GIT-SEED: Estrategia para el trabajo colaborativo en cursos de Desarrollo de software

Marco A. Adarme Jaimes, Ph.D [1] y Judith de Pilar Rodríguez Tenjo, Ph. D[1], and Oscar A. Gallardo P, MSc[1]
[1]Universidad Francisco de Paula Santander, Departamento de Sistemas e Informática Colombia, madarme@ufps.edu.co, judithdelpilarrt@ufps.edu.co y oscargallardo@ufps.edu.co

Resumén– El desarrollo de software es considerado una actividad compleja debido a que intervienen un conjunto de personas especializadas en lenguajes y herramientas de programación que realizan tareas bien definidas dentro de la implementación del software. En este contexto de trabajo, se hace necesario que el estudiante en su inicio de formación como desarrollador de software, tome habilidades en buenas prácticas de desarrollo, una de los cuales comprende la utilización de sistemas de controles de versiones que permiten el registro y control de cambios de proyectos de software en repositorios específicos de código de programación donde interactúan varios usuarios. El propósito de este trabajo es presentar una estrategia de aprendizaje denominada GIT-SEED basada en casos donde el estudiante trabaje de forma colaborativa con el uso de un sistema de control de versiones bajo la plataforma GitLab, desde esta perspectiva se ofrece un mecanismo de apoyo para las habilidades de trabajo en equipo, análisis y pruebas de código de programación, despliegue de aplicaciones en entornos reales de trabajo y manejo de procesos de desarrollo de software ágiles. Los escenarios de prueba de la estrategia fueron aplicados a los estudiantes de estructura de datos y programación web del Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, los resultados demuestran la gran afinidad y apropiación que tienen los estudiantes en el uso de estas herramientas y la facilidad de integración de sus aplicaciones con los componentes creados en su equipo de trabajo.

Palabras claves -- Desarrollo de software, gitlab, sistema de control de versiones, trabajo colaborativo.

I. INTRODUCCIÓN

Dentro de la formación del estudiante en el área de desarrollo de software se hace necesario que no solo adquiera habilidades en resolución de problemas y razonamiento lógico, sino que, adquiera apropiación en el uso de herramientas tecnológicas para mantener y validar los productos software que realiza en cada una de sus actividades[1]; por lo tanto, es necesario una integración del trabajo que se realiza con la plataforma de despliegue que ejecutará su software, esto permite que el estudiante comprenda desde el inicio de su formación que el producto realizado tiene unas implicaciones considerables a nivel de requerimientos no funcionales y de despliegue sin dejar a un lado la satisfacción del requerimiento funcional para el cual su producto debe responder. Esta visión holística del desarrollo de software permite que el estudiante tenga contacto con la dinámica que es vista desde un entorno empresarial de trabajo[2], donde se irá encontrar con situaciones de: trabajo en equipo, despliegues continuos, pruebas e implementación de nuevos requisitos funcionales sobre una solución ya construida[3].

El concepto de trabajo en equipo y su importancia en el desarrollo de software han sido ampliamente estudiados en [4], [5] en ellos se destaca que son habilidades genéricas que debe tener todo profesional en está área y que en muchas ocasiones debido a la importancia que se le da a la apropiación de las habilidades técnicas en su formación, muchos graduados se les dificulta aprender a trabajar asertivamente en equipo[6][7]. Si bien, esta problemática pareciese ser cubierta con las diferentes estrategias que un profesor coloca en sus cursos de desarrollo o programación de software, no siempre son efectivas y carecen de orientaciones técnicas que ofrezcan un entorno de trabajo colaborativo a través de herramientas computacionales que el estudiante se irá a enfrentar en su práctica profesional[8], [9].

Desde el inicio de su formación en desarrollo de software, el estudiante adquiere habilidades para solucionar un problema en un dominio dado a través de un lenguaje computacional especifico, por lo general de alto nivel(Java, C#, phyton, entre otros), interactúa con un entorno de desarrollo integrado (IDE) que ofrece un conjunto de operaciones que ayudan a la escritura de su código fuente a través de sugerencias sintácticas y de escritura automática de código; no obstante, el proceso de creación de código dentro de un equipo de trabajo solo se basa en acciones de copiar y unir código, aumentando los esfuerzos a la hora de integrar su solución con la de sus compañeros, ya que no son tomados en cuenta características de cohesión y acoplamiento de código[10].

El objetivo principal de este trabajo de investigación es la creación de una estrategia de aprendizaje basada en casos que ofrezca al estudiante la posibilidad de apropiarse en temas de despliegue continuo y trabajo colaborativo a través de las siguientes acciones:

- Integrar la IDE a Gitlab como sistema de control de versiones (VCS).
- Registrar en repositorios todo código fuente en el VCS.
- Compartir su código fuente a diferentes usuarios identificando los roles respectivos.

Digital Object Identifier: (only for full papers, inserted by LACCEI).
ISSN, ISBN: (to be inserted by LACCEI).
DO NOT REMOVE

- Realizar integración continua de trabajos colaborativos.
- Auditar el proceso de creación de un código fuente por parte de cada uno de los integrantes de su equipo.
- Trabajar con metodologías de desarrollo ágiles que ofrecen la identificación de roles y trabajos dentro de la creación de un producto software.

II. GIT-SEED

Existen diferentes plataformas para usar VCS, las más frecuentes son GitHab y GitLab, ambos proveedores ofrecen a sus usuarios cuentas de acceso académico a través de un email con dominio ".edu" y con opciones de almacenamiento ilimitado y cierta cantidad de procesos para el despliegue continuo; con estas opciones un estudiante puede trabajar e integrar fácilmente a sus IDE los componentes necesarios para la comunicación de estos servicios. Para el propósito del trabajo de investigación se toma como referente GitLab, la IDE Netbeans versión 9.0 a la 11.0, principalmente por su facilidad de uso[11] y la integración desde la IDE como del sistema operativo que tenga el estudiante instalado en sus dispositivos personales (Linux, Windows o IOS).

La utilización de VCS en entornos educativos han demostrado que su uso continuo permite a los estudiantes adquirir habilidades en el desarrollo de software[12][13] basado en cinco pilares principales:

1. Interacción y comunicación asertiva de su equipo de trabajo
2. Lectura y entendimiento del código fuente escrito por cada integrante.
3. Realización de un proyecto software basado en contratos de uso que permita estandarizar las nombres entidades y operaciones (en el caso de un entorno orientado a objetos de métodos y/o clases
4. Control y ejecución de pruebas de aceptación e integración.
5. Reutilización de código.

En adición a esto, desde la perspectiva del aprendizaje, los ambientes colaborativos se refieren a un conjunto de personas que interactúan con sus conocimientos y habilidades en la resolución de un problema particular[8]. Este proceso requiere de un trabajo armonizado donde cada integrante desarrolla parte de la solución[14]; el reto en este punto es que el estudiante debe identificar efectivamente sus miembros y el rol que cada uno va a ejecutar dentro del desarrollo del proyecto.

Para el desarrollo de la estrategia, la investigación toma las experiencias ofrecidas en el área del aprendizaje colaborativo asistido por computador(CSCL) [10] [11] cuyo objetivo es estudiar como las personas construyen conocimiento de forma colaborativa en actividades sociales de interacción. Las soluciones a nivel de herramientas computacionales para CSCL son variadas para enfoques de aprendizaje de desarrollo de software, no obstante, el trabajo centra su atención en el uso en IDEs conocidos por los estudiantes y su forma de integrarse a un VCS, esta característica de diseño de la estrategia extiende la utilización del entorno de desarrollo, y de la ejecución y aprendizaje a través de modos de consola para la actualización de los diferentes repositorios que tengan los estudiantes.

A. Diseño de la estrategia para trabajo colaborativo

Por lo general, los estudiantes comienzan a adquirir habilidades en trabajo colaborativo en cursos de nivel superior donde son aplicadas técnicas y metodologías especificas dadas por la ingeniería de software convencional, que exige que se tengan en cuenta aspectos de análisis de requerimientos, desarrollo de modelos de datos, implementación, pruebas y despliegue; no obstante como se ha mencionado, en los inicios de la formación el estudiante realiza un trabajo simple en equipos que redundan en errores de cohesión y acoplamiento[17], debido a estos factores, la estrategia se divide en dos aspectos basados en dos actores: el profesor y el estudiante, tomando en cuenta esto, el trabajo no es solo de un actor , sino de la interacción que tenga directa mediada por los aspectos pedagógicos y técnicos involucrados en una solución software.

B. Actividad del profesor

La Fig. 1 muestra las actividades del profesor conducentes a la preparación y seguimiento del caso problémico planteado en un esquema de trabajo basado en la especificación de requerimientos funcionales.

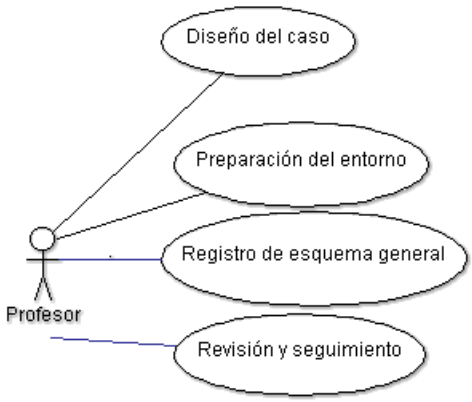

Fig1. Actividades del professor

El objetivo de la actividad del profesor es ofrecer el entorno o esquema completo que permita al estudiante identificar claramente los componentes software que interactúan con cada uno de los requerimientos funcionales proporcionados en la plantilla inicial de trabajo. De esta forma, sus actividades la conforman:

- Diseño del caso: El caso problémico se basa de una plantilla de trabajo que especifica:
    - Nombre del problema: Este debe ser un nombre llamativo y de un contexto real de trabajo.
    - Cantidad de integrantes: Se especifica la obligatoriedad de la cantidad máxima de participantes.
    - Descripción: Se describe el problema en términos del área a solucionar y los modelos matemáticos y/o abstractos que operan sobre la solución.
    - Especificación de los requerimientos funcionales: Se especifica uno a uno los requerimientos funcionales del problema. Estos requerimientos engloban backend y frontend.
    - Modelo de clases: Se realiza el modelo inicial de clases y/o paquetes de la aplicación.
    - Protocolo de entrega: Se establece una plantilla de trabajo donde el estudiante debe diligenciar la url de su proyecto en gitlab y asociar los requerimientos que irá a solucionar cada uno de los integrantes. Así mismo, se establece el integrante que servirá de integrador de la solución.
- Preparación del entorno: Se crea el entorno de trabajo a través de una plantilla de código donde solo se deja el esqueleto de las clases iniciales junto con interfaces asociadas a cada requerimiento funcional. La Fig.2, presenta el metamodelo de un esquema de trabajo típico para la resolución de un problema, esta idea fue tomada del proyecto de investigación SEED[18] para la implementación de estructuras de datos usando programación por capas.

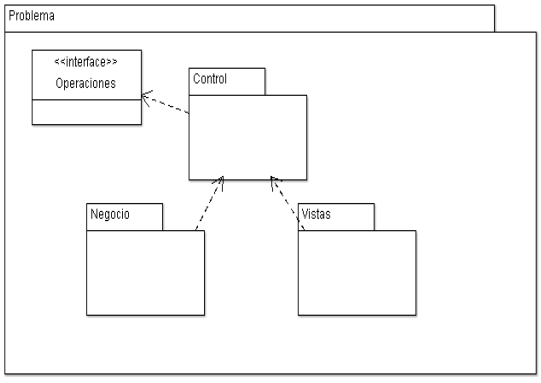

Fig2. Metamodelo de paquetes para trabajo colaborativo

De la Fig2, la clase interfaz "Operaciones" corresponde a la especificación de los requerimientos funcionales, ellas irán a interactuar dentro un entorno multicapa con los controles a su backend o frontend según sea el caso.

- Registro de esquema general: Después de creado el modelo de clases del paso anterior, el profesor crea las clases y paquetes iniciales y es registrado en un repositorio de código donde los estudiantes deben "clonar" a sus espacios de trabajo.
- Revisión y seguimiento: A medida que los estudiantes van registrando sus integrantes y URL de su proyecto de GitLab el profesor irá a realizar seguimiento de la implementación del código fuente de manera individual, analizando cada rama del proyecto y grupal cuando el proyecto se integre en su totalidad.

C. Actividad del estudiante

La Fig.3, presenta las actividades del estudiante, en él se destaca que es responsabilidad del integrador de la solución asignar las funciones de implementación directa por cada requerimiento funcional dado por el profesor. El historial de ramas creadas por cada proyecto será adjuntado al protocolo de entrega como muestra del desarrollo incremental que debe tener cada problema en particular.

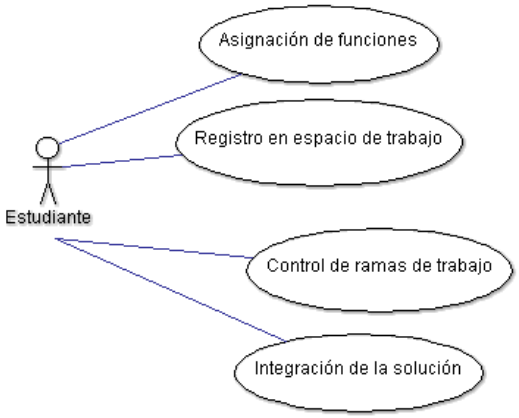

Fig3. Metamodelo de paquetes para trabajo colaborativo

De la Fig.3, la asignación del integrador de la solución y la asociación de cada requerimiento a ser implementado por cada integrante del equipo de trabajo. El registro de espacio corresponde a la clonación o copia del proyecto inicial de la plantilla de trabajo dada por el profesor, en este punto, el integrador de la solución debe asignar los roles respectivos propios de Gitlab para la interacción con el código fuente por parte de todos los equipos de trabajo.

El control de ramas de trabajo, este punto es propio de los VCS, donde cada rama corresponde a una copia del proyecto que es realizado por uno o más integrantes del equipo, de este modo, en Gitlab el trabajo es almacenado de forma incremental[19];es decir, no se simula un entorno de trabajo donde dos o más personas interactúan directamente sobre el proyecto (haciendo similitud a un archivo de texto compartido

en Google drive), sino que cada uno trabaja de forma independiente y a medida que la solución va creciendo las ramas se van actualizando hasta integrarla a la rama principal denominada "master"[20]. La confirmación de rama realizada, el integrante procede a realizar un "commit" para registrar sus cambios, estos mensajes contienen datos del autor y de la descripción de las tareas que ejecuto. De tal forma que una rama corresponde a un "commit" que será aprobado o no hasta llegar a la rama master o momento de integración.

III.RESULTADOS

Para la realización de la evaluación de la estrategia se escogieron 42 estudiantes de los cursos de Estructuras de Datos y 30 programación web de 3 y 6 semestre respectivamente del Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander (Cúcuta, CO) durante el I semestre 2020, para un total de 72 estudiantes.

Se realizaron 3 ejercicios para cada grupo funcional (estructuras de datos y programación web), usando como lenguaje de programación Java con la IDE Netbeans en sus versión 8 a la 11. Los equipos de trabajo son de tres estudiantes. Al finalizar cada ejercicio se analizaron las siguientes premisas a través de una escala de likert que califica el grado de dificultad para la utilización de gitlab en la IDE, de la configuración del entorno y de la integración de la solución (1= muy difícil, 2= difícil, 3= aceptable, 4= fácil, 5= muy fácil):

P1. El registro e interacción en gitlab de usuarios
P2. Registro del repositorio inicial con netbeans.
P3. Actualización de repositorios a través de netbeans.
P4. Trabajo con modo través de modo consola.
P5. Seguimiento de ramas en Gitlab
P6. Integración de la solución.

Las Tablas I y II presentan los resultados del grupo de estructuras de datos y de programación web.

TABLA I. Resultados estructuras de datos

Satisfacción	P1	P2	P3	P4	P5	P6
1	0%	0%	2%	19%	9%	0%
2	40%	0%	0%	38%	18%	23%
3	10%	5%	36%	12%	44%	27%
4	30%	25%	37%	11%	27%	33%
5	20%	70%	25%	20%	2%	17%

TABLA II. Resultados programación web

Satisfacción	P1	P2	P3	P4	P5	P6
1	0%	0%	0%	5%	5%	0%
2	0%	0%	0%	20%	9%	3%
3	11%	12%	17%	37%	13%	0%
4	43%	23%	39%	12%	39%	33%
5	46%	65%	44%	26%	34%	64%

De las tablas anteriores, con respecto al registro e interacción de usuarios los estudiantes de estructuras de datos denotan un 40% que es difícil de manejar, aunque el 50% entiende el manejo de la plataforma; a su vez esté número para los estudiantes de programación web que son de dos semestres más avanzados mas del 80% considera fácil su registro, en este punto se resalta que estos estudiantes ya vienen interactuando con SVC desde semestres anteriores. Del registro inicial del repositorio ambos grupos manifiestan que su interacción es fácil, este hecho es atribuible a que la IDE cuenta con módulos especializados para esta tarea. Con respecto a la actualización de repositorios, los estudiantes de estructuras de datos denotan dificultan, por lo general debido a que algunos estudiantes aún no comprenden el uso del repositorio y lo asocian a un trabajo donde interpretan su uso como la interacción directa de dos o más usuarios sobre el mismo proyecto, los estudiantes de programación web mas del 80% tiene un manejo de los procesos de actualización.

Como tarea adicional al trabajo con Gitlab a través de la IDE, también se realizan prácticas en entornos modo consola para la interacción de los repositorios, en este punto, los estudiantes de estructuras de datos presentan dificultades en comparación con los de programación web, no obstante, cabe resaltar que ambos grupos prefieren utilizar la IDE con respecto al modo consola.

Con respecto al trabajo con ramas del proyecto y su integración, ambos grupos demuestran un alto grado de usabilidad de GitLab, el porcentaje es mayor en el grupo de programación web.

IV.CONCLUSIONES

Este trabajo presenta una estrategia de trabajo colaborativo usando GitLab para los cursos de desarrollo de software, su objetivo fue el de proporcionar una serie de actividades encaminadas a la formalización de actividades en el ámbito de la programación de computadores. Los resultados demuestran la afinidad de los estudiantes con respecto al uso de este tipo de iniciativa, la motivación y compromiso al realizar las practicas evidenciaron la importancia de la innovación en procesos de enseñanza aprendizaje en está área. Así mismo, la utilización de herramientas de libre acceso para entornos académicos permite que los estudiantes puedan explorar opciones avanzadas en el campo del desarrollo de aplicaciones con integraciones continuas.

AGRADECIMIENTOS

Al Grupo De Investigación Y Desarrollo De Ingeniería Del Software de la Universidad Francisco de Paula Santander , por su apoyo logístico y academico para la ejecución del Proyecto.

REFERENCIAS

[1] G. Chen, "Programming Language Teaching Model Based on Computational Thinking and Problem-based Learning," vol. 156, no. Seiem, pp. 128–131, 2018, doi: 10.2991/seiem-17.2018.31.

[2] A. Ekuban, A. Mikroyannidis, A. Third, and J. Domingue, "Using GitLab Interactions To Predict Student Success When Working As Part Of A Team," 2020.

[3] J. Feliciano, M.-A. Storey, and A. Zagalsky, "Student experiences using GitHub in software engineering courses: a case study," in 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), 2016, pp. 422–431.

[4] T. Dingsoyr, T. E. Faegri, T. Dyba, B. Haugset, and Y. Lindsjorn, "Team performance in software development: Research results versus agile principles," IEEE Softw., vol. 33, no. 4, pp. 106–110, 2016, doi: 10.1109/MS.2016.100.

[5] A. B. Soomro, N. Salleh, E. Mendes, J. Grundy, G. Burch, and A. Nordin, "The effect of software engineers' personality traits on team climate and performance: A Systematic Literature Review," Inf. Softw. Technol., vol. 73, pp. 52–65, 2016, doi: 10.1016/j.infsof.2016.01.006.

[6] F. Ahmed, L. F. Capretz, S. Bouktif, and P. Campbell, "Soft skills and software development: A reflection from the software industry," Int. J. Inf. Process. Manag., 2013.

[7] G. Pan and P.-S. Seow, "Preparing accounting graduates for digital revolution: A critical review of information technology competencies and skills development," J. Educ. Bus., vol. 91, no. 3, pp. 166–175, 2016.

[8] L. M. Serrano-Cámara, M. Paredes-Velasco, C. M. Alcover, and J. Á. Velazquez-Iturbide, "An evaluation of students' motivation in computer-supported collaborative learning of programming concepts," Comput. Human Behav., vol. 31, no. 1, pp. 499–508, 2014, doi: 10.1016/j.chb.2013.04.030.

[9] J. F. V. Serrano, A. P. Abril, F. G. Bellas, and Á. S. Calle, Diseñar y programar, todo es empezar: Una introducción a la programación orientada a objetos usando UML y Java. Dykinson, 2010.

[10] M. A. Constanzo, S. I. Casas, and C. A. Marcos, "Comparación de modelos de calidad, factores y métricas," Inf. Científicos Técnicos-UNPA Técnicos-UNPA, vol. 6, no. 1, pp. 1–36, 2014.

[11] Y. Perez-Riverol et al., "Ten simple rules for taking advantage of Git and GitHub." Public Library of Science San Francisco, CA USA, 2016.

[12] S. Eraslan et al., "Integrating GitLab Metrics into Coursework Consultation Sessions in a Software Engineering Course," J. Syst. Softw., p. 110613, 2020.

[13] J. C. C. Ríos et al., "A methodology for using GitLab for software engineering learning analytics," in 2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), 2019, pp. 3–6.

[14] S. Azmi, N. A. Iahad, and N. Ahmad, "Gamification in online collaborative learning for programming courses: A literature review," ARPN J. Eng. Appl. Sci., vol. 10, no. 23, pp. 1–3, 2015.

[15] P. Dillenbourg, S. Järvelä, and F. Fischer, "The evolution of research on computer-supported collaborative learning," in Technology-enhanced learning, Springer, 2009, pp. 3–19.

[16] J. Andriessen, M. Baker, and D. D. Suthers, Arguing to learn: Confronting cognitions in computer-supported collaborative learning environments, vol. 1. Springer Science & Business Media, 2013.

[17] R. G. B. Valdivia, "Collaborative Learning Using Git with GitLab in Students of the Engineering Programming Course," 2019.

[18] M. Adarme and D. Jabba Molinares, "SEED: A software tool and an active-learning strategy for data structures courses," Comput. Appl. Eng. Educ., 2017, doi: 10.1002/cae.21885.

[19] J. Van Baarsen, GitLab Cookbook. Packt Publishing Ltd, 2014.

[20] K. Engwall and M. Roe, "Git and GitLab in library website change management workflows," Code4Lib J., no. 48, 2020.