apertura
Volumen 12, número 1, pp. 39-55 | Universidad de Guadalajara
http://dx.doi.org/10.32870/Ap.v12n1.1791

Programación individual, por pares o colectiva: ¿cuál conviene utilizar en la universidad?

Solo, pair or Mob programming: Which should be used in university?

Ramón Ventura Roque Hernández* | Sergio Armando Guerra Moya** | Adán López Mendoza***

Recepción del artículo: 26/9/2019 | Aceptación para publicación: 11/12/2019 | Publicación: 30/3/2020

RESUMEN

El objetivo de esta investigación fue comparar la programación individual, por pares y colectiva a través de las percepciones de 24 estudiantes del curso universitario Programación intermedia con Visual Basic.Net. Los participantes fueron expuestos a las tres modalidades de trabajo en el desarrollo de proyectos de *software* durante sesiones regulares de clase. El diseño de investigación fue mixto explicativo secuencial. Para el componente cuantitativo se aplicaron cuestionarios, y para el apartado cualitativo se realizaron entrevistas. Los resultados mostraron que los alumnos prefieren la programación por pares porque la perciben como un punto medio entre no programar con nadie y hacerlo con un grupo numeroso. Programar de manera individual puede provocarles estrés y bloqueo intelectual, mientras que hacerlo con demasiadas personas al mismo tiempo les genera distracción y desbalance de trabajo. Una limitación del estudio es el tamaño de la muestra; sin embargo, hace aportaciones cuantitativas y cualitativas a un área de conocimiento poco explorada en la literatura formal. Los hallazgos sugieren promover el trabajo por parejas en los cursos universitarios de programación, pues se implementa con facilidad, con pocos recursos y buenos resultados.

Palabras clave
Desarrollo de *software*; educación universitaria; enseñanza de la tecnología; estrategias de enseñanza; programación

Abstract

The aim of this research is to compare individual, pair and Mob programming in university programming courses taking into account the perceptions of the students. 24 students participated in a course of Intermediate programming with Visual Basic. Net. They worked with the three modalities in the development of software projects during regular class sessions. A sequential explanatory mixed research design was used. For the quantitative component, questionnaires were administered. For the qualitative section, interviews were conducted. The results showed that students prefer pair programming because they perceive it as a midpoint between programming alone and doing it with a large group. Solo programming may cause stress and intellectual blockage, and doing so with too many people at the same time may generate distraction and imbalance of work among the participants. A limitation of the study is the sample size. However, the work makes quantitative and qualitative contributions in an area of knowledge little explored in formal literature. The findings suggest promoting work in pairs in university programming courses, as it is easily implemented with few resources and good results.

Keywords
Software development; university education; teaching technology; teaching strategies; programming

* Doctor en Ingeniería Telemática por la Universidad de Vigo, España. Doctor en Educación por la Universidad José Martí de Latinoamérica, México. Profesor investigador de la Universidad Autónoma de Tamaulipas, México. ORCID: https://orcid.org/0000-0001-9727-2608 | ** Doctor en Filosofía con Especialidad en Administración por la Universidad Autónoma de Nuevo León, México. Profesor investigador de la Universidad Autónoma de Nuevo León, México. ORCID: https://orcid.org/0000-0002-3369-8527 | *** Doctor en Educación Internacional por la Universidad Autónoma de Tamaulipas, México. Profesor investigador de la Universidad Autónoma de Tamaulipas, México. ORCID: https://orcid.org/0000-0003-4801-640X

INTRODUCCIÓN

El desarrollo de *software* es una actividad que debe realizarse con el apoyo de algún enfoque metodológico que facilite la organización de tareas, el trabajo colaborativo y el entendimiento de los requerimientos de las aplicaciones. No existe un enfoque que sea universalmente reconocido como el mejor para todos los escenarios, por lo que las diversas formas de trabajo deben evaluarse para elegir la más indicada de acuerdo con las necesidades de cada proyecto y cada equipo de personas.

En los cursos universitarios de programación es habitual trabajar de manera individual para llevar a cabo ejercicios y prácticas de clase (Umapathy & Ritzhaupt, 2017). Esto se debe a la disposición física de los laboratorios de cómputo y a la cantidad de alumnos que cursan estas asignaturas. En algunas ocasiones los estudiantes comparten una sola computadora debido a la escasez de equipos de cómputo disponibles o porque elaboran trabajos conjuntos; sin embargo, es fácil ver en estos casos que una persona conduce el proceso por tiempo indefinido, mientras que la otra solo observa, opina escasamente o adopta una actitud pasiva.

Karthiekheyan, Ahmed y Jayalakshmi (2018) explican que la programación por pares es una práctica que consiste en dos personas que trabajan en un mismo equipo de cómputo y se intercambian el teclado y el ratón en intervalos regulares. Ambas personas también cambian entre sí los roles de líder y seguidor, o conductor y navegante, si lo expresamos en la terminología propia de la programación por pares. Por su parte, Zuill (2015) define la programación colectiva (Mob) como tres o más personas trabajando en un mismo equipo de cómputo, espacio y proyecto. Esta forma de trabajo requiere un medio de amplificación de

la imagen proveniente de la computadora, lo cual puede hacerse a través de una pantalla de gran tamaño o con un proyector, como lo describe autor.

Aunque el término *programación Mob* surgió por primera vez a principios de la década pasada, su estudio y aplicación son recientes (Balijepally, Chaudhry & Sridhar, 2017). Algunas empresas han empleado este enfoque y han obtenido buenos resultados; sin embargo, si la bibliografía sobre este tema es escasa para el mundo de los negocios, lo es todavía más para el ámbito educativo. Estos dos entornos son distintos; por ejemplo, en las empresas existe mayor libertad para adecuar espacios de trabajo de acuerdo con lo requerido. En las universidades, por el contrario, las instalaciones tradicionales de los laboratorios de cómputo para la docencia son utilizadas por muchos alumnos; además, están limitadas a un espacio fijo, lo que podría dificultar la implementación de la programación Mob como estrategia de aprendizaje.

Si en los cursos universitarios de programación se puede trabajar de manera individual, por pares o en modalidad Mob, surgen las siguientes preguntas: ¿cuál de estos enfoques es el más apropiado?, ¿cuál es la percepción de los alumnos respecto a estas tres formas de trabajo?, ¿cuáles son las diferencias entre estas modalidades? En este artículo presentamos nuestra experiencia al aplicar los tres enfoques en un grupo de estudiantes universitarios con el objetivo de comparar sus percepciones y brindar respuestas a estas preguntas. Planteamos la hipótesis de que existen diferencias entre las percepciones de los alumnos acerca de las tres modalidades, y que los enfoques colaborativos (par y Mob) se privilegiarán sobre la programación individual.

El trabajo está organizado en cuatro secciones: la primera contiene los antecedentes que han sido reportados previamente en la bibliografía, la segunda describe la metodología que condujo nuestra investigación, la tercera comparte los resultados y su discusión, y la cuarta expone las conclusiones, las recomendaciones y los trabajos futuros.

ANTECEDENTES

Programación por pares

La programación por pares o por parejas fue propuesta como parte del enfoque ágil denominado "programación extrema" a finales de la década de los noventa; desde entonces, su popularidad ha crecido y su estudio se ha realizado desde diferentes ángulos de interés. Aottiwerch y Kokaew (2018), por ejemplo, consideran que cada pareja de programadores debe ser cuidadosamente elegida para que el trabajo sea efectivo; por esta razón, crearon un algoritmo que selecciona al mejor compañero de una persona con base en los siguientes criterios: actitudes, competencias de programación y comportamientos de aprendizaje.

A lo anterior, Poonam y Yasser (2018) añaden que los factores humanos y las características de la personalidad son también importantes para el éxito de los proyectos cuando se programa por parejas. En su investigación, identificaron que los aspectos técnicos de los proyectos se han estudiado más extensamente que los factores humanos. Al conducir un experimento, los autores encontraron una relación significativa entre las personalidades de los programadores, su ubicación de trabajo, local o remota, y el desempeño de las parejas.

Otro aspecto poco estudiado de la programación por pares es el uso de entornos integrados de desarrollo (IDE). Gómez y Aguileta (2018) hicieron aportaciones a esta área con una investigación cuyos resultados indican que, en proyectos sencillos, el uso de un IDE aumenta el número de defectos para programadores solitarios y parejas; sin embargo, en programas más complicados las parejas tienen significativamente menos defectos con el IDE. Estos autores concluyen que los estudiantes deberían realizar sus prácticas iniciales con un simple procesador de textos y un compilador, pero enseguida deben incorporar el uso de un IDE, en especial si trabajan en la modalidad de parejas.

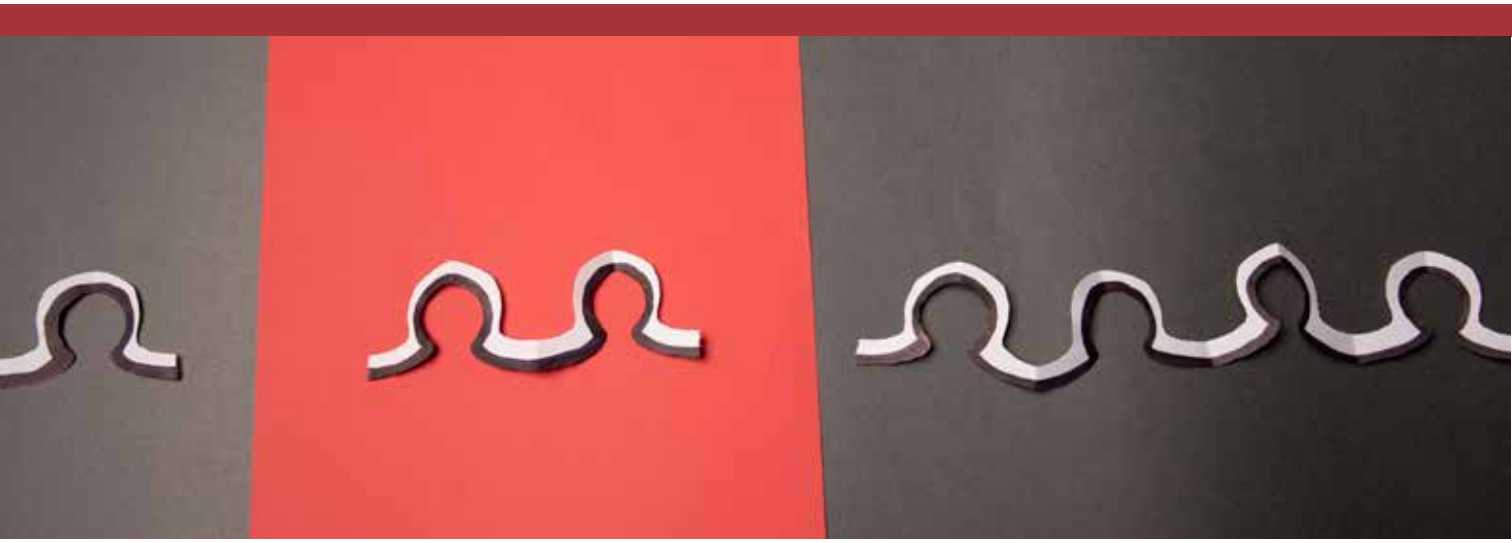
Sadath, Karim y Gill (2018) destacan la brecha que existe entre la enseñanza de la ingeniería de *software* en las aulas y la realidad actual en el mundo de los negocios. Por este motivo, ellos centran su estudio en la educación universitaria y proponen un marco de trabajo que reúne las mejores prácticas que han probado ser efectivas en la industria. El marco de trabajo incluye la programación por pares como práctica fundamental para potenciar los conocimientos compartidos entre los participantes.

Smith, Giugliano y DeOrio (2018) coinciden en la importancia de trabajar por parejas en los cursos universitarios de ingeniería y programación, pues prepara a los estudiantes para las actividades del mundo real, las cuales no se realizan con frecuencia en solitario. Estos autores efectuaron un estudio para determinar el efecto de la programación por pares en el desempeño académico de los alumnos a largo plazo. Sus resultados muestran que esta manera de programar tuvo un efecto positivo: quienes trabajaron en pares en los cursos introductorios obtuvieron después calificaciones más altas en cursos más avanzados.

Lee *et al.* (2016) llevaron a cabo una investigación para relacionar las horas de estudio y el desempeño académico de los estudiantes que programan en parejas. Encontraron que se obtiene un mejor desempeño para los estudiantes en riesgo de reprobar con mayores horas de estudio independiente antes de la programación por pares en el laboratorio. Concluyen que la preparación individual aunada a la programación en parejas resulta benéfica para evitar el bajo desempeño en estos cursos.

De acuerdo con Du *et al.* (2015), la programación por pares es una herramienta útil que mejora la comunicación entre los estudiantes de cada pareja, así como la comprensión de los temas académicos, sin importar las calificaciones de cada integrante. Estos autores también encontraron que el trabajo en parejas desencadena nuevas ideas. Para su estudio, utilizaron lenguaje C con los temas de flujo de control, funciones, apuntadores y archivos. Sus evaluaciones estuvieron basadas en entrevistas a estudiantes y en la percepción de los maestros. Por su parte, Saltz y Shamshurin (2017) destacan los resultados positivos de la programación por pares al aplicarla al área del análisis de datos con R. Estos autores realizaron una investigación con estudiantes organizados en parejas y observaron que los participantes mejoraron sus niveles de comunicación; además, escribieron mejor código en menos tiempo y obtuvieron buenas percepciones sobre los resultados que alcanzaron.

Swamidurai y Umphress (2015), por otra parte, argumentan que el desarrollo de *software*



no tiene que estar siempre acompañado de otra persona; por eso, proponen un enfoque llamado "programación por pares invertida", que consiste en dos programadores que inician diseñando soluciones juntos, pero se separan durante la implementación y después se reúnen de nuevo para realizar pruebas. Estos autores validaron su propuesta a través de dos experimentos, los cuales arrojaron que la programación por pares tradicional puede llegar a ser costosa y que la programación por pares invertida logra mayores o iguales niveles de calidad a un costo menor.

Para Meyer (2018), la programación por pares es una práctica de interés que debería usarse ocasionalmente, en especial en las partes más complicadas del *software* que se esté desarrollando. Él opina que no hay razón para imponerla como una modalidad única para crear *software* y recomienda no confundirla con la tutoría o acompañamiento, que es una actividad diferente. En este sentido, Meyer explica que los métodos ágiles no son una panacea, pues todavía tienen desventajas y retos que enfrentar. Asimismo, destaca la importancia de las mediciones y evaluaciones reales de estos enfoques para no caer en exageraciones o crear falsas expectativas.

Para Meyer (2018), la programación por pares es una práctica de interés que debería usarse ocasionalmente, en especial en las partes más complicadas del *software* que se esté desarrollando

Programación Mob

La programación Mob, también llamada *mobbing*, ha comenzado a ser vista con interés en el desarrollo de *software* orientados a los negocios. Zuill (2015) ha sido uno de los pioneros en dar a conocer detalles sobre este enfoque, al que él se refiere como "un paso más allá de la programación por pares". Zuill, basado en su propia experiencia, propone principios para que la programación Mob funcione con éxito, como tratar a los demás con amabilidad, consideración y respeto; aplicar los roles de conductor y navegador, en los que el conductor utiliza el teclado mientras el navegador le expresa su idea y lo guía para concretarla en el programa; rotar ambos roles cada quince minutos; utilizar el teléfono y el correo electrónico como si se tratara de una sola persona, lo que implica tener una sola cuenta de correo y atender las llamadas en altavoz en una única extensión telefónica. De igual modo, recomienda organizar reuniones retrospectivas para reflexionar sobre lo que se ha hecho bien y aquello que se puede mejorar.

Pyhäjärvi y Falco (2018), creadores del primer material en formato de libro dedicado completamente a la programación Mob, explican que la persona que está utilizando el teclado no debe pensar, solo debe dejar que las ideas, los pensamientos y las reflexiones del resto del equipo fluyan para capturarlas e implementarlas en la tarea que está ejecutando. Así, quienes no usan el teclado también adoptan una postura activa.

Actualmente, quienes han empleado la programación Mob en entornos de negocios expresan que, aunque pasaron por etapas de adaptación, han obtenido beneficios al trabajar en equipos de más de tres personas; sin embargo, todavía no queda claro cuáles son estos, en qué medida se obtienen y cómo se alcanzan, pues los aportes provienen, principalmente, de relatos vivenciales que son difíciles de cuantificar.

Schartman (2014) menciona que su equipo de desarrollo de *software* adoptó la modalidad Mob tras motivarse con una charla en la que se decía que se podía llegar a ser hasta diez veces

más productivo con este enfoque de trabajo. No obstante, apenas lograron ser igual de productivos como lo eran con otros enfoques que utilizaban antes. Schartman reflexiona sobre la importancia de contar con mediciones concretas sobre la programación Mob para que las expectativas estén apegadas a la realidad.

Por otra parte, Lilienthal (2017) destaca la falta de experimentos científicos orientados a conocer la utilidad de la programación Mob. Balijepally *et al.* (2017) coinciden en que, en la actualidad, apenas se cuenta con evidencia inicial de quienes han empezado a usar programación Mob, y que se requiere validación empírica tanto de ingenieros de *software* como de académicos. Por otro lado, identifican, a través del trabajo de Zuill (2015), los siguientes beneficios en la programación Mob: reduce los trámites administrativos, elimina obstrucciones en la comunicación de los participantes, disminuye la cantidad de decisiones que tomar para situaciones futuras; además, existe menor desperdicio, menos parches rápidos en el código, menos interrupciones externas, menos políticas, menos juntas extensas, más aprendizaje continuo, mayor satisfacción en los integrantes del equipo y mejor calidad en el *software*. Como posibles riesgos que pueden afectar la manera en la que el equipo trabaja, se señalan: la cultura organizacional, la falta de familiaridad con el desarrollo ágil, las personalidades dominantes, así como el cansancio físico y mental de los integrantes.

Balijepally *et al.* (2017) coinciden en que en la actualidad apenas se cuenta con evidencia inicial de quienes han empezado a usar programación Mob, y que se requiere validación empírica tanto de ingenieros de *software* como de académicos

Buchan y Pearl (2018) describen su experiencia en un equipo de desarrollo de *software* que trabajó con la programación Mob durante un mes para crear un producto dirigido al sector de servicios financieros. Ellos encontraron los siguientes beneficios en esta modalidad: las tareas se concluyeron más rápido, el equipo tuvo un mejor sentido de pertenencia hacia el código, el estilo de diseño y codificación fue más consistente, el uso de las herramientas fue más productivo, las personas conocieron más ampliamente el sistema en el que trabajaban, la confianza en el código se incrementó, los nuevos integrantes se incorporaron más rápido, y la estimación de tiempo requerido para las tareas fue más acertada.

Buchan y Pearl (2018) reportaron los siguientes riesgos y retos de la programación Mob: las personas pueden mostrarse renuentes a aceptar este enfoque de trabajo, también pueden estar motivadas al inicio y luego perder el interés, el código se genera lentamente cuando se comienza a programar en Mob, las relaciones interpersonales se incrementan y pueden alterar las capacidades del equipo para terminar el trabajo, algunas personas se aíslan y se les dificulta comunicarse con sus compañeros, puede ser problemático conseguir las instalaciones físicas y el equipo necesario.

Wilson (2015) reporta la experiencia de su equipo al trabajar con programación Mob y explica que el enfoque puede ser adaptado a las necesidades de los proyectos. Ellos, por

ejemplo, evolucionaron de una forma de trabajo puramente Mob a una híbrida que involucra programación por pares; utilizaron el enfoque Mob para modificar código de importancia crítica, encontraron que no todas las tareas requerían la atención de todo el equipo y se dieron cuenta de que, para ellos, este enfoque era más útil en escenarios donde se desconoce la solución que en los que esta es conocida, pero demanda mucho tiempo para implementarla.

Kerney (2015) describe cómo, a pesar de su personalidad fuerte, pudo convivir y trabajar con sus compañeros en la programación Mob y, al hacerlo de manera cotidiana, algunos rasgos de su carácter se transformaron de modo positivo; por ejemplo, mejoró su capacidad de escuchar y aumentó su consideración hacia los demás.

Arsenovski (2016) narra su experiencia con el desarrollo de *software* colectivo en un proyecto heredado con muchos defectos. A su enfoque de trabajo particular lo denomina *swarm* (enjambre), término que alude a varios programadores que colaboran para un proyecto común. Arsenovski menciona que utilizaron varios patrones como la "ramificación hacia afuera" (*branch out*), en la que un participante se separa con el objetivo de solucionar algún problema que afecta al equipo completo y luego se reincorpora. Por su parte, Lilienthal (2017) hace la diferenciación entre programación Mob y arquitectura Mob. Esta última se refiere a la creación y mejora de la arquitectura de los sistemas de *software*; esto lo realiza el equipo completo a través de herramientas automatizadas y de un navegador externo que inspecciona código fuente, modela vistas de arquitectura, solicita refactorizaciones y establece prioridades, siempre acompañado del equipo de desarrollo en el rol de observadores.

Tabla 1. Resumen de los principales hallazgos reportados en la literatura analizada sobre la programación por pares y Mob

Programación por pares	Programación Mob
• La programación por pares en cursos introductorios ayuda a que los alumnos obtengan calificaciones más altas en cursos avanzados (Smith *et al*., 2018) • La preparación individual aunada a la programación en parejas es benéfica para evitar el bajo desempeño académico de los alumnos (Lee *et al*., 2016)	• Hacen falta experimentos científicos orientados a conocer más sobre la programación Mob (Lilienthal, 2017) • Se requiere la validación empírica de la programación Mob (Balijepally *et al.*, 2017)
• La programación por pares es una herramienta útil que mejora la comunicación entre los estudiantes, la comprensión de los temas y, además, desencadena nuevas ideas (Du *et al*., 2015) • La programación en parejas mejora los niveles de comunicación y reduce el tiempo en el que se escribe el código (Saltz & Shamshurin, 2017) • El desarrollo de *software* no tiene que llevarse a cabo todo el tiempo entre dos personas. Se pueden intercalar actividades en solitario y en pares (Swamidurai & Umphress, 2015) • La programación por pares puede usarse ocasionalmente, en especial en las partes más complicadas del *software* (Meyer, 2018)	• La programación Mob reduce los trámites administrativos y las juntas extensas; promueve el aprendizaje continuo, la satisfacción en los integrantes del equipo y la calidad del *software* (Zuill, 2015) • Con la programación Mob, las personas conocen más ampliamente el sistema en el que trabajan. Sin embargo, al inicio avanzan lentamente y pueden mostrar renuencia, perder el interés y aislarse (Buchan & Pearl, 2018) • Con la programación Mob, se puede mejorar la capacidad de escuchar y la consideración hacia los demás (Kerney, 2015) • Puede pasar tiempo antes de encontrar las configuraciones óptimas de tiempo, espacio, recursos y estrategias para trabajar con programación Mob (Bohekhout, 2016)

Fuente: elaboración propia.

Bohekhout (2016) utilizó el enfoque Mob con buenos resultados para la programación y el diseño de flujos de procesos. Reconoce las bondades de esta forma de trabajar, explica que tardó en encontrar las configuraciones óptimas de tiempo, espacio, recursos y estrategias; al lograrlo combinó con éxito las modalidades individual y Mob. En la tabla 1 incluimos los hallazgos más importantes identificados en la literatura que consultamos sobre la programación por pares y Mob.

METODOLOGÍA

Propósito y preguntas de investigación

Nuestra investigación tuvo como objetivo comparar las percepciones de los estudiantes acerca de los enfoques de programación individual, por pares y Mob, con el propósito de contar con elementos validados empíricamente que permitan valorar la factibilidad de la implementación de alguno de estos tres enfoques. De esta manera, será posible establecer en el futuro acciones y estrategias que faciliten su adopción en cursos universitarios. Esto es importante porque en la actualidad las mediciones para determinar comparaciones son escasas en este contexto, en especial en el caso de la programación Mob.

Las preguntas orientadoras del estudio fueron: ¿cuál de estos tres enfoques (individual, por pares, Mob) es el más apropiado en el contexto de la docencia universitaria?, ¿cuál es la percepción de los alumnos respecto a estos?, y ¿cuáles son las diferencias que existen entre estas modalidades?

Diseño de investigación

Para esta investigación, utilizamos un diseño explicativo secuencial (Hernández, 2014), que consiste en aplicar un enfoque cuantitativo, analizar los datos y obtener conclusiones para, posteriormente, emplear un enfoque cualitativo orientado a profundizar en estos resultados y lograr la interpretación del análisis completo (ver figura 1).

Aplicamos también un cuestionario y un análisis estadístico de datos para el enfoque cuantitativo, además de entrevistas en profundidad para el enfoque cualitativo.

Participantes

Participaron 24 estudiantes que cursaban la materia de Programación intermedia en el tercer semestre de la Licenciatura de Tecnologías de la Información. Su edad oscilaba entre 19 y 25 años, con una media de 20.70 y desviación estándar de 2.01. De estos, 20

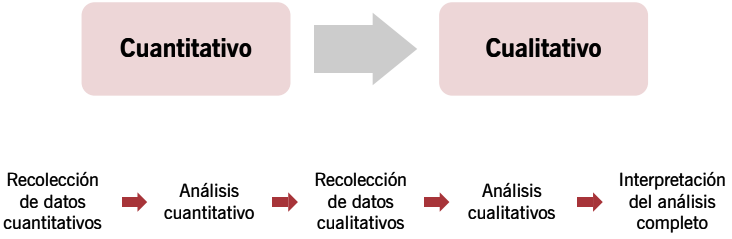


Figura 1. Diseño explicativo secuencial.
Fuente: Hernández (2014).

correspondían al género masculino y cuatro, al femenino. Durante el desarrollo de este trabajo, los alumnos asistieron a sesiones regulares de clase y no sabían que esta investigación se estaba efectuando; por ello, no les ofrecimos algún incentivo o remuneración por sus respuestas. Todos los participantes habían cursado y aprobado previamente los cursos universitarios Fundamentos de la informática y metodología de la programación y Programación básica. El primero se concentró en la lógica necesaria para realizar algoritmos computacionales y el segundo, en la solución de problemas con estructuras básicas de programación en el lenguaje Visual Basic.Net.

Escenario

Este trabajo se realizó en el laboratorio de cómputo al que los estudiantes asisten con regularidad a sus clases de programación. En este lugar tienen acceso a 30 computadoras con procesador Intel i5, monitores de 21 pulgadas y disco duro de 1 tb. Estos equipos tienen instalado el sistema operativo Windows 8 y el entorno integrado de desarrollo Visual Studio con el lenguaje Visual Basic.NET.

Procedimiento

Todos los participantes tenían experiencia de trabajo individual, pues esa era la manera habitual de realizar sus prácticas hasta ese momento; sin embargo, desconocían los enfoques de programación por pares y Mob. Para fines de esta investigación, primero, en una sesión de clase, les solicitamos a los participantes programar de manera individual. En otra sesión diferente, les pedimos trabajar en parejas y, en una tercera sesión, con programación Mob. Finalmente, los invitamos a responder el cuestionario basándose solo en la experiencia de estos tres momentos de trabajo.

Cada sesión duró dos horas. El tiempo se distribuyó de esta forma: durante los primeros quince minutos esperamos a que todos los alumnos llegaran al laboratorio; después, pasamos lista y explicamos las reglas del trabajo de cada metodología empleada; en seguida, les presentamos el programa que debían realizar y asignamos, de manera aleatoria, los alumnos a las computadoras del laboratorio. En el caso de las modalidades por parejas y Mob, los equipos fueron creados también de modo aleatorio. Todas estas actividades se ejecutaron en veinte minutos. Los alumnos tuvieron 85 minutos para desarrollar el programa planteado. Durante la investigación, ningún alumno podía contactar a personas externas a su equipo. Tampoco les permitimos copiar el trabajo de alguien más; sin embargo, sí podían consultar información a través de internet.

Sesión de programación individual

En la modalidad individual, cada alumno fue asignado a una computadora. El programa

Durante la investigación, ningún alumno podía contactar a personas externas a su equipo. Tampoco les permitimos copiar el trabajo de alguien más; sin embargo, sí podían consultar información a través de internet

que se solicitó utilizó una arquitectura de *software* orientada a objetos de tres capas e implementó una lista de registros almacenados en una base de datos con una tabla.

Sesión de programación por pares

Cada pareja de alumnos utilizó una sola computadora, donde cada participante tuvo que usar el teclado durante cinco minutos y luego cederlo a su compañero. Los investigadores cronometraron el tiempo y avisaron cuando era momento de cambiar el teclado. El programa solicitado utilizó una arquitectura de *software* orientada a objetos de tres capas e implementó diversas listas selectivas de registros almacenados en una base de datos con varias tablas.

Sesión de programación Mob

Los estudiantes fueron asignados aleatoriamente a cuatro equipos de seis personas que se ubicaron en espacios de trabajo distintos, pero dentro de la misma área del laboratorio de prácticas al que ellos asisten a clase. A cada equipo se le asignó una sola computadora con las características descritas en el apartado "escenario" de este artículo. Además, dispusieron de seis sillas, un proyector, una superficie para proyectar, y un espacio de trabajo. Cada equipo se autoorganizó para trabajar. Ellos tuvieron esta libertad, siempre y cuando el equipo actuara de manera unida, respetuosa y afín a los principios de la programación Mob. El programa solicitado fue una interfaz gráfica para realizar diversas consultas de registros almacenados en una base de datos y utilizó una arquitectura de *software* orientada a objetos de tres capas.

Método para el enfoque cuantitativo

Diseñamos un cuestionario con las preguntas que se muestran en la tabla 2 y lo entregamos a los estudiantes, les pedimos que evaluaran cada aspecto para cada modalidad de trabajo utilizando una escala del uno al diez, donde uno es la puntuación más baja, que representa "malo" o "poco", y diez es la más alta, que significa "excelente" o "mucho". Decidimos recurrir a esta escala por la sencillez que los alumnos encuentran al realizar evaluaciones con esta, pues la relacionan con las calificaciones escolares que ellos reciben en la universidad. Además, en una investigación anterior conducida por nuestro equipo, algunos alumnos mostraron dificultad para evaluar y expresar sus percepciones usando una escala del uno al cinco.

Tabla 2. Evaluaciones solicitadas a los participantes para trabajo individual, por pares y Mob

Pregunta	Aspecto a evaluar
1	Calificación global de la metodología
2	Facilidad de entender la metodología
3	Facilidad de implementar la metodología
4	Facilidad de adaptar la metodología
5	Facilidad para trabajar con la metodología
6	Experiencia al trabajar con la metodología
7	Detección de errores en el programa
8	Rapidez para finalizar el programa
9	Calidad del proceso de desarrollo
10	Facilidad para comunicarse
11	Motivación para trabajar en el proyecto
12	Organización para trabajar
13	Nivel de confianza en el éxito del proyecto
14	Nivel de satisfacción con el trabajo realizado

Fuente: elaboración propia.

Todas las respuestas fueron capturadas en el paquete SPSS, en el cual se realizó un proceso de revisión y limpieza de datos. No encontramos valores perdidos o atípicos. Posteriormente, las respuestas se transformaron como se indica en la tabla 3. Estos valores en la nueva escala del uno al cinco fueron utilizados para todos los procedimientos y análisis efectuados en esta investigación.

Tabla 3. Respuestas proporcionadas por los estudiantes y valores categorizados utilizados en el análisis cuantitativo

Valores antiguos (respuestas proporcionadas por los estudiantes)	Valores nuevos (nuevos valores transformados)
1,2	1
3,4	2
5,6	3
7,8	4
9,10	5

Fuente: elaboración propia.

Después se realizó un análisis factorial considerando de la pregunta 2 a la 14. Encontramos tres dimensiones que identificamos con nombres representativos. Luego, calculamos el alfa de Cronbach para cada una de estas. En este proceso obtuvimos también la participación de cada pregunta en su respectiva dimensión. En seguida, calculamos un valor ponderado para cada dimensión; para ello, multiplicamos cada respuesta categorizada de los estudiantes por el porcentaje correspondiente a la participación de esta pregunta en esa dimensión.

Finalmente, realizamos pruebas de Friedman para distinguir diferencias estadísticas significativas entre los valores de cada una de las dimensiones para los tres enfoques metodológicos estudiados. También con la prueba de Friedman buscamos diferencias entre la calificación global que los estudiantes asignaron a cada uno de los tres enfoques metodológicos. La figura 2 presenta un esquema global de los pasos que comprendió el análisis de los datos cuantitativos.

Método para el enfoque cualitativo

Para el componente cualitativo de esta investigación tomamos como referencia los resultados cuantitativos obtenidos: la programación por pares obtuvo las más altas puntuaciones, mientras que la programación Mob y la modalidad individual recibieron las más bajas. Por esta razón, realizamos entrevistas con preguntas abiertas orientadas a conocer más sobre las razones de estos hallazgos. Formulamos las siguientes preguntas: ¿cuáles son las ventajas que percibes de la programación por pares sobre la programación Mob y la programación individual?, ¿cuáles son los principales problemas que encontraste

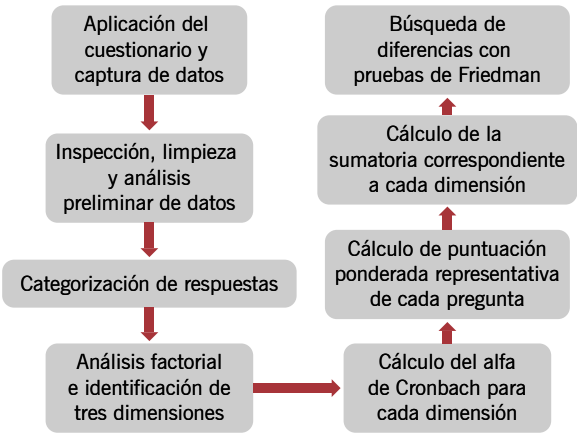


Figura 2. Esquema del análisis cuantitativo realizado en esta investigación.
Fuente: elaboración propia.

al trabajar con la programación Mob?, ¿cuáles son los principales problemas que encontraste al trabajar con la programación individual?, ¿qué opinas sobre la posibilidad de implementar la programación por pares o la programación Mob como práctica cotidiana en los cursos universitarios?

De los 24 estudiantes que intervinieron en la fase cuantitativa, cinco estuvieron disponibles y dispuestos a participar en las entrevistas de la fase cualitativa. Aunque solo dos brindaron entrevistas extensas y ricas en contenido, mientras que los otros tres aportaron ideas en frases cortas que estaban contenidas en las respuestas de los dos estudiantes que se presentan en este artículo.

En ese sentido, Hernández (2014) menciona que en enfoques cualitativos no es necesario que la muestra sea estadísticamente representativa de la población de estudio y que su tamaño se determina por tres factores: la capacidad operativa y de recolección de datos, la accesibilidad de las unidades de estudio y la saturación de categorías, que implica el número de casos que permitan responder las preguntas de investigación. Así, en estudios cualitativos también es posible la reformulación de la muestra. Esto quiere decir que en el proceso se pueden agregar casos no previstos al inicio, o bien, excluir otros que sí lo estaban.

RESULTADOS

Resultados cuantitativos

Para encontrar las tres dimensiones que mostramos en la tabla 4, llevamos a cabo un análisis factorial exploratorio con el método de extracción de máxima verosimilitud basado en autovalores mayores de 1. El método de rotación fue Promax con Kappa = 4. Obtuvimos una medida KMO de 0.704; la prueba de esfericidad de Barlett arrojó los valores de $Chi^2 = 364.60$, gl = 78, y sig. = 0. La varianza total explicada con tres factores fue del

Tabla 4. Dimensiones encontradas en el cuestionario

Dimensión	Pregunta	Aspecto a evaluar
Aspectos de gestión del proceso y del equipo	2	Facilidad de entender la metodología
	4	Facilidad de adaptar la metodología
	9	Calidad del proceso de desarrollo
	10	Facilidad para comunicarse
	12	Organización para trabajar
	13	Nivel de confianza en el éxito del proyecto
Aspectos de implementación	3	Facilidad de implementar la metodología
	5	Facilidad para trabajar con la metodología
	6	Experiencia al trabajar con la metodología
	11	Motivación para trabajar en el proyecto
	14	Nivel de satisfacción con el trabajo realizado
Aspectos de depuración	7	Detección de errores en el programa
	8	Rapidez para finalizar el programa

Fuente: elaboración propia.

80.32%. El análisis factorial confirmatorio para encontrar exactamente tres factores dio los mismos resultados. Los cálculos del alfa de Cronbach y los porcentajes de aportación de cada variable a su dimensión se incluyen en las tablas 5, 6 y 7.

La prueba de Friedman aplicada para establecer diferencias en la primera dimensión (aspectos de gestión del proceso y del equipo) resultó significativa (PValue = 0.005, Jí cuadrado = 10.564, n = 24, gl = 2); las puntuaciones más altas fueron para la programación por pares (rango promedio = 2.46) y las más bajas, para la programación Mob (rango promedio = 1.63). Aunque no encontramos diferencias estadísticas significativas en la prueba de Friedman, en la segunda dimensión (aspectos de implementación) (PValue = .108, Jí cuadrado = 4.447, n = 24, gl = 2), observamos tendencias aritméticas de la programación por pares para obtener las puntuaciones más altas (rango promedio = 2.31) y

Tabla 5. Porcentaje de participación para cada pregunta de la dimensión 1: aspectos de gestión del proceso y del equipo (alfa de Cronbach = 0.94)

Id	Aspecto a evaluar	Correlación total de elementos corregida	Participación de cada pregunta en esa dimensión
2	Facilidad de entender la metodología	0.846	0.170
4	Facilidad de adaptar la metodología	0.866	0.174
9	Calidad del proceso de desarrollo	0.795	0.160
10	Facilidad para comunicarse	0.872	0.175
12	Organización para trabajar	0.850	0.171
13	Nivel de confianza en el éxito del proyecto	0.732	0.147

Fuente: elaboración propia.

Tabla 6. Porcentaje de participación para cada pregunta de la dimensión 2: aspectos de implementación (alfa de Cronbach = 0.925)

Id	Aspecto a evaluar	Correlación total de elementos corregida	Participación de cada pregunta en esa dimensión
3	Facilidad de implementar la metodología	0.799	0.199
5	Facilidad para trabajar con la metodología	0.792	0.197
6	Experiencia al trabajar con la metodología	0.831	0.207
11	Motivación para trabajar en el proyecto	0.775	0.193
14	Nivel de satisfacción con el trabajo realizado	0.823	0.205

Fuente: elaboración propia.

Tabla 7. Porcentaje de participación para cada pregunta de la dimensión 3: aspectos de depuración (alfa de Cronbach = 0.939)

Id	Aspecto a evaluar	Correlación total de elementos corregida	Participación de cada pregunta en esa dimensión
7	Detección de errores en el programa	0.889	0.500
8	Rapidez para finalizar el programa	0.889	0.500

Fuente: elaboración propia.

de la programación Mob para las puntuaciones más bajas (rango promedio = 1.83).

En la tercera dimensión (aspectos de depuración) sí reconocimos diferencias significativas con la prueba de Friedman (PValue = .031, Jí cuadrado = 6.969, n=24, gl=2); las puntuaciones más altas fueron para la programación por pares (rango promedio = 2.23) y las más bajas, para la programación individual (rango promedio = 1.65). En la calificación global que los estudiantes otorgaron a las metodologías no distinguimos diferencias significativas con un nivel de confianza del 95% como referencia, pero puede considerarse significativa con un 90% de confianza como referencia, al tratarse de una investigación inicial. Los resultados de esta comparación con la prueba de Friedman fueron: PValue = 0.062, Jí cuadrado = 5.548, gl = 2. Las puntuaciones más altas correspondieron a la programación por pares (rango promedio = 2.29) y las más bajas, a la programación Mob (rango promedio = 1.75).

Resultados cualitativos

Las entrevistas para el apartado cualitativo revelaron que a los alumnos les gusta programar por pares porque lo perciben como un punto medio entre no programar con nadie y hacerlo con un grupo numeroso. Programar de manera individual puede provocar equivocaciones, tensión, bloqueo intelectual y aprehensión por la responsabilidad del trabajo; por otra parte, programar con demasiadas personas simultáneamente puede generar desorden, estrés, distracción y desbalance de trabajo entre los participantes. La tabla 8 contiene un resumen de las respuestas proporcionadas por los participantes.

Tabla 8. Resumen de las entrevistas realizadas

Pregunta	Resumen de respuestas del participante 1 (hombre, 20 años, alumno regular de tercer semestre)	Resumen de respuestas del participante 2 (hombre, 21 años, alumno regular de tercer semestre)
• ¿Cuáles son las ventajas que percibes de la programación por pares sobre la programación Mob y la programación individual?	• Tener un compañero con quien resolver los problemas • Trabajo y comunicación están bajo mayor control • Se organiza mejor el trabajo • Se cuenta con el apoyo de otra persona en todo el proceso	• Responsabilidad compartida • Programar en parejas es un punto medio entre estar completamente solo y estar rodeado de muchas personas Así se superan las desventajas de trabajar solo y en Mob • Mayor capacidad de lograr acuerdos de trabajo
• ¿Cuáles son los principales problemas que encontraste al trabajar con la programación Mob?	• Genera más estrés trabajar con tantas personas al mismo tiempo • El ruido afecta la concentración • Falta un espacio más privado para trabajar en equipo • El trabajo más pesado lo hacen unas pocas personas • No todos explican su trabajo a los demás	• Desorganización para trabajar y tomar decisiones • No todos colaboran con el mismo interés • Diversos niveles de habilidades y conocimientos • Las actitudes inmaduras de algunos participantes

Pregunta	Resumen de respuestas del participante 1 (hombre, 20 años, alumno regular de tercer semestre)	Resumen de respuestas del participante 2 (hombre, 21 años, alumno regular de tercer semestre)
• ¿Cuáles son los principales problemas que encontraste al trabajar de manera individual?	• En ocasiones produce tensión o bloqueo especialmente si se desconoce cómo resolver los problemas que se presentan. Se tiene que pedir ayuda a otro compañero o al profesor	• Se siente la responsabilidad total del proyecto • Se cometen más errores y es más tardado encontrarlos y corregirlos
• ¿Qué opinas sobre la posibilidad de implementar la programación por pares o la programación Mob como práctica cotidiana en los cursos universitarios?	• Estaría bien implementar la programación por pares • Sería bueno que los alumnos puedan elegir a su compañero de trabajo • La programación Mob quizás podría implementarse en materias más avanzadas, con alumnos de semestres superiores que se conocen más entre sí y se enfocan más en su trabajo	• No estaría bien implementar la modalidad Mob si no hay espacios de trabajo más privados ni suficientes proyectores para cada equipo. A lo mejor también habría que entrenar a los alumnos para trabajar en equipo • La programación por pares estaría bien, pero se tendría que cuidar que cada pareja sea compatible para trabajar en un mismo proyecto

Fuente: elaboración propia.

DISCUSIÓN

Los resultados de este trabajo indican que la programación por pares podría ser el enfoque más apropiado como estrategia didáctica, ya que cuenta con una buena aceptación por parte de los alumnos y, además, su implementación no requiere equipos adicionales o adecuaciones al espacio de trabajo. Los alumnos podrían organizarse en equipos de dos personas para utilizar las computadoras que ya se encuentran instaladas y distribuidas en los laboratorios de cómputo. La percepción de los estudiantes privilegia la programación por pares sobre la individual y la modalidad Mob, pues consideran que es bueno compartir las responsabilidades pero no con demasiadas personas al mismo tiempo, porque las situaciones podrían salirse de control con facilidad. En este sentido, en la programación Mob, la actitud y el comportamiento de algunos estudiantes afectan aspectos como la comunicación, el entendimiento del problema y la organización de las tareas. Por otra parte, al programar individualmente, los alumnos sienten que cometen más errores y que tardan más tiempo en encontrarlos y corregirlos.

Las posturas de los autores de este trabajo concuerdan con las de Sadath *et al.* (2018), quienes señalan que la enseñanza de la ingeniería del *software* en las aulas universitarias está desfasada respecto a la realidad que viven las empresas en la actualidad. Por esta razón, es relevante alinear los contenidos temáticos de los cursos y, al mismo tiempo, promover un aprendizaje significativo mediante formas dinámicas de desarrollar *software* que sean directamente aplicables en las empresas. Por esto, es necesario realizar investigaciones sobre los enfoques utilizados en las aulas para que los estudiantes aprendan a desarrollar *software*. En el caso concreto de la programación Mob, coincidimos con Scharman (2014), Lilienthal (2017) y Balijepally *et al.* (2017) en la necesidad de contar con más mediciones y evaluaciones orientadoras.

Nuestros resultados son semejantes a los de Du *et al.* (2015) y Saltz y Shamshurin (2017). Al igual que ellos, encontramos que la programación por pares mejora la comunicación entre los estudiantes. También son análogos a los de Gómez y Aguileta (2018), pues los participantes sintieron que fue más fácil encontrar y corregir errores con

la programación por pares, auxiliados por un entorno integrado de desarrollo. También coincidimos con Swamidurai y Umphress (2015) y Meyer (2018) en que los alumnos no tienen que programar en parejas todo el tiempo. Un enfoque híbrido evolutivo, producto de la continua adaptación a las necesidades, como el que menciona Wilson (2015), podría resultar benéfico para el aprovechamiento académico de los estudiantes.

Para la correcta interpretación de nuestros resultados debe considerar que este estudio se realizó con una muestra pequeña de un curso universitario de nivel principiante-intermedio. Se requiere mayor investigación para comprobar estos hallazgos y profundizar en ellos.

CONCLUSIONES Y RECOMENDACIONES

Los alumnos que experimentaron con las tres modalidades de trabajo estudiadas en este artículo mostraron una preferencia por la programación por pares. El uso de la programación Mob no fue reportado como favorable en estos aspectos debido a la interacción múltiple de los participantes y a la falta de consenso en la organización y en la toma de decisiones. La programación individual, por el contrario, limita la interacción continua con otras personas y esto puede ser un impedimento si los participantes no tienen clara la manera de resolver los problemas planteados. Recomendamos promover el trabajo por parejas en los cursos universitarios de programación. Esta práctica puede implementarse fácilmente en las instalaciones existentes de los laboratorios de cómputo en las universidades.

Es necesario continuar con las investigaciones para conocer las maneras más adecuadas de enseñar a programar. En este sentido, se debe profundizar en la aplicación de la programación colaborativa en sus diversas modalidades e involucrar a estudiantes principiantes, intermedios y avanzados.

REFERENCIAS BIBLIOGRÁFICAS

Aottiwerch, N. & Kokaew, U. (2018). The analysis of matching learners in pair programming using K-Means, en *2018 5th International Conference on Industrial Engineering and Applications* (362-366). http://doi.org/10.1109/IEA.2018.8387125

Arsenovski, D. (2016). *Swarm: Beyond pair, beyond scrum.* Agile Alliance.

Balijepally, V.; Chaudhry, S. & Sridhar, N. (2017). Mob programming–A promising innovation in the agile toolkit, en *Twenty-third Americas Conference on Information Systems, Boston, 2017-Systems Analysis and Design (SIGSAND)* (1-9). Boston, Estados Unidos: AIS Electronic Library.

Bohekhout , K. (2016). Mob programming: Find fun faster, en *Lecture Notes in Business Information Processing* (185-192). Haarlem, The Netherlands. http://doi.org/10.1007/978-3-319-33515-5_15

Buchan, J. & Pearl, M. (2018). Leveraging the mob mentality: An experience report on mob programming, in *EASE 2018 -Evaluation and Assessment in Software Engineering* (1-6). Christchurch, Nueva Zelanda: Editorial de la Universidad de Canterbury. http://doi.org/10.1145/3210459.3210482

Du, W.; Ozeki, M.; Nomiya, H.; Murata, K. & Araki, M. (2015). Pair programming for enhancing communication in the fundamental C language exercise, in *2015 IEEE 39th Annual International Computers, Software & Applications Conference* (664-665). IEEE Computer Society. http://doi.org/10.1109/COMPSAC.2015.67

Gómez, O. & Aguileta, A. (2018). Influence on the use of an IDE as tool support in the pair programming: A controlled experiment. *IEEE Latin America Transactions, 16*(3), 948-956. http://doi.org/10.1109/TLA.2018.8358678

Hernández Sampieri, R. (2014). *Metodología de la investigación*. Ciudad de México: McGraw-Hill.

Karthiekheyan, K.; Ahmed, I. & Jayalakshmi, J. (2018). Pair programming for software engineering education: An empirical study. *The International Arab Journal of Information Technology, 15*.

Kerney, R. (2015). *Mob programming-My first team.* Estados Unidos: Agile Alliance.

Lee, L. K.; Au, O.; So, R. & Nga-Inn, W. (2016). Being well-prepared for regular pair-programming helps at-risk

students, in *2016 International Symposium on Educational Technology* (65-68). http://doi.org/10.1109/ISET.2016.22

Lilienthal, C. (2017). From pair programming to mob programming to mob architecting, in *International Conference on Software Quality -SWQD2017-Software, Quality, Complexity and Challenges of Software* (3-12). Vienna, Austria: Springer. http://doi.org/10.1007/978-3-319-49421-0_1

Meyer, B. (2018). Making sense of agile methods. *IEEE Software*, *35*(2), 91-94. http://doi.org/10.1109/MS.2018.1661325

Poonam, R. & Yasser, C. (2018). An experimental study to investigate personality traits on pair programming efficiency in extreme programming, in *2018 5th International Conference on Industrial Engineering and Applications* (95-99). https://doi.org/10.1109/IEA.2018.8387077

Pyhäjärvi, M. & Falco, L. (2018). *The mob programming guidebook.* n/d: LeanPub.

Sadath, L.; Karim, K. & Gill, S. (2018). Extreme programming implementation in academia for software engineering sustainability, in *2018 Advances in Science and Engineering Technology International Conferences (ASET)* (1-6). Abu Dhabi. http://doi.org/10.1109/ICASET.2018.8376925

Saltz, J. & Shamshurin, I. (2017). Does pair programming work in a data science context? An initial case study, in *2017 IEEE International Conference on Big Data (BIGDATA)* (2348-2354). http://doi.org/10.1109/BigData.2017.8258189

Schartman, M. (2014). *My experience with mob programming.* Appfolio Engineering.

Smith, M.; Giugliano, A. & DeOrio, A. (2018). Long term effects of pair programming. *IEEE Transactions on Education*, *61*(3), 1-8. http://doi.org/10.1109/TE.2017.2773024

Swamidurai, R. & Umphress, D. (2015). Inverted pair programming, in *Proceedings of the IEEE SoutheastCon 2015* (1-6). Fort Lauderdale, Florida: IEEE. http://doi.org/10.1109/SECON.2015.7133010

Umapathy, K. & Ritzhaupt, A. D. (2017). A meta-analysis of pair-programming in computer programming courses: Implications for educational practice. *ACM Transactions on Computing Education*, *17*(4), 1-13. https://doi.org/10.1145/2996201

Wilson, A. (2015). *Mob programming-What works, what doesn´t.* Helsinki, Finlandia: Springer Link. http://doi.org/10.1007/978-3-319-18612-2 33

Zuill, W. (2015). *Mob programming-A whole team approach.* Recuperado de: https://www.agilealliance.org/wp-content/uploads/2015/12/ExperienceReport.2014.Zuill_.pdf

CÓMO CITAR ESTE ARTÍCULO:

Roque Hernández, Ramón Ventura; Guerra Moya, Sergio Armando y López Mendoza, Adán. (2020). Programación individual, por pares o colectiva: ¿cuál conviene utilizar en la universidad? *Apertura*, *12*(1), pp. 39-55. http://dx.doi.org/10.32870/Ap.v12n1.1791