La evaluación continua y la autoevaluación en el marco de la enseñanza de la programación orientada a objetos

E. Mosqueira-Rey
Departamento de Computación
Universidad de A Coruña
Campus de Elviña, s/n
15071 A Coruña
eduardo@udc.es

Resumen

La implantación del crédito ECTS nos lleva a adoptar un enfoque de evaluación orientado al proceso basado, fundamentalmente, en la evaluación continua. En dicho proceso los estudiantes reciben refuerzo constante sobre la corrección de las tareas que llevan a cabo. Como acompañamiento a esta técnica se plantea el desarrollo de técnicas de autoevaluación, que permiten a los alumnos participar o conocer de antemano la evaluación de sus trabajos. En este trabajo se comentan las experiencias sobre autoevaluación y evaluación continua llevadas a cabo en la asignatura de Programación Orientada a Objetos, tanto en el ámbito de la evaluación de la teoría (a través de tests o talleres de autoevaluación) como en el de la práctica (a través del uso de foros o la autocorrección de prácticas a través de tests de unidad y herramientas de cobertura).

1. Introducción

Uno de los inconvenientes de la evaluación continua es la correcta estimación del esfuerzo que el alumno realiza a la hora de desarrollar los trabajos planteados, de forma que estos logren el objetivo del aprendizaje sin caer en una excesiva carga de trabajo. La experiencia acumulada de años anteriores puede permitir obtener un indicador más o menos objetivo que sirva para recalcular dicho esfuerzo y acercarlo a la realidad. Sin embargo, no debe olvidarse que en un sistema de evaluación continua también aumenta el trabajo y la dedicación horaria del docente, no sólo por la preparación de las actividades sino también por la corrección y calificación de los resultados, especialmente cuando el número de estudiantes es elevado (lo cual no siempre va acompañado del merecido reconocimiento a nivel institucional, académico o retributivo [5]).

Una forma de lograr que la evaluación continua sea efectiva es combinarla con técnicas de autoevaluación. La autoevaluación consiste en que el propio alumno participe en la decisión de la evaluación del propio trabajo. También puede plantearse la alternativa de la co-evaluación, en la que son los propios compañeros los que se encargan de dicha tarea. Dentro de las ventajas de la autoevaluación y/o la co-evaluación podemos citar: los estudiantes hacen suyos los criterios de evaluación, ven soluciones alternativas a sus trabajos, se esfuerzan más si saben que van a a ser evaluados por compañeros, se fomenta el hábito de reflexión sobre el trabajo realizado y el ser crítico con el trabajo de los demás y, además, ayudan al profesor en la tarea de dar retroalimentación a tiempo [10].

Una autoevaluación realizada de forma que no suponga una carga excesiva a los alumnos, sino entendida como parte del proceso de aprendizaje, y apoyada en entornos virtuales de aprendizaje/enseñanza [6], permite conseguir los objetivos de la evaluación continua sin resultar en una carga excesiva para la labor del docente. En este trabajo comentaremos las experiencias de evaluación continua y autoevaluación en el marco de la enseñanza de la Programación Orientada a Objetos (POO), tanto dentro del ámbito de la teoría (mediante tests y talleres prácticos), como dentro del ámbito práctico (a través de la utilización de foros virtuales o la comprobación de ejercicios prácticos mediante el uso de utilidades como JUnit o Cobertura).

2. Evaluación de la parte teórica y la parte práctica

Una asignatura como Programación Orientada a Objetos (POO) tiene una importante carga práctica, como es propio de cualquier asignatura de programación, en la que se plantean ejercicios que tienen que

ser resueltos por los alumnos. Pero también es necesario recordar que la POO tiene un aspecto teórico nada desdeñable. En nuestro caso la asignatura se imparte en segundo o tercer año (dependiendo del plan de estudios) y tiene dos partes diferenciadas: programación orientada a objetos y diseño orientado a objetos. En el nuevo plan aprobado la asignatura pasa a denominarse "Diseño de Software"pero los descriptores continúan siendo los mismos y se imparten en segundo año de la carrera. En la parte de programación se describen los aspectos típicos de la POO (herencia, polimorfismo, etc.) y en la parte de diseño se habla de temas como UML, principios y patrones de diseño, y en general de cómo usar las características de la POO vistas previamente para generar código de calidad.

Para la evaluación continua de la parte teórica usamos tests de autoevaluación y talleres de auto y coevaluación. Aunque se enmarcan dentro de la parte teórica tienen un contenido bastante práctico en el sentido de que muchas pruebas consisten en mostrar un software e indicar cuál es el resultado de la ejecución del mismo, o mostrar un diagrama UML e indicar qué software es una correcta representación del mismo (o al contrario a partir de un software obtener el diagrama UML que lo represente).

Para la evaluación continua de la parte práctica se utilizan foros de preguntas y respuestas, que es muy similar a un cuestionario pero en este caso la respuesta no es cerrada sino libre, por lo que se les puede pedir a los alumnos que desarrollen un pequeño programa y, finalmente, boletines de ejercicios tradicionales pero autoevaluados en parte usando herramientas como *JUnit* o *Cobertura*.

En el curso 09-10 el reparto final de peso en la evaluación de la asignatura quedó de la siguiente forma: Examen de teoría (30 %), práctica de diseño (30 %), actividades del campus virtual (10 %) - estas actividades incluían tests de autoevaluación para la teoría y foros de preguntas-respuestas para la práctica -, boletines de ejercicios (30 %). Es decir, el 40 % de la evaluación se basaba en la evaluación continua. Es de destacar que, como se comenta más adelante, en este curso no pudieron realizarse talleres, por lo que es de esperar de que este porcentaje aumente en futuros cursos.

3. Tests de autoevaluación

3.1. Planteamiento

La primera solución evidente para desarrollar la evaluación continua y la autoevaluación dentro de la parte teórica de la asignatura es la realización de tests por parte de los alumnos. Los tests son de respuesta cerrada, es decir, se plantea una pregunta y se dan varias opciones de respuesta de las cuales el alumno debe escoger la correcta. Si se quiere complicar un poco más la prueba se puede aumentar el número de posibles respuestas y hacer que varias de ellas sean correctas. Los tests se planifican para que su desarrollo se realice al término de cada tema visto en clase, lo cual permite reforzar lo explicado en las clases magistrales. Además el alumno sabe que dichos tests no estarán disponibles después de un periodo determinado de tiempo (normalmente una semana) por lo que no debe obviarlos durante el curso y utilizarlos solo al final como repaso de la asignatura, ya que no es este el fin que persiguen.

3.2. Evaluación

Cualquier sistema de evaluación debe ofrecer al alumno los siguientes atributos [10]: *fiabilidad*(el resultado de la evaluación es el mismo con independencia de la persona que la realice), *precisión* (el resultado tiene poco margen de error) y *prontitud* (el resultado está en manos del alumno lo antes posible). Usar una plataforma de enseñanza virtual como Moodle ([8]) permite conseguir los atributos antes mencionados a la hora de realizar los tests. Esto exige un esfuerzo previo por parte del docente a la hora de prepararlos, ya que las respuestas correctas o incorrectas deben ir acompañadas de una explicación para que su tarea de aprendizaje sea efectiva. Pero una vez realizado dicho esfuerzo la evaluación de los alumnos es automática, por lo que es una solución muy adecuada para enfrentarse a clases con un número elevado de alumnos. En la Figura 1 podemos ver un ejemplo de test extraído directamente de la plataforma Moodle.

3.3. Resultados obtenidos

En la asignatura de POO los tests los podían realizar los alumnos en el momento y situación que ellos pensaran que eran más adecuados, dentro de los lí-

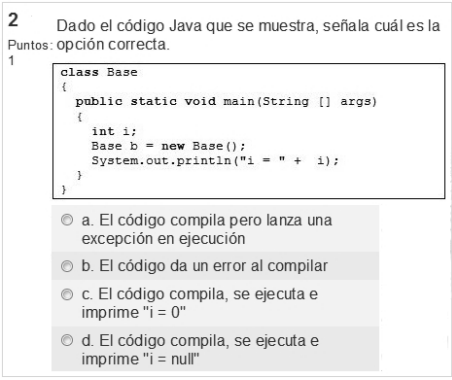

Figura 1: Pregunta de test sobre la inicialización de variables en Java

mites temporales expuestos, y podían consultar los apuntes si así lo consideraban necesario. Los tests se podían repetir cuantas veces se quisiera pero sólo la calificación inicial contaba para la nota.

La idea de los mismos era forzarlos a repasar la teoría durante el curso y no sólo en las vísperas del examen. Para tratar de evitar que los alumnos se centraran sólo en la nota a sacar y no en repasar los conocimientos aprendidos se les notificó que los tests de autoevaluación serían un fiel reflejo del test de teoría que se encontrarían en el examen de la asignatura, por lo que copiarlo de un compañero a la larga les perjudicaría. Los resultados obtenidos (que presentaban una amplia variabilidad en las notas) eran indicativos de que no se habían producido copias entre los compañeros.

4. Talleres

4.1. Planteamiento

El taller es probablemente la actividad más completa y compleja de Moodle. Permite como pocas actividades el aprendizaje y la evaluación cooperativa, introduciendo a los estudiantes en un proceso de evaluación conjunta y de autoevaluación. Cada estudiante observa cómo han resuelto el mismo problema otros compañeros/as, enriqueciendo así sus puntos de vista y sus posibilidades de aprendizaje. Además, debe ser crítico y realizar una evaluación rigurosa del trabajo de los demás, según unos criterios previamente establecidos, dejando menos margen a la intuición.

4.2. Evaluación

En la asignatura de POO existen partes de programación pura y dura, pero también existe una parte importante de aprender a hacer un diseño correcto orientado a objetos (normalmente basado en patrones de diseño) y de aprender a plasmar luego este diseño mediante diagramas UML, tanto estáticos (diagrama de clases) como dinámicos (diagramas de secuencia). Nuestro planteamiento para los talleres fue proponer un trabajo que debería resolverse mediante patrones de diseño y diagramas UML, los alumnos entregaban los trabajos y, posteriormente, el profesor publicaba una guía detallada de los términos que debería tener una práctica para ser considerada correcta. La guía de corrección procuraba entrar en la forma (los gráficos son correctos, son legibles, etc.) como en el fondo (se han creado las clases adecuadas, se ha realizado una estructura de herencia en un sitio determinado, etc.)

Existen dos opciones de evaluación de los talleres, la autoevaluación en la que el alumno analiza su propio trabajo y la co-evaluación, en el que el alumno evalúa el trabajo de los compañeros (elegidos aleatoriamente por la propia herramienta Moodle). La autoevaluación es preferida para los primeros trabajos (para que los alumnos se familiaricen con el mecanismo) y la co-evaluación para trabajos posteriores. En este último caso, es necesario abrir un periodo de revisión para que el profesor revise las evaluaciones de aquellas personas que se sientan injustamente evaluados y el profesor tiene la posibilidad de evaluar las evaluaciones de los alumnos.

4.3. Resultados obtenidos

Debido a problemas técnicos con la plataforma Moodle este año fue imposible realizar talleres en la asignatura de POO. De todas formas comentaremos los resultados basándonos en las experiencias de años anteriores en esta y otras asignaturas.

Como conclusión general sobre estas pruebas podemos decir que los alumnos desarrollan el hábito de la reflexión, y la identificación de los propios errores, cuestión fundamental cuando se trata de formar personas con capacidad para aprender de for-

N.	Criterio (Valorar como B, R+, R, R- o M)	Valor
	Diagrama de clases	
1	¿Es legible el gráfico?	
2	¿Es correcta la notación?	
3	¿Responde fielmente al código?	
4	¿Se muestran los métodos de forma detallada?	
5	¿Se muestran los atributos de forma detallada?	
6	¿Se muestran las asociaciones de forma detallada incluyendo: nombres de rol, multiplicidades, navegabilidad, visibilidad, etc.?	
7	¿Se muestran relaciones de dependencia?	
	Diagrama de secuencia	
8	¿Es legible el gráfico?	
9	¿Es correcta la notación?	
10	¿Responde fielmente al código?	
11	¿Se utilizan fragmentos? ¿De forma correcta?	
12	¿Se incluye el evento que dispara la secuencia?	

Figura 2: Tabla de evaluación para la realización de un modelo de dominio

ma autónoma. Además, en el caso concreto de la co-evaluación podemos decir que los alumnos se esfuerzan más, impulsados por la motivación de quedar bien ante los ojos de sus compañeros y desarrollan el hábito de criticar de forma constructiva el trabajo realizado por compañeros con los que van a tener que continuar colaborando. Ésta es también una habilidad fundamental que se echa en falta con frecuencia en el mundo profesional.

La principal queja de los alumnos es que muchas veces se muestran inseguros a la hora de aplicar los criterios de corrección, para ello es necesario que los criterios creados por el profesor sean claros y detallados y no den lugar a ambigüedades. Por ello es necesario o conveniente echar mano de escalas de calificación prediseñadas (lo que en inglés se conoce como *rubrics* [1] y que se suele traducir no demasiado correctamente como rúbricas en castellano).

En la Figura 2 se muestra un ejemplo de tabla de evaluación en un ejercicio que consistía en hacer ingeniería inversa y, dado un código en Java, obtener el diagrama de clases y el diagrama de secuencia UML que representara fielmente dicho código. Como vemos las preguntas pretenden ser sencillas y ser fáciles de evaluar. Los posibles valores de contestación son (B)ien, (R)egular y (M)al, a los que se añaden las categorías intermedias de (R-) y (R+). En caso de que fuera necesario puede detallarse más la tabla especificando cuándo hay que dar una valoración u otra (por ejemplo, asignar un R en la pregunta 6 si las asociaciones muestran nombres de rol y multiplicidades pero no la navegabilidad o la visibilidad).

5. Foros de preguntas y respuestas

5.1. Planteamiento

Un foro de preguntas y respuestas es un recurso docente existente en la herramienta Moodle. Es básicamente un foro normal en el que el docente incluye una entrada en la que plantea un problema, los alumnos deben dar la solución a ese problema como una respuesta a dicha entrada en el foro. Lo novedoso del sistema es que los alumnos sólo podrán ver las respuestas de sus compañeros cuando hayan enviado las suyas propias, de esa forma podrán comprobar de inmediato si su respuesta coincide con la mayoría de la clase y, posteriormente, podrán comprobar si su respuesta coincide con la ofrecida por el profesor.

5.2. Evaluación

Realmente un foro de este tipo es más bien un cuestionario de respuesta abierta con la peculiaridad de poder ver las respuestas de los compañeros una vez enviada la propia. En la asignatura de POO han sido utilizados para plantear lo que denominamos *puzles* de programación [2], es decir, código que parece que hace una cosa cuando realmente hace la contraria. Suelen ser muy útiles para aprender de forma práctica los aspectos más complejos del lenguaje de programación usado.

En la Figura 3 podemos ver un ejemplo de puzle Java típicamente usado en los foros de preguntas y respuestas. Se trata de adivinar de antemano cuál va a ser el resultado del software que se muestra a continuación. El alumno puede sentirse tentado de copiar el software y ejecutarlo para ver la respuesta correcta, pero se indica claramente que además de dar la respuesta correcta hay que explicarlo convenientemente. La mayoría de las respuestas a este ejercicio indicaban que los Strings eran IGUALES porque la llamada a `toUpperCase` no altera al String `s1`. Sin embargo se olvidaban de mencionar de que el operador == realiza una comparación de identidad, por lo que las cadenas de texto no sólo son iguales sino que son el mismo objeto. Este ejercicio sirve para aprender cómo Java utiliza constantes para representar cadenas de texto y por su naturaleza complicada es más útil en un ejercicio de este tipo que en un examen tipo test.

Debido a este carácter de análisis de aspectos complejos en los foros de preguntas y respuestas se

```
public class Clase
{
  public static void main(String argv[])
  {
    String s1 = "abcde";
    String s2 = "abcde";
    s1.toUpperCase();
    if (s1 == s2)
      System.out.println("IGUALES");
    else
      System.out.println("DISTINTOS");
  }
}
```

Figura 3: Ejemplo de puzle Java

premia más la participación que otra cosa, teniendo en cuenta de todas formas a quién da la respuesta correcta a la pregunta planteada. Para forzar el trabajo continuo los foros se escalan de forma temporal relacionándolos con los temas vistos en teoría.

5.3. Resultados obtenidos

Este tipo de foros son bien aceptados por los estudiantes ya que muchas veces tienen un componente lúdico nada desdeñable. De todas formas, sí se ha observado que cuando un alumno tiene la respuesta correcta a un problema sus compañeros pueden copiarla para introducirla en el foro. Este comportamiento es distinto al observado en los tests de autoevaluación, en los cuales al ser conscientes de que el examen de teoría también tiene formato test, prefieren no copiar las respuestas y autoevaluar su capacidad de hacer correctamente una prueba de este tipo. Para paliar este problema se trató de poner plazos de entrega cortos, de forma que una vez la solución correcta era conocida por varios alumnos se limitara la posibilidad de que ésta solución se *propagara* por el resto de la clase. Además, también se indicó que, como los tests de autoevaluación, los problemas propuestos son similares a los que aparecen en el examen.

6. Empleo de pruebas unitarias y herramientas de cobertura

6.1. Planteamiento

De todas las pruebas de evaluación continua y autoevaluación la que plantea más problemas a la hora de su realización es aquella que involucra al software. Si bien habíamos visto como los tests, talleres o foros podían orientarse hacia analizar programas o desarrollar diagramas de UML basados en dichos programas también es cierto que una asignatura de programación tiene que involucrar, indiscutiblemente, la necesidad de programar ante un compilador programas sencillos pero que sean un fiel reflejo de la programación que el alumno tendrá que hacer en su trabajo habitual una vez termine sus estudios.

En la asignatura de POO las prácticas se dividen en dos partes: boletines de ejercicios y práctica de diseño. Los boletines de ejercicios se plantean como problemas breves para analizar partes concretas de la orientación a objetos (herencia, polimorfismo, genericidad, etc.). La práctica de diseño se orienta a hacer un miniproyecto con su documentación incluida, centrado en el diseño, y que les sirva a los alumnos para obtener habilidades que luego deberán demostrar en su proyecto fin de carrera. Como la práctica de diseño es el resultado final de las prácticas de la asignatura centraremos nuestros esfuerzos sobre la evaluación continua en los boletines de ejercicios.

6.2. Evaluación

El problema de plantear boletines de ejercicios en un esquema de evaluación continua es la carga de trabajo que supone para el profesor poder dar la retroalimentación necesaria al alumno sobre si está haciendo el ejercicio de la forma correcta. Por ello en POO intentamos darle la vuelta a la situación y que sean los propios alumnos los que demuestren que su ejercicio es correcto, es decir, todo ejercicio incluido en el boletín debe incluir pruebas de unidad que utilicen el framework JUnit [7] y que demuestren, en la medida de lo posible, que el código desarrollado cumpla las especificaciones.

Como el hecho de pasar las pruebas de unidad no implica necesariamente que el código sea correcto se le añade un requisito a mayores que es presentar un documento de Cobertura al estilo de los que genera la herramienta *Cobertura* [4]. Esta herramienta muestra de forma gráfica cuántas líneas del código original y ramas condicionales han sido cubiertas por los tests. De esta forma se garantiza no solo que se hayan realizado pruebas sino que los casos de prueba introducidos tengan las condiciones de cantidad y representatividad requeridas (los resultados de cobertura debían ser superiores al 80 %, prestando especial atención a la cobertura de ramas).

```
@Test
public void testEsPalindromo()
{
  System.out.println("TestEsPalindromo\n");
  Palindromo instance = new Palindromo();
  assertTrue(instance.esPalindromo("Radar"));
  assertFalse(instance.esPalindromo("Radares"));
  assertTrue(instance.esPalindromo("3n3"));
  assertFalse(instance.esPalindromo("hola"));
}
```

Figura 4: Prueba para el método esPalindromo

Package	# Classes	Line Coverage		Branch Coverage	
All Packages	16	97%	389/399	91%	143/156
ejercicio1	1	93%	15/16	90%	9/10
ejercicio2	1	100%	16/16	100%	10/10
ejercicio3	5	93%	30/32	100%	4/4
ejercicio4	1	100%	32/32	100%	12/12
ejercicio5	5	95%	116/121	86%	50/58
ejercicio6	4	100%	30/30	100%	11/11
ejercicio7	4	98%	89/90	93%	29/31
ejercicio8	3	98%	61/62	90%	18/20

Figura 5: Valores de cobertura

En este contexto el alumno puede comprobar de antemano que el software que está realizando es correcto quedando como trabajo del profesor el comprobar que las pruebas se realizan de forma correcta y que el ejercicio se ha desarrollado siguiendo las directrices propias de la orientación a objetos.

Como ejemplo podemos ver en la Figura 4 el código correspondiente a una prueba de unidad para comprobar el funcionamiento correcto de un método que determina si una cadena de caracteres es un palíndromo o no. Analizando la información de cobertura de todos los ejercicios (Figura 5) podemos ver que los valores son muy altos, sin embargo, en el ejercicio del palíndromo (ejercicio 1) a pesar de ser un código muy sencillo no se llega al 100 % de cobertura. Esto puede ser debido a que el juego de prueba es incompleto o que, como es este caso, una condición innecesaria se ha incluido en el código y los tests de unidad no han tenido la necesidad de pasar por dicha condición tal y como se puede ver en el código original (Figura 6). También ocurre que, en ocasiones, el código que queda sin comprobar es código *demasiado simple para ser erróneo*, por ejemplo métodos simples de lectura o escritura de atributos.

```
16      public boolean esPalindromo(String s)
17      {
18  4     String p1 = s.toLowerCase();
19  4     String p2 = reverse(p1);
20  4     int i = 0;
21
22  4     if (p1.length() == p2.length())
23        {
24          while (i<p1.length()
25 12         && p1.substring(i, i+1).equalsIgnoreCase(
26              p2.substring(i, i+1)))
27  8           i++;
28  4       if (i==p1.length()) return true;
29  2         else return false;
30        }
31  0     else return false;
32      }
```

Figura 6: Método esPalindromo con las líneas no recorridas por las pruebas marcadas en rojo

6.3. Resultados obtenidos

Si bien es cierto que el hecho de usar pruebas de unidad y herramientas como JUnit o Cobertura añade una complejidad extra a los boletines de ejercicios, también es cierto que es una buena labor de aprendizaje enseñar a los alumnos a responsabilizarse de su propio trabajo. Por lo general los alumnos han recibido positivamente la medida aunque se han dado casos evidentes de intentar simplificar su complejidad haciendo casos de prueba deficientes o simplemente engañosos.

Para su implementación exitosa consideramos necesario, por un lado, dar instrucciones detalladas de cómo deben realizarse y entregarse los ejercicios para no dar lugar a equívocos. Por ejemplo, ejecutar los tests y obtener los valores de cobertura debe ser algo sencillo de realizar para el profesor, por lo que los alumnos tienen que adherirse a las estrictas especificaciones de los ejercicios. Por otro lado, el hecho de tener un foro para responder dudas, y en el que se invita (y premia) a los alumnos a responder las dudas de los compañeros evita, en gran medida, estar resolviendo recurrentemente los mismos problemas a alumnos diferentes.

También es cierto que un test correcto con una cobertura adecuada no garantiza al cien por cien que el código desarrollado sea correcto, pero es un buen indicativo del mismo. Aunque en los últimos tiempos se han publicado propuestas para la evaluación automática de prácticas de programación [9] estas propuestas no soy de fácil aplicación a la asignatura de POO en donde normalmente no existe una *solución correcta* al ejercicio planteado contra la que se pueda hacer una corrección automática, sino que

existen muchos diseños que podrían ser tenidos en cuenta como soluciones correctas.

7. Conclusión

Como conclusión final de lo expuesto en este trabajo podemos decir que la evaluación continua es un experiencia positiva en el campo del aprendizaje, los alumnos comprueban que se valora su trabajo a lo largo de todo el curso y no solo en una prueba puntual en la que pueden fallar por diversas razones y que puede no ser representativa de lo que han aprendido. Sin embargo, la evaluación continua requiere un mayor esfuerzo, tanto por parte de los alumnos como por parte de los docentes. Este esfuerzo puede ser reducido si nos basamos en la utilización de herramientas de enseñanza virtual como puede ser la plataforma Moodle. A través de las evaluaciones automáticas de los tests, los foros de preguntas y respuestas, los foros de participación del alumnado, etc. la carga que exige al profesor el nuevo sistema de evaluación puede ser reducida, y al alumno se le facilita el seguimiento de las distintas tareas que tienen que llevar a cabo durante el curso.

Por otro lado, la autoevaluación y/o la coevaluación permite no solo abandonar un aprendizaje más basado en las técnicas memorísticas sino que permite que los alumnos interioricen y examinen los criterios del profesor y adopten su rol al evaluar sus prácticas o las de sus compañeros. Eso hace que los errores sirvan para aprender y no sólo sirvan para bajar la calificación final en la asignatura. Además, al verse descargado de las tareas más rutinarias de corrección el docente puede fijar sus esfuerzos en otros ámbitos de la docencia (como pueden ser la tutorías personalizadas).

Para comprobar el impacto de las nuevas técnicas de evaluación en la asignatura de POO en la Figura 7 y en la Figura 8 se pueden ver las estadísticas de las calificaciones en este curso y el anterior. En general puede verse una tendencia a la mejora de las notas (descienden los suspensos y aumentan los notables) que viene a demostrar que cuando se hace trabajar a los alumnos a lo largo del curso estos no se lo juegan todo a una baza (el examen) y en general las notas mejoran. De todas formas también existen otros factores que han influido en la mejora de las notas: el número de los alumnos a disminuido, por lo que la atención ha sido más personalizada; se ha reducido

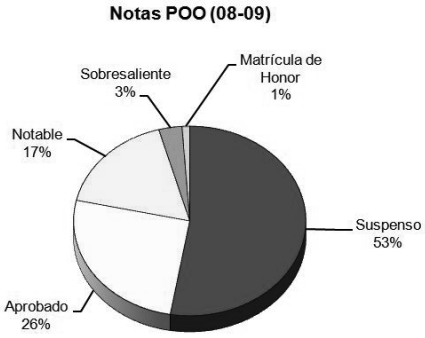

Figura 7: Estadística POO (08-09)

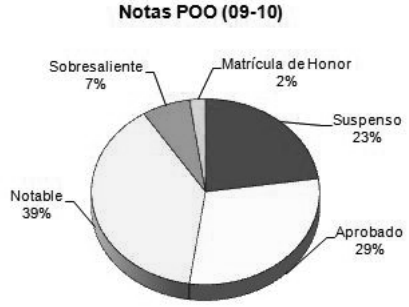

Figura 8: Estadística provisional POO (09-10)

la carga de trabajo de los boletines de ejercicios permitiendo que se realizaran por grupos de 2 alumnos (sugerencia recogida de los alumnos del año anterior), etc.

Este último hecho, el realizar grupos de dos personas en los boletines, ha sido motivado también por una de las mayores preocupaciones de la estrategia seguida en la asignatura, la posibilidad de que los alumnos se copien entre sí, algo desgraciadamente demasiado habitual en las titulaciones de informática y para lo que no existe una solución sencilla [3]. A este problema se le pueden poner parches a nivel personal pero que mientras no haya una política clara que penalice la *deshonestidad académica* al estilo de las universidades estadounidenses, será algo contra lo que se tendrá que seguir luchando.

Referencias

[1] Andrade, H.G., *Using rubrics to promote thinking and learning*, 57 (5), pp. 13–19, 2000

[2] Bloch, J. and Gafter, N. *Java Puzzlers: Traps, Pitfalls, and Corner Cases*, Addison-Wesley Professional, 2005

[3] P.J. Clemente, A. Gómez, J. González *La copia de prácticas de programación: el problema y su detección*. Congreso JENUI-2004. `http://www.dccia.ua.es/jenui2004/actas/ponencias/ponencia23.pdf`

[4] *Cobertura: Java code coverage analysis tool*, `http://cobertura.sourceforge.net/`

[5] Delgado, A. M. y Oliver, R., *La evaluación continua en un nuevo escenario docente*, Revista de Universidad y Sociedad del Conocimiento, 3 (1), 2006.

[6] García Beltrán, A. et al. *La autoevaluación como actividad docente en entornos virtuales de aprendizaje/enseñanza*. RED. Revista de Educación a Distancia, no. M6, 2006. Consultado (08/01/2010) en `http://www.um.es/ead/red/M6`

[7] *JUnit.org Resources for Test Driven Development*, `http://www.junit.org`

[8] *Moodle.org: open-source community-based tools for learning*, `http://www.moodle.org`

[9] Rodríguez del Pino, J. C. Díaz Roca, M., Hernández Figueroa, Z., González Domínguez, J. C. *Hacia la Evaluación Continua Automática de Prácticas de Programación*. Congreso JENUI-2007 `http://bioinfo.uib.es/~joemiro/aenui/procJenui/Jen2007/rohaci.pdf`

[10] Valero-García, M., y Díaz de Cerio, L. M., *Autoevaluación y co-evaluación: estrategias para facilitar la evaluación continuada*, CEDI 2005, `http://epsc.upc.edu/projectes/usuaris/miguel.valero/materiales/docencia/articulos/SINDI2005.pdf`