UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE GUAYAQUIL

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE:
INGENIERO DE SISTEMAS

CARRERA:
INGENIERÍA DE SISTEMAS

TEMA:
“ANÁLISIS DE LA PRODUCTIVIDAD DE LA PROGRAMACIÓN EN PAREJA FRENTE A LA PROGRAMACIÓN INDIVIDUAL EN UN AULA UNIVERSITARIA”

AUTOR:
CRISTHIAN JONATHAN VILLEGAS VIZHÑAY

TUTOR:
Msg. RICARDO NARANJO SANCHEZ

JUNIO 2021
GUAYAQUIL-ECUADOR

DECLARATORIA DE RESPONSABILIDAD

Yo, **Cristhian Jonathan Villegas Vizhñay**, declaro que los conceptos y análisis desarrollados y las conclusiones del presente trabajo son de exclusiva responsabilidad del/los autor/es.

Firma del autor

Cristhian Villegas

Ricardo Naranjo Sánchez

ANÁLISIS DE LA PRODUCTIVIDAD DE LA PROGRAMACIÓN EN PAREJA FRENTE A LA PROGRAMACIÓN INDIVIDUAL EN UN AULA UNIVERSITARIA

Ricado Naranjo Sánchez [1] and Cristhian Villegas Vizhñay [2]

[1] Universidad Politécnica Salesiana, Campus Centenario, Robles 107 y Chambers, Guayaquil, Ecuador
rnaranjo@ups.edu.ec
[2] Universidad Politécnica Salesiana, Campus Centenario, Robles 107 y Chambers, Guayaquil, Ecuador
cvillegasv@est.ups.edu.ec

Resumen. El presente estudio explica el desarrollo de un proceso experimental sobre la comparación de estilos de programación, que tuvo como objetivo medir la productividad en la creación de software aplicando dos estilos diferentes: Programación en Pareja y Programación Individual. Se efectuó una búsqueda de información por diferentes bases de datos online relevantes sobre estudios experimentales que abarquen la comparación en el uso de las metodologías ágiles conocidas. Para cada estudio se verificó que aborden experimentos sobre la implementación de los estilos de programación para el desarrollo de software. Los resultados muestran que la experimentación en este campo, generan en la mayoría de estudios, resultados significativos en el desarrollo de software realizando la comparación de la productividad de ambos estilos de programación, para determinar la factibilidad que puede tener la programación en pareja sobre la programación individual y validar que esta sea aplicable en empresas dedicadas al desarrollo de software. Además, este estudio proporciona sugerencias para futuras experimentaciones donde se establezcan parámetros más detallados y tiempos de ejecución más largos para obtener resultados diferentes sobre la implementación de estilos de programación pertenecientes a la Programación Extrema.

Abstract. The present study explains the development of an experimental process about the comparison of programming styles; which was objective to measure productivity in software creation applying two different styles: Pair Programming and Solo Programming. There was a search for information by different relevant online databases about experimental studies that cover the comparison in the use of known agile methodologies. For each study it was verified that the address experiments about the implementation of programming styles for software development. The results show that experimentation in this area, in most studies, generates significant results in software development by comparing of the productivity of both programming styles, to determinate the feasibility that programming in pairs can have on individual programming and validate that it is applicable in companies dedicated to software development. In addition, this study provides suggestions for futures experimentation where more detailed parameters and longer execution times are established to obtain different results on

the implementation of programming styles belonging to the Extreme Programming.

Keywords: Pair Programming, Solo Programming, Agile Methodology, Software development.

1 Introducción

La ingeniería de software es la aplicación de un enfoque sistemático de los conocimientos, orientado a los métodos, diseño, implementación, operación, prueba, documentación y mantenimiento del software [1].

La introducción de diferentes metodologías ágiles en el desarrollo de software se ha vuelto relevante en la industria, volviéndose exitosa en varios países [2]–[5] . Estas metodologías ágiles nacieron con la necesidad de acelerar procesos de desarrollo en empresas que necesitaban con urgencia sistemas para mantener la competitividad de las mismas. No cuenta aún con evidencia necesaria para apoyar la mayoría de las prácticas actuales aplicables al desarrollo de software [6]. En esencia esto se debe a que no se posee una base teórica sólida. Una vía para obtener este conocimiento científico, es a través de la experimentación.

El desarrollo de software es realizado bajo un enfoque guía para el desarrollador, otorgando facilidad para la organización y el trabajo colaborativo para satisfacer los requerimientos de los usuarios finales [7], [8].

En las materias de programación de las universidades en general, es habitual que se trabaje de forma individual los ejercicios y prácticas de la clase [9], debido al espacio físico de los espacios de trabajo de los laboratorios de cómputo y a la cantidad de estudiantes inscritos en la materia [10].

La programación extrema creada por Kent Beck [11] a finales de los 90, es uno de los más conocidos enfoques ágiles [12], hace referencia a la programación en pareja como una práctica que brinda beneficios. Se han realizado diversos estudios empíricos que resaltan los efectos de la implementación de esta práctica [13]–[17].

Se propone realizar el diseño, ejecución y análisis de un experimento de desarrollo de software, teniendo como principal objetivo, observar la influencia que tiene sobre la productividad la aplicación de dos estilos de programación diferentes. Para ello, se contará con un grupo de sujetos de experimentación integrado por los alumnos de la materia de Ingeniería de Software de la Universidad Politécnica Salesiana Guayaquil (UPS-G), donde se plantea la siguiente interrogante:

- ¿Existen diferencias en la Productividad al aplicar los estilos de programación en pareja y programación individual?

Para este proceso se utilizaron dos estilos de programación: Programación en Pareja (PP) y Programación Individual (PI). En la PP, dos programadores diseñan, codifican y prueban el software de forma conjunta en un ordenador [2], mientras que en la PI, una persona se encarga de todo el trabajo de desarrollo.

2 Trabajo relacionado

Entre los trabajos que se han realizado analizando PP frente a PI, se pudo observar en algunos que la tarea en parejas mostró un menor tiempo de duración en completar una tarea a diferencia de realizarlo de forma individual, de igual manera se relatan otros beneficios (Lui et al., 2008; Nawrocki & Wojciechowski, 2001; Nosek, 1998; Ventura et al., 2017).

Algunos estudios mostraron resultados diversos, [19] utilizó 15 profesionales que fueron distribuidos en cinco parejas y cinco individuos, quienes programaron un script para bases de datos, donde los valores otorgados por el experimento reflejaron aceptación del PP mostrando una disminución del 29% en el tiempo de duración para culminar las tareas frente a la PI.

En el estudio de [18] emplearon 16 estudiantes para la experimentación que fueron distribuidos en cinco parejas y seis individuos, quienes resolvieron cuatro ejercicios. Los creadores del experimento no hallaron diferencias relevantes entre los estilos de PP y PI.

El experimento de, [16] empleó cinco parejas y cinco individuos, donde los creadores del experimento mostraron aceptación del PP reportando una disminución del 52% en el tiempo de finalización de la tarea a diferencia de la PI.

El estudio de (Ventura et al., 2017) utilizó 94 estudiantes como sujetos de estudio divididos en 54 parejas y 40 individuales, desarrollando el mismo software bajo las mismas condiciones. Los resultados reflejan que los alumnos que trabajaron en pareja tuvieron mejores resultados y percibieron un trabajo más fácil, a diferencia de los estudiantes que programaron de forma individual, pudiendo detectar con mayor rapidez los errores.

En el estudio de [20] se utilizaron 24 estudiantes quienes experimentaron tres modalidades de trabajo durante las sesiones normales de clase. Los resultados reflejaron que la programación en pareja fue más productivo y aceptado por los estudiantes, a diferencia de la programación individual y programación Mob (colectiva).

Los estudios mostraron resultados diversos: algunos reportaron beneficios con resultados cuantitativos diferentes, otros no encontraron diferencias significativas; es por ello que resultará positivo realizar otro experimento a fin de tener una mayor apreciación.

3 Metodología

La planificación o diseño experimental de software es la parte fundamental previa a la ejecución y análisis del experimento, en esta fase se definió y diseñó el experimento que se llevó a cabo. En la definición del experimento se determinó: hipótesis, selección de variables, selección de sujetos de experimentación, objetos; mientras que en el diseño se especificó: instrumentación, diseño del experimento, procedimiento de recopilación y análisis, y la evaluación de la validez.

En el experimento realizado se consideró el factor productividad y los estilos de Programación en Pareja y Programación Individual, como tratamientos del factor, para determinar el efecto de estos estilos en la productividad del desarrollo de software.

En este apartado se detalla el proceso del diseño del experimento, que de acuerdo a la variable respuesta "productividad" y los estilos de PP y PI, siguió un diseño de bloqueo aleatorio. Se diseñó la asignación aleatoria de 30 sujetos de experimentación en dos grupos, cada grupo correspondiente a un tratamiento.

3.1 Objetivos

Determinar si el estilo de programación empleado produce un efecto sobre la productividad de los sujetos en el desarrollo de software de baja complejidad.

3.2 Hipótesis

Para el planteamiento de las hipótesis, tanto nula como alternativas, se usó la notación descrita por Jedlitschka A. en [21], donde para cada objetivo se denota la hipótesis nula como H_0 y sus correspondientes hipótesis alternativas como H_1.

En este experimento se tuvo únicamente una hipótesis nula y una hipótesis alternativa, por lo tanto:

- H_0 = "No hay diferencia en la productividad obtenida por el desarrollo de software de baja complejidad con el uso de dos distintos estilos, Programación en Pareja y Programación Individual".
- H_1 = "Si hay diferencia en la productividad obtenida por el desarrollo de software de baja complejidad con el uso de dos distintos estilos de programación, Programación en Pareja y Programación Individual".

3.3 Variables

En este experimento se consideraron un conjunto de factores que pueden resultar relevantes para el experimento, es decir, pueden tener incidencia en los resultados del mismo siendo clasificados como se verá a continuación en variables y parámetros.

Las variables son aquellas que representan la causa y efecto del estudio:

- *Variable Dependiente:* Dentro del presente experimento solo se cuenta con una variable respuesta que se ha definido como "Productividad".
- *Variable Independiente:* Para la evaluación de la Productividad se tuvo en cuenta "Estilo de Programación" como variable independiente, que tiene dos niveles: Programación en Pareja y Programación Individual.
- *Variable de Bloqueo:* Se determinó la creación de un software que fue desarrollado en una sesión de "1 hora".

3.4 Parámetros

Los parámetros son las variables de control que deberían mantenerse constantes en la prueba del experimento, o aleatorias en su defecto para evitar que los resultados del estudio puedan resultar sesgados:

- *Tiempo para la ejecución del experimento:* Para controlar la influencia de este parámetro, el tiempo definido para la sesión del experimento es de 1 hora. En caso de que se complete el ejercicio en un tiempo menor, se indicará la hora en la que se ha terminado.
- *Lenguaje y herramienta de Programación:* Como estrategia de control para este aspecto, se ha definido que el desarrollo del ejercicio será realizado en el lenguaje de programación Python mediante la herramienta en la nube Google Colaboratory (Colab) para los estilos PP y PI.

3.5 Diseño del experimento

En este experimento se propuso utilizar un diseño de bloqueo aleatorio, haciendo un análisis estadístico y organizando dos grupos de experimentación, el grupo 1 conformado por 10 parejas y el grupo 2 conformado por 10 sujetos, para obtener los resultados del experimento. Además, se contempla el desarrollo de un software de baja complejidad en un tiempo estimado de 1 hora de experimentación. Se planteó solo un problema, cuyos requerimientos fueron especificados a los participantes del experimento al inicio de la sesión.

Para la configuración del experimento, se planteó la siguiente distribución en la siguiente tabla:

Tabla 1. Configuración del diseño experimental.

Sesión	Ejercicio	Tratamiento	Tiempo
1	A	Programación en Pareja Programación Individual	1 hora

Para este diseño planteado, con el fin de mantener una distribución equitativa en ambos grupos de experimentación, en relación a la cantidad de participantes, contando con un tiempo límite de 60 minutos (1 hora), se plantearon las siguientes características:

- Para la sesión, los sujetos experimentales fueron distribuidos aleatoriamente en dos grupos de trabajo, uno para aplicar Programación en Pareja y otro para aplicar Programación Individual.
- Se contó con 1 ejercicio durante la sesión, para que los que sujetos de ambos grupos de trabajo lo desarrollen

Tabla 2. Distribución de participantes en la sesión.

Sesión	Cantidad de sujetos Individual	Cantidad de sujetos Pareja	Total Sujetos
1	10	10	2*10 + 10 = **30**

3.6 Diseño del experimento

Se buscó que durante la ejecución del experimento todos los sujetos se encuentren en igualdad de condiciones técnicas y ambientales, por lo cual se definió la realización del mismo en uno de los laboratorios de computación de la UPS-G, donde todos los sujetos tienen acceso a computadores de escritorio con iguales características técnicas. De esta forma se controló la influencia sobre la variable respuesta que puede derivarse de algún problema técnico. En la Figura 1 se puede observar el diseño del experimento.

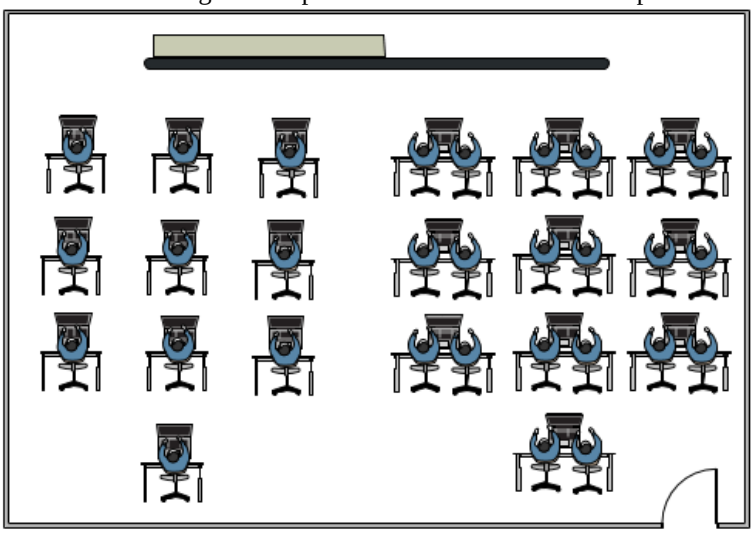

Fig. 1. Distribución de los grupos de experimentación en el laboratorio.

3.7 Participantes

Los participantes son alumnos de la Universidad Politécnica Salesiana, quienes tienen conocimientos básicos del lenguaje de programación Python y fueron distribuidos y organizados en equipos de desarrollo para la sesión, que estuvieron configurados por PP o PI. La Figura 2 muestra cómo fue la división de los participantes en base al tipo de programación empleado.

Fig. 2. Configuración Programación en Pareja y Programación Individual.

4 Resultados

4.1 Análisis Estadístico

Para realizar el análisis de este experimento, empleamos la Prueba T de Student, con el fin de evaluar si los grupos de experimentación (dos), muestran diferencias significativas en el resultado. Esta prueba se la realizó mediante el software SPSS Statistics.

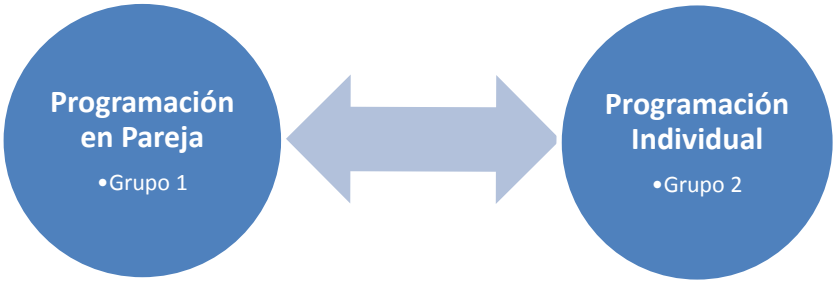

Fig. 3. Análisis de comparación entre la Programación en Pareja y Programación Individual.

Al ingresar los datos del experimento en el software SPSS Statistics, correspondiente a los sujetos de experimentación, éste nos arrojó los primeros resultados para verificar que los datos ingresados están correctos y no hay existencia de falta de información, tomando en cuenta que para el primer grupo "Programación en Pareja" fueron 10 parejas y para el segundo grupo "Programación Individual" fueron 10 sujetos. Se puede apreciar este enunciado con mayor detalle en la Figura 4.

Resumen de procesamiento de casos

		Casos					
		Válido		Perdidos		Total	
	Estilo	N	Porcentaje	N	Porcentaje	N	Porcentaje
Funcionalidad_Entregada	1	10	100,0%	0	0,0%	10	100,0%
	2	10	100,0%	0	0,0%	10	100,0%

Fig. 4. Verificación de los datos ingresados.

En la Figura 5 se puede apreciar un cuadro descriptivo donde se especifican algunos cálculos estadísticos de ambos estilos de programación, Programación en Pareja (1) y Programación Individual (2). El valor referencial en mencionada tabla es el valor de la Media el cual nos indica la diferencia numérica entre ambos estilos, teniendo la programación en pareja una media de 37.78 y la programación individual una media de 36.41. La diferencia entre los valores es de 1.37, donde aparentemente la programación en pareja resultó tener ventaja sobre el otro estilo de programación.

	Estilo			Estadístico	Desv. Error
Funcionalidad_Entregada	1	Media		37,3850	8,68561
		95% de intervalo de confianza para la media	Límite inferior	17,7368	
			Límite superior	57,0332	
	2	Media		36,4100	7,46623
		95% de intervalo de confianza para la media	Límite inferior	19,5202	
			Límite superior	53,2998	

Fig. 5. Cuadro descriptivo del análisis estadístico de Programación en Pareja y Programación Individual.

En el siguiente apartado se relatan los pasos utilizados para llevar a cabo la comparación del experimento y obtener el resultado de la variable respuesta "Productividad".

Paso 1: Redacción de Hipótesis.

- **Hipótesis:** La productividad en el Grupo 1 "PP" es mayor que la del Grupo 2 "PI".
  - **H_1= Existe** una diferencia significativa en la media de resultados de ambos grupos.
  - **H_0= No Existe** una diferencia significativa en la media de resultados de ambos grupos.

Paso 2: Determinar Alfa (α).

Nuestro experimento tuvo un análisis cuantitativo con una confianza del 95%, por lo tanto, se determina el porcentaje de error que tendrá el experimento:

$$\text{Alfa } (\alpha) = 5\% = 0.05$$

Paso 3: Lectura del Valor de la Prueba (P-Valor).

- **Normalidad:** La variable aleatoria (VA) debe seguir una distribución normal en los dos grupos. Se utilizó la prueba "Shapiro Wilk" para una muestra de tamaño **<30.** Para determinar la distribución normal de la VA, se determinó:
  - **P-Valor ≥ α** Aceptar **H_0**= Los datos provienen de una distribución **normal.**
  - **P-Valor < α** Aceptar **H_1**= Los datos **no** provienen de una distribución **normal.**

Pruebas de normalidad

		Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estilo	Estadístico	gl	Sig.	Estadístico	gl	Sig.
Funcionalidad_Entregada	1	,209	10	,200*	,890	10	,168
	2	,194	10	,200*	,951	10	,676

Fig. 6. Valores estadísticos prueba Shapiro-Wilk.

Tabla 3. Comparación de valores entre la prueba de Normalidad y el porcentaje de error.

Igualdad de varianza		
P-Valor (Grupo 1) = 0.890	>	α = 0.05
P-Valor (Grupo 2) = 0.951	>	α = 0.05

Conclusión: La variable productividad en los dos grupos se comportan normalmente.

- **Igualdad de Varianza:** Corroborar la igualdad de varianza mediante la prueba de Levene.
  - **P-Valor ≥ α** Aceptar **H_0**= Las varianzas son **iguales**.
  - **P-Valor < α** Aceptar **H_1**= Existe **diferencia** significativa entre las **varianzas.**

		Prueba de Levene de igualdad de varianzas	
		F	Sig.
Funcionalidad_Entregada	Se asumen varianzas iguales	,058	,813
	No se asumen varianzas iguales		

Fig. 7. Prueba de Levene.

Tabla 4. Comparación de valores entre la igualdad de varianza y el porcentaje de error.

Igualdad de varianza		
P-Valor (Grupo 1) = 0.813	>	α = 0.05

Conclusión: La igualdad de varianza en ambos grupos es la misma.

Paso 4: Resolución Estadística.

- **Calcular el Valor de la Prueba (P-Valor)**
  - Si la probabilidad obtenida **P-Valor ≤ α**, rechace H_0.
  - Si la probabilidad obtenida **P-Valor > α**, no rechace H_0.

Prueba de muestras independientes

		Prueba de Levene de igualdad de varianzas		prueb		
		F	Sig.	t	gl	Sig. (bilateral)
Funcionalidad_Entregada	Se asumen varianzas iguales	,058	,813	,085	18	,933
	No se asumen varianzas iguales			,085	17,603	,933

Fig. 8. Análisis de la Prueba T Student para la igualdad de medias.

Tabla 5. Comparación de valores de Prueba T Student y el porcentaje de error.

Prueba T Student		
P-Valor (Grupo 1) = 0.813	>	$\alpha = 0.05$

Conclusión Análisis SPSS: No Existe una **diferencia significativa** entre la media de resultados de ambos grupos.

5 Discusión

Una vez finalizado el experimento, se tomaron en cuenta las indicaciones dadas al inicio de la sesión para evaluar la programación realizada por los grupos de trabajo y verificar que el código entregado sea correcto y funcional. Los resultados del experimento mostraron que la programación en pareja puede ser el enfoque más apropiado para su implementación ya que presenta una diferencia de 1.37 en la media tal como se analiza en la Figura 8, sobre la programación individual. La percepción a simple vista parece ser factible con el primer método, pero aplicando el análisis estadístico mediante el software SPSS Statistics al ingresar las variables, se pueden evidenciar resultados más precisos mediante la Prueba de T Students para muestras independientes que nos permitió determinar la si la productividad es significativa en los tipos de programación aplicados en el experimento.

Nuestros resultados se asemejan a los de [2], [18], [22] al no encontrar diferencias en cuanto al tipo de programación aplicada (Programación en Pareja y Programación Individual). A diferencia de otros estudios como los de [10], [20] donde sus resultados reflejaron que la programación en pareja es el método aceptado por estudiantes y es un estilo que puede ser aplicado para una nueva modalidad de enseñanza en las universidades.

Hay que considerar que el estudio fue realizado con una muestra pequeña de participantes y un tiempo límite de 1 hora, por lo que será necesario, para una mejor interpretación de los resultados de este experimento, realizar una investigación que sea más profunda y que permita comprobar estos hallazgos.

6 Conclusión

El proceso experimental es un proceso complejo en el que intervienen una gran diversidad de factores que deben ser considerados en la fase de planificación, lo que representa una demanda de trabajo colaborativo y de coordinación entre los actores del proceso, para lo cual se hace necesario una comunicación constante por diferentes medios tecnológicos síncronos o asíncronos.

Los estudiantes que fueron evaluados en las dos modalidades de programación estudiadas en este experimento mostraron resultados similares con lo que podemos concluir que no existe una diferencia significativa en la variable productividad. La programación en pareja otorga beneficios al tener una segunda persona para poder resolver problemas durante la ejecución de una actividad en específico, pero la programación individual limita la interacción continua con otras personas y puede significar un bajo rendimiento si no se tiene clara la resolución de un determinado ejercicio.

Es necesario continuar realizando experimentos, donde se profundice y se establezcan parámetros más detallados que permitan obtener un resultado diferente sobre la implementación de estos estilos de programación pertenecientes a la Programación Extrema como parte de las Metodologías Ágiles aplicadas al desarrollo de software.

References

1. I. Xplore, "INTERNATIONAL STANDARD ISO / IEC / IEEEXplore, I. (2017). INTERNATIONAL STANDARD ISO / IEC / IEEE. 2017.," vol. 2017, 2017.
2. O. S. Gomez, A. A. Aguileta, R. A. Aguilar, J. P. Ucan, R. H. Rosero, and K. Cortes-Verdin, "An Empirical Study on the Impact of an IDE Tool Support in the Pair and Solo Programming," *IEEE Access*, vol. 5, pp. 9175–9187, 2017, doi: 10.1109/ACCESS.2017.2701339.
3. O. Salo and P. Abrahamsson, "Agile methods in European embedded software development organisations: A survey on the actual use and usefulness of Extreme Programming and Scrum," *IET Softw.*, vol. 2, no. 1, pp. 58–64, 2008, doi: 10.1049/iet-sen:20070038.
4. G. S. Matharu, A. Mishra, H. Singh, and P. Upadhyay, "Empirical Study of Agile Software Development Methodologies," *ACM SIGSOFT Softw. Eng. Notes*, vol. 40, no. 1, pp. 1–6, 2015, doi: 10.1145/2693208.2693233.
5. P. Rodríguez, J. Markkula, M. Oivo, and K. Turula, "Survey on agile and lean usage in finnish software industry," *Int. Symp. Empir. Softw. Eng. Meas.*, pp. 139–148, 2012, doi: 10.1145/2372251.2372275.
6. N. Juristo, A. M. Moreno, and S. Vegas, "A survey on testing technique empirical studies: How limited is our knowledge," *ISESE 2002 - Proceedings, 2002 Int. Symp. Empir. Softw. Eng.*, pp. 161–172, 2002, doi: 10.1109/ISESE.2002.1166935.
7. B. Choudhary and S. K. Rakesh, "An approach using agile method for software development," *2016 1st Int. Conf. Innov. Challenges Cyber Secur. ICICCS 2016*, no. Iciccs, pp. 155–158, 2016, doi: 10.1109/ICICCS.2016.7542304.
8. F. Nurullah, G. Wang, E. R. Kaburuan, and A. N. Fajar, "The Collaboration of DevOps Automation and SOA to Accelerate Software Development Culture," *1st 2018 Indones. Assoc. Pattern Recognit. Int. Conf. Ina. 2018 - Proc.*, pp. 262–266, 2019, doi: 10.1109/INAPR.2018.8627022.

9. K. Umapathy and A. D. Ritzhaupt, “A Meta-Analysis of Pair-Programming in Computer Programming Courses,” *ACM Trans. Comput. Educ.*, vol. 17, no. 4, pp. 1–13, 2017, doi: 10.1145/2996201.
10. R. Ventura, R. Hernández, J. Antonio, H. Izaguirre, A. L. Mendoza, and D. Bravo Herrera, “Comparación de la programación individual y por pares en los cursos universitarios,” *Temas Cienc. y Tecnol.*, vol. 21, no. 61, pp. 11–22, 2017.
11. K. Beck, *Extreme Programming Explained , Second Edition*. 2004.
12. M. M. Müller, “Two controlled experiments concerning the comparison of pair programming to peer review,” *J. Syst. Softw.*, vol. 78, no. 2, pp. 166–179, 2005, doi: 10.1016/j.jss.2004.12.019.
13. M. A. Domino, R. W. Collins, and A. R. Hevner, “Controlled experimentation on adaptations of pair programming,” *Inf. Technol. Manag.*, vol. 8, no. 4, pp. 297–312, 2007, doi: 10.1007/s10799-007-0016-8.
14. G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. A. Visaggio, “Evaluating performances of pair designing in industry,” *J. Syst. Softw.*, vol. 80, no. 8, pp. 1317–1327, 2007, doi: 10.1016/j.jss.2006.11.004.
15. T. Bipp, A. Lepper, and D. Schmedding, “Pair programming in software development teams - An empirical study of its benefits,” *Inf. Softw. Technol.*, vol. 50, no. 3, pp. 231–240, 2008, doi: 10.1016/j.infsof.2007.05.006.
16. K. M. Lui, K. C. C. Chan, and J. Nosek, “The effect of pairs in program design tasks,” *IEEE Trans. Softw. Eng.*, vol. 34, no. 2, pp. 197–211, 2008, doi: 10.1109/TSE.2007.70755.
17. S. Xu and V. Rajlich, “Empirical validation of test-driven pair programming in game development,” *Proc. - 5th IEEE/ACIS Int. Conf. Comput. Info. Sci., ICIS 2006. conjunction with 1st IEEE/ACIS, Int. Work. Component-Based Softw. Eng., Softw. Arch. Reuse, COMSAR 2006*, vol. 2006, pp. 500–505, 2006, doi: 10.1109/ICIS-COMSAR.2006.34.
18. J. Nawrocki and A. Wojciechowski, “Experimental Evaluation of Pair Programming,” *Eur. Softw. Control Metrics*, pp. 99–101, 2001, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.1689.
19. J. T. Nosek, “The Case for Collaborative Programming,” *Commun. ACM*, vol. 41, no. 3, pp. 105–108, 1998, doi: 10.1145/272287.272333.
20. V. R. Hern, S. Armando, and G. Moya, “Ramón Ventura Roque Hernández* | Sergio Armando Guerra Moya** | Adán López Mendoza***,” pp. 39–55, 2020.
21. A. Jedlitschka and A. Tn, “Reporting Guidelines for Controlled Experiments in Software Engineering Dietmar Pfahl,” pp. 95–104, 2005.
22. A. A. Aguileta and O. S. Gómez, “Estudio de calidad y eficiencia de un enfoque de desarrollo software secuencial con programadores solos y en pareja,” *Ingeniare. Rev. Chil. Ing.*, vol. 27, no. 2, pp. 304–318, 2019, doi: 10.4067/s0718-33052019000200304.