Uso de sistemas de control de versiones en el seguimiento continuo del trabajo del alumno

Miguel Riesco Albizu, M. Ángeles Díaz Fondón
Departamento de Informática
Universidad de Oviedo
c/ Calvo Sotelo, s/n 33007 Oviedo
{albizu, fondon}@uniovi.es

Resumen

El seguimiento continuo del trabajo del alumno suele ser complicado cuando se trata prácticas de desarrollo de software. Además de ser tedioso y de difícil realización, puede ser una tarea considerablemente larga cuando el número de alumnos es elevado.

En este trabajo se propone una combinación de técnicas, basadas en la utilización de un sistema de control de versiones, para hacer posible este seguimiento incluso con un número alto de alumnos.

1. Introducción

La realización de prácticas que implican el desarrollo de software es común en muchas asignaturas de las titulaciones informáticas. Si bien este tipo de prácticas está consolidado, para llevar a cabo su evaluación nos encontrarnos con muchas variantes, pudiéndose realizar en función de muchos productos del trabajo del alumno:

- Evaluar únicamente el resultado, utilizando pruebas "de caja negra" al producto final de la práctica.
- Evaluar el "aspecto" del código final entregado, aplicando criterios de estructuración, normas de codificación, comentarios, etc.
- Evaluar la documentación asociada al desarrollo del software.
- Etc.

Además de lo que se evalúa, una cuestión crucial es cuándo se lleva a cabo dicha evaluación:

- Evaluar únicamente al final de la práctica.
- Establecer puntos de control intermedios para observar cómo va realizando el trabajo el alumno y corregirlo en su caso.

Desde el punto de vista didáctico es mucho más constructiva la segunda opción, dado que el proceso de evaluación se integra dentro del proceso de aprendizaje con un valor formativo, no únicamente evaluador. Además, cuantos más puntos de control haya mejor será el seguimiento que se hace al alumno y más provechoso será para su formación.

Sin embargo, es bien conocido lo aburrido y largo que puede llegar a resultar el corregir trabajos de programación. Si además se cuenta con un número de alumnos elevado, el multiplicar ese trabajo por el número de controles intermedio hace inviable un seguimiento continuo del trabajo del alumno. La consecuencia de esto suele ser que únicamente se evalúa el resultado final y, como mucho, se "mira por encima" algún producto intermedio.

Viéndonos afectados por esta problemática en nuestra asignatura, durante el curso 2009-10 se ha puesto en marcha un sistema que trata de hacer lo más continuo posible el seguimiento del trabajo del alumno sin que ello suponga un costo excesivo para el profesor.

2. Entorno de la experiencia

Esta experiencia se ha realizado en la asignatura de Sistemas Operativos, asignatura de segundo curso de la titulación de Ingeniería Técnica en Informática, tanto de gestión como de sistemas de la Escuela Universitaria de Ingeniería Técnica en Informática de Oviedo.

El número de alumnos matriculados en la asignatura en el curso 2009-10 es de 90, si bien los alumnos que realmente siguen la asignatura son 70.

Dentro de las prácticas de esta asignatura hay tres que se corresponden a desarrollar distintas aplicaciones relacionadas con los sistemas operativos. Una se desarrolla en lenguaje Java mientras

que las dos restantes se implementan en lenguaje C [3].

3. Objetivos

El objetivo principal que se persigue con esta experiencia es lograr un sistema de seguimiento continuo del trabajo del alumno, lo que nos permitiría:

- Realizar una evaluación más acorde con el trabajo real del alumno. Si nos basamos en el resultado final únicamente evaluamos sólo el producto final, no el proceso de realización del trabajo.
- Detectar posibles errores en la realización de la práctica cuanto antes, pudiendo así dar un valor formativo al proceso de evaluación.
- Fomentar un desarrollo racional del trabajo. Si sólo se evalúa el resultado final resulta frecuente encontrar a alumnos que acumulan el trabajo al final del periodo de realización del mismo. Esto provoca que, en ocasiones, el alumno se ve desbordado, dado que un ejercicio que es relativamente sencillo realizado semana a semana se vuelve inabordable cuando se trata de resolver de manera global.
- Detectar comportamientos fraudulentos. En ocasiones nos encontramos con alumnos que durante el desarrollo de la práctica apenas realizan trabajo alguno pero que, a la hora de entregar, aparecen con un código completo y funcional. A veces será fruto de trabajo legítimo del alumno fuera del aula, pero no siempre es así.

Obviamente, una característica fundamental que debe tener la solución que buscamos es que su costo de realización por parte del profesor sea lo más pequeño posible, aun cuando el número de alumnos sea elevado.

4. Modo de trabajo

Para intentar lograr los objetivos antes expuestos, se diseño una estrategia de realización y corrección de las prácticas de acuerdo con los siguientes puntos:

1. Se utilizará un sistema de control de versiones (*Version Control Systems - VCS*), que deberá ser usado por todos los alumnos. En nuestro caso usaremos *Subversion* [1]. Cada alumno tendrá su propia cuenta en el sistema, mientras que el profesor tendrá capacidades de administración del repositorio.
2. Siempre que el alumno realice una modificación sustancial de sus ficheros, deberá actualizarlos en el VCS.
3. Se establecerán controles intermedios en la realización de la práctica. Habrá un control por cada semana de clase.
4. Al final de la práctica habrá un punto de control final.
5. Todos los ejercicios a realizar tendrán claramente definidas las entradas que debe aceptar el programa y las salidas que debe generar en cada caso.

5. Modo de evaluación

Dado que se trata de la asignatura de Sistemas Operativos y al no ser ésta una asignatura específica de programación, se decidió que se evaluaría principalmente el funcionamiento de las aplicaciones, sin entrar a valorar otros aspectos del código o de documentación adicional.

Teniendo en cuenta la forma de trabajo descrita, el profesor tiene en todo momento una copia de todo el trabajo que ha ido haciendo cada alumno.

Así, en la fecha prevista para cada punto de control puede pasar a evaluar los programas de los alumnos. Los siguientes puntos describen las pruebas que se realizarán.

Como se observará, las tres primeras pruebas son automatizables por completo, con lo que el grueso de la evaluación se llevará a cabo con un mínimo trabajo para el profesor. La última prueba sí necesita intervención directa del profesor, pero en nuestro caso es muy corta e incluso puede limitarse únicamente a algunos alumnos.

5.1. Pruebas de caja negra

Dado que los ejercicios tienen bien definidos las entradas que aceptan y las salidas que producen es posible llevar a cabo pruebas de caja negra con ellos.

Mediante un conjunto de *scripts* se llevarán a cabo este tipo de pruebas para todos los alumnos, viéndose así los que cumplen con la especificación dada y los que no lo hacen.

5.2. Pruebas de originalidad

En cada punto de control se llevará a cabo una prueba para intentar detectar plagios por medio de la herramienta JPlag [2], para verificar que el trabajo del alumno es original.

Es muy frecuente que los alumnos que acuden a la misma academia presenten programas muy similares. Utilizando este tipo de herramientas desde el primer punto de control se puede advertir a los implicados para que, independientemente del trabajo que hagan en la academia, realicen personalmente su práctica.

Además de este tipo de comportamientos, ocasionalmente nos encontramos con alumnos que copian el trabajo de sus compañeros (con o sin su autorización), plagios de prácticas de otros años, etc. Estas pruebas servirán para detectarlos cuanto antes y obrar en consecuencia.

5.3. Pruebas de "continuidad"

Al disponer del histórico de trabajo de los alumnos se puede evaluar el número de versiones realizadas y la diferencia entre las mismas, pudiendo así hacernos una idea bastante aproximada del modo de trabajar del alumno.

De esta manera podemos detectar comportamientos anómalos (por ejemplo, que en una sola sesión de trabajo aparezca ya el ejercicio completo y correcto), permitiéndonos corregir ese tipo de comportamientos desde ese momento sin esperar al final de la práctica.

Para llevar a cabo este seguimiento se han desarrollado una serie de programas que examinan las distintas versiones que el alumno ha ido almacenando y calculan una serie de datos cuantitativos para cada versión (número de líneas, número de líneas nuevas, modificadas y eliminadas, número de comentarios, tiempo transcurrido entre cada versión, ...).

5.4. Código

Al disponer el profesor de una manera sencilla de la versión con la que está trabajando el alumno en todo momento puede, si se considera necesario, inspeccionar el código de cualquiera de las versiones de los programas de cualquier alumno.

En general en nuestra asignatura no se suele hacer más que una ligera revisión del "aspecto" general del código, en cuanto a estructuración, estructuras de datos utilizadas, comentarios, etc.

En otras asignaturas podrían realizarse otros estudios con el código de los ejercicios (o con el de las distintas versiones) si así se considerara oportuno.

6. Justificación del modo de trabajo

La idea fundamental de este modo de trabajo es el disponer del mayor número de versiones de cada programa (sean funcionales o no) para poder realizar así el seguimiento continuo del trabajo del alumno.

Se pensó el utilizar un sistema de control de versiones, frente a otras posibilidades como entregas periódicas a través de herramientas de campus virtual o similar, porque es mucho más cómodo y natural para el alumno trabajar de este modo: a medida que va editando el programa y antes de compilarlo puede, ejecutando una simple instrucción, almacenar una nueva versión en el sistema.

También es ventajoso para el profesor. En todo momento puede ver cómo va evolucionando el trabajo de los alumnos, pudiendo intervenir en caso necesario emitiendo avisos, recordatorios, etc.

Asimismo, el profesor también puede ir controlando el trabajo de los alumnos antes de las entregas pactadas, lo que puede permitirle descubrir errores generalizados que puede corregir a tiempo.

7. Ventajas adicionales

Además de las señaladas, la utilización de un sistema como el indicado ofrece otra serie de ventajas, tanto para los alumnos como para el profesor:

- Los alumnos conocen y trabajan con un sistema de control de versiones, cosa que habitualmente no hacen y que es deseable que utilicen [4] .
- El sistema mantiene una copia de seguridad de todo el código desarrollado, evitando así problemas de pérdidas o averías en los equipos del alumno.
- El profesor tiene disponible en todo momento el estado de desarrollo de las prácticas de todos los alumnos, con lo que puede responder mejor a las dudas o problemas que pueda presentar un alumno en su práctica, sin necesidad, además, de que este le facilite el código.

- El profesor puede estudiar los hábitos de trabajo de cada alumno de manera individual o de todos los alumnos de manera colectiva. Estos estudios pueden ser interesantes para muy diversos fines (planificación de la asignatura, recomendaciones a los alumnos, etc).

8. SVN y su adecuación a la experiencia

Subversion (*SVN*) es un sistema de control de versiones de código abierto y de uso muy difundido. De entre las ventajas que aporta su uso a nosotros básicamente nos interesaba su utilidad para almacenar en un servidor central las distintas revisiones de los programas que los alumnos iban realizando, pudiendo el profesor recuperar cualquiera de ellas en cualquier momento.

Desde este punto de vista el sistema ha cumplido perfectamente con su cometido, pudiendo el profesor consultar cuántas versiones había desarrollado cada alumno, cuándo las había almacenado, qué contenía cada una, etc.

El inconveniente más notorio que se ha encontrado es que no facilita la extracción de información del repositorio para realizar los estudios mencionados anteriormente. Se han tenido que realizar multitud de pequeños programas para hacerlo. No obstante, para años sucesivos pueden desarrollarse programas más elaborados que automaticen todas estas tareas.

En este mismo sentido, la mayor parte del trabajo del profesor se ha centrado en la realización de estos programas. Una vez realizados estos, la corrección de cada una de las entregas ha supuesto muy poco trabajo para el profesor.

9. Conclusiones

Se ha mostrado en este trabajo un método para intentar llevar un seguimiento continuo del trabajo del alumno cuando este consiste en el desarrollo de software. Este método permite llevar a cabo el seguimiento incluso de grupos numerosos utilizando un tiempo razonable.

La utilización de una herramienta de control de versiones nos facilita estudiar la manera en que el alumno ha ido desarrollando su trabajo, además de poder realizar otro tipo de pruebas (pruebas de caja negra, pruebas de originalidad, examen manual del código) sobre cualquier fase del trabajo del alumno y sin necesidad de formalizar ninguna entrega por otros medios.

Esta experiencia se ha puesto en marcha este curso y todavía no disponemos de ninguna conclusión sobre su repercusión en el proceso de enseñanza-aprendizaje.

Referencias

[1] Collins-Sussman, B.; Fitzpatrick, B. W.; Pilato, C.M. Version *Control with Subversion: Next Generation Open Source Version Control*. Ed. O'Reilly Media, Junio 2004. Disponible en http://svnbook.red-bean.com/ (última consulta:8/2/10)

[2] *JPlag Home Page*. https://www.ipd.uni-karlsruhe.de/jplag/ (última consulta: 8/2/10)

[3] Riesco Albizu, Miguel; Díaz Fondón, M. Ángeles. *Propuesta de contenido de prácticas en la materia de Sistemas Operativos*. X Jornadas de Enseñanza Universitaria de la Informática (JENUI 04). Actas de las Jornadas. Ed. Thomson,. Alicante, Julio 2004

[4] Ruiz-Bertol, F. J.; Zarazaga-Soria, F. J. *El control de versiones en el aprendizaje de la ingeniería informática: Un enfoque práctico*. XIII Jornadas de Enseñanza Universitaria de Informática (JENUI 2007). Actas de las Jornadas. Ed. Thomson. Teruel, Julio 2007.