Creación de un servidor de integración continua para gestión y corrección de entregas de prácticas*

Jesús Aransay, Francisco García, Jónathan Heras, Adrián Inés, Gadea Mata
Departamento de Matemáticas y Computación
Facultad de Ciencia y Tecnología
Universidad de La Rioja
jesus-maria.aransay@unirioja.es,
francisco.garcia@unirioja.es,
jonathan.heras@unirioja.es,
adrian.ines@unirioja.es, gadea.mata@unirioja.es

Resumen

En este trabajo se presenta un servidor web público (`http://iscima.unirioja.es/`) desarrollado para permitir, tanto a estudiantes como a profesores, comprobar que los envíos de prácticas de programación cumplen una serie de requisitos mínimos (en cuanto a estructura de directorios, compilación y comprobación de tests). Para completar el trabajo se muestra un caso de uso de aplicación de la herramienta en una asignatura y las conclusiones de su uso. Además del servidor, que automatiza y reduce el trabajo de los profesores, también se ha aprovechado su utilización para inculcar a los estudiantes conceptos y metodologías propias de Ingeniería del Software como la entrega continua y el uso de tests. En este trabajo se presenta dicha herramienta informática, sus soluciones tecnológicas, y una valoración de los cambios detectados en el aprendizaje de los estudiantes por el uso de la misma.

Abstract

In this work, we present a web application (`http://iscima.unirioja.es/`) devoted to allow both students and teachers to check that lab submissions satisfy a set of minimal requirements (related to structure, compilation and satisfiability of tests). Moreover, we present a case study where the tool is applied to a programming module and the conclusions obtained from its use. The web application is not only used to automatize and reduce the teachers' work, but it also serves to introduce concepts about continuous integration and testing to the students. In this paper, we introduce the web application, its technological solutions, and an evaluation of the tool based on the students' improvements.

Palabras clave

Prácticas de programación, Automatización de entregas, Entrega continua, Testing.

1. Motivación

En este trabajo se presenta una experiencia llevada a cabo en el presente curso, curso 17/18, en una asignatura de Programación Orientada a Objetos (POO) — esta asignatura se imparte de manera conjunta en el segundo curso del Grado en Ingeniería Informática y en el Grado en Matemáticas de la Universidad de La Rioja, y es la tercera de seis asignaturas (cuatro en el Grado en Matemáticas) dedicadas a la programación. En la misma se introduce a los estudiantes en los conceptos de POO usando como lenguajes para las prácticas de la asignatura Java y C++. Es importante reseñar que la asignatura supone también el primer contacto de los estudiantes con Java (C++ es usado, de manera procedural, en el primer curso). A lo largo de los años, la asignatura ha constado de 6 créditos ECTS, de los cuales 3,2 corresponden a clases de grupo grande y otros 2,8 a grupos de laboratorio. Los 2,8 créditos de grupos de laboratorio se articulan por medio de 14 sesiones (prácticas) de grupo informático en las que los estudiantes, por lo general, deben programar en C++ y Java diversos casos de uso de ocultación de la información, herencia, polimorfismo, clases abstractas, interfaces, tipos paramétricos, excepciones, etc. El planteamiento de las prácticas suele consistir en facilitar a los estudiantes uno o varios diagramas UML, junto a

*Trabajo parcialmente financiado por el Vicerrectorado de Profesorado de la Universidad de La Rioja a través de un proyecto de innovación docente, y por el Plan de financiación de los grupos de investigación de la Universiad de La Rioja [REGI2018/52].

una especificación informal de los métodos a programar (el diseño de clases y métodos se trata en asignaturas posteriores, como Especificación y Desarrollo de Sistemas Software). La especificación facilitada a los estudiantes facilita (y se presta) al uso de tests unitarios para validar la corrección de las prácticas. De esta forma, los estudiantes completan 11 prácticas en la asignatura; cada una consta de dos o tres ejemplos, y casi todas ellas deben ser programadas tanto en Java como en C++.

El número de estudiantes en la asignatura ha sido variable a lo largo de los años, pero siempre ha estado por encima de los 50 estudiantes (llegando en algunos cursos, como el actual, hasta los 93). El volumen de trabajo que requiere la gestión y la evaluación de todo el trabajo completado en las sesiones de grupo informático por los estudiantes es costoso, a pesar de que el trabajo en algunos cursos ha sido compartido entre dos o tres profesores.

Por otra parte, el desarrollo de software en los últimos tiempos, así como la necesidad de actualizar código en uso de manera continua, ha dado lugar a la introducción y visibilización de ciertas técnicas como el testing o la entrega y despliegue continuo del software. Si bien en algunos casos se pueden llegar a estudiar en Ingeniería del Software, a día de hoy no suelen formar parte de la "caja de herramientas" de los estudiantes de informática en sus primeros cursos.

Por tanto, la motivación de este trabajo es, al menos, doble. En primer lugar, facilitar a estudiantes y profesores una herramienta que les simplifique el proceso de comprobación de algunos requisitos que las entregas de prácticas deben satisfacer. En segundo lugar, mejorar las competencias de los estudiantes en algunos campos de Ingeniería del Software, por medio del uso de herramientas y técnicas propias del área.

2. Objetivos

Los objetivos que se persiguen en este trabajo son:

- Familiarizar a los estudiantes con las técnicas de programación con tests y de integración continua.
- Crear una herramienta que facilite a estudiantes y profesores la comprobación de la corrección de entregas de estudiantes.
- Facilitar a los estudiantes una herramienta que les permita verificar que sus entregas cumplen ciertos criterios de corrección.
- Aumentar la calidad de las entregas de los estudiantes.

Estos objetivos, y los pasos que se han dado para completarlos, se presentan a lo largo de las siguientes secciones.

3. Programación con tests

Los tests, también llamados pruebas, son imprescindibles para controlar la calidad del código de una aplicación. Cuando se utilizan tests, se pueden añadir, modificar o eliminar partes de la aplicación con la certeza de saber que si se rompe algo, nos daremos cuenta al instante gracias a los tests. Aunque existen diversos tipos de tests (entre los que destacan los unitarios, funcionales, de integración y de aceptación), en el contexto que nos ocupa nos centramos en los tests unitarios ya que permiten comprobar si funciones y clases individuales tienen el comportamiento esperado. En este tipo de tests, el creador de los mismos implementa para cada función o método a validar un conjunto de casos de prueba. En estos casos de prueba se proporcionan un conjunto de aserciones que especifican la salida esperada ante unas entradas dadas. Si al ejecutar los tests no se obtienen los resultados esperados se dice que el test falla o no pasa, y por lo tanto se considera que el código probado es incorrecto.

Para trabajar con tests (ya sean unitarios, funcionales, o de cualquier otro tipo), se utilizan frameworks de testing, que proporcionan la funcionalidad necesaria para crear y ejecutar los tests. La mayoría de los frameworks de testing siguen la convención xUnit [16], y entre ellos destacamos, por ser los utilizados en la asignatura de POO, JUnit [25] para Java, y Catch [19] para C++ (también existen frameworks de esta familia para otros lenguajes como PHPUnit para PHP o unittest para Python).

El uso de tests puede llevarse al extremo y ser la base del desarrollo software. En concreto, a partir del movimiento de la programación ágil [22] surgieron un número de prácticas para garantizar la producción de código de calidad, entre ellas el desarrollo guiado por tests (*Test Driven Development*, o *TDD*): una aproximación al desarrollo software en la cual se escriben los tests antes que el código de la aplicación [4]. El objetivo esencial del TDD es diseñar y especificar los comportamientos de un sistema con respecto a las expectativas del cliente, y además ser capaz de probar de manera constante que esos comportamientos se satisfacen mientras el producto viva.

Esta aproximación al desarrollo software ha sido la aplicada durante el curso 17/18 en las prácticas de la asignatura POO. Para cada práctica, el profesor facilita a los estudiantes el enunciado tradicional, que proporciona la especificación de los ejercicios a realizar, junto a una batería de tests que los programas de los estudiantes deben pasar. Estos tests permiten comprobar a los estudiantes, sin la intervención explícita del profesor, si van por el buen camino recibiendo un *feedback* inmediato; y además, les obliga a considerar casos problemáticos de sus programas en los cuales puede que

no hubieran pensado inicialmente.

El uso de tests no es solo beneficioso para los estudiantes, sino que simplifica la tarea de evaluación del profesor. Los tests permiten centrarse en evaluar aspectos de más alto nivel, como el diseño de los programas o la adopción de soluciones eficientes, ya que los tests permiten verificar que la solución proporcionada por los estudiantes satisface una gran cantidad de requisitos que son tediosos de comprobar manualmente.

4. Creación de una herramienta de comprobación de entregas

La entrega de las 11 prácticas de POO se divide en tres bloques. Para cada uno de estos bloques se pide a los estudiantes que entreguen un fichero comprimido que contenga el código fuente de sus prácticas siguiendo una estructura de ficheros prefijada. Al proporcionar una estructura de ficheros se pretende conseguir que los estudiantes aprendan a discriminar la parte relevante de sus prácticas (los ficheros de código fuente) de todo lo que es superficial o reproducible por el profesor, como ejecutables o ficheros de configuración propios del entorno de desarrollo (IDE) utilizado durante las prácticas; y, además, se facilita la corrección de las prácticas por parte del profesorado.

Sin embargo, los estudiantes encontraban dificultades a la hora de realizar los envíos de este modo. Por un lado, en un gran porcentaje de las entregas (ver Sección 7) no se satisfacía la estructura de ficheros solicitada, los estudiantes se olvidaban de ficheros o no seguían la organización de carpetas pedida. Por otro lado, los estudiantes, acostumbrados a trabajar utilizando un IDE, no eran capaces de comprobar si su código compilaba y funcionaba de la manera esperada una vez lo habían organizado en carpetas; esto era causa de intranquilidad para algunos estudiantes que temían entregar versiones antiguas de sus prácticas. Es por ello que durante el curso 17/18 se han desarrollado dos herramientas que abordan los problemas anteriores y sirven para validar las entregas de prácticas.

La primera herramienta es una aplicación Java encargada de comprobar si una carpeta contiene la estructura de ficheros solicitada. En caso de que falte algún fichero o carpeta la aplicación avisa al usuario; en caso contrario, informa al estudiante de que la estructura de ficheros es correcta y crea un fichero comprimido listo para enviar. La aplicación es multiplataforma y se configura mediante un fichero de texto que contiene la estructura de ficheros a enviar. Esto permite utilizar la misma aplicación para validar distintas entregas. La aplicación es libre y está disponible en `https://github.com/joheras/CompruebaEstructura`. Esta herramienta ha estado disponible para los estudiantes en las tres entregas del curso 17/18 y, en principio (veremos que esto no es del todo cierto en la Sección 7) sirve para resolver el problema de las entregas que no satisfacían la estructura de ficheros solicitada.

Para dar solución al problema de la intranquilidad de los estudiantes a la hora de realizar sus envíos (con respecto a la compilación del código que entregaban), se ha desarrollado una aplicación web utilizando Java servlets [12], disponible en `http://iscima.unirioja.es/`. A partir de un fichero comprimido, dicha aplicación comprueba (1) que la estructura de ficheros es correcta, (2) que los ficheros compilan, y (3) que el código pasa los tests proporcionados en las prácticas. El resultado final es un informe que indica a los estudiantes si se cumplen las tres condiciones anteriores, y en caso contrario, los problemas encontrados. La implementación del servidor sigue el principio *open/closed* [15]; es decir, la aplicación permite añadir nuevas entregas a validar, sin necesidad de modificar el código fuente, subiendo unos ficheros de configuración al servidor. El servidor web ha estado disponible a partir de la segunda entrega del curso 17/18, permite envíos múltiples y está pensado solo para validar las entregas. Los estudiantes deben realizar la entrega definitiva utilizando la plataforma Blackboard Learn [3] que se usa como herramienta habitual de comunicación de la asignatura.

Además de estas dos herramientas destinadas a los estudiantes, también se ha desarrollado una herramienta pensada para facilitar el trabajo de corrección de los profesores. La plataforma Blackboard Learn permite descargar un fichero comprimido con todas las entregas de todos los estudiantes. Una vez descomprimido dicho fichero se obtiene una carpeta que contiene los ficheros comprimidos enviados. Hacer la gestión manual de todos esos ficheros y la comprobación de su corrección es complicado y costoso, por lo que la misma funcionalidad disponible en el servidor web está disponible de manera local para procesar todos los envíos recibidos. Esto simplifica la tarea del profesorado ya que los informes de prácticas generados por la aplicación permiten comprobar de manera rápida si las entregas cumplen una serie de requisitos mínimos.

5. Uso de la herramienta por parte de los estudiantes

Nos centramos ahora en el uso del servidor web por parte de los estudiantes. El servidor desarrollado estaba guiado por tres principios fundamentales para que cumpliera con su fin.

En primer lugar, *la sencillez de uso*. Si queremos promover que los estudiantes incluyan el servidor en

su "caja de herramientas" el primer requisito que debe satisfacer el mismo es ser fácil de usar. En este caso, el servidor solo requiere del usuario "cargar" un fichero comprimido y el servidor genera automáticamente una URL en la que el estudiante puede encontrar un informe con el resultado de comprobar que su entrega tiene la estructura pedida, compila y pasa los tests facilitados por los profesores. Conviene recordar que en algunas entregas el número de ejemplos a entregar puede ser superior a la veintena (y el de ficheros superior al centenar). Las comprobaciones realizadas no son inmediatas, lo que aconseja que las peticiones sean atendidas de forma asíncrona. Así, tras subir el fichero comprimido, la aplicación facilita al estudiante una URL donde, pasado un breve tiempo (generalmente menos de 30 segundos) puede obtener su informe. Para lograr que la aplicación tenga una apariencia amigable y sea fácil de usar, se ha empleado Bootstrap [23] para desarrollar su "frontend". El uso de Bootstrap tiene la ventaja adicional de hacer que la aplicación sea *responsive*.

En segundo lugar, *la disponibilidad del servicio*. Si bien no se considera que en la herramienta la disponibilidad sea crítica (a día de hoy los estudiantes y profesores lo usamos como herramienta de validación, no de entrega) sí que se considera que debe estar disponible el mayor tiempo posible para incentivar su uso y aumentar la confianza de los usuarios. También es importante recordar que el servidor es público, lo cual hace que pueda sufrir ciertos tipos de ataques. Este objetivo se ha cubierto por medio del uso de balanceadores de carga y del encolado de mensajes (peticiones) de los usuarios. En nuestro caso el balanceador de carga usado ha sido *HAProxy* [26].

El tercer objetivo buscado era la *seguridad*. Los usuarios pueden subir ficheros comprimidos, y si estos ficheros tienen la estructura exigida, el código que contienen los ficheros va a ser compilado y ejecutado junto a los tests en el servidor. En nuestro caso, la forma de conseguir que estas ejecuciones no afectaran al propio servidor es por medio del uso de contenedores Docker [17]. Las limitaciones que imponemos a los contenedores para evitar problemas en el servidor son que los contenedores no se ejecutan con el usuario *root*, que los contenedores se ejecutan con una limitación de tiempo de uso de la CPU (lo cual nos ayuda a evitar también errores desintencionados, como los bucles infinitos) y que los contenedores se ejecutan con una limitación de uso de memoria RAM.

Aparte de las configuraciones aplicadas en el servidor y de las pruebas realizadas por nosotros, por parte de los estudiantes no hemos recibido quejas relevantes sobre el correcto funcionamiento del servidor.

		Bloque 01	
Curso	Estructura	Compilación	Tests/Correctas
16/17	0 % (0/58)	29 % (17/58)	8 % (5/58)
17/18	42 % (36/85)	35 % (30/85)	32 % (27/85)
		Bloque 02	
Curso	Estructura	Compilación	Tests/Correctas
16/17	1 % (1/53)	40 % (21/53)	3 % (2/53)
17/18	81 % (64/79)	52 % (41/79)	54 % (43/79)
		Bloque 03	
Curso	Estructura	Compilación	Tests/Correctas
16/17	47 % (23/49)	90 % (44/49)	20 % (10/49)
17/18	94 % (72/76)	88 % (67/76)	79 % (60/76)

Cuadro 1: Porcentaje de entregas correctas

6. Caso de estudio

En esta sección valoramos la calidad de las entregas en los cursos 16/17 y 17/18. Las prácticas en ambos cursos han sido muy similares, cubriendo los mismos conceptos. La diferencia radica en que durante el curso 17/18 se introdujeron tanto los tests como la aplicación de validación.

A la hora de evaluar la calidad de las entregas hemos considerado tres parámetros: estructura, compilación y corrección. El primer parámetro, la estructura, indica si los estudiantes han realizado la entrega siguiendo la estructura de ficheros pedida, es decir, que no falta ningún fichero, que no se incluyen ficheros adicionales, y que la organización de ficheros en carpetas es la solicitada. El segundo punto, compilación, sirve para medir el porcentaje de prácticas que tiene errores de compilación. Por último, la corrección nos indica si el comportamiento de las entregas es el esperado. Durante el curso 16/17 la corrección se revisaba manualmente, mientras que en el curso 17/18 esto se ha hecho mediante los tests. Los resultados obtenidos con respecto a estos tres parámetros se presentan en el cuadro 1.

Como se puede ver en el cuadro 1, los resultados obtenidos con respecto a los tres parámetros medidos han mejorado o se han mantenido del curso 16/17 al curso 17/18. Si nos centramos en la estructura de ficheros podemos ver que durante el curso 16/17, casi ningún estudiante entregaba los ficheros siguiendo la estructura pedida: 0 % en la primera entrega, 1 % en la segunda, y 47 % en la tercera (la última entrega consta de muchos menos ficheros que las dos primeras, por lo que es razonable que, tras insistirles en la importancia de entregar las prácticas con la estructura adecuada, el porcentaje de entregas con estructura correcta aumentara). Esto ha mejorado considerablemente durante el curso 17/18 (42 % en la primera entrega, 80 % en la segunda y 94 % en la tercera). Se puede ver como la aplicación de escritorio que comprueba la estructura ha ayudado a mejorar esto, ya que desde la primera entre-

ga, el porcentaje de entregas correctas es muy superior al curso anterior. En esta mejora también ha tenido que ver la introducción de la aplicación web, en la cual si no se envía un fichero con la estructura correcta, no es posible comprobar si las prácticas compilan y pasan los tests. Gracias a esto se ha pasado de un 42 % de entregas con la estructura correcta en el primer bloque a un 81 % en el segundo y a un 94 % en el tercero.

La exigencia de los estudiantes respecto a la compilación de sus prácticas no ha decrecido. Esto es debido, posiblemente, a que para ejecutar los tests se necesita que las prácticas compilen, y por lo tanto los estudiantes deben esforzarse en solucionar problemas que antes pasaban por alto. La introducción de la aplicación web también aporta un valor añadido con respecto a la compilación de las prácticas en diversos entornos. En concreto, las prácticas solicitadas son independientes del sistema operativo, y debería ser posible compilarlas en cualquier entorno sin introducir cambios en las mismas. La mayoría de estudiantes trabaja en un entorno Windows, pero la aplicación web está alojada en un servidor Linux, por lo que los estudiantes deben ser cuidadosos con sus entregas para que sea posible compilarlas en diversas plataformas.

Finalmente, la corrección de las prácticas ha aumentado considerablemente. Mientras que durante el curso 16/17 el porcentaje de entregas calificadas como correctas fue muy bajo, ahora ha pasado al 32 % en la primera entrega, al 54 % en la segunda y al 79 % en la tercera. Esto es claramente debido a los tests que fuerzan a los estudiantes no solo a compilar sus programas, sino a ejecutarlos y comprobar que el comportamiento es el esperado. Durante el curso 16/17 muchos estudiantes no ejecutaban sus programas, o en caso de ejecutarlos, los probaban con muy pocos casos, lo que hacía que pasaran errores por alto. Con la incorporación de los tests, el estudio de los programas es mucho más exhaustivo. También aquí la aplicación web ha supuesto una mejora, ya que ayuda a los estudiantes a comprobar que todo es correcto antes de hacer el envío.

Como se puede ver, en la segunda entrega del curso 17/18, el número de prácticas que pasan los tests es superior al número de prácticas que compila. Esto se debe a que los tests no prueban los ficheros "main" de las prácticas, ya que estos ficheros suelen necesitar de interacción con el usuario, y es aquí donde encontramos entregas que pasan todos los tests pero no compilan. Por lo tanto, aunque con la introducción de los tests hemos conseguido aumentar el número de entregas correctas, todavía hay estudiantes que no ejecutan sus prácticas salvo para pasar los tests. Esto también lo hemos visto en las tutorías donde muchos estudiantes estaban más interesados en que las prácticas pasaran los tests, en lugar de preocuparse de que realmente estuvieran correctas.

A modo de conclusión de esta sección podemos decir que tanto la introducción de los tests como la disponibilidad de diversas aplicaciones que ayudan a validar la corrección de las entregas ha servido para mantener o mejorar la calidad de las mismas, aunque todavía quedan aspectos que se podrían perfeccionar.

7. Satisfacción con las herramientas

Con el propósito de conocer la opinión de los estudiantes con respecto al uso de los tests y a las diversas herramientas puestas a su disposición se ha elaborado, utilizando Google Forms, una encuesta individual, anónima y voluntaria. En esta sección analizamos los resultados obtenidos a través de dicha encuesta. La encuesta consistía de 5 secciones (información general, valoración de los tests, valoración aplicación estructura de ficheros, valoración aplicación web, y comentarios). Las secciones de valoración consistían en una serie de afirmaciones que seguían una escala de Likert de 4 puntos que iban de 1 (muy en desacuerdo) a 4 (muy de acuerdo). Las preguntas eran del tipo: "Me ha resultado fácil usar los tests" o "La aplicación web me resultaba útil para comprobar que mi entrega compila". Por último la sección de comentarios permitía a los estudiantes introducir comentarios adicionales. En la encuesta participaron 42 estudiantes.

Empezamos analizando la opinión de los estudiantes sobre el uso de tests (ver Figura 1). La valoración de los tests es muy positiva en todos los aspectos evaluados, y en general casi un 75 % de los estudiantes están satisfechos con el uso de los mismos. Considerando la utilidad de los mismos, el 84 % de los estudiantes afirman que los tests les han servido para detectar fallos, el mismo porcentaje opina que el nivel de exigencia con respecto a las prácticas ha aumentado gracias a los tests, y un 79 % considera que los tests dan más seguridad a la hora de evaluar las prácticas. Por último, desde el punto de vista de la usabilidad, un 69 % de los estudiantes considera que son fáciles de usar y un 63 % opina que el uso de tests les ha ahorrado tiempo.

La valoración de la herramienta encargada de comprobar la estructura de ficheros es también muy positiva (ver figura 2). Un 71 % de los estudiantes está satisfecho con dicha aplicación, y un 76 % piensa que cumple con su cometido. Además la mayoría de los estudiantes indica que la aplicación es sencilla de usar (un 81 %) e intuitiva (un 76 %). Por último, al 76 % de los estudiantes les gustaría disponer de una herramienta similar en otras asignaturas.

De manera similar, la aplicación web también recibe valoraciones positivas (ver figura 3). Un 73 % de

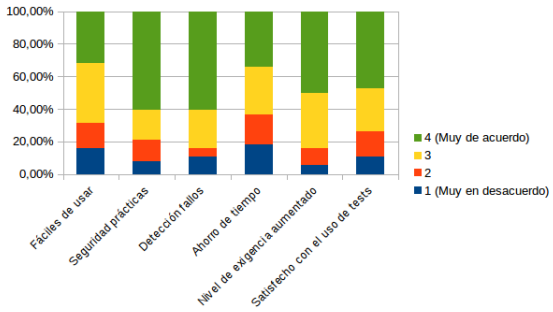

Figura 1: Valoración de los tests

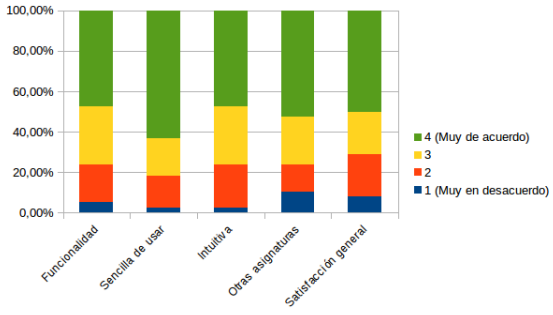

Figura 2: Valoración de la aplicación estructura ficheros

los estudiantes está satisfecho con la aplicación web, y un 71 % piensa que cumple con su cometido. Además la mayoría de los estudiantes indica que la aplicación es sencilla de usar (un 79 %) e intuitiva (un 76 %). De nuevo, al 73 % de los estudiantes les gustaría disponer de una herramienta similar en otras asignaturas. Una cuestión a mejorar son las sugerencias ante errores proporcionadas por la aplicación ya que solo el 55 % las considera útiles.

Por último si desgranamos las tres funcionalidades (comprobación de estructura, compilación y tests) de la aplicación web, los estudiantes opinan que en general la aplicación es útil (ver figura 4). Para el 76 % la aplicación sirve para comprobar si la estructura de

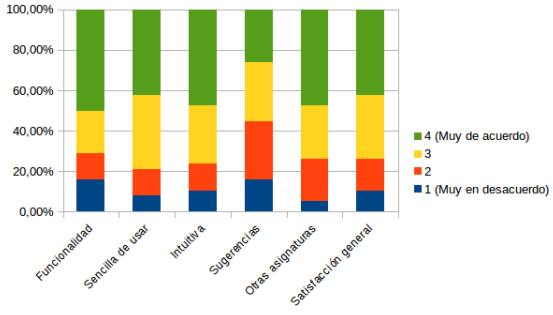

Figura 3: Valoración de la aplicación web

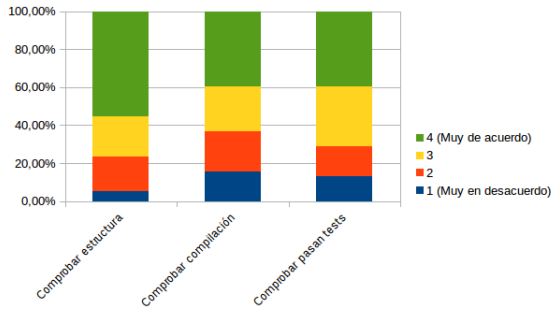

Figura 4: Valoración de la funcionalidad de la aplicación web

ficheros es correcta. El 63 % de los estudiantes opina que la aplicación web ayuda a comprobar si las prácticas compilan. Finalmente, el 71 % de los estudiantes cree que la aplicación es útil a la hora de comprobar si las entregas pasan todos los tests.

Desde el punto de vista de los profesores, el uso de tests y de herramientas para validación de entregas ha facilitado en gran medida la corrección de las prácticas, ya que permiten comprobar de manera rápida si las entregas cumplen unos requisitos mínimos. El tiempo invertido en crear los casos de prueba y las distintas herramientas de validación se compensa con la considerable reducción de tiempo dedicado a corregir las prácticas.

8. Trabajos relacionados

En la categoría de trabajos relacionados, deberíamos destacar al menos tres grandes grupos.

En primer lugar, el uso de testing para la autoevaluación de código por parte de los estudiantes y también por parte de los profesores. Este tópico, sus ventajas y sus debilidades han sido tratados en múltiples trabajos previos [6, 7, 8]. Nuestras aportaciones en este ámbito se podrían reducir a la novedad tecnológica de usar simultáneamente JUnit (para Java), Catch (para C++) y contenedores Docker para ejecutar de manera segura el código de los estudiantes. Los datos muestran que los tests han elevado el nivel de autoexigencia de los estudiantes.

En segundo lugar, podemos identificar el uso de herramientas informáticas para la gestión y corrección de entregas de estudiantes. El número de trabajos en esta categoría es ingente (entre ellos, algunos de los citados en el párrafo anterior). En nuestro caso, las herramientas presentadas han sido empleadas tanto para facilitar la tarea de evaluación de las prácticas como también para aumentar la autoexigencia de los estudiantes; de cara a la evaluación, su uso ha consistido en eliminar las partes más tediosas de la misma (comprobación de

ficheros, de que estos compilan correctamente, y que satisfacen los requisitos estipulados en los tests). Una idea que barajamos de cara al futuro es desarrollar dos grupos de tests, unos que entreguemos a los estudiantes comprobando funcionalidades básicas, y otros más exhaustivos que nos permitan determinar de manera más precisa la calidad de los trabajos. Nuestra solución tecnológica permitiría integrar esta solución de manera sencilla. En esta línea, existen herramientas tanto en C++ (por medio de ficheros makefile) como en Java (por medio de, por ejemplo, arquetipos en Maven) que permitirían facilitar la gestión y ejecución de los tests por parte de los estudiantes.

En tercer lugar, también podemos destacar algunos trabajos que fomentan el uso de herramientas de integración continua dentro de las aulas de Informática. En esta categoría posiblemente es donde menos trabajos reseñables podamos identificar. Los dos grandes servidores de integración continua en la empresa son Travis CI y Jenkins. En los trabajos académicos en la literatura, la herramienta preferida es Travis CI por su simplicidad de uso, configuración y por las posibilidades de gestión de repositorios de usuarios de manera individual o colectiva [13, 21]. Sí nos gustaría destacar el trabajo de Heckmann y otros en Jenkins [10], ya que más allá del testing, también integra otras herramientas para evaluar la calidad y cobertura del código (una ventaja adicional de las herramientas de integración continua con respecto a nuestra propuesta es que se pueden integrar herramientas o plugins externos de manera sencilla). Por otra parte, estas herramientas se basan en la utilización de gestores de versiones (como GitHub), lo cual hace que su adopción requiera de los estudiantes un salto tecnológico más amplio que por el momento hemos preferido evitar. Sin embargo, pensamos que las distintas herramientas que les hemos facilitado les ayudan a disponer de un marco de trabajo muy similar al que les facilitaría una herramienta de integración continua como Travis CI o Jenkis si solo van a aplicar testing.

9. Conclusiones y Trabajo Futuro

En este trabajo hemos presentado una experiencia en la asignatura Programación Orientada a Objetos donde se han introducido los tests como forma de aumentar la autoexigencia de los estudiantes a la hora de realizar las prácticas; además se ha desarrollado una aplicación web que sirve no solo para validar la corrección de las prácticas antes de ser enviadas, sino también para inculcar a los estudiantes principios básicos sobre integración continua y desarrollo basado en tests. Los resultados obtenidos en esta experiencia han sido positivos tanto para los estudiantes como para el profesorado de la asignatura. Desde el punto de vista de los estudiantes, gracias a la introducción de los tests y las herramientas de validación se ha mejorado la calidad de las entregas. Desde el punto de vista del profesorado, el tiempo dedicado a corregir las entregas se ha reducido gracias a las herramientas encargadas de comprobar los requisitos mínimos.

A partir de la experiencia de este curso, y el *feedback* recibido de parte de los estudiantes, hemos detectado una serie de puntos que pueden mejorarse para los próximos años. En primer lugar queremos aumentar la calidad de los tests, ya que algunos de ellos eran demasiado restrictivos (por ejemplo, al devolver una función un *string*, los tests de esa función esperaban una correspondencia completa con la solución esperada) y forzaban a los estudiantes a programar de una manera concreta dejando, en algunos casos, poco espacio a la imaginación. También queremos mejorar los informes de resultados devueltos por la aplicación web (por ejemplo, modificar el color de los warnings que hasta ahora aparece en rojo) e incluir informes estadísticos como los proporcionados por Allure [27]. También consideramos relevante la integración del servidor con el uso de gestores de versiones por parte de los estudiantes; el planteamiento sería generar repositorios con los tests de partida para que los estudiantes los clonaran y gestionaran sus propios repositorios. En un escenario ideal, el gestor de versiones estaría conectado (o "enganchado") con nuestro servidor para que se comprobara la corrección del código de manera automática y transparente al estudiante. Por último, sería interesante implantar la aplicación web en otras asignaturas del grado.

Referencias

[1] ACM/IEEE-CS Joint Curriculum Task Force. Computing curricula 1991. N.Y., febrero 1991. Disponible en `http://www.acm.org`.

[2] R. Alberich y J. Miró. La colaboración en el Jenui *revisited*: La convergencia europea. In *Actas de las XV Jornadas de Enseñanza Universitaria de Informàtica, Jenui 2009*, páginas 431 – 434, Barcelona, Julio 2009. Póster.

[3] P. Bradford, M. Porciello, N. Balkon, y D. Backus The Blackboard Learning System: The Be All and End All in Educational Instruction? *Journal of Educational Technology Systems*, 35(3):301–314, 2007.

[4] K. Beck. *Test Driven Development: By Example*. The Addison-Wesley Signature Series, 2002.

[5] S S. Center and S Starr. *Network Programming*. Sun Microsystems, San Jose, CA, mayo 1988. Disponible en `http://www.sun.com/manuals/np.html`.

[6] A. Cernuda del Río, M. García Rodríguez, N.

García Fernández, y M. González Rodríguez Uso de JUnit para evaluación en laboratorio de Estructuras de Datos En *Actas de las XX Jornadas de Enseñanza Universitaria de Informàtica, Jenui 2014*, páginas 237 – 243, Barcelona, Julio 2009. Póster.

[7] A. García Dopico, S. Rodríguez de la Fuente y F. J. Rosales García. Automatización de prácticas en entornos masificados. En *Actas de las IX Jornadas de Enseñanza Universitaria de Informática, Jenui 2003*, páginas 119 – 126, Cádiz, Julio 2003.

[8] P. P. Garrido Abenza. Sistema de evaluación automatizada de prácticas para Tecnología de Computadores En *Actas de las XI Jornadas de Enseñanza Universitaria de Informática, Jenui 2005*, páginas 138 – 144, Villaviciosa de Odón (Madrid), Julio 2005.

[9] M. A. Gómez Martín, G. Jiménez Díaz y P. P. Gómez Martín Test de unidad para la corrección de prácticas de programación, ¿una estrategia win-win? En *Actas de las XVI Jornadas de Enseñanza Universitaria de Informàtica, Jenui 2010*, páginas 51 – 58, Santiago de Compostela, Julio 2010.

[10] S. Heckman, J. King y M. Winters. Automating Software Engineering Best Practices Using an Open Source Continuous Integration Framework (Abstract Only) En *Proceeding SIGCSE '15 Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, páginas 677–677, Kansas City (EEUU), Marzo 2015.

[11] J. L. Hennessy y D. A. Patterson. *Computer Architecture. A Quantitative Approach*. Morgan Kaufman, San Francisco, segunda edición, noviembre 1996.

[12] J. Hunter y Crawford Java Servlet Programming O'Reilly & Associates, segunda edición, 2001.

[13] C. Kästner. Teaching Software Construction with Travis CI. `https://www.cs.cmu.edu/~ckaestne/travis/` Carnegie Mellon University, 2014.

[14] R. Lister y I. Box. A citation analysis of the ace2005-2007 proceedings, with references to the june 2007 core conferences and journal rankings. En *Tenth Australasian Computing Education Conference*, enero 2008.

[15] R. C. Martin. *Agile Software Development, Principles, Patterns, and Practices.* Prentice Hall, 2003.

[16] G. Meszaros. *xUnit Test Patterns: Refactoring test code.* Addison-Wesley, 2007.

[17] D. Merkel. *Docker: lightweight linux containers for consistent development and deployment*. Linux Journal 239(2), 2014.

[18] C. R. Merlo, J. M. Merlo, L. Hoeffner y R. Moscatelli. An analysis of computer science education publication using lotka's law. *J. Comput. Small Coll.*, 26:85–92, enero 2011.

[19] P. Nash. Catch2: C++ Automated Test Cases in a Header `https://github.com/catchorg/Catch2`, 2017.

[20] L. A. Rosado, F. J. Rodríguez, R. M. Luque-Baena y F. Luna Experiencia con una herramienta de pruebas de caja negra para el aprendizaje de asignaturas de programación en evaluación continua En *Actas de las XXIII Jornadas de Enseñanza Universitaria de Informàtica, Jenui 2017*, páginas 61 – 68, Caceres, julio 2017.

[21] O. Shaikh. Real-time feedback for students using continuous integration tools. `https://github.com/blog/2324-real-time-feedback-for-students-using-continuous-integration-tools` San Francisco State University.

[22] J. Shore y S. Warden. *The Art of Agile Development*. O'Reilly Media, Inc. 2007.

[23] J. Spurlock. *Bootstrap: Responsibe Web Development*. O'Reilly, 2013.

[24] R. V. Solé y R. Pastor-Satorras. *Complex networks in genomics and proteomics.* En Stefan Bornholdt and Heinz Georg Schuster, editors, Handbook of Graphs and Networks: From the Genome to the Internet, volume 2, chapter 3. Wiley, enero 2003.

[25] P. Tahchiev, F. Leme, V. Massol y G. Gregory. *JUnit in Action*. Manning Publications, 2008.

[26] W. Tarreau. *HAProxy-the reliable, high-performance TCP/HTTP load balancer*. `http://www.haproxy.org/`, 2011.

[27] The Yandex testing team. *Allure Test Report*. `http://allure.qatools.ru/`, 2013.