Actas de las Jenui, vol. 4. 2019.
Páginas: 239-246

Metodología para el desarrollo y evaluación de competencias de programación software en la nube

Amanda Calatrava, J. Damián Segrelles Quilis
Instituto de Instrumentación para Imagen Molecular (I3M)
Universitat Politècnica de València (UPV)
46022 Valencia
amcaar@i3m.upv.es, dquilis@dsic.upv.es

Resumen

Hoy en día, las competencias relativas a la programación de software son fundamentales en las titulaciones STEM (*Scientific, Technology, Engineering and Mathematics*). Por ello, habitualmente éstas incorporan asignaturas específicas enfocadas a su adquisición y evaluación, en las que se llevan a cabo actividades educativas de carácter práctico que sumergen al alumno en entornos realistas de desarrollo software.

Para la ejecución de dichas actividades, se requiere de instrumental específico comúnmente utilizado en entornos profesionales, tales como repositorios de versiones o entornos de desarrollo y testeo software, entre otros, que necesitan de una configuración y orquestación adecuada para que sean efectivos y que en ocasiones supone al docente una sobrecarga inasumible.

En este artículo, se presenta una metodología de trabajo para sesiones prácticas que permitirá a los docentes desarrollar y evaluar las competencias relativas a la programación software. Además, se presenta un recurso docente que da soporte a dicha metodología mediante el despliegue y uso del instrumental software específico necesario en la nube, con el objeto de disponer de forma sencilla y rápida de un entorno integrado para la aplicación de la metodología sin que suponga una sobrecarga al docente.

Abstract

Nowadays, the competences related to software programming are fundamental in the STEM (*Scientific, Technology, Engineering and Mathematics*) degrees. For that, they usually incorporate specific subjects focused on the acquisition and evaluation of software programming skills. These subjects commonly have practical educational activities that try to immerse the student in realistic environments.

For the appropriate execution of those type of activities, we require specific software tools commonly used in professional environments, such as version-control repositories for tracking changes in the source code or environments to develop and test code, among others. These tools require an adequate configuration and orchestration to be effective, sometimes involving the teacher in a unapproachable overload.

In this article, we present a work methodology for practical sessions that will allow teachers to develop and evaluate the competences related to software programming. In addition, a teaching resource is presented that supports this methodology through the deployment and use of specific software tools necessary in the cloud, in order to have a simple and fast way of an integrated environment for the application of the methodology without involving an overload to the teacher.

Palabras clave

La nube, programación sofware, competencias

1. Introducción

Hoy en día, las competencias relativas a la programación de software son fundamentales en muchas de las titulaciones STEM [13, 1](*Scientific, Technology, engineering and Mathematics*). Es por ello, que dichas competencias se integran dentro de sus planes de estudio mediante asignaturas específicas donde se realizan Actividades Educativas (AE), principalmente de carácter práctico, diseñadas para la aplicación de métodos de trabajo que conllevan de forma intrínseca a su desarrollo y evaluación. Para una aplicación efectiva de dichos métodos, se requiere de herramientas e instrumental software específico que necesitan de una configuración y orquestación adecuada, más si cabe si hablamos de entornos individualizados centrados en el alumnado.

Estas herramientas e instrumental específico se traducen en entornos como repositorios de versiones

(ej. Git[1], Apache Subversion [2]), herramientas testeo e integración continua (CI/CD *Continuous Integration and Continuous Delivery*) (Ej. Jenkins[3]) y IDE (*Integrated Deployment Enviroments*) (Ej. Devc++ [4], Code::Blocks [5]) entre otras. Su uso permite a la comunidad docente poner en marcha metodologías de aprendizaje en las que obligan al alumnado a sumergirse en entornos reales de desarrollo software, lo que mejora de forma significativa los procesos de enseñanza aprendizaje que se producen [5, 6], y por lo tanto les permite adquirir competencias específicas (ej. Lenguaje C, Python o Java, programación modular, o programación orientado a objetos) y transversales ("Instrumental Especifica", "Análisis y Resolución de Problemas", "Trabajo en Grupo") de una forma mas eficaz.

En este contexto, configurar y orquestar todas estas herramientas en un entorno integrado e individualizado para cada estudiante o grupo, supone un esfuerzo inasumible por el personal técnico de los laboratorios, y mucho más para el personal docente, dado que estas herramientas están más dirigidas a entornos de producción software y no para escenarios de aprendizaje académico, por lo que su configuración suele resultar costosa en estos ámbitos. Por ese mismo motivo, al alumnado le resulta prácticamente imposible acceder a estas herramientas configuradas e integradas entre sí fuera de los laboratorios docentes, siendo esto otra dificultad para su puesta en marcha en el ámbito académico, sobre todo en lo que se refiere a actividades no presenciales. Por desgracia, esto se suele traducir en que métodos de aprendizaje que necesitan de estas herramientas no se realicen en el aula, al menos de forma completa y eficaz. Sin embargo, las tecnologías en la nube ya han demostrado a través de numerosas trabajos los beneficios de su uso en la educación superior [8], tanto desde el punto de vista de los profesores [15], estudiantes y centros educativos [14]. Entre dichos beneficios se encuentra: i) la capacidad de la nube de proporcionar despliegues automatizados de instrumental específico software y hardware orquestados y configurados adhoc para una actividad educativa; ii) la posibilidad de acceder a dichos instrumentos de forma ubicua y con una accesibilidad 24x7; iii) la posibilidad del alumnado de utilizar sus propios dispositivos para acceder a las herramientas, siendo la nube la que proporciona la capacidad de computo y almacén, y no los propios dispositivos.

En este artículo se presenta, por una parte, una metodología de trabajo para el desarrollo y evaluación de competencias de programación software mediante instrumental software específico, y por otra, un recurso docente que permite a los profesores la aplicación de dicho método en el aula, mediante herramientas para desplegar y configurar el entorno de desarrollo software en la nube para aplicar dicho método, con el objeto de disponer de una forma sencilla y rápida de un entorno integrado para el desarrollo de competencias relacionadas con la programación.

La estructura del artículo es la siguiente. En primer lugar, en la sección 2 se presenta un escenario de referencia donde actualmente se aplica una metodología docente para la adquisición de competencias relacionadas con la programación software. Sobre dicho escenario, se describen las necesidades y carencias existentes en su aplicación, para a continuación proponer unos objetivos dirigidos al diseño de un método mejorado que incluya nuevo instrumental específico software en el flujo de trabajo. Seguidamente, la sección 3 presenta como resultado la metodología de trabajo y en la sección 4 se describe un recurso docente que facilita la aplicación del nuevo método propuesto. Finalmente, se presentan la discusión, conclusiones y trabajos futuros.

2. Escenario de referencia

El escenario de referencia ocurre actualmente en la Escuela Técnica superior de Ingeniería del diseño (ETSID) de la Universitat Politècnica de València (UPV), en el Grado de Ingeniería Electrónica y Automática (GEIA), concretamente en la asignatura obligatoria del primer cuatrimestre *Informática* de primer curso. La asignatura se divide en dos bloques, por una parte 3 créditos a impartir mediante clases teóricas (2 horas semanales) y por otra parte 3 créditos a impartir mediante clases prácticas de laboratorio (2 horas semanales) a lo largo de un total de 16 semanas.

Las sesiones de prácticas de laboratorio son de vital importancia en el proceso de aprendizaje del alumno en la adquisición de las habilidades y competencias de programación, dado que es donde los alumnos ponen en práctica los conceptos teóricos adquiridos en las clases de teoría a través de un instrumental software específico. Actualmente, los alumnos resuelven problemas planteados a través de boletines, que deben desarrollar de forma presencial o autónoma fuera del aula. En la actualidad, se utiliza el entorno Devc++ como IDE de programación para la resolución de los problemas planteados.

2.1. Carencias y necesidades

La principal carencia en las clases prácticas, es que el alumnado se limita a desarrollar programas software

[1] Git: A free and open source distributed version control system. `https://git-scm.com/`

[2] Apache Subversion: Enterprise-class centralized version control for the masses. `https://subversion.apache.org/`

[3] Jenkins: Build great things at any scale. `https://jenkins.io/`

[4] DevC++: A free, portable, fast and simple C/C++ IDE. `https://sourceforge.net/projects/orwelldevcpp/`

[5] Code:Blocks. `http://www.codeblocks.org/`

en lenguaje C. El único instrumental software especifico utilizado es el IDE Devc++, que se utiliza para desarrollar los programas planteados en los boletines de prácticas, bien en el aula (boletines presenciales) o fuera de ella (boletines autónomos). Por tanto, el método actual, queda lejos de entornos realistas de producción software debido a que no utilizan herramientas como repositorios de versiones o herramientas de testeo e integración continua. Es por ello que surge la necesidad de modificar el método de trabajo actual, e integrar nuevos flujos en la metodología que permitan acercarles a un entorno profesional de desarrollo software a través de nuevo instrumental software específico.

Otra de las carencias existentes, es la accesibilidad al instrumental específico utilizado dado que no todos los equipos propios del alumnado son compatibles (los Mac no disponen de una versión de DevC++), dificultando la realización por parte del alumno del las prácticas autónomas fuera del aula. Este hecho se agravaría con la inclusión de nuevo instrumental específico, no solo por incompatibilidad de versiones sino también por su complejidad en su configuración y requerimientos hardware. Por ello, surge la necesidad de proporcionar todas estas herramientas a través de la nube, proporcionando todos los recursos hardware y software requeridos a través de proveedores Cloud y que estos se puedan utilizar desde cualquier dispositivo (portátil o PC) que dispongan.

2.2. Objetivos

Para cubrir las necesidades identificadas anteriormente se plantea como objetivo general integrar nuevos flujos de trabajo a la metodología empleada actualmente, con el objeto de que el alumnado utilice instrumental específico ampliamente utilizado en el día a día del desarrollo software profesional. Por ello, y para la consecución de este objetivo general, se plantean los siguientes objetivos específicos:

- Integrar en el flujo de trabajo un software de control de versiones ampliamente empleado en el mundo laboral, como es Git, concretamente su implementación más utilizada, GitHub [6]. También, se utilizará GitHub Classroom [7], como una herramienta asociada a GitHub que proporciona facilidades para la gestión de múltiples repositorios de cara a su aplicación en el aula.
- Integrar en el flujo de trabajo un entorno de validación de desarrollos software, como es Jenkins [8], concretamente una de las herramientas de código abierto más empleadas en la actualidad.
- Proporcionar *feedback* continuo en los desarrollos de prácticas a través de Jenkins, de forma que permita al alumnado conocer al instante y sin requerir interacción por parte del docente del estado de sus desarrollos, acercándoles a entornos realistas de producción software.
- Proporcionar todo el instrumental especifico y recursos hardware requeridos a través de la nube, permitiendo un acceso 24x7 ubicuo a través de sus propios dispositivos (portátiles y tabletas).
- Desarrollar un recurso docente que automatice la configuración y orquestación en la nube de todas estas herramientas, de forma que por una parte proporcionen un entorno personalizado para cada alumno, y por otra parte que facilite y haga viable el despliegue y la gestión del entorno de prácticas por parte de los docentes.

[6] GitHub. https://github.com/
[7] GitHub Classroom. https://classroom.github.com/
[8] The Jenkins project. https://jenkins.io/

3. Metodología de trabajo

La figura 1 muestra la arquitectura general de la metodología de trabajo que se propone en esta contribución. En ella se destacan las fases o acciones principales que han de llevar a cabo tanto el docente (en naranja) como los alumnos (en verde), de las cuales se presentan más detalles a continuación.

3.1. Despliegue del entorno de prácticas en la nube

En esta primera fase, el docente tiene que desplegar en la nube el instrumental software especifico requerido para las prácticas (fase 1 naranja en la figura 1). Esto incluye el despliegue de una Máquina Virtual (MV) por estudiante configurada con las herramientas software necesarias, de forma que los alumnos no tengan que configurar nada en sus máquinas para ponerse a trabajar. En concreto, el entorno de prácticas que necesita el alumnado está compuesto por máquinas Windows configuradas con el IDE DevC++ y un cliente gráfico para GitHub, así como una MV central con Jenkins instalado que servirá para llevar a cabo el testeo de los ejercicios de prácticas de cada alumno. Las MV de los estudiantes y la MV central se encontrarán en una misma subred virtual para optimizar las comunicaciones entre ellas.

3.2. Crear la organización en GitHub

A su vez, es necesario configurar el espacio que se utilizará para la asignatura en GitHub. GitHub ya ha sido utilizado en entornos de aula, como se demuestra en trabajos como [2] y [11]. Éste es un servicio en la nube que ayuda a los desarrolladores a almacenar y administrar su código fuente, al igual que llevar

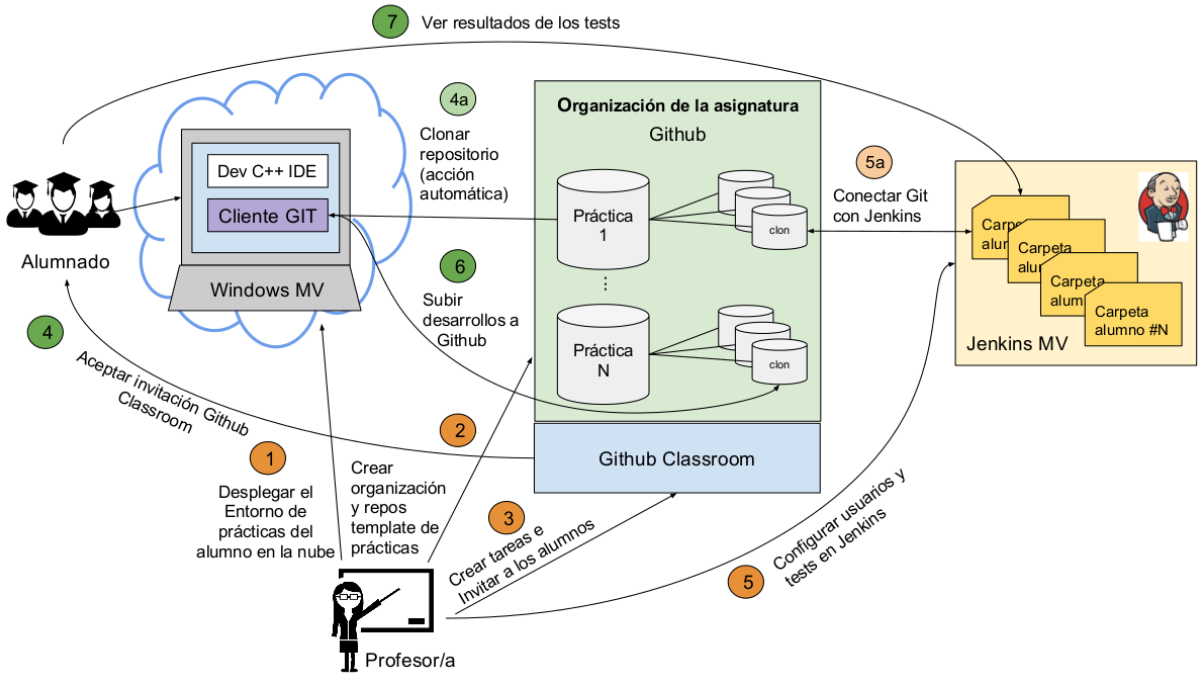


Figura 1: Diagrama de la metodología propuesta. En naranja se destacan las acciones a llevar a cabo por el docente, mientras que en verde las acciones llevadas a cabo por los alumnos.

un registro y control de cualquier cambio sobre este. El uso de Git requiere de conocimiento y uso de línea de comandos que puede complicar el acercamiento de la herramienta al estudiante. Por ello, la interfaz gráfica de GitHub es bastante más asequible para iniciarse en este tipo de repositorios de código. Además de las cuentas de usuario, GitHub ofrece cuentas para organizaciones, las cuales representan un grupo de usuarios que comparten proyectos, y permite gestionar a estos usuarios como equipos de trabajo.

Así, el docente tiene que crear una organización específica para la asignatura en GitHub que servirá para recopilar los repositorios de prácticas de todos los alumnos de forma centralizada (fase 2 naranja en la figura 1). Para ello, el profesor/a tiene que crear una cuenta de organización en su cuenta de usuario en GitHub. Tras ello, para guiar al alumno/a en cada práctica, el docente crea un repositorio ”modelo” o ”plantilla” que contendrá el código inicial de los ejercicios, sirviendo como punto de partida a los alumnos/as para cada una de las prácticas.

3.3. Crear tareas en GitHub Classroom

El siguiente paso consiste en crear tareas para cada una de las prácticas en GitHub Classroom (fase 3 naranja en la figura 1). Esta plataforma asiste tanto al docente como al estudiante automatizando la creación de repositorios y el control de acceso a los mismos, facilitando así a los profesores la distribución del código inicial de las prácticas y la recolección de sus desarrollos. Así, el docente, tras darse de alta en GitHub Classroom y crear una organización para la asignatura, creará una tarea (*assingment* en GitHub Classroom) para cada práctica de la asignatura, indicando el repositorio ”modelo” creado anteriormente con el código inicial (plantilla) de la práctica. El/la docente obtendrá un enlace que deberá enviar a los alumnos para que acepten la invitación a la tarea. Esta tarea puede ser individual o grupal. En este punto, el único requisito para el docente es una lista con los correos de sus estudiantes. Cuando el alumno acepta la tarea (simplemente acceden al enlace que les proporciona su profesor/a), se genera automáticamente el clon del repositorio ”modelo” de la práctica correspondiente a la tarea en la cuenta del alumno, dentro de la organización inicialmente creada por el docente (fase 4 y 4a verde en la figura 1).

3.4. Configuración de Jenkins

Una vez creados de forma automática los repositorios de cada estudiante a través de GitHub Classroom, el docente tendrá que configurar Jenkins para testearlos automáticamente (fase 5 y 5a naranja en la figura 1). Esto conlleva diferentes acciones a realizar con Jenkins para: (i) crear una cuenta de usuario para cada

estudiante; (ii) crear una jerarquía de carpetas por estudiante; (iii) configurar un test por ejercicio de práctica vinculado al repositorio correspondiente de cada estudiante; y (iv) llevar a cabo una configuración de permisos adecuada para que los estudiantes solo tengan acceso a sus prácticas y no a las del resto de compañeros.

Los resultados de los tests de Jenkins servirán al alumnado para conocer el acierto en cada ejercicio de prácticas, y al docente, facilitando su evaluación.

3.5. Desarrollo de las prácticas

Finalmente, los estudiantes podrán desarrollar sus prácticas de laboratorio integrando en las mismas el *workflow* de GitHub, clonando localmente su repositorio, desarrollando los ejercicios de cada práctica, subiendo los cambios al repositorio (fase 6 verde en la figura 1) y accediendo a Jenkins para comprobar los resultados de los tests (fase 7 verde en la figura 1). Estos tests pueden ser configurados para que se ejecuten cada vez que un alumno haga cambios en su repositorio o pueden ser ejecutados bajo demanda cuando el alumno se conecte a la interfaz de Jenkins y éste se lo pida a la herramienta de forma explícita.

3.6. Dificultades para la implantación en el aula

A la hora de implantar en el aula la metodología propuesta, existen una serie de limitaciones que pueden dificultar su puesta en marcha. Estas dificultades las enumeramos a continuación, para posteriormente proporcionar un recurso docente que ayude a paliarlas.

- Es necesario que el profesorado tenga conocimiento de entornos virtualizados y de computación en la nube para crear las máquinas necesarias que componen el entorno de prácticas.
- El profesorado necesita conocer aspectos avanzados de Jenkins, para configurar de forma apropiada e individualizada a cada estudiante los tests de las prácticas y los permisos de los usuarios.
- Se necesita que el profesorado y el alumnado tengan una cuenta de GitHub. El primer día de prácticas el profesor puede guiar a los alumnos en este proceso a aquellos que no dispongan de una.
- El uso de GitHub puede complicar la identificación del trabajo de cada estudiante, dado que cada alumno/a tendrá una cuenta con un nombre de usuario que puede no coincidir con su verdadero nombre o correo de la organización/universidad a la que pertenece. En este problema, GitHub Classroom proporciona un mecanismo al docente (llamado *roster*) que permite asociar las cuentas de GitHub de los alumnos con una lista de nombres o correos para facilitar la identificación de la autoría de cada repositorio. El docente necesitará por tanto, aplicar una solución similar en Jenkins, para identificar los tests de cada alumno.
- Necesitamos que los repositorios de GitHub que utilicen los estudiantes sean privados, para evitar que vean el trabajo del resto de compañeras y compañeros. Esto supone un coste que puede no ser asumible en todos los casos. Sin embargo, GitHub ofrece la posibilidad de solicitar una beca para poder crear los repositorios de los alumnos en la organización de forma privada[9].

4. Recurso docente

Para tratar de resolver las dificultades identificadas en la sección anterior para la implantación de la metodología de trabajo en el aula, en esta sección presentamos un recurso docente que, apoyándose en diferentes herramientas de código abierto bien conocidas como las ya mencionadas GitHub, GitHub Classroom y Jenkins; y otras como Infrastructure Manager (IM) [4] y Google Spreadsheets [10]; ayudará al docente en el proceso de su puesta en marcha, ejecución y mantenimiento. Todos los desarrollos presentados en este artículo están disponibles como código abierto en GitHub [11].

4.1. Automatización del despliegue del entorno de prácticas en la nube

Para ayudar al profesor/a con el despliegue inicial del entorno de prácticas, utilizaremos el IM, una herramienta que facilita el despliegue y configuración de infraestructuras complejas en plataformas de computación en la nube. El IM es compatible con la mayoría de proveedores actuales, tanto públicos (como AWS, Google Cloud Engine o Microsoft Azure), como privados (como OpenNebula y Openstack), entre otros, y proporciona un lenguaje de descripción propio para que los usuarios construyan "recetas" compatibles con Ansible [12] con el objeto de describir la infraestructura y la configuración requerida. Ansible es una plataforma software que automatiza el aprovisionamiento de software, la gestión de configuraciones y el despliegue de aplicaciones. Para este caso específico, hemos desarrollado una receta de Ansible para el IM que se encarga de desplegar los recursos necesarios, y de instalar y configurar automáticamente el entorno de prácticas de los alumnos, por lo que el profesor no necesi-

[9] Github Education program. `https://education.github.com/`

[10] Spreadsheets. `https://spreadsheets.google.com/`

[11] https://github.com/dsegrelles/autoassesment-programming

[12] Ansible. `https://www.ansible.com/`

taría dominar Ansible para utilizar el recurso, solo utilizar la receta que se proporciona. Este entorno incluye las MVs para cada alumno configuradas con el cliente de GitHub gráfico (*GitHub Desktop* [13]) y el Entorno de Desarrollo Integrado (IDE) Dev C++ en un sistema operativo Windows. A su vez, la receta despliega y configura una máquina adicional con Jenkins instalado, donde el docente será el usuario administrador.

Con esto, el docente solo necesita utilizar el cliente del IM para disponer del despliegue necesario en las prácticas de laboratorio. Cabe destacar que, para otra asignatura que requiera de otro tipo de entorno (por ejemplo una máquina Linux con un IDE diferente o con librerías específicas), se puede desarrollar una nueva receta específica para ese entorno y seguir automatizando el despliegue con un solo comando. La sintaxis para desplegar el entorno es la siguiente:

```
./im_client.py create <receta_ansible>
-a <fichero_credenciales>
```

4.2. Automatización de la configuración de Jenkins

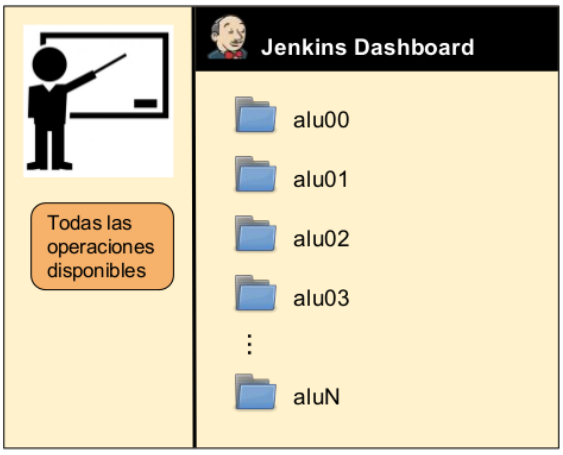
El segundo desarrollo principal del recurso docente son dos scripts que automatizan la configuración de Jenkins y así dar soporte al *feedback* en tiempo real al alumnado sobre los test realizados en sus desarrollos de prácticas. El objetivo es lograr que la interfaz de Jenkins tenga una vista para alumno y profesor como la que se muestra, esquematizada, en la Figura 2.

El primer script, *registerStudents.sh*, ayuda al docente a crear los usuarios, la estructura de carpetas y la configuración pertinente de permisos de las mismas para una lista de alumnos y alumnas. La lista estará compuesta por los nombres de las cuentas de GitHub de cada estudiante y su email. El script también necesita conocer el número de prácticas que tendrá la asignatura, para crear la estructura de carpetas necesaria que albergará los diferentes tests por ejercicio de cada práctica programada. Así, el script automatiza la creación de los usuarios en Jenkins y envía a cada estudiante sus credenciales de acceso. La gestión de los permisos de cada usuario se realiza a través de roles. Para ello, el script utiliza el plugin *Role-based Authorization Strategy*, para crear y asignar los roles para cada usuario y carpeta. Esta es la sintaxis de invocación del script:

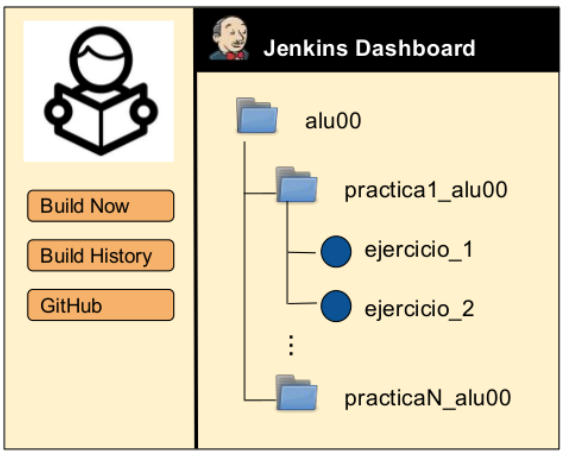
```
./registerStudents.sh -f <names_file>
-j <jenkins_URL> -a <num_practs>
-u <user> -p <password>
```

Es necesario destacar que este script solo es necesario ejecutarlo una vez, al inicio del curso, una vez los alumnos tengan creada su cuenta en GitHub.

[13] GitHub Desktop. https://desktop.github.com/



Vista del profesor



Vista del alumno

Figura 2: Vistas del docente y el estudiante en Jenkins y acciones disponibles.

El segundo script, *createTests.sh*, es el encargado de crear, para cada práctica, los tests (*jobs* en Jenkins) que evaluarán cada ejercicio de la misma. Este script recibe un fichero que contiene la lista de usuarios de GitHub de cada alumno junto con la URL del repositorio de la práctica para la que se van a crear los tests. Para generar este fichero se proporciona un script auxiliar llamado *get_repos_url.sh*, que proporciona la lista de URLs de los repositorios dentro de una organización que cumplen con un prefijo dado. Además, el script requiere de la plantilla del test en formato XML (el formato soportado por Jenkins) a partir de la cual la herramienta configurará los tests para cada alumno. Cada test se creará adecuadamente dentro de la estructura de carpetas creada por el script anterior para cada uno de los alumnos. La invocación a este script es la siguiente:

```
./createTests.sh -f <repos_file>
-j <jenkins_URL> -a <num_pract>
-n <test_name> -t <test_template>
```

Acciones a llevar a cabo	Docente	Alumnado	¿Asistido?
Desplegar el entorno de prácticas en la nube	X		Completamente por el recurso docente
Crear organización y repositorios modelo en GitHub	X		Por la interfaz de GitHub
Crear las tareas en GitHub Classroom e invitar a los alumnos	X		Por la interfaz de GitHub Classroom
Crear cuenta en GitHub y aceptar invitación		X	Por la interfaz de GitHub
Configurar Jenkins apropiadamente	X		Completamente por el recurso docente
Hacer práctica y subirla a GitHub		X	Por el docente y GitHub Desktop
Ver resultados de las prácticas		X	Por el recurso docente y Jenkins
Evaluar las prácticas	X		Parcialmente por los tests de Jenkins

Cuadro 1: Resumen de acciones a llevar a cabo por parte del docente y del alumnado para implantar la metodología descrita en clase y soporte proporcionado para las mismas.

```
-u <user> -p <password>
```

Finalmente, para resolver los posibles problemas de identificación de la autoría de los desarrollos con cada estudiante, se propone el uso de Google Spreadsheets. Spreadsheets es una herramienta accesible a través de Internet (no requiere más que un navegador para acceder a ella) para crear hojas de cálculo, permitiendo que múltiples personas accedan y editen de forma colaborativa la misma hoja. Así, en la primera sesión de prácticas, tras la creación de las cuentas de los alumnos en GitHub, se puede solicitar a los estudiantes que accedan a una hoja de cálculo donde encontrarán la lista de nombres y correos de la clase (preparada previamente por el profesor/a) y que rellenen una columna con su nombre de usuario de GitHub. Así, el docente podrá relacionar los desarrollos de cada repositorio con el estudiante y los correspondientes tests en Jenkins.

Como resumen, el cuadro 1 recoge las principales acciones a llevar a cabo por el docente y por el alumnado para implantar la metodología descrita en las sesiones de prácticas de laboratorio, así como el soporte proporcionado por el recurso desarrollado del que disponen ambos roles. Cabe decir que no podemos automatizar el uso de GitHub Classroom, ya que no dispone de ningún API para interaccionar con la herramienta. Así, el docente tendrá que crear la tarea en el portal de GitHub Classroom, e invitar a los alumnos mediante el portal que proporciona la propia herramienta.

5. Discusión

En este artículo se presenta una metodología de trabajo que acerca a los alumnos a entornos profesionales de programación software mediante la inclusión en el flujo de trabajo del aula, de instrumental software específico. Esto, combinado con estrategias y métodos de aprendizaje apropiados [7], permite mejorar los procesos de enseñanza-aprendizaje, dado que el alumnado se familiariza con herramientas, entornos y tecnologías que se van a encontrar en el mundo empresarial, e incluso permite simular contextos laborales reales a través de simuladores[9] o laboratorios virtuales [12].

La aplicación de herramientas de desarrollo software de uso común en el entorno profesional en el aula aporta importantes beneficios para el alumnado, pero a la vez una carga docente inasumible para el profesor/a. Es por ello que el uso del recurso docente que se ha presentado en este artículo es de vital importancia para el docente, permitiéndole implantar esta metodología en clase con un esfuerzo razonable.

Otro de los beneficios relevantes que aporta la metodología descrita en este artículo es la retroalimentación o *feedback* continuo [3]. Gracias al testeo automático de las prácticas, el alumnado obtendrá información para mejorar en el proceso de aprendizaje y comprender los nuevos conocimientos. Así, conseguimos implantar un proceso de enseñanza-aprendizaje mucho más rico tanto para el docente como para el alumno.

En la literatura existe un trabajo [10] que propone una solución similar a la presentada en este artículo, pero sin proporcionar herramientas para la automatización del despliegue del entorno de prácticas y del servidor Jenkins, solo aportan una guía de instalación y configuración, por lo que no resuelve el incremento de carga de trabajo para el docente. Además, no utilizan Google Classroom, sino que implementan un servicio propio que hace funciones similares. Aun así, los autores de [10] han introducido herramientas como GitHub y Jenkins a más de 3.000 estudiantes, lo que refuerza la viabilidad de la metodología propuesta en este artículo.

6. Conclusiones

Este trabajo presenta una metodología apoyada en un recurso docente que facilita al profesor el empleo de

herramientas del ámbito profesional para la programación software en clases prácticas de laboratorio, permitiendo a los estudiantes aprender buenas prácticas sobre ingeniería software. Además, ayudan a que reciban *feedback* continuo de sus desarrollos sin necesidad de interacción continua con el docente.

Como trabajo futuro se pretende desarrollar una interfaz gráfica de usuario para el profesor que englobe los scripts desarrollados y que gestione la obtención de las listas que necesitan de entrada ambos scripts a partir de la hoja de Google Spreadsheets. Además, se pretende dotar a la herramienta de la capacidad de generar informes que enriquezcan el *feedback* que recibe el alumno de la plataforma y que ayuden a la posterior evaluación de las prácticas por parte del docente. A partir del curso que viene, se pretende poner en marcha el uso de este recurso docente en el aula, para poder evaluar de forma cuantitativa y cualitativa la satisfacción de docentes y alumnos, el impacto en la carga docente y los resultados académicos para detectar posibles dificultades no identificadas hasta ahora.

Agradecimientos

Los autores agradecen la financiación recibida por el Vicerrectorado de Estudios, Calidad y Acreditación de la Universitat Politècnica de València para desarrollar el Proyecto de Innovación y Mejora Educativa “Comunidades de Aprendizaje como servicios en la nube para el desarrollo y evaluación automática de Competencias Transversales y Objetivos Formativos específicos”, con referencia B29.

Referencias

[1] Mohammed Al-Bow, Debra Austin, Jeffrey Edgington, Rafael Fajardo, Joshua Fishburn, Carlos Lara, Scott Leutenegger y Susan Meyer. Using game creation for teaching computer programming to high school students and teachers. *ACM SIGCSE Bulletin*, 41(3):104–108, 2009.

[2] Miguel A Angulo y Ozgur Aktunc. Using github as a teaching tool for programming courses. *Proceedings of the 2018 ASEE Gulf-Southwest Section Annual Conference*, 2018.

[3] David Boud y Elizabeth Molloy. *El feedback en educación superior y profesional: Comprenderlo y hacerlo bien*, volumen 42. Narcea Ediciones, 2015.

[4] Miguel Caballer, Ignacio Blanquer, Germán Moltó y Carlos Alfonso. Dynamic management of virtual infrastructures. *Journal of Grid Compututing*, 13(1):53–70, marzo 2015.

[5] J. Olin Campbell, John R. Bourne, Pieter J. Mosterman y Arthur J. Brodersen. The Effectiveness of Learning Simulations for Electronic Laboratories. *Journal of Engineering Education*, 91(1):81–87, enero 2002.

[6] D. M. Fraser, R. Pillay, L. Tjatindi y J. M. Case. Enhancing the Learning of Fluid Mechanics Using Computer Simulations. *Journal of Engineering Education*, 96(4):381–388, octubre 2007.

[7] DM Fraser, R Pillay, L Tjatindi y JM Case. Enhancing the learning of fluid mechanics using computer simulations. *Journal of Engineering Education*, 96(4):381–388, 2007.

[8] José A. González-Martínez, Miguel L. Bote-Lorenzo, Eduardo Gómez-Sánchez y Rafael Cano-Parra. Cloud computing and education: A state-of-the-art survey. *Computers & Education*, 80:132–151, 2015.

[9] Esra Güney, Ziya EkşI y Murat ÇAkıRoğLu. Webecg: A novel ecg simulator based on matlab web figure. *Advances in Engineering Software*, 45(1):167–174, 2012.

[10] Sarah Heckman y Jason King. Developing software engineering skills using real tools for automated grading. En *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, SIGCSE ’18, pp 794–799, New York, NY, USA, 2018. ACM.

[11] Coromoto León Hernández, Gara Miranda Valladares, Casiano Rodríguez León, Eduardo Segredo González y Carlos Segura González. Integración de las herramientas"github education.[en] el aula. En *De la innovación imaginada a los procesos de cambio*, pp 333–346. Servicio de Publicaciones, 2018.

[12] Jakub Kolota. A remote laboratory for learning with automatic control systems and process visualization. *The International journal of engineering education*, 27(5):1130–1138, 2011.

[13] Miguel A Rubio, Carolina Mañoso Hierro y APDY Pablo. Using arduino to enhance computer programming courses in science and engineering. En *Proceedings of EDULEARN13 conference*, pp 1–3, 2013.

[14] J. Damian Segrelles y Germán Moltó. Assessment of Cloud-based Computational Environments for Higher Education. En IEEE, editor, *2016 IEEE Frontiers in Education Conference Proceedings*, 2016.

[15] J. Damian Segrelles, Germán Moltó y Miguel Caballer. Remote Computational Labs for Educational Activities via a Cloud Computing Platform. En *2015 Proceedings of the Information Systems Education Conference (ISECON)*, pp 309–321, 2015.