Como citar este artículo: G. Wolf, "Using the Git Version Control System to Replace a Learning Management System," in IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 19, pp. 24-32, 2024, doi: 10.1109/RITA.2024.3368293.

Uso del sistema de control de versiones *Git* en reemplazo de un sistema de administración de la enseñanza

Gunnar Wolf

***Resumen*—Desde hace varios años, pero con especial fuerza a partir del inicio de la contingencia por COVID, los profesores de educación superior han acudido a herramientas telemáticas que ayuden al docente a gestionar los trabajos con que sus alumnos practican y demuestran su avance en clase — Tareas, prácticas, proyectos, exposiciones y demás. Los sistemas que, en el transcurso de las últimas dos décadas, más se han utilizado para tal fin toman el nombre de Sistemas de Administración del Aprendizaje (LMS, por sus siglas en inglés); el más conocido es indudablemente Moodle, pero hay una gran cantidad de opciones a éste.**

A partir de la observación empírica de un rechazo por parte de los alumnos al uso de estos sistemas, los autores tomaron la decisión de reemplazar por completo su uso por el de un sistema de control de versiones ampliamente empleado para el desarrollo de software, *Git*, específicamente para la enseñanza a alumnos de la carrera de Ingeniería en Computación, para la materia de Sistemas Operativos.

Este artículo presenta la justificación para la elección de esta plataforma, y la evaluación de la experiencia tras ocho semestres empleándola.

I. Introducción

La experiencia que presenta este trabajo se ubica dentro de la enseñanza de la materia *Sistemas Operativos* de la carrera de *Ingeniería en Computación*, impartida en la Facultad de Ingeniería de la Universidad Nacional Autónoma de México.

El artículo inicia con una revisión de conceptos. La presente Sección presenta los conceptos de sistemas de administración del aprendizaje (LMS) así como de los sistemas de control de versiones (SCVs). Naturalmente, en los últimos 30 años se han documentado varias experiencias del uso de SCVs como parte del flujo de trabajo de los grupos lectivos; la Sección II documenta trabajos relacionados, contrastando sucintamente sus contribuciones o las diferencias con el modelo reportado por el presente trabajo. La Sección III describe el funcionamiento, planteamiento e interacción básica de los alumnos con el SCVs. A efectos de validar la experiencia como útil o exitosa a ojos de los alumnos, en noviembre de 2020 se aplicó una encuesta, misma que fue respondida por el 13 % de los alumnos. La Sección IV presenta los resultados de dicha encuesta. Por último, la Sección V presenta conclusiones generales sobre la experiencia reportada y delinea posibles caminos futuros para profundizar en ella.

I-A. Antecedentes: Sistemas de administración del aprendizaje (LMS)

Las herramientas de enseñanza a distancia, sea para permitir que los alumnos aprendan al ritmo y horario que les sea posible, o sea para hacer llegar la enseñanza a lugares carentes de cobertura del sistema educativo en cuestión, no son ni una idea novedosa ni un desarrollo reciente. Antes del desarrollo de las tecnologías de la información y las redes de datos de escala global, diversos centros de aprendizaje surgieron por todo el mundo con respuestas variadas; ejemplo de esto son los cursos por correspondencia o las *telesecundarias* [1].

Con la masificación del acceso a Internet en la década de 1990 surgieron diversos sistemas que, utilizando Internet, permitieron la distribución de materiales educativos a los alumnos siguiendo el ritmo y la lógica del cursado, y permitieron también que éstos emplearan el mismo medio de forma bidireccional — foros para expresar sus dudas, espacios de entrega para tareas y exámenes. Si bien esto inició empleando simples sistemas de gestión de contenido (CMS, por sus siglas en inglés), la posibilidad de gestionar aspectos administrativos del cursado (definir cursos, matricular alumnos, registrar profesores, definir relaciones entre ellos, llevar informes de progreso y calificaciones) y el desarrollo de módulos para una interacción específica al entorno educativo fue definiendo a sistemas especializados, denominados *Sistemas de administración del aprendizaje* [2].

Dadas las facilidades que proporcionan al trabajo docente, la adopción de LMS pronto excedió al planteamiento original de utilizarse para educación a distancia o mixta, para convertirse en herramienta habitual también para el cursado tradicional en el aula. Un LMS puede informar a los alumnos de nuevas tareas o de cuándo es la fecha máxima para entregar sus trabajos. Proporciona al docente un punto único de entregas digitales, reemplazando al tradicional portafolio lleno de tareas. Permite al docente diseñar cuestionarios y exámenes que los alumnos pueden responder, incluso fuera de las horas de clase — e incluso permite que dichos exámenes se califiquen de forma automatizada, resultando en importantes ahorros de tiempo para el docente [3].

I-B. Antecedentes: Marco tecnológico de los SCVs

El desarrollo de software, a cualquier escala seria, siempre ha requerido de la coordinación de esfuerzos entre desarrolladores, y resultaría natural que herramientas con éste fin hayan sido desarrolladas desde muy temprano en la historia de la computación. El modelo de desarrollo que se seguía hasta la década de 1960 era muy diferente, pero conforme se popularizó el uso interactivo de las computadoras mediante los sistemas de acceso compartido y las terminales interactivas, es-

ta necesidad se hizo patente. El primer documento encontrado que cubre SCVs menciona (traducción propia):

> Dado que *sccs* representa un cambio tan radical de los métodos convencionales de control de código fuente, resultó claro cuando comenzamos a desarrollarlo (a fines de 1972) que presentar un artículo con la especificación no sería suficiente para "vender" el sistema a los proyectos de software para los que estaba pensado; debíamos tener un prototipo funcional. [4]

Tal como lo supusieron los autores, la idea tardó varios años en lograr la tracción necesaria para ser común en el desarrollo de proyectos de software; hacia mediados de la década siguiente, las principales herramientas para este fin eran *sccs* y *RCS* [5]; hacia fines de los ochenta, y motivado porque los tres participantes de un proyecto tenían horarios incompatibles entre sí, se desarrolló *CVS*, inicialmente un conjunto de funciones ayudantes sobre *RCS*, que facilitó el desarrollo paralelo y sin coordinación explícita [6]. Varios años más tarde, resulta claro: "una forma de predecir si un proyecto de software tendrá éxito es preguntar, ¿utilizan sus desarrolladores un sistema de control de versiones para coordinar su trabajo?" [7]

Los grandes proyectos de software libre iniciados a principios de los noventa (particularmente, los sistemas operativos Linux, FreeBSD, NetBSD y OpenBSD, y una gran cantidad de aplicaciones) iniciaron su desarrollo empleando *CVS* como punto nodal. Cabe mencionar que, además de proyectos de desarrollo técnicos, los aquí mencionados comparten un factor socioideológico: el movimiento ideológico del software libre busca la creación de un cuerpo de software que no imponga a sus usuarios y posibles desarrolladores las limitaciones derivadas de la ley de derecho de autor, y muy particularmente, permita la reapropiación, modificación y redistribución del código fuente [8].

La década de los noventa fue testigo de un crecimiento vertiginoso tanto en los proyectos de software libre como en la capacidad de los equipos de cómputo y la universalidad del acceso a red (puede argumentarse que el primero se debió a la conjunción de los otros dos). El modelo de uso de *CVS* comenzó a resultar insuficiente; *CVS* carecía de capacidad para representar acciones tan comunes como la eliminación o el cambio de nombre de un archivo e imponía un alto costo a trabajar con archivos que no fueran de texto plano.

Entre el 2000 y 2004 se desarrolló *Subversion*; éste sistema seguía el mismo modelo de interacción de *CVS*, corrigiendo estas y otras debilidades [9]. *Subversion* creció en pocos años y se convirtió en una de las principales herramientas de desarrollo en el ámbito del software libre.

Sin embargo, todos los SCVs mencionados hasta este momento operan de forma centralizada. Con el crecimiento de algunos proyectos a escalas de desarrollo inimaginables hasta ese momento (8,000 desarrolladores en 20 años, 15 millones de líneas de código y más de 37,000 archivos [10]), en 2002 Linus Torvalds tomó la controvertida decisión de dejar *CVS* y adoptar un SCV *distribuido* (SCVsD): un sistema en que no existiera una única copia maestra o servidor central, sino que cada copia del proyecto guardara toda la historia y estado, permitiendo sincronización entre ramas relacionadas. Sin embargo, dado que no había ningún SCV que cumpliera con los postulados del software libre e implementara el modelo distribuido, Torvalds tomó la decisión pragmática y polémica de adoptar *BitKeeper*.

Si bien políticamente la adopción de *BitKeeper* fue difícil y un punto recurrente de fricción con los puristas de la libertad de software, es indudable que ayudó tremendamente a recuperar la velocidad en el desarrollo de Linux, que venía sufriendo por varios años [11]. *BitKeeper* ofrecía una licencia gratuita (no libre) para los desarrolladores de software libre, siempre y cuando no se utilizara para competir con *BitKeeper* mismo.

Para 2005, se suscitó una discusión acerca de si la funcionalidad que Andrew Tridgell estaba agregando a Linux consistía una violación de la licencia [12], lo cual eventualmente llevó a que Torvalds mismo iniciara el desarrollo de un SCVsD, al que llamó *Git*.

En el mismo periodo de tiempo se desarrollaron varios otros SCVsDs libres, como *Monotone*, *Darcs*, *Mercurial* o *Bazaar*. Existen además otras alternativas propietarias, como *Team Foundation Server* de Microsoft. Sin embargo, *Git* ha resultado claramente ganador sobre de los demás, y hoy puede verse como *lingua franca*, ya no únicamente entre los desarrolladores de software libre, sino que en el mundo de la programación en general; ha aglutinado no únicamente a una clara mayoría de proyectos entre los SCVsDs, sino que ha logrado atraer a una gran cantidad de proyectos que tenían incluso más de veinte años de historia sobre SCVs centralizados [13].

De forma paralela a lo ya presentado, desde fines de los noventa comenzaron a aparecer las *forjas*, sitios Web dedicados al hospedaje de proyectos de software libre, a los cuales brindan recursos administrados como un espacio Web, seguidor de fallos, listas de correo — y un SCV. Según la información disponible en el mismo sitio primera de estas forjas, *SourceForge*, hospeda al día de hoy más de 500,000 proyectos y tiene "varios millones" de usuarios registrados. Sin embargo, el flujo de interacción en que está basado es poco amigable, lo cual lo hace apto únicamente para usuarios que ya son profesionales del desarrollo de software.

En 2008, y ya viendo un rápido crecimiento en la adopción de *Git* para proyectos de todo tamaño, nació *GitHub*: Una forja con un flujo de trabajo simplificado, y fuertemente centrado en el modelo de desarrollo de *Git*. Apenas tres años más tarde era ya la forja con mayor actividad en el mundo del software libre [14]. A la fecha de escritura del presente texto, según la información disponible en el sitio Web, hospeda a más de 100 millones de proyectos y más de 50 millones de desarrolladores.

Ahora, si bien *GitHub* se ha vuelto nodal para el desarrollo de esta impresionante cantidad de proyectos libres, causa una cierta disonancia cognitiva que *GitHub* mismo no es libre: El software con el cual opera el sitio Web, y que integra las diferentes herramientas que lo componen, es un desarrollo propietario. Hay otros servicios comparables, como *GitLab*, que incluso ofrece un modelo de colaboración y una semántica perfectamente mapeable contra la de *GitHub*. Resultaría probablemente más coherente con los principios

personales de uso y promoción del software libre con que se inició el proyecto que este texto reporta el desarrollar la experiencia que se relata a continuación en *GitLab* o alguna plataforma similar; la decisión de hacerlo en *GitHub* no fue tomada a la ligera, y se centra en la importancia que dicho sitio tiene para el desarrollo de software en general. Al día de hoy, prácticamente todos los proyectos libres de desarrollo están alojados en *GitHub* (sea que éste provee su espacio principal de desarrollo o que es elegido como un almacenamiento de respaldo o conveniencia).

I-C. Acerca del documento

Este trabajo se presenta como una re-elaboración y actualización de una versión muy preliminar, presentada en el Encuentro en Línea de Educación y Software Libre (EDUSOL) 2017, y publicado en extenso [15]; los principales aportes desde donde llega dicho texto son:

- Estudio realizado sobre un periodo mucho mayor
- Mayor profundización sobre trabajos relacionados
- Elaboración y reporte de una encuesta a los alumnos, validando varios de los puntos presentados

II. Trabajos relacionados

El uso de SCVs como mecanismo para la entrega de trabajos de clase no es una idea nueva. Pueden encontrarse reportes de experiencia desde 1989 [16], aunque en esa ocasión, al docente no le fue posible concluir la implementación por la dificultad de los alumnos en comprender el sistema. Citando al autor, en traducción propia:

> Se le dio a los alumnos una copia legible por computadora del código fuente, bajo un sistema de control de versiones. Para la desilusión de los instructores, el sistema de control de versiones no fue utilizado, y se dará mayor énfasis a los beneficios de dicho sistema en futuras iteraciones del curso.

Sin embargo, no fue posible localizar información respecto a experiencias posteriores respecto a esta experiencia temprana.

Varios trabajos reportan experiencias en el mismo sentido que el nuestro, aunque comprensiblemente, partiendo de diferentes herramientas, obedeciendo al punto en el tiempo en que se presentaron: En todos los casos, se refieren a SCVs centralizados [7], [17], [18]. Tras varios años de no encontrar literatura al respecto, una referencia mucho más cercana al trabajo que aquí se presenta puede encontrarse en el texto de Glassey [19].

Es importante recalcar que un SCVs no cubre toda la funcionalidad que presentan los LMS. La experiencia que el presente artículo busca reseñar se centra en el componente para comunicación de consignas del docente a los alumnos y de entregables en sentido inverso; no considera roles como supervisores, tutores y administradores, ni pretende cubrir funcionalidad como foros de participación, herramientas de retroalimentación, módulos educativos de contenidos, mecanismos de comunicación, etc. [20, Sección 2.1]

II-A. Cursos empleando SCVs centralizados

Casi todas las experiencias reseñadas presentaban el uso de un SCV siguiendo un modelo de un repositorio independiente por alumno. Esto resulta antitético con el uso obtenido que se espera de la implementación para trabajar con desarrollo colaborativo: Si cada uno de los alumnos tiene un repositorio independiente, ¿cómo se gestionan las entregas de trabajos grupales? ¿Debe cada miembro del equipo enviar por separado al profesor la misma entrega? ¿O uno sólo de ellos hace la entrega, notificando fuera de banda al profesor dónde buscar el trabajo? Al discutir respecto a este punto, Reid y Wilson mencionan incluso un importante punto que se considera nodal para la discusión [7]: "prevenir que los alumnos vean uno el trabajo del otro". Los sistemas de desarrollo colaborativo son, entonces, presentados en un entorno más bien adversarial, trabajando en contra de la colaboración natural que deberían implementar.

La implementación de Milentijevic presenta una experiencia construida sobre una base tecnológica de *Subversion* o *CVS*, pero necesariamente mediados por una interfaz Web [18]. No se maneja un único repositorio a lo largo del curso, sino que avanza separando cada entrega: a lo largo del curso, se genera un nuevo repositorio para cada tarea o proyecto; cada alumno debe obtener el repositorio. Las entregas, explican los autores, incluyen el ciclo de vida de cada uno de estos desarrollos: Definición y asignación de tareas, trabajo en la tarea, terminación y evaluación. Cada una de estas etapas requiere de un diferente tipo de interacción entre supervisor y alumnos, lo cual supone una gran cantidad de trabajo burocrático (cubierto por el sistema Web) para el modelo presentado.

La propuesta de Glassy contrasta con las dos anteriores, dado que su metodología no busca el uso de un SCV como un mecanismo para la entrega de tareas y proyectos, sino que como una herramienta para evaluar las prácticas de desarrollo de sus alumnos [17]. Cada alumno crea su repositorio *Subversion*, realiza el trabajo, y entrega por correo electrónico al repositorio entero. Glassy analiza los patrones de *commits* (conjuntos de cambios enviados) que forman parte del repositorio, y documenta patrones no demasiado distintos de los que se presentarán en la Figura 2.

II-B. Cursos empleando SCVs distribuidos

La sección de Trabajos relacionados de Glassy hace referencia a un curso en su misma universidad (Montana Tech) en primavera de 2005, basado en Darcs [17]. Este sería el primer caso de un curso sobre un SCVsD, y la experiencia sería muy interesante de analizar, pero no fue posible encontrar mayor información al respecto.

GitHub opera una comunidad denominada *GitHub Education Community*, dedicada a generar contacto entre docentes y alumnos y a discutir el uso de la tecnología de SCVsDs para fines educativos. Ofrece además ayuda organizacional para repositorios *Git*, mediante organizaciones que operan Salones *GitHub*; el flujo que presenta es específico a la configuración propuesta y no refleja al trabajo colaborativo en un espacio profesional. Cada alumno trabaja y realiza sus entregas en su

propio repositorio. *GitHub Classrooms* se desarrolló originalmente como software libre, pero fue convertido en un proyecto interno y cerrado de *GitHub* a inicios de 2020 [21].

Glassey publica una comparación de herramientas para llevar control de grupos, enfocándose en cursos implementados sobre *Git* y que utilizan *GitHub*, pero enfocado en las herramientas desarrolladas por los distintos docentes para facilitar el seguimiento de las entregas por parte de los alumnos [19]. El trabajo que se presenta aquí no tiene ni requiere de herramientas externas para este fin — si bien sólo se ha empleado con grupos medianos (hasta 40 alumnos), es opinión del autor que, con mucho, la mayor carga en tiempo para el docente es la calificación de las entregas, no el dar seguimiento a si los proyectos ya fueron entregados o no.

Una parte particularmente relevante del artículo de Glassey es la comparación entre los modelos de distribución de los diferentes sistemas: Siete de ocho flujos que revisa distribuyen los trabajos a los alumnos siguiendo el modelo que denomina ORpSpA (*One Repository per Student per Assignment*, un repositorio por alumno por entrega), y el restante, OBpP (*One Branch per Problem*, una rama por problema). La implementación aquí reseñada es independiente y novedosa respecto a todos los casos de la revisión de Glassey.

III. Implementación

La Facultad de Ingeniería ofrece varias instalaciones de sistemas LMS, particularmente el cluster llamado EducaFI, que cuenta con un Moodle para cada una de las divisiones de la Facultad, administrado por la Unidad de Servicios de Cómputo Académico (UNICA).

La experiencia docente que relata el presente trabajo inicia en enero de 2013 con el semestre 2013-2. Desde el primer curso, se tomó la decisión de emplear EducaFI como herramienta parcial para la entrega de tareas y para comunicar a los alumnos de bibliografía relacionada y otras actividades que se fueran cubriendo; el uso que se dio a Moodle como LMS fue relativamente limitado, con únicamente dos ejercicios realizados a partir de cuestionarios desarrollados con calificación asignada, y sin emplear ningún módulo adicional.

En general, la administración realizada por el grupo de becarios que conforma la atención a usuarios en UNICA es buena y ágil, pero circunscripta a sus políticas de operación; particularmente, el docente solicitaba que los cursos ya impartidos se preservaran como memoria y para poder comparar el progreso con siguientes experiencias. Por esto, y por ocasionales problemas de algunos alumnos con problemas en el manejo de sus cuentas, se tomó la decisión de migrar el LMS a otra instancia de Moodle, administrada por quien escribe, del Instituto de Investigaciones Económicas. Se consideró agregar algunos módulos de utilidad para la materia, como GeSHi para la presentación de fragmentos de código o el Virtual Programming Lab (VPL) para ofrecer un entorno de desarrollo donde pudieran resolver planteamientos de programación sin depender de un entorno específico; es necesario reconocer que esto quedó en mera intención por el tiempo que requiere conocer y aprovechar el entorno Moodle más allá de un uso casual.

Hacia el final del semestre 2016-2, se tomó la decisión de, en vez de buscar ofrecer a los alumnos herramientas que *simulen* a aquellas que encontrarán en un entorno realista de desarrollo de software, llevarlos a emplearlas en realidad, lo cual condujo a la adopción de *Git*. Tras evaluar los puntos presentados en el marco tecnológico, en la preparación del semestre 2017-1 se preparó un repositorio *Git* llamado *clase-sistop-2017-01*, mismo que se alojó en el espacio personal *GitHub* del docente, recibiendo la dirección https://github.com/gwolf/clase-sistop-2017-01; en iteraciones posteriores, la dirección se simplificó para ayudar a los alumnos a encontrarla, y a partir del semestre 2020-1, se ubicaron en el espacio de la organización de la Facultad, como puede apreciarse en el Cuadro I:

Cuadro I
URLs de los repositorios por semestre

2017-2	https://github.com/gwolf/sistop-2017-2
2018-1	https://github.com/gwolf/sistop-2018-1
2018-2	https://github.com/gwolf/sistop-2018-2
2019-1	https://github.com/gwolf/sistop-2019-1
2019-2	https://github.com/gwolf/sistop-2019-2
2020-1	https://github.com/unamfi/sistop-2020-1
2020-2	https://github.com/unamfi/sistop-2020-2

Con respecto a la estructura interna que se dio al repositorio, en un principio, contempló únicamente a los tres únicos directorios del repositorio para las entregas de los alumnos (*tareas*, *prácticas* y *proyectos*), con cada una de las entregas ubicada en un subdirectorio numerado dentro de éstos. Poco tiempo después, a ésto se agregó un directorio para las exposiciones, que si bien siguen lógicas de entrega distintas, pueden enmarcarse en el flujo general. Al poco tiempo, se hizo obvio que el repositorio *Git* resultaba también un medio ideal para transmitir contenido a los alumnos — Los ejemplos de código realizados en clase, referencias a sitios Web u otros recursos mencionados, etc.

Al presentar este esquema de trabajo, en la práctica 1 se explicita a los alumnos bajo qué supuestos se estructuran las entregas de la forma que se hace, mediante el siguiente planteamiento:

> «El repositorio del curso es un proyecto de desarrollo al que te interesa contribuir. Cuando el docente genera una nueva actividad, naturalmente "aparece" un importante defecto (un *bug*, en términos de desarrollo de software): tu participación no está registrada para esa actividad. Lo que debes hacer es proveer un parche para ese defecto. Para hacerlo, actualiza tu copia local del proyecto, desarrolla los pasos que corrijan al bug, y envíaselo al docente para que éste lo incorpore al proyecto principal mediante un *pull request*.»

El primer semestre que se utilizó *Git* no se pidió a los alumnos seguir un estándar de nombres para sus entregas, por lo que se presentó la complicación de rastrear la relación entre los archivos y el alumno que los había generado; esto puede apreciarse en alguno de los directorios de entrega de tareas, como el que presenta la Figura 1.

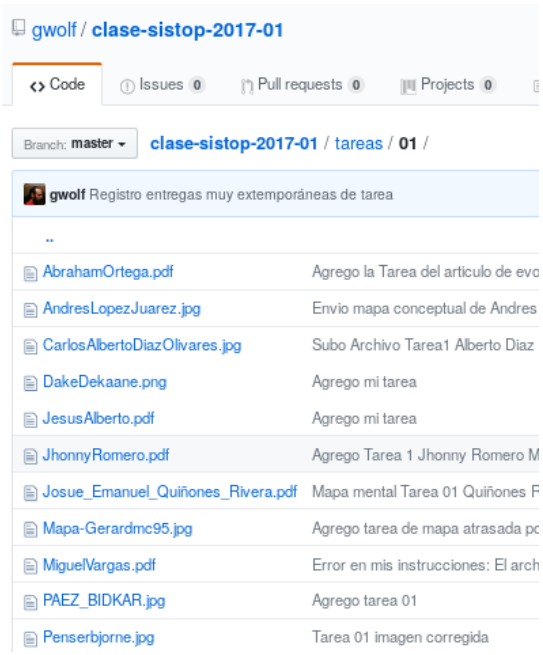

Figura 1. Ejemplo de directorio de entrega de tarea en el primer semestre de aplicación. Nótese que varios de los alumnos denominaron a sus archivos empleando pseudónimos, no nombres reales, dificultando la relación de las calificaciones.

Además de dicha adecuación, la estructura de los repositorios utilizados semestre a semestre se ha mantenido básicamente inalterada para los semestres subsecuentes.

Dado que, en general, los alumnos no están familiarizados con *Git* al iniciar el curso (ni con ningún otro SCV), fue considerado necesario el presentar su funcionamiento básico mediante tres prácticas; se explica a los alumnos que, para efectos de calificación, si bien la entrega de tareas es obligatoria (un alumno que no entregue la totalidad de tareas pierde derecho a exención, y un alumno que no entregue el 80 % de tareas pierde derecho al examen final en primera vuelta), las prácticas son opcionales, y su efecto únicamente es el de subir la calificación obtenida.

Las prácticas relativas al uso de *Git* que se llevan como parte del curso son:

1. **Uso de *Git* y *GitHub***: Consta de ocho pasos, con instrucciones completas y detalladas, al estilo tutorial. Pide al alumno que genere una cuenta en *GitHub* en caso de no tenerla, que haga un *fork* (copia en su espacio personal, donde tiene permisos de modificación) del repositorio principal, lo clone a su computadora personal, genere un archivo con su nombre dentro del directorio de entrega de dicha práctica, lo envíe a su *fork* en *GitHub*, y notifique al docente de sus cambios mediante un pull request (solicitud de incorporar los cambios de un *fork* en la rama principal).
2. **Ramas paralelas de desarrollo**: Una característica central del éxito de *Git* es que facilita trabajar en distintos aspectos de forma muy eficiente: Mediante el uso de ramas temáticas, un desarrollador puede estar trabajando sobre distintas características de forma independiente. El manejo de ramas permite que los alumnos puedan comenzar desde temprano a desarrollar sus proyectos y exposiciones (actividades que se busca que se repartan a lo largo del tiempo) sin que esto les impida hacer entregas menores, como otras prácticas o tareas.
3. **Eliminando archivos innecesarios**: Una importante provisión de todo SCV es la capacidad de ignorar los archivos autogenerados, resultantes de la compilación o ejecución de un programa. Esta práctica se incluye para evitar que los alumnos se confundan con los cambios en dichos archivos, y para llevarlos hacia buenas prácticas del desarrollo colaborativo.

En las experiencias que se ha tenido con el manejo de *Git* se ha dedicado también tiempo de clase para explicar el modelo; si bien no se hicieron mediciones precisas, el tiempo total dedicado se estima en menos de dos horas y media – Aproximadamente una sesión de clase en el transcurso del semestre. La Figura 2 ilustra una instantánea de uno de los repositorios mostrando cómo cada alumno diverge de la rama principal del repositorio para realizar sus entregas y cómo éstas van siendo incorporadas de vuelta a la rama principal; aunque se invita a consultar la versión pública e interactiva de dicha gráfica para explorarla y comprender mejor los diversos eventos que registra, desde las direcciones presentadas en el Cuadro I, haciendo click en *Insights* → *Network*.

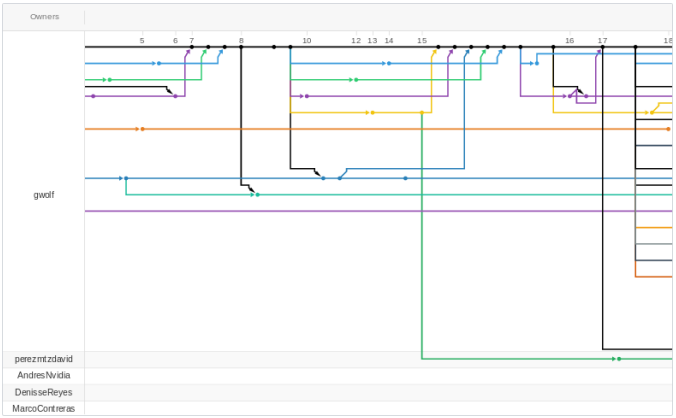

Figura 2. Gráfica de la red de interacciones entre repositorios para el semestre 2017-1. Cada punto representa un *commit*. Pueden observarse patrones que indican fechas de entrega (entre los días 15 y 16), el anuncio de una nueva tarea a realizar (día 17), manejo de líneas paralelas de desarrollo por parte de los alumnos (bifurcación de la línea azul alrededor del día 10 y de la amarilla el 17). Las líneas inferiores corresponden a alumnos que dejaron algún trabajo pendiente, al no haber enviado un *pull request*, o no haber cumplido con los cambios requeridos en alguno de ellos.

Fuera de las tres prácticas mencionadas y de la explicación que se realiza en clase, nada caracteriza en particular a la interacción de los alumnos con *Git* como específica al salón de clases: los alumnos realizan exactamente las mismas operaciones que realizarían en un proyecto no-académico.

IV. Resultados

Git no es un LMS, ni con el presente trabajo se busca presentar su aplicabilidad de forma genérica como si lo fuera. La experiencia que aquí se relata se circunscribe necesariamente al escenario planteado: Una materia orientada al desarrollo de software, y un compromiso personal del docente de formar

a los alumnos en el uso de las herramientas de desarrollo colaborativo que muy probablemente serán valiosas para su desempeño profesional.

La unidad de entrega en *Git* es el *commit*: Cuando un usuario determina que los cambios que realizó constituyen un *hito*, los agrupa bajo un *commit* (una traducción literal sería que el usuario *compromete* o que *adquiere un compromiso* sobre un conjunto de cambios). Para entregar un objeto (práctica, tarea, etc.), un alumno debe realizar un mínimo absoluto de un *commit*, después de lo cual notifica al docente mediante un *pull request* (solicitud de incorporación de cambios); el docente realiza un segundo *commit* por cada una de las entregas.

El Cuadro II muestra algunos datos respecto a los *commits* realizados en los repositorios de los siete semestres que aquí se reportan. La columna «Alumnos» indica el total de alumnos matriculados en el curso. C_{Tot} indica el total de *commits* que recibió el repositorio, de los cuales C_{Alum} indica cuántos fueron realizados por los alumnos, y C_{Doc} cuántos fueron realizados por el docente. Las últimas dos columnas presentan el número promedio de *commits* realizado por cada uno de los alumnos y el porcentaje de commits totales que fue realizado por los alumnos.

Cuadro II
COMMITS REALIZADOS EN LOS REPOSITORIOS, REPORTADOS POR SEMESTRE.

Semestre	Alumnos	C_{Tot}	C_{Alum}	C_{Doc}	$\frac{C_{Alum}}{Alum}$	$\frac{C_{Alum}}{C_{Tot}}$
2017-2	27	316	178	138	6.59	0.56
2018-1	21	473	294	179	14.00	0.62
2018-2	32	331	158	173	4.94	0.48
2019-1	31	599	388	211	12.52	0.65
2019-2	30	964	613	351	20.43	0.64
2020-1	33	1273	970	303	29.39	0.76
2020-2	43	777	528	249	12.28	0.68

Es de esperarse que el número total de *commits* por alumno varíe según el total de entregas que les fueron solicitadas; en la mayoría de los casos, los *commits* por alumno varían entre 12 y 20. Sin embargo, dado que el total de entregas requerido a los alumnos se ha mantenido entre 9 y 11, llaman la atención los semestres 2017-2 y 2018-2 por tener un promedio de *commits* por alumno cercano a la mitad de lo observado en otros semestres, y el semestre 2020-1 por ser cercano al doble.

Respecto los dos primeros casos, al explicar el modo de trabajo en *Git*, se invita a los alumnos a realizar varios *commits* en el proceso de desarrollo cada una de de sus entregas, como se haría con un proyecto real de desarrollo. Los alumnos, sin embargo, tienden a hacer *commits* únicos, dado que no ven motivación para hacerlo a lo largo del desarrollo. Es por esto que, en las entregas de proyectos más elaborados, se ha incluido como criterio de calificación el hacer más de cinco *commits*. Puede observarse que esto elevó la actividad total por alumno.

Y respecto al 2020-1, su valor elevado se debe a que el curso fue tomado por dos alumnos que ya tenían experiencia utilizando SCVs: Uno de ellos realizó 194 commits y el otro 152, subiendo fuertemente el promedio grupal.

Respecto a la apreciación de utilidad del conocimiento adquirido con el empleo de *Git*, si bien la experiencia había sido ya exitosa desde el punto de vista subjetivo del docente y en conversaciones informales con algunos alumnos, para tener certeza al respecto se aplicó una encuesta anónima a los alumnos de los semestres 2017-2 a 2020-2; no se envió al primer grupo en usar *Git* (2017-1) por no contar con sus direcciones electrónicas. La encuesta se generó en la plataforma gratuita en línea *QuestionPro*. La participación en la encuesta se buscó enviando un único correo a los 224 alumnos que formaron parte de alguno de los grupos en cuestión; la encuesta se mantuvo abierta para su llenado por una semana durante el mes de noviembre de 2020, y 30 de los alumnos enviaron sus respuestas. La distribución de respuestas sobre los grupos se presenta en el Cuadro III.

Cuadro III
PARTICIPACIÓN EN LA ENCUESTA

Semestre	Alumnos totales	Respuestas recibidas	% respuesta
2017-2	28	2	7 %
2018-1	22	4	18 %
2018-2	33	2	6 %
2019-1	32	3	9 %
2019-2	31	8	26 %
2020-1	34	10	29 %
2020-2	44	5	11 %

La encuesta consta de dos preguntas de selección múltiple, una pregunta de selección única, cinco preguntas de escala de Likert, y dos preguntas de texto abiertas. Se le presentaron a los alumnos divididas en tres grupos: 1. Antes del cursado, preguntas enfocadas a conocer a partir de qué puntos base llegaron al curso; 2. Durante el cursado, cómo fue su relación con *Git* en el transcurso del semestre, y 3. Después del cursado, cómo sienten que les haya resultado el aprendizaje de *Git* después de cursar la materia.

Las preguntas realizadas a los alumnos fueron redactadas de forma informal y divertida, buscando conectar con ellos y lograr una mayor tasa de respuestas que la que se estimó podría obtenerse con una escala Likert tradicional, orientando las preguntas a un rango de valores de «muy en desacuerdo» a «muy de acuerdo».

IV-A. Antes del cursado

Varios alumnos han cursado más de una vez la materia, razón por la cual la primera pregunta es de opción múltiple: ¿En qué semestre llevaste la materia de Sistemas operativos? (ver Figura 3). Los 30 asistentes indicaron 35 participaciones semestrales, lo cual indica que hasta cinco de ellos fueron repitentes.

Se le preguntó también a los alumnos qué tan familiarizados estaban con *Git* antes de cursar la materia; los resultados se presentan en la Figura 4.

Puede apreciarse que, si bien la generalidad de los alumnos sabía ya algo respecto a *Git*, su familiaridad era muy baja; únicamente el 26 % de los encuestados lo habían empleado activamente.

IV-B. Durante el cursado

Las preguntas de esta sección se presentaron en escala Likert de cinco puntos. El texto habitual de la escala Likert

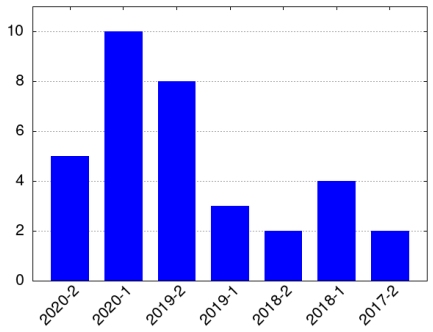

Figura 3. ¿En qué semestre llevaste la materia?

Figura 4. ¿Conocías a *Git* antes de cursar la materia?

se reemplazó por equivalentes semánticos para faciltiar el acercamiento a los alumnos.

Como se describió con anterioridad, el manejo básico de *Git* se presenta con tres prácticas únicamente. ¿Es esto suficiente introducción como para conocer suficiente la herramienta para su uso a lo largo del semestre? La Figura 5 muestra que, efectivamente, los alumnos lo consideraron suficiente.

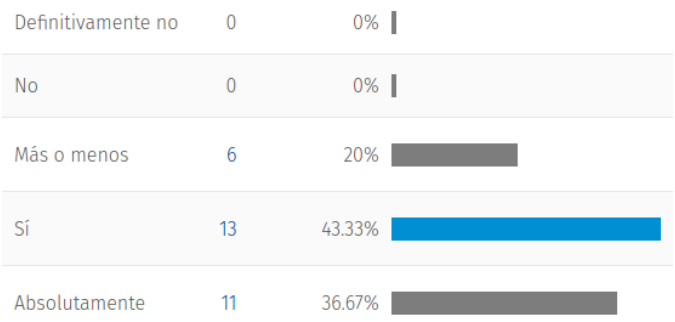

Figura 5. ¿Te parecieron suficiente presentación de *Git* las tres prácticas, mas las explicaciones en clase?

No es lo mismo el que una explicación sea suficiente a que el uso de la herramienta resulte natural. La Figura 6 muestra que, si bien el 16 % consideró difícil o muy difícil usar la herramienta, la opinión mayoritaria sigue siendo claramente positiva.

¿Qué tan bien respondió el uso de *Git* a la motivación original de presentar una alternativa a Moodle u otros LMS? La Figura 7 apunta a que los alumnos resultaron, en general, muy satisfechos con este cambio.

El uso de *Git* es sólo una parte (si bien fundamental) de la experiencia. El planteamiento se hizo siguiendo la terminolo-

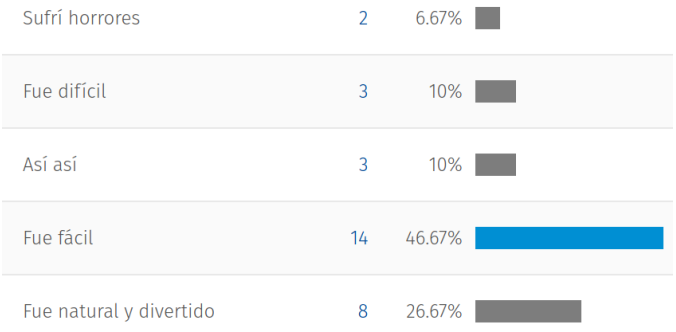

Figura 6. ¿Recuerdas qué tan difícil te resultó hacer tus entregas y coordinarte con tus compañeros de equipo?

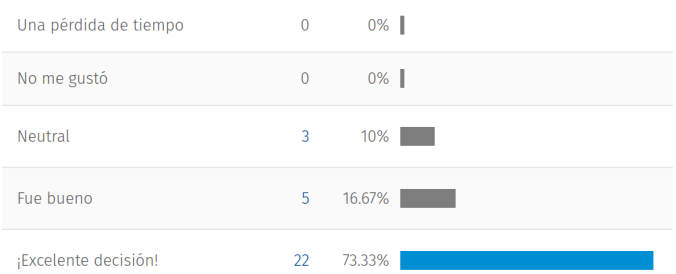

Figura 7. } ¿Qué te pareció el uso de *Git* en comparación con el de un LMS tradicional como Moodle?

gía y tipos de interacción habituales en la plataforma *GitHub*. A excepción de uno, los alumnos indicaron que pudieron comprender el modelo de *GitHub*, y que mayoritariamente le ven valor futuro, como lo indica la Figura 8.

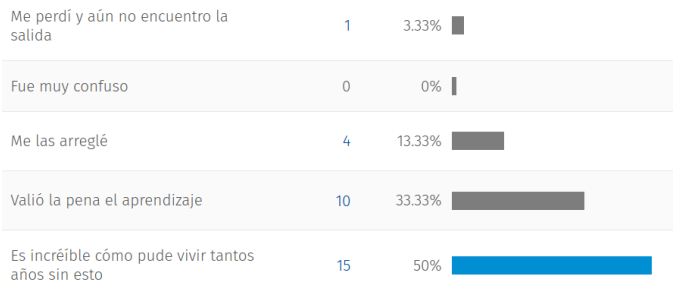

Figura 8. No únicamente fue *Git*: usamos este sistema mediante la interacción con el sitio Web *GitHub*. ¿Te pareció suficientemente clara la manera de trabajo, con las múltiples ramas, *clones* manejo de usuarios, *pull requests*, *pushes* y demás?

La última pregunta de esta sección es de texto abierto, pidiendo a los alumnos que mencionen casos de uso o características que hubieran deseado que hubieran sido cubiertas. Si bien algunos mencionan características que no fueron abordadas como el manejo de *issues*, los *stashes*, resolución de conflictos al hacer *merge* u otras, el sentimiento general es de satisfacción con el nivel alcanzado en lo que se refiere al cursado de la materia.

IV-C. Después del cursado

El proceso enseñanza-aprendizaje, sin embargo, debe evaluarse ante los resultados a largo plazo para los alumnos:

¿Qué tan útil les resultó el conocimiento adquirido? Para esto, una última pregunta presentada con un esquema Likert permite llegar con amplio optimismo al cierre de este reporte de adopción de herramienta: Como lo muetra la Figura 9, la totalidad de los alumnos le ven valor para su vida profesional, 70 % de ellos aseverando que *es/será mi pan de todos los días*.

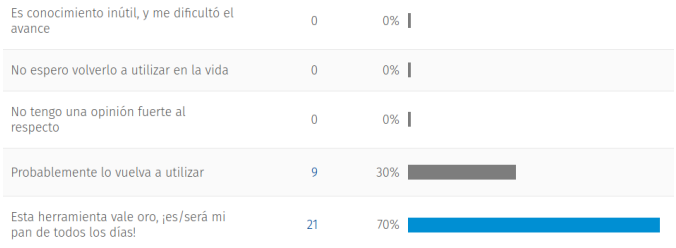

Figura 9. ¿Sientes que aprender el uso de *Git* te capacitó mejor para tu vida profesional?

En el tiempo transcurrido entre el cursado de la materia y el momento de aplicación de la encuesta, ¿han vuelto los alumnos a utilizar *Git*? ¿En qué entorno? Esta pregunta *no sigue* un esquema tipo Likert, sino que presenta cinco posibilidades, permitiendo seleccionar varias de las mismas. Como se puede constatar en la Figura 10, el 37 % de los participantes han utilizado a *Git* para sus proyectos personales y el 27 % para otras materias de su carrera.

Figura 10. ¿Has vuelto a utilizar *Git*? ¿En qué entorno?

Cerrando la encuesta, se da una última pregunta abierta pidiendo comentarios adicionales. El sentimiento es abrumadoramente positivo, con únicamente tres alumnos mencionando la dificultad que les presentó en su momento la herramienta; me permito citar a uno de ellos:

> Ja ja pues es curioso porque lo odié durante la materia porque no entendía realmente mucho, sin embargo una vez que le agarré la onda le tome un cariño inmenso y ahora ya es como mi uña y mugre, aun no soy un máster pero ya me defiendo, de hecho en mi actual trabajo no se usaba y fui yo quien lo propuso y prácticamente capacite a mis compañeros para que aprendieran a usarlo, así que en definitiva me alegro haber tenido un acercamiento antes donde podía equivocarme y me explicaban donde fue su materia.

Esas palabras resumen perfectamente, a fin de cuentas, la intención del docente al presentar el uso de *Git*: Brindar un reto intelectual que haga de los alumnos profesionales mejor capacitados para la vida profesional.

IV-D. Retos para la validez

El autor reconoce que el reporte de experiencia que aquí se presenta es meramente eso, una experiencia. Puede apreciarse en la lectura del texto que la adopción del modelo de interacción no deriva de un planteamiento riguroso de mejoría de prácticas educativas, sino que de una reflexión informal; el planteamiento no fue hecho *a obscuras*, y particularmente se consideró la arquitectura propuesta por *GitHub Classrooms* [21] y se consultó con compañeros docentes buscando las mejores ideas y prácticas, pero la revisión de trabajos relacionados que detalla la Sección II fue consecuencia, más que preparación, del trabajo realizado.

Si bien el total de alumnos que participaron en los cursos que utilizaron Git no es bajo (224), el que este trabajo no sea fruto de una planeación desde el principio del cursado causó que el total de alumnos que respondieron la encuesta fuera de apenas 30 — y no puede descartarse que la aplicación de la encuesta con hasta tres años de demora respecto al cursado inicial haya introducido un sesgo en sus resultados.

V. Conclusión

Después de ocho semestres concluidos y reportados en el presente trabajo, puede afirmarse que la prueba ha resultado claramente exitosa. El uso de *Git* no es, como ya fue dicho, natural y claro para alumnos de cualquier disciplina, pero por lo menos para las materias intermedias y avanzadas de Ingeniería en Computación, se invita a los docentes a adoptar los aprendizajes aquí reportados.

V-A. Trabajo futuro

Parte importante de la importancia de la adopción de SCVsD estiba en la colaboración entre desarrolladores; particularmente, los SCVsD han sido instrumentales para facilitar la colaboración en proyectos con alta dispersión geográfica. Considero que podría ser benéfico agregar alguna práctica que requiera que dos o más alumnos colaboren en un desarrollo, cada cual con su usuario, y realizar una entrega que conjunte sus trabajos. Si bien esto se ha observado espontáneamente en el caso de trabajo en equipo, no es una habilidad con la que cuenten todos. No se ha desarrollado una práctica en este sentido por considerarse innecesariamente complejo para el trabajo del curso, pero indudablemente ayudaría en la adopción de *Git* como una herramienta de colaboración.

La gestión de las entregas requiere cierto compromiso de tiempo por parte del docente, particularmente, asegurar que las entregas de los alumnos se realizan siguiendo la convención de nombres requerida. Se ha considerado el desarrollo de *hooks* (validadores de ejecución automática) que simplifiquen esto para el docente, pero no se han implementado por el potencial que tienen de confundir al alumno con mensajes de error inesperados.

V-B. Agradecimientos

El autor desea reconocer y agradecer al LIDSOL (Laboratorio de Investigación y Desarrollo en Software Libre) de la Facultad de Ingeniería, UNAM, por su ayuda y apoyo en este proyecto.

Referencias

[1] Z. Navarrete Cazales y P. A. López Hernández, "La telesecundaria en México", *Perfiles educativos*, vol. XLIV, n.º 178, págs. 63-78, 2022. dirección: https://doi.org/10.22201/iisue.24486167e.2022.178.60673.

[2] "Estado actual de los sistemas e-learning", *Education in the Knowledge Society (EKS)*, vol. 6, n.º 2, 2005. dirección: https://doi.org/10.14201/eks.18184.

[3] N. A. Alias y A. M. Zainuddin, "Innovation for better teaching and learning: Adopting the learning management system", *Malaysian online journal of instructional technology*, vol. 2, n.º 2, págs. 27-40, 2005. dirección: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=d5f63457fb7c53eb83bd2149d860753c3ebca662.

[4] M. J. Rochkind, "The source code control system", *IEEE Transactions on Software Engineering*, n.º 4, págs. 364-370, 1975. dirección: https://ieeexplore.ieee.org/abstract/document/6312866 (visitado 12-06-2019).

[5] W. F. Tichy, "RCS—a system for version control", *Software: Practice and Experience*, vol. 15, n.º 7, págs. 637-654, 1985. dirección: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.4380150703 (visitado 12-06-2019).

[6] B. Berliner et al., "CVS II: Parallelizing software development", en *Proceedings of the USENIX Winter 1990 Technical Conference*, vol. 341, 1990, pág. 352. dirección: https://www.tiffe.de/Robotron/PDP-VAX/rtVAX300/NetBSD6.0/usr/src/external/gpl2/xcvs/dist/doc/cvs-paper.pdf (visitado 12-06-2019).

[7] K. L. Reid y G. V. Wilson, "Learning by doing: introducing version control as a way to manage student assignments", en *Acm Sigcse Bulletin*, ACM, vol. 37, 2005, págs. 272-276. dirección: https://dl.acm.org/citation.cfm?id=1047441 (visitado 12-06-2019).

[8] S. Williams, *Free as in Freedom [Paperback]: Richard Stallman's Crusade for Free Software*. " O'Reilly Media, Inc.", 2011.

[9] C. M. Pilato, B. Collins-Sussman y B. W. Fitzpatrick, *Version control with subversion: next generation open source version control*. " O'Reilly Media, Inc.", 2008.

[10] J. Brockmeier, *Counting Contributions: Who Wrote Linux 3.2?*, 4 de abr. de 2012. dirección: https://www.linux.com/learn/counting-contributions-who-wrote-linux-32 (visitado 12-06-2019).

[11] V. Henson y J. Garzik, "Bitkeeper for kernel developers", en *Ottawa Linux Symposium*, 2002, págs. 197-212. dirección: http://courses.cs.vt.edu/cs5204/fall05-gback/papers/ols2002_proceedings.pdf#page=197 (visitado 12-06-2019).

[12] J. Barr, *Bitkeeper and Linux: The end of the road?*, 11 de abr. de 2005. dirección: https://www.linux.com/news/bitkeeper-and-linux-end-road.

[13] B. De Alwis y J. Sillito, "Why are software projects moving from centralized to decentralized version control systems?", en *2009 ICSE Workshop on Cooperative and Human Aspects on Software Engineering*, IEEE, 2009, págs. 36-39.

[14] K. Finley, *Github Has Surpassed Sourceforge and Google Code in Popularity*, jun. de 2011. dirección: https://readwrite.com/github-has-passed-sourceforge/ (visitado 12-06-2019).

[15] G. Wolf, "De Moodle a Git: Experiencia con el uso de un sistema de control de versiones (SCV) para reemplazar a un sistema de administración de la enseñanza (LMS)", en *Prácticas Abiertas*, A. Miranda, ed., ép. Educación y Cultura Libre, 2019, págs. 92-108. dirección: http://ru.iiec.unam.mx/4574/.

[16] B. J. Cornelius, M. Munro y D. J. Robson, "An approach to software maintenance education", *Software Engineering Journal*, vol. 4, n.º 4, págs. 233-236, 1989. DOI: 10.1049/sej.1989.0030. (visitado 25-06-2019).

[17] L. Glassy, "Using version control to observe student software development processes", *Journal of Computing Sciences in Colleges*, vol. 21, n.º 3, págs. 99-106, 2006. dirección: https://dl.acm.org/citation.cfm?id=1089195 (visitado 12-06-2019).

[18] I. Milentijevic, V. Ciric y O. Vojinovic, "Version control in project-based learning", *Computers & Education*, vol. 50, n.º 4, págs. 1331-1338, 2008. dirección: https://www.sciencedirect.com/science/article/pii/S0360131506001977 (visitado 12-06-2019).

[19] R. Glassey, "Adopting Git/Github within Teaching: A Survey of Tool Support", en *Proceedings of the ACM Conference on Global Computing Education*, ACM, 2019, págs. 143-149. dirección: https://dl.acm.org/citation.cfm?id=3309518 (visitado 25-06-2019).

[20] M. Pérez-Mateo Subirà y M. Guitert Catasús, *Aprender y enseñar en línea*, 2011. dirección: http://cv.uoc.edu/annotation/e5274644a40912f5e2fbad5191bd9123/564161/PID_00173067/modul_1.html.

[21] GitHub, *GitHub Classroom*, 2019. dirección: https://github.com/education/classroom/.