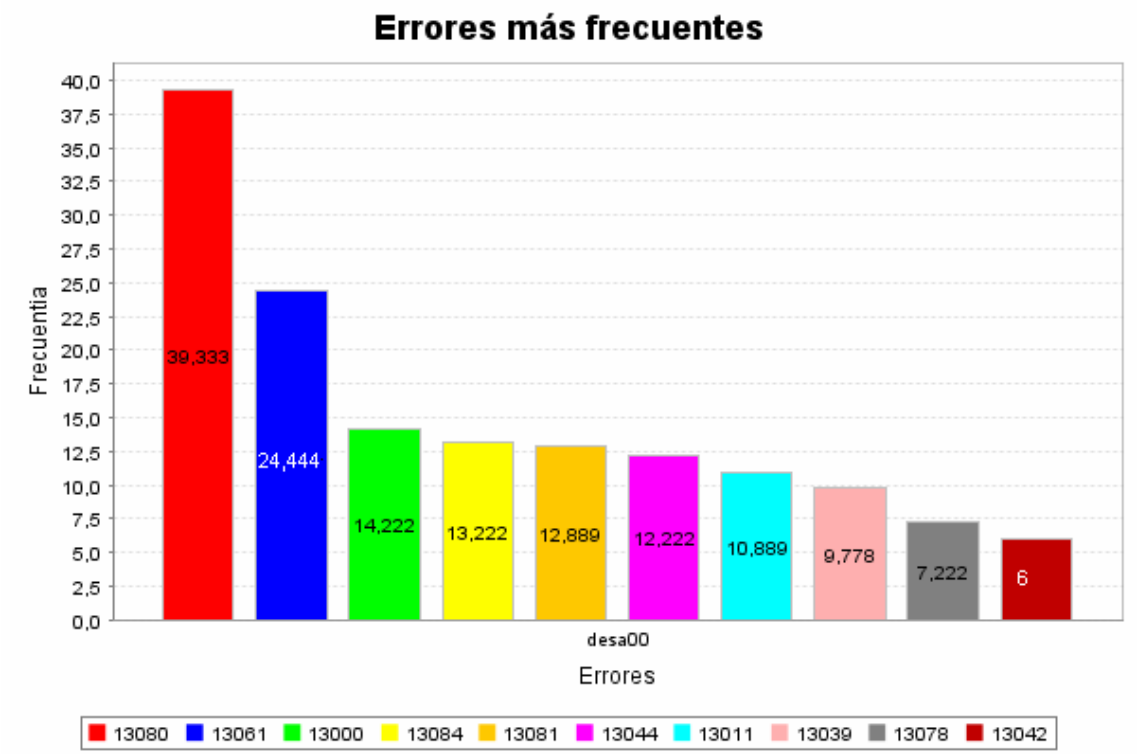

Figura 70. Avisos más frecuentes para el Proyecto Fin de Carrera

10.7 Comparación de los avisos y conclusiones

En la Tabla 22 listamos todos los avisos recogidos en las tablas previas de avisos frecuentes; pero ahora los ordenamos dependiendo del curso en el que aparecen para poder apreciar las diferencias. El número que aparece en las columnas de la derecha corresponde al número medio de errores en los proyectos, de ese curso, que contienen ese tipo de error. Analizando los datos de la tabla podemos considerar dos grandes clases de avisos:

- Los que aparecen en todos los cursos.
- Los que van evolucionando.

Tabla 22. Avisos encontrados en cada grupo de proyectos

Código error	Descripción del error	curso 1	curso 2	curso 3	curso 4	pfc
13080	The same String literal appears several times in this file.	4,53	7,84	6,53	12,50	88,50
13084	Avoid unused local variables.	4,29	6,83	8,69	21,33	29,75
13039	Avoid unnecessary comparisons in boolean expressions	8,11	8,30	3,00	16,50	29,33
13000	Avoid empty catch blocks	1,76	8,52	4,20	10,00	64,00
11004	Comparison of String objects using "==" or "!="	2,53				
13016	An empty statement (semicolon) not part of a loop	1,26				
13040	Switch statements should have a default label	2,72	5,04			
12002	Local variable shadows component of class.	5,87	6,43			
13044	Avoid calls to overridable methods during construction	9,45		6,06		55,00
12001	Component in this class shadows one in base class.	15,87			7,00	
12014	Compare strings as object references.	6,46			9,00	
11021	Unread field: should this field be static?		17,97			
13046	This final field could be made static		18,58			
13042	Avoid reassigning parameters.		5,03			18,00
10013	May be wrong assumption about ELSE branch association		6,14			
13078	A method should'nt have Exception in throws declaration.		13,86	17,93	11,25	65,00
13061	Avoid unused imports			4,46		220,00
13059	Avoid duplicate imports			3,17		
11011	Redundant comparison of a reference value to null.			12,50		
13083	Avoid unused private fields.			3,00		
11023	This field is never read. Consider removing it from the class.			8,69		
13081	Avoid instantiating String objects; this is usually unnecessary.				23,50	29,00
11003	Innecesary calls to methods.				11,33	
12005	Value of referenced variable may be NULL.				23,00	
13011	Avoid returning from a finally block					32,67
25,00	Total errores especificados	11,00	11,00	11,00	10,00	10,00

10.7.1 Errores que aparecen en todos los cursos

Existen una serie de errores que aparecen en todos los cursos analizados. Estos errores denotan dos tipos de problemas:

- Problemas a los que los docentes de las asignaturas no dan importancia, podría ser el caso de: ***The same String literal appears several times in this file***, ***Avoid***

unused local variables y ***Avoid unnecessary comparisons in boolean expressions***.

- Problemas que no se detectan nunca o en pocas ocasiones ante el poco tiempo para hacer una revisión del código de la práctica que existe actualmente. Es el caso de: ***Avoid empty catch blocks***. Además, este es un claro ejemplo de un problema en el que un análisis estático se muestra mucho más efectivo que la realización de pruebas.

10.7.2 Errores exclusivos de un curso

En la Tabla 22 aparecen algunos errores como exclusivos de un único curso. Los más significativos son:

- ***Comparison of String objects using "==" or "!="*** que aparece únicamente en primer curso. La media de errores por proyecto con errores es baja; sin embargo, este es un claro error conceptual a la hora de realizar el tratamiento de los objetos String. Parece claro que este problema se aborda por parte de los profesores de la asignatura, ya que en los siguientes cursos no aparece.
- ***This final field could be made static***, aparece únicamente en segundo. Este error parece evidenciar problemas en los conceptos de final y de static. En curso el concepto de static no se trata en profundidad es en segundo cuando aparece y se trata más ampliamente.
- ***Redundant comparison of a reference value to null***, aparece en tercer curso. Este error se detecta cuando hay una comparación entre referencias donde alguno de ellos es siempre null, normalmente porque se ha eliminado la referencia anteriormente. Por tanto, parece indicar que el programador ha cambiado código sin darse cuenta de los efectos sobre otra parte del código y por tanto seguramente habrá algún error lógico. Por otra parte, otros errores exclusivos de este curso están relacionados con campos no usados: ***Avoid unused private fields*** y ***This field is never read. Consider removing it from the class***. Esto tiene clara relación con la asignatura elegida en este tercer curso: base de datos. El tipo de proyectos que se realizan en esta asignatura, implica clases que se corresponden con tablas en la base de datos y por tanto habrá campos en cada clase por cada uno de los campos de la tabla, si posteriormente las operaciones que se piden para una tabla sólo implican operar con un campo los demás quedan sin utilizar.
- ***Value of referenced variable may be NULL***. Aparece en cuarto curso. Este error se produce cuando se utiliza una referencia que en determinadas condiciones del flujo de control se asigna a null. Por tanto, o hay determinadas condiciones del programa que nunca se ejecutan y por tanto se podrían eliminar o bien habría que comprobar si la referencia es null antes de utilizarla.

10.7.3 Errores que evolucionan con los cursos

Hay otro conjunto de errores que no aparecen en todos los cursos; pero tampoco son exclusivos de un único curso. Por ejemplo, ***Local variable shadows component of class***, aparece en primer y segundo cursos y luego desaparece.

10.7.4 Conclusiones finales

En este capítulo hemos abordado el estudio de los errores a partir de los análisis de los proyectos en distintos cursos de Ingeniería Informática.

Se ha realizado un estudio con un gran número de proyectos y además no se ha limitado a un único nivel sino que se han obtenido proyectos de cinco niveles de formación de los estudiantes.

En el análisis se han filtrado los mensajes relacionados con las convenciones de código y nombres que por un lado sólo se abordan directamente en alguna de las asignaturas y por otro lado no consideramos que puedan ser indicadores directos de errores.

Se ha utilizado el sistema propuesto en esta tesis: SICODE a través del subsistema PBA para obtener las tablas de los diez o doce errores que aparecen más frecuentemente en cada uno de los cursos y se proporcionan los datos de si los errores están concentrados en pocos proyectos o repartidos entre todos los proyectos y la media de errores de ese tipo en cada uno de los proyectos. Además, se incluye una gráfica de barras que permite comparar la frecuencia de cada uno de los errores por curso.

Por último, se ha elaborado una tabla global que permite comparar los errores en distintos cursos. En esta tabla hemos caracterizado errores dependiendo de los cursos en los que aparecen y su evolución: errores que aparecen en todos los cursos, errores que son exclusivos de un curso (en todos los cursos hay este tipo de errores) y errores que evolucionan con los cursos.

Este estudio proporciona datos que permitirán construir las bases para un modelo de usuario y de esta forma que el sistema se adapte a cada usuario dependiendo su perfil.