Actas de las Jenui, vol. 3. 2018.
Páginas: 173-180

Experiencia con una herramienta para automatizar la comprobación de requisitos de los trabajos HTML-CSS de Sistemas Informáticos*

Jose Divasón, Ana Romero, Eduardo Sáenz-De-Cabezón

Departamento de Matemáticas y Computación
Universidad de La Rioja
26006 Logroño
{jose.divason,ana.romero,eduardo.saenz-de-cabezon}@unirioja.es

Resumen

La llegada del Plan Bolonia supuso un importante cambio en la enseñanza universitaria en España, dotando de más peso a las prácticas de laboratorio y a los trabajos (individuales o en grupo). En el ámbito de la Ingeniería Informática esto tiene varias consecuencias: para el profesorado la corrección de código se convierte en una tarea ardua y pesada, y para los estudiantes no es fácil comprobar si sus entregas cumplen los requisitos mínimos exigidos. Por ello se extendió el uso de tests automáticos, especialmente para ejercicios de programación. Sin embargo, la comprobación de requisitos en otras asignaturas a veces no es fácil con las herramientas ya existentes, incluso en ocasiones no es posible y conviene desarrollar herramientas *ad-hoc*. En este artículo se presenta una herramienta desarrollada para facilitar a los estudiantes la comprobación de los requisitos exigidos en la realización de un trabajo de la asignatura de Sistemas Informáticos (impartida en primer curso del Grado en Ingeniería Informática y del Grado en Matemáticas): el diseño y desarrollo de una página web. Dicha herramienta ha permitido reducir el esfuerzo de los profesores al corregir, además de mejorar notablemente el número de estudiantes que cumplen con los requisitos exigidos en la entrega.

Abstract

Bologna Process imposed an important change with respect to the usual way of teaching at Spanish universities, mainly by increasing the number and importance of practical lessons and exercises. In the Degree in Computer Science, this has many consequences: correcting code exercises becomes tedious and cumbersome, whereas students can find it difficult to check whether their practical work satisfy the established requirements. To this end, the use of automatic tests was widely adopted, specially involving programming-exercises. However, when other subjects are involved this could not be so easy with the existing tools, even sometimes is not possible and could be desirable to develop *ad-hoc* tools. In this work, we present a tool devoted to easing the check of the requirements of an exercise of the Computer Systems subject (taught at the first course in the Degree in Computer Science and the Degree in Mathematics): the design and development of a website. This tool allows us to decrease teacher's effort when correcting the exercises as well as it significantly improves the number of students that satisfy the requirements.

Palabras clave

Corrección automática, retroalimentación automática, validación automática de requisitos.

1. Introducción

Históricamente en todos los niveles de educación en España el proceso de enseñanza-aprendizaje siempre ha tenido como elemento principal la figura del profesor, mientras que el estudiante estaba considerado como un simple receptor pasivo de este proceso, recibiendo retroalimentación (o feedback) tan solo al final del proceso. El ámbito universitario no ha sido ninguna excepción, e incluso probablemente sea el contexto donde más se extendió este fenómeno.

Sin embargo, todo cambió con la llegada del plan Bolonia y la implantación del Espacio Europeo de Educación Superior (EEES): este proceso ahora está centrado en el aprendizaje del estudiante, lo que ha

*Trabajo parcialmente financiado por el Vicerrectorado de Profesorado de la Universidad de La Rioja a través de un proyecto de innovación docente.

propiciado que tanto la forma de evaluar como los procesos de enseñanza-aprendizaje hayan tenido que evolucionar para adaptarse.

Si antes prácticamente la evaluación de una asignatura universitaria se centraba en un examen final escrito tras una serie de lecciones magistrales, ahora las manera de impartir la docencia han cambiado y durante los últimos años se han popularizado nuevas metodologías y recursos docentes como la clase invertida, el aprendizaje basado en problemas y la gamificación que incluso han sido exitosamente aplicadas también en el contexto de la enseñanza de la informática [1, 2, 8, 9, 10]. Además, gran parte de la evaluación ahora también se reparte entre otras actividades, como la entrega de prácticas y la realización de trabajos, quitando de esta manera peso al examen final. Con estos cambios en los sistemas de evaluación, los docentes debemos conseguir mecanismos de retroalimentación que permitan a los estudiantes conocer el grado de consecución de los objetivos y competencias de cada una de las pruebas de evaluación que componen nuestras asignaturas. Desafortunadamente, no siempre es fácil conseguir esto. El elevado número de estudiantes por clase implica que la carga del profesorado para corregir los trabajos y entregables de prácticas de laboratorio haya aumentado notablemente en los últimos años. Este hecho tiene como consecuencia que las correcciones se demoren, y que por tanto el estudiante reciba retroalimentación de sus soluciones pasado un tiempo desde su realización, pudiendo olvidar los detalles que le llevaron a cometer tales errores.

Todos estos problemas se acentúan aún más en el Grado en Ingeniería Informática, puesto que es un grado eminentemente práctico en el cual por lo general al menos la mitad de los créditos ECTS de cada asignatura consisten en la realización de prácticas en aulas de informática.

Dependiendo del profesor y la asignatura, en la Universidad de La Rioja hay varias formas de evaluar las prácticas y los trabajos del Grado en Ingeniería Informática. Algunos ejemplos son:

- Las prácticas se entregan semanalmente y se corrigen profundamente. No hay examen de prácticas (ejemplo: Sistemas Distribuidos).
- Se entrega un subconjunto de las prácticas. No hay examen final de prácticas (ejemplo: Especificación y Desarrollo de Sistemas de Software).
- No se entregan las prácticas semanalmente, pero hay examen final y una entrega de un trabajo final (ejemplo: Sistemas Operativos).
- Las prácticas consisten en el desarrollo gradual de una aplicación. No hay examen final de prácticas, pero la evaluación se lleva a cabo mediante entrevistas con el profesor acerca del diseño, soluciones alternativas, etc (ejemplo: Programación de Aplicaciones Web).

En general no hay examen de prácticas en las distintas asignaturas, sino que su evaluación se realiza a través de entregas periódicas de prácticas, además de la realización de algún trabajo. En el contexto de la enseñanza de informática, la corrección de prácticas y trabajos es una tarea ardua, repetitiva y laboriosa: hasta hace poco lo habitual era que los estudiantes hiciesen sus entregas a través del campus virtual de la universidad, y que el profesor las imprima y corrija en papel. Sin embargo, revisar un código línea por línea es algo costoso, aunque en determinadas asignaturas todavía lo seguimos haciendo, bien sea por su importancia, por costumbre, o por querer dar una corrección muy detallada e individualizada a cada estudiante. Por poner un ejemplo concreto en nuestra universidad, la media de líneas de código que emplea cada estudiante para resolver una práctica de Sistemas Distribuidos es de 500 (con lo cual, si hay 30 estudiantes, el profesor debe revisar unas 15.000 líneas de código en cada entrega).

Algunos expertos en docencia universitaria afirman incluso que la corrección manual y profunda de los ejercicios entregados por los estudiantes no es provechosa y que el tiempo invertido por el profesor en hacer esta tarea no compensa [4] puesto que los estudiantes reciben las correcciones pasado un tiempo, cuando lo más adecuado es dar feedback inmediato aunque no sea tan profundo, al menos inicialmente (y después si es posible más adelante proveer de una corrección detallada). De esta manera, se considera que lo que hace que los estudiantes aprendan es la calidad de su implicación en las actividades de aprendizaje, no que los docentes realicen gran cantidad de correcciones.

Para solucionar parcialmente esto y poder dar retroalimentación automática de manera inmediata, en la gran mayoría de universidades españolas se ha buscado integrar en las prácticas y trabajos de programación alguna ayuda para su evaluación, principalmente a través de las herramientas surgidas de los concursos de programación y el empleo de los tests unitarios que están disponibles en todos los lenguajes de programación (como por ejemplo, las librerías Catch para C++[1], JUnit para Java[2] y Unittest para Python[3]). De esta forma, estos tests unitarios permiten tener cierta seguridad de que el algoritmo programado cumple la especificación, y se proporciona información automática a los estudiantes acerca de sus errores. A su vez esto tiene una gran ventaja, puesto que se descarga de trabajo de corrección al profesorado. Como algunos ejemplos de herramientas de este estilo usadas y consolidadas, cabe destacar entre otras que se han desarrollado ser-

[1] https://github.com/catchorg/Catch2
[2] https://junit.org/junit4/
[3] https://docs.python.org/2/library/unittest.html

vicios web para la corrección automática de consultas SQL en asignaturas de bases de datos, que incluso existen herramientas integradas en Moodle que permiten hacer test unitarios automáticamente a ejercicios de programación además de revisar similitudes de código entre estudiantes al realizar la entrega de prácticas [11] y que también existen los llamados jueces online, plataformas software que contienen descripciones de diferentes problemas de programación, junto con baterías de pruebas secretas, con las que testear si una determinada solución es o no correcta, siendo el más famoso *¡Acepta el reto!* [7].

Dichas ayudas centradas en la evaluación en programación están muy extendidas y se han ido presentando en las JENUI de años anteriores [6], e incluso hay una sesión específica de este tema en dichas jornadas [5].

Sin embargo surge una pregunta. ¿Qué hacemos cuando no estamos en una asignatura de programación? Es decir, cuando queremos hacer un test pero el test no consiste en comprobar si un algoritmo cumple una especificación o si una consulta SQL devuelve lo esperado. En este caso, puede ocurrir que muchos de los requisitos de nuestras prácticas y trabajos no puedan ser comprobados por ninguna de las utilidades existentes y tengamos que desarrollar alguna herramienta *ad-hoc*.

El trabajo se estructura de la siguiente manera. En la Sección 2 presentamos el contexto, el sistema de evaluación y los requisitos del trabajo que demandamos en la asignatura de Sistemas Informáticos en la Universidad de La Rioja. El análisis y discusión detallada de la herramienta que hemos desarrollado para automatizar parcialmente la validación de dichos requisitos se muestra en la Sección 3. En la Sección 4 presentamos las encuestas que hemos llevado a cabo para evaluar la herramienta, además de mostrar la percepción de los estudiantes acerca de ella junto con una explicación de los resultados que hemos obtenido. Finalmente, en la Sección 5 se presentan las principales conclusiones del trabajo y se incluye posibles líneas de actuación futura.

2. El caso de Sistemas Informáticos

Nuestra asignatura de Sistemas Informáticos se encuentra en el caso que hemos comentado anteriormente: posee un trabajo cuya comprobación de requisitos no se puede llevar a cabo con herramientas existentes ni con pruebas unitarias, pues no consiste en programación de algoritmos. Sistemas Informáticos es una asignatura de primer curso común al Grado en Ingeniería Informática y al Grado en Matemáticas. Está pensada para que sea una asignatura eminentemente práctica, consta de 6 créditos ECTS repartidos en una hora semanal de teoría y 3 horas de prácticas en las aulas de informática. Cada año la cursan alrededor de 75 estudiantes, de los cuales aproximadamente 50 pertenecen al Grado en Ingeniería Informática y 25 del Grado en Matemáticas que están entremezclados indistintamente en los diferentes grupos de la asignatura.

Concretamente, en el curso 2017-2018 ha habido 3 grupos de prácticas con 23, 23 y 25 estudiantes respectivamente, haciendo un total de 71 estudiantes matriculados, de los cuales 68 asistieron con cierta regularidad a las prácticas y 64 se presentaron al examen final. Los tres autores de este trabajo somos los encargados de la docencia de dicha asignatura. Una parte importante de la evaluación de la asignatura, concretamente un 20 %, consiste en la realización en grupos de una página web utilizando los lenguajes HTML y CSS. El resto de la evaluación consiste en la asistencia y aprovechamiento de las prácticas (20 %) y un examen final teórico-práctico (60 %).

Para la realización del trabajo, se divide aleatoriamente a los estudiantes en grupos de tres personas (y excepcionalmente de dos). A cada grupo se le asigna un tema relacionado con los conceptos vistos en las clases de teoría. Los componentes del grupo deben realizar una página web con los mismos contenidos (todos los miembros del grupo deben tener exactamente los mismos archivos HTML), pero el diseño debe ser distinto (distintos CSS). Es decir, no se presenta un único trabajo por grupo, sino que cada miembro debe presentar el suyo. Hay una serie de requisitos que deben cumplir obligatoriamente los trabajos. A continuación enumeramos algunos[4] de ellos:

- Se debe respetar la diferenciación entre contenido y aspecto. Las páginas HTML únicamente deben centrarse en el contenido y su estructura y no deben incluir ningún elemento sobre el aspecto o presentación de la información. No se podrán usar atributos de estilo en los ficheros HTML.
- Al menos habrá 8 ficheros HTML diferentes.
- Todos los ficheros HTML deben verse afectados por alguna hoja de estilo y, al menos dos, deben verse afectados por al menos 2 hojas de estilo.
- Todos los ficheros HTML deben pasar la validación de HTML 5 según las especificaciones de la W3C.
- Al menos habrá 2 ficheros CSS diferentes.
- Los ficheros CSS deben pasar la validación, al menos, de la especificación CSS 2.1.

Tradicionalmente se ha fijado una fecha para realizar una primera entrega, en la cual los estudiantes deberían entregar la página web cumpliendo todos los requisitos. Desafortunadamente, la gran mayoría de trabajos

[4] La lista completa de requisitos puede verse en `https://goo.gl/Rk5VQY`

no cumplen con dichos requisitos en esta primera entrega, a pesar de que ello debería suponer la no superación del trabajo. El principal motivo es que hay muchos requisitos y los estudiantes tienen que que trabajar con multitud de ficheros, de modo que les resulta relativamente fácil despistarse y cometer errores. Poniendo en contexto la situación con datos concretos: en los cursos anteriores aproximadamente 2 de cada 3 trabajos no cumplían los requisitos exigidos. Los porcentajes detallados pueden verse en el Cuadro 1, donde los resultados se muestran en función de los trabajos entregados (y no del número total de estudiantes, puesto que hay un pequeño porcentaje de estudiantes que abandonan la asignatura y no entregan el trabajo). Tras esta primera entrega, se les comunicaba a los estudiantes los requisitos que no cumplían sus trabajos y se les daba un plazo para corregirlos en una segunda entrega (sin añadir más cambios a los trabajos que la simple corrección). En este curso, el trabajo se propuso a principios de noviembre, siendo la fecha límite para la primera entrega el 15 de diciembre y para la segunda el 10 de enero.

	2016	**2017**	**2018**
Número de entregas	73	75	68
Cumplían los requisitos	20	25	58
Porcentaje de éxito	27,4 %	33,33 %	85,29 %

Cuadro 1: Entregas que cumplían los requisitos del trabajo en el curso actual y los dos anteriores.

Como se deduce a partir de la lista de requisitos presentada anteriormente, no existe una herramienta que permita comprobar automáticamente todos ellos por ser muy específicos. La experiencia de años anteriores que teníamos los autores acerca de la revisión y corrección de los requisitos es que era pesada y extremadamente manual: por ejemplo, para comprobar que todos los documentos HTML de una entrega son válidos sintácticamente (el primer requisito de la lista anterior), la forma habitual de hacerlo ha consistido en comprobarlo manualmente, archivo a archivo (teniendo en cuenta que hay alumnos que entregan más de 50 archivos) a través de la página web oficial para validar [5].

La característica clave que buscábamos en una herramienta era que al menos validase automáticamente todos los HTML y los CSS del trabajo, sin tener que ir fichero por fichero, ya que esto era la parte más pesada y en la que se invertía más tiempo. Lo más parecido que encontramos fue la aplicación web WDG HTML Validator [6]. Sin embargo, poseía varias desventajas: solamente valida los HTML (no las hojas de estilo CSS) y no permite la validación de múltiples ficheros HTML en local sino únicamente a través de la URL de la página web ya subida a un servidor (lo cual supondría trabajo extra para los estudiantes, ya que tendrían que subir la página web al servidor cada vez que quisiesen validar los HTML). Además, dicha herramienta tan solo comprobaría uno de los requisitos exigidos del trabajo (la validación de los HTML), teniendo que comprobar el resto manualmente o dejándolo en manos de otras herramientas distintas cuando fuese posible (por ejemplo, la comprobación de alguno de los requisitos restantes puede hacerse a través del comando `grep`).

Para evitar tener que estar revisando manualmente los distintos requisitos y no tener que recurrir a distintas herramientas para cada uno de ellos, decidimos desarrollar una única herramienta *ad-hoc* que automatizase dicha tarea de comprobación de los requisitos del trabajo. Además, decidimos no solo ponerla a disposición de todos los profesores de la asignatura, sino también a los estudiantes para que puedan hacer uso de ella durante la realización del trabajo. De esta forma, los estudiantes podrán estar seguros de que sus páginas web cumplen los requisitos antes de realizar la entrega, de forma que obtengan feedback inmediato y puedan corregir errores sin tener que esperar a que el profesor les corrija. Aunque la comprobación de los requisitos también seguiría siendo tarea del profesor, al igual que en los casos previos, el uso de esta herramienta permitiría liberar de carga de trabajo a los docentes ya que facilita y automatiza al menos una parte bastante importante de la corrección.

Más aún, para aprovechar las sinergias se propuso que una pequeña parte del desarrollo de la herramienta fuese parte de un trabajo de otra asignatura de la titulación, Sistemas Distribuidos, perteneciente al tercer curso, cuando los estudiantes ya han adquirido más conocimientos. Dicha asignatura es impartida desde el curso pasado por uno de los autores de este trabajo. La opinión de este autor (por su reciente época como estudiante [3]) es que a veces los estudiantes no ven determinadas asignaturas como algo interesante. Claro que saben que son muy importantes para ellos, incluso absolutamente fundamentales e imprescindibles dentro de la titulación, pero no les resultan interesantes ni motivantes. Y de ahí que nuestros estudiantes a veces se pregunten ¿y esto que me están explicando realmente sirve para algo útil y tiene aplicación real?

El profesor de Sistemas Distribuidos siempre hace lo mismo el primer día de clase con sus estudiantes, se les ofrece algo muy simple pero que casi nunca habían tenido la oportunidad de hacer anteriormente: la posibilidad de expresar por escrito anónimamente lo que quisieran sobre la asignatura antes de empezar (qué esperaban, qué contenidos les gustaría ver con detalle,

[5] `https://validator.w3.org/`
[6] `http://htmlhelp.com/tools/validator/`

etc.). En general, la respuesta siempre es la misma: "tener la sensación de que he aprendido algo útil para mi futuro laboral, no un contenido teórico que se me va a olvidar antes de acabar la carrera". Es decir, los estudiantes esperan problemas reales, aunque no requieran de unos conocimientos muy avanzados, pero sí que sientan que van a desarrollar algo útil aplicando los conocimientos adquiridos durante el curso y dejando a un lado las prácticas y trabajos guiados.

Por tanto, estábamos en una situación *win-win* entre el profesorado y los estudiantes de dicha asignatura: los estudiantes de Sistemas Distribuidos se encontraban ante un problema real, es más, el trabajo de la página web ya la habían tenido que hacer cuando cursaron Sistemas Informáticos en el primer curso y por tanto conocían de primera mano la dificultad para comprobar los requisitos manualmente. Esto hizo que el problema les resultase atractivo y motivante, puesto que normalmente los trabajos de dicha asignatura no habían consistido en la resolución de problemas del mundo real. Además, se impartían los conocimientos suficientes acerca de servicios web, sincronización y concurrencia como para que fuese factible su desarrollo. Por otra parte, para el profesorado de Sistemas Informáticos suponía una ventaja: se iba a desarrollar algo que iba a tener una utilidad real y realmente facilitaba la tarea de revisión de los requisitos.

3. Validador automático de requisitos

Como se ha explicado en la sección anterior, en la asignatura de Sistemas Distribuidos se desarrolló en Java el primer prototipo del validador de requisitos. El motivo de hacerlo en Java fue simplemente porque las prácticas de dicha asignatura se impartían en dicho lenguaje. El funcionamiento del primer prototipo era muy simple: recibía un fichero comprimido y comprobaba si todos los documentos HTML y CSS contenidos en él pasaban la validación conectándose automáticamente a los servicios web remotos y procesando los resultados.

A partir de dicho prototipo, decidimos incluir nuevas funciones para la autocorrección de los requisitos mostrados anteriormente. En la etapa de análisis de nuestro programa, decidimos que debía cumplir ciertas características para que resultase útil a los estudiantes: no solo tenía que comprobar los requisitos, sino que era muy importante que lo hiciese lo más rápido posible y debía ser muy fácil de utilizar. El requisito de que la ejecución fuese rápida no es trivial: normalmente hay que conectarse con el servicio web remoto y subir hasta 50 archivos, hay que además examinar cada uno de ellos para comprobar que no hay atributos de estilo en los HTML, etc. Esto requirió un uso cuidadoso de la concurrencia y el paralelismo, la sincronización entre hilos y el uso de estructuras de datos eficientes. En cuanto a la sencillez de uso, decidimos hacer una interfaz gráfica con un único botón: el de seleccionar la ruta donde está alojada la página web (es decir, los ficheros que hay que examinar) [7].

Es importante remarcar que desarrollar una herramienta para comprobar la totalidad de los requisitos exigidos es muy difícil, por no decir imposible. Un ejemplo de ello sería comprobar que no se ha utilizado ningún generador o herramienta de diseño web para la creación de las páginas. Por tanto, nuestro validador no comprueba este tipo de requisitos, siendo responsabilidad de los estudiantes asegurarse de que los cumplen y del profesor revisarlo manualmente.

La Figura 1 muestra la interfaz gráfica y un ejemplo de ejecución del programa.

Como puede apreciarse en la captura, aparecen en verde los requisitos comprobados y cumplidos, en rojo los que no se cumplen (dando información de qué fichero falla), y debajo una serie de requisitos que la herramienta no comprueba debido a su naturaleza, y que por tanto los estudiantes y el profesorado debería comprobar manualmente.

Los estudiantes tuvieron acceso a la herramienta a través del aula virtual de la asignatura durante todo el periodo para la realización del trabajo. Hay que remarcar que en ningún caso obligamos a los estudiantes a hacer uso de la herramienta, sino que simplemente nos limitamos a explicarla y mostrarles su utilidad, dejando a voluntad de cada estudiante usarla o no durante el desarrollo de su trabajo. No hubo problemas para su utilización, salvo algún pequeño número de alumnos (especialmente pertenecientes al Grado en Matemáticas) que no tenía Java instalado. La herramienta está además adaptada para que el profesorado también pueda obtener informacion automáticamente que pueda resultar útil para la calificación de los trabajos (es decir, por ejemplo detectando si se han incluido vídeos, detectar el número de tablas o idiomas utilizados, etc.).

4. Resultados

Tras la primera entrega de los trabajos, decidimos realizar una encuesta a todos los estudiantes que asistían a prácticas para conocer su opinión acerca de la utilidad del validador de requisitos. Todos los datos fueron tratados de forma anónima y confidencial, hecho que fue además anunciado a los estudiantes previamente a su participación en las encuestas. Aunque

[7] En la versión utilizada en el curso 2017-2018 la carga de archivos se hace en local por ser la más utilizada por los estudiantes, pero está previsto que próximamente se pueda incluir directamente la dirección URL donde esté alojado y a través de una araña-web navegar por ella para conseguir el resto de ficheros.

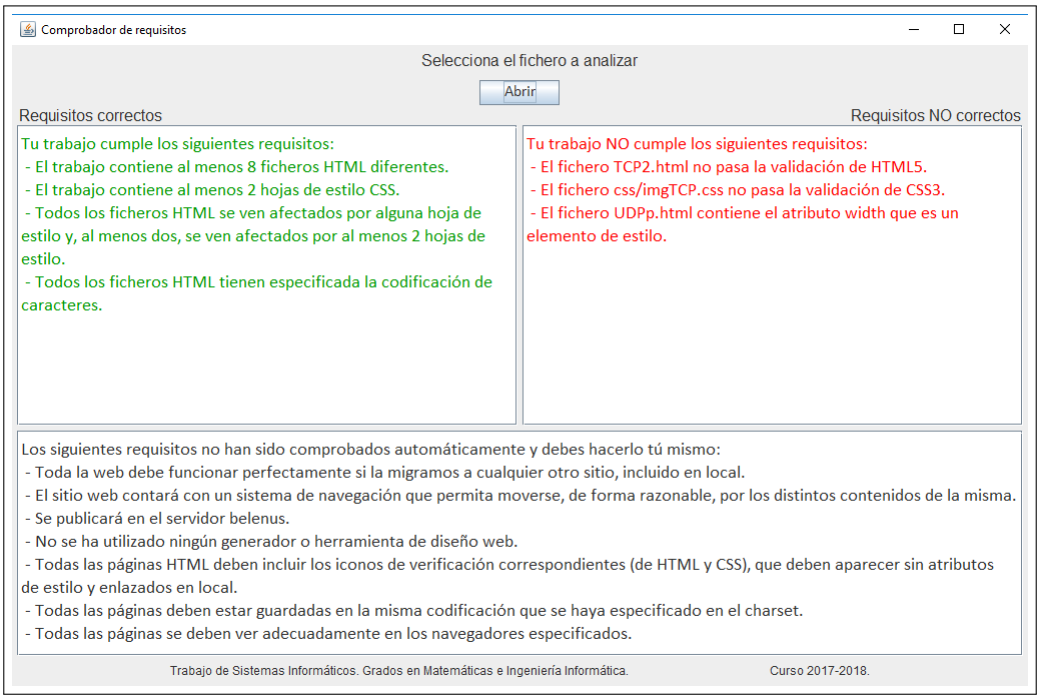

Figura 1: Ejemplo de funcionamiento del validador.

el aula virtual Blackboard Learn© que dispone nuestra universidad permite elaborar formularios anónimos para llevar a cabo esta tarea, la necesidad de autenticación previa en la plataforma hace que los estudiantes puedan sentir una falta en el anonimato. Por ello, realizamos directamente un cuestionario anónimo en papel a los distintos grupos de prácticas de la asignatura.

El modelo de encuesta era intencionadamente simple, con tan solo 4 preguntas.

1. ¿Has utilizado la herramienta para comprobar los requisitos?
2. Valora numéricamente si te ha resultado útil y te ha facilitado la comprobación de requisitos.
3. Valora numéricamente la facilidad de uso.
4. Número de veces que has hecho uso de la herramienta hasta conseguir que el trabajo cumpla los requisitos.

Además, se daba la posibilidad de realizar un comentario libre acerca de la herramienta y proponer posibles mejoras. La valoración numérica se realizaba a través de una escala de tipo Likert de 5 puntos, siendo 1 lo peor valorado y 5 lo mejor valorado.

Los resultados de dichas encuestas pueden verse en el Cuadro 2. Como puede apreciarse, los resultados son bastante homogéneos entre los tres grupos y en general la opinión de los estudiantes ha sido muy positiva: en términos globales un 91,94 % de ellos utilizaron la herramienta, valorándola como muy útil (4,44 puntos sobre 5) y fácil de usar (4,23 puntos sobre 5).

En los comentarios libres de las encuestas también se veía que su uso por parte de los estudiantes había resultado muy satisfactorio y que valoraban muy positivamente la reducción de tiempo para comprobar los requisitos. Además, ellos mismos propusieron alguna mejora para el programa, como por ejemplo que el programa permitiese arrastrar y soltar directamente los ficheros que quieres comprobar a la interfaz. Cabe destacar que no se aprecian diferencias en cuanto a valoración y uso ni entre grupos ni entre titulaciones.

En cuanto a los resultados, el Cuadro 1 muestra el notable incremento en el número de estudiantes que cumplen los requisitos del trabajo, demostrando que la implantación de la herramienta ha supuesto una clara mejora con respecto a años anteriores. Concretamente, hemos pasado de tener un porcentaje de éxito de alrededor del 30 % en cursos anteriores a más de un 85 %. Es importante remarcar también que, al igual que pasaba con la valoración de los estudiantes, no hemos encontrado grandes diferencias en cuanto a resultados entre los estudiantes de los distintos grupos y titulaciones.

Cabe recordar que había principalmente dos objetivos en el desarrollo de esta herramienta: que se produjese una mejora en los resultados de los estudiantes y que a su vez se redujese el tiempo empleado por los profesores en la corrección. Como ya se ha visto a lo largo de esta sección, la herramienta ha tenido un impacto muy positivo en los resultados. Por otra parte, la opinión personal del profesorado también es muy po-

Pregunta	Grupo 1	Grupo 2	Grupo 3	Total
Estudiantes matriculados	23	23	25	71
Estudiantes participantes	22	19	21	62
Estudiantes que utilizan el validador	20	18	19	57
Porcentaje de estudiantes que utilizan el validador	90,9 %	94,74 %	90,48 %	91,94 %
Utilidad	4,35	4,56	4,42	4,44
Facilidad de uso	4,1	4,22	4,37	4,23
Número de veces usado	3,8	3,95	2,63	3,46

Cuadro 2: Resultados de las encuestas.

sitiva, puesto que la reducción en cuanto a la carga de trabajo a la hora de corregir ha sido muy evidente, estimándola en alrededor de un 70 %. Más aún, tenemos la percepción de que sobre todo ha mejorado la calidad de las correcciones, puesto que anteriormente era bastante probable que incluso a nosotros también se nos pasase algún requisito entre la multitud de páginas que había que comprobar manualmente.

5. Conclusiones y trabajo futuro

La corrección automática de ejercicios de programación por medio de tests y servidores de prácticas está ampliamente extendido y es de sobra conocido por la comunidad docente de informática, puesto que permiten dar feedback inmediato a los estudiantes, a la vez que facilitar la tarea de corrección. Prácticamente en todas las universidades existen servidores, aplicaciones y servicios web para resolver esto. Sin embargo, ¿qué hacemos cuando hablamos de trabajos más específicos o que no son de programación, donde no existan herramientas que faciliten dicha tarea? En este caso, quizás pueda resultar muy conveniente y útil que se desarrolle una herramienta *ad-hoc* para facilitar la tarea. Es más, para aprovechar las sinergias con otras asignaturas, nuestra experiencia nos lleva a animar a plantear la herramienta como una práctica o aplicación a desarrollar en cursos más avanzados, de esta manera los estudiantes se mostrarán motivados ante el reto puesto que se enfrentan a un problema real que además conocen de primera mano. El desarrollo de este tipo de herramientas beneficia tanto a los profesores, puesto que se automatiza al menos en parte el proceso de corrección, como a los estudiantes, ya que obtienen información acerca de sus trabajos y prácticas de un modo totalmente inmediato, pudiendo corregir errores antes de realizar la entrega.

En nuestro caso, hemos aplicado estas ideas desarrollando una herramienta que ayude a la comprobación de los requisitos de trabajos de HTML y CSS en la asignatura de Sistemas Informáticos. La herramienta ha tenido un impacto muy positivo tanto entre el alumnado, que valoran favorablemente la herramienta y les ha permitido mejorar notablemente sus resultados, y entre el profesorado, puesto que se ha reducido el número de horas empleadas en corregir, además de poder ofrecer una corrección mucho más precisa y menos subjetiva.

Nuestra percepción es que hay que incidir que el hecho de invertir tiempo en desarrollar herramientas *ad-hoc*, aunque sean específicas, merece la pena. El desarrollo de una herramienta que automatice ciertas tareas pesadas puede marcar una importante diferencia en la tasa de éxito de nuestros estudiantes, como se ha comprobado en nuestro caso. Nuestra herramienta es pública y puede descargarse de la siguiente dirección web:

`https://goo.gl/T2T9Ed`

Está específicamente diseñada para su uso con nuestros trabajos y nuestros requisitos específicos, aunque en cualquier caso el código fuente de la herramienta es de acceso libre y permitimos su modificación para adaptarla a las necesitades de otras asignaturas. Incluso contactando con los autores ofrecemos además acceso total al repositorio que contiene el desarrollo.

Actualmente la herramienta es un programa Java independiente. Como trabajo futuro, nos planteamos la integración de la herramienta directamente en el campus virtual de nuestra universidad, que es proporcionado por Blackboard Learn©. De esta forma, los estudiantes podrían realizar la comprobación de requisitos online y en el mismo momento que hacen la entrega, de modo que el sistema impidiese entregar si no se cumplen las condiciones requeridas en el trabajo. Hemos consultado esta posibilidad con el responsable de Blackboard Learn© en nuestra universidad, llegando a la conclusión de que esta integración es factible, aunque quizás no es tan fácil y directa, requiriendo más trabajo que si por ejemplo se usase Moodle. En cualquier caso, una vez que hemos visto que la herramienta es útil tanto para estudiantes como profesores, creemos que esta integración supondría una mejora notable en cuanto a su facilidad de uso, y que sobre todo ayudaría a evitar los típicos problemas que surgen con las

versiones que suben los estudiantes respecto a las versiones sobre las que ellos habían comprobado los requisitos. De hecho, al menos 3 de los estudiantes de este curso que no cumplían los requisitos era por esto: comprobaron los requisitos, tenían todo correcto, realizaron un pequeño cambio de última hora, subieron la nueva versión y ese último cambio provocaba que hubiese errores.

Referencias

[1] Jesús R. Campaña, Ana E. Marín, María Ros, Daniel Sánchez, Juan M. Medina, M. Amparo Vila, M. Dolores Ruiz, Manuel P. Cuéllar y Maria J. Martin-Bautista. Metodologías activas y gamificación en las asignaturas de iniciación a la programación En *Actas de las XXII Jornadas de Enseñanza Universitaria de Informática, Jenui 2016*, páginas 245 – 252, Universidad de Almería, 2016.

[2] Carlos Catalán y Raquel Lacuesta. Aprendizaje Basado en Problemas: una experiencia interdisciplinar en Ingeniería Técnica en Informática de Gestión. En *Actas de las X Jornadas de Enseñanza Universitaria de Informática, Jenui 2004*, páginas 305 – 311, Universidad de Alicante, 2004.

[3] Jose Divasón. La experiencia de un docente que hace no tanto era alumno. En *Actas del Simposio-Taller previo a las XXIII Jornadas de Enseñanza Universitaria de Informática, Jenui 2017*, páginas 11 – 17, Universidad de Extremadura, 2017.

[4] Graham Gibbs y Claire Simpson. Conditions under which assessment supports students' learning. *Learning and teaching in higher education*, (1): 3–31, 2005.

[5] Alberto Gómez Mancha y Roberto Rodríguez-Echeverría, editores. *Actas de las XXIII Jornadas sobre la Enseñanza Universitaria de la Informática*. Universidad de Extremadura, 2017.

[6] Marco Antonio Gómez Martín, Guillermo Jiménez Díaz y Pedro Pablo 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 Informática, Jenui 2010*, páginas 51–58, Universidad de Santiago de Compostela, 2010.

[7] Pedro Pablo Gómez Martín y Marco Antonio Gómez Martín. ¡Acepta el reto!: juez online para docencia en español. En *Actas de las XXIII Jornadas de Enseñanza Universitaria de Informática, Jenui 2017*, páginas 77 – 84, Universidad de Extremadura, 2017.

[8] Antoni Jaume-i-Capó, Isaac Lera, Francisco Juan Vives, Biel Moyà-Alcover y Carlos Guerrero Tomé. Experiencia piloto sobre el uso de la gamificación en estudios de grado de ingeniería en informática. En *Actas del Simposio-Taller previo a las XXII Jornadas de Enseñanza Universitaria de Informática, Jenui 2016*, páginas 35 – 40, Universidad de Almería, 2016.

[9] Mercedes Marqués. Qué hay detrás de la clase al revés (flipped classroom). En *Actas de las XXII Jornadas de Enseñanza Universitaria de Informática, Jenui 2016*, páginas 77 – 84, Universidad de Almería, 2016.

[10] Alberto Prieto Espinosa, Beatriz Prieto Campos y Begoña del Pino Prieto. Una Experiencia de flipped classroom. En *Actas de las XXII Jornadas de Enseñanza Universitaria de Informática, Jenui 2016*, páginas 237 – 244, Universidad de Almería, 2016.

[11] Juan Carlos Rodríguez del Pino, Enrique Rubio Royo y Zenón J. Hernández Figueroa. VPL: laboratorio virtual de programación para Moodle. En *Actas de las XVI Jornadas de Enseñanza Universitaria de Informática, Jenui 2010*, páginas 429 – 435, Universidad de Santiago de Compostela, 2010.