Paradigma Experimental en la Ingeniería de Software: Caso Programación en Pareja

Omar S. Gómez, Julio C. Díaz, Raúl A. Aguilar
Facultad de Matemáticas
Universidad Autónoma de Yucatán
Mérida, Yucatán, México
omar.gomez@uady.mx

***Resumen*—Contexto: La Ingeniería de Software (IS) es una disciplina compleja donde a día de hoy el resultado de la aplicación de alguna tecnología para apoyar el desarrollo de software es impredecible. De manera similar a otras disciplinas ingenieriles, la IS requiere de construcción de modelos, experimentación y aprendizaje. La experimentación es un recurso valioso que permite evaluar de manera objetiva las distintas tecnologías disponibles para desarrollar software. Objetivo: Con la finalidad de promover el paradigma experimental en la IS, en este artículo se aplica el paradigma experimental para estudiar algunos efectos de la programación en pareja. Método: A través de un diseño experimental cuadrado latino con dos factores de bloqueo, se emplearon 21 estudiantes del programa de licenciatura en IS de la Universidad Autónoma de Yucatán (UADY) para examinar la duración y esfuerzo que conlleva la programación en pareja. Los estudiantes se agruparon en 7 parejas y 7 individuos donde codificaron durante dos sesiones dos programas pequeños. Resultados: Los resultados del experimento sugieren una diferencia significativa (en un nivel α=0.1) en favor de la programación en pareja con respecto a la duración de codificación de los ejercicios asignados (28% decremento en tiempo). Por el contrario, los resultados sugieren una diferencia significativa (en un nivel α=0.1) en favor de la programación individual con respecto al esfuerzo (30% decremento en esfuerzo). Conclusiones: En este artículo se presenta el paradigma experimental aplicado a la IS. A través del paradigma experimental es posible obtener observaciones de manera objetiva para acumular un cuerpo de conocimientos que pueda ser usado de manera confiable por profesionales en IS.**

***Palabras Clave*—ingeniería de software; programación en pareja; paradigma experimental; experimento controlado**

I. INTRODUCCIÓN

A día de hoy el resultado de la aplicación de una determinada tecnología para el desarrollo de software es impredecible [1]; no contamos con evidencia suficiente que apoye la mayoría de las creencias en que basamos la construcción de software [2]. A menudo en IS se adoptan tecnologías nuevas sin contar con evidencia convincente de que serán efectivas [3].

A través de la experimentación es posible evaluar de manera objetiva las distintas tecnologías disponibles para la construcción de software. La experimentación contrasta las creencias y las opiniones con la realidad convirtiéndolas en conocimiento científico o desechándolas. En la historia de todas las disciplinas ingenieriles se ha producido una transición desde las creencias, especulaciones y aciertos casuales hasta el conocimiento científico mediante el cual una ingeniería alcanza resultados predecibles [4,5].

La aplicación del paradigma experimental aplicado a la investigación en IS cuenta ya con varias décadas. Los orígenes de la experimentación en IS se remontan a los trabajos de Weinberg y Gressett [6], Grant y Sackman [7], Kolence [8] y Hanson [9] donde los autores empleaban aproximaciones científicas basadas en la observación y evaluación de tecnologías para la construcción de software. No obstante, los primeros experimentos así como la aplicación del paradigma experimental a la IS fue hecha por Basili en la década de 1980 [10]. La experimentación en IS hace posible la identificación de variables relevantes en la construcción de software y la comprensión de relaciones existentes entre ellas [11].

Siguiendo el paradigma experimental aplicado a la IS, en este artículo se presenta un experimento que analiza los efectos de la programación en pareja con respecto a la programación individual. El resto del artículo se organiza de la siguiente manera. En la sección II se describe el contexto del experimento. En la sección III se presenta el diseño experimental empleado. En la sección IV se presentan los resultados. En la sección V se discuten los resultados. Por último, en la sección VI se presentan las conclusiones.

II. CONTEXTO DEL EXPERIMENTO

La programación en pareja es una de las doce prácticas principales de la programación extrema creada a finales de la década de 1990 por Kent Beck [12,13]. Básicamente en esta práctica dos programadores trabajan en conjunto sobre una misma tarea usando una computadora. Uno de los programadores, quien toma el rol de controlador, escribe el código mientras que el otro programador, identificado como observador, revisa de manera activa el trabajo realizado por el controlador con el fin de detectar posibles defectos. El observador puede hacer anotaciones o definir estrategias para llevar a cabo la tarea a realizar, también puede buscar referencias para ayudar a resolver alguna cuestión que se presente durante la realización de la tarea.

Se han realizado algunos experimentos donde se examinan algunos efectos de la programación en pareja [14]-[19] De manera general, en estos experimentos se concluye que esta práctica ayuda a producir programas más pequeños, ayuda a implementar mejores diseños y que los programas contienen menos defectos que aquellos escritos de manera individual. Se dice también que la duración en completar una tarea en pareja es menor que realizar la tarea de manera individual.

Bajo un contexto académico el experimento que a continuación se describe se centra en analizar la duración y esfuerzo que implica codificar programas pequeños en pareja e indivi-

dualmente. Siguiendo el enfoque Goal-Question-Metric [20] la definición de este experimento se describe como:

"Estudiar la duración y el esfuerzo que conlleva la programación en pareja respecto a la programación individual con el propósito de evaluar posibles diferencias entre estas dos formas de programar. El estudio se realiza desde el punto de vista del investigador en un contexto académico compuesto por alumnos del tercer año de licenciatura en IS de la UADY donde codifican en pareja o individualmente un par de programas".

Las hipótesis de trabajo para este experimento se definen como:

- $H_{0(1)}$: El tiempo requerido para codificar un programa utilizando programación en pareja es equivalente al tiempo requerido para codificarlo de manera individual o, programar en pareja = a programar individualmente con respecto a la duración de tiempo.
- $H_{a(1)}$: El tiempo requerido para codificar un programa utilizando programación en pareja es diferente al tiempo requerido para codificarlo de manera individual o, programar en pareja ≠ a programar individualmente con respecto a la duración de tiempo.
- $H_{0(2)}$: El esfuerzo requerido para codificar un programa empleando programación en pareja es equivalente al esfuerzo requerido para codificar un programa de manera individual o, programar en pareja = a programar individualmente con respecto al esfuerzo.
- $H_{a(2)}$: El esfuerzo requerido para codificar un programa empleando programación en pareja es diferente al esfuerzo requerido para codificarlo de manera individual o, programar en pareja ≠ a programar individualmente con respecto al esfuerzo.

III. Diseño del Experimento

Las hipótesis planteadas en la sección anterior serán contrastadas a través de diferentes mediciones que se recogerán de los sujetos durante la ejecución del experimento. Básicamente, las mediciones pertenecen a dos grupos de sujetos: aquellos que realizan una tarea en pareja y aquellos que la realizan individualmente. Con estas mediciones se realizará entonces un análisis estadístico de acuerdo a un diseño experimental.

A. Diseño Cuadrado Latino

La característica principal del diseño experimental cuadrado latino es el manejo de factores de bloqueo para aumentar la precisión en las mediciones obtenidas. De manera particular, este diseño emplea dos factores o variables de bloqueo donde cada tratamiento se encuentra presente en cada nivel del primer factor de bloqueo y también se encuentra presente en cada nivel del segundo factor de bloqueo.

El arreglo de este diseño se conforma por un número igual de filas (factor de bloqueo uno) y columnas (factor de bloqueo dos). Los tratamientos (en este caso dos, programación en pareja y programación individual) se representan por caracteres latinos como símbolos donde cada símbolo se encuentra presente sólo una vez en cada fila y sólo una vez en cada columna. En la Tabla I se muestra un ejemplo del arreglo de un diseño cuadrado latino.

TABLA I. Diseño cuadrado-latino

A	**B**	**C**
B	**C**	**A**
C	**A**	**B**

En un diseño cuadrado latino el bloqueo es usado para aislar de manera sistemática las fuentes de variación no deseadas en la comparación entre tratamientos. Los dos factores de bloqueo usados en este experimento son el programa y la herramienta usada para codificar. En la Tabla II se muestra el arreglo usado para este experimento.

TABLA II. Arreglo cuadrado-latino usado en el experimento

Programa / Herramienta	**IDE**	**Editor Texto**
Calculadora	Individual	En pareja
Codificador	En pareja	Individual

El primer factor de bloqueo identificado como "programa" consta de dos niveles: calculadora y codificador, mientras que el segundo factor de bloqueo identificado como "herramienta" se conforma por dos niveles: IDE (En Inglés, Integrated Development Environment) y editor de texto. Los tratamientos a examinar son: programación en pareja y programación individual.

B. Sujetos, Tareas y Objetos

En este experimento participaron 21 estudiantes (sujetos) pertenecientes a la carrera de IS de la UADY (Experimento realizado en el periodo académico de agosto-diciembre de 2012). La mayoría de los sujetos cursaban el tercer año de la licenciatura, el resto de ellos (tres sujetos) comenzaban su cuarto año. De acuerdo a la clasificación sobre experiencia en programación creada por Dreyfus y Dreyfus [21], los sujetos de este experimento se clasifican como novicios avanzados, es decir, cuentan con experiencia práctica y conocimiento sobre los aspectos fundamentales del lenguaje de programación Java.

Se formaron dos grupos de manera aleatoria con los 21 sujetos disponibles, el grupo que trabajará en pareja y el grupo que trabajará individualmente. El experimento se dividió en dos sesiones, en cada sesión los sujetos codificaron un programa distinto en el lenguaje de programación Java. En la primera sesión los sujetos que trabajaron individualmente usaron el NetBeans IDE para codificar el primer programa mientras que los sujetos que trabajaron en pareja usaron como herramienta un editor de texto. En la segunda sesión se intercambió la herramienta de soporte, los sujetos que antes trabajaron individualmente con el NetBeans IDE ahora trabajaron con un editor de texto mientras que los sujetos que trabajaron en pareja codificaron el segundo programa con el NetBeans IDE.

Antes de realizar el experimento, en una sesión de clases se dio una charla sobre la programación en pareja. En esta charla se explicaron los conceptos que subyacen en esta práctica. También se explicó cómo compilar un programa en Java usando sólo un editor de texto. Por último, se les explicó a los sujetos cómo deben obtener las mediciones durante las sesiones del experimento. En una hoja de papel se les pidió a los sujetos registrar la hora de inicio en que comenzaron a codificar el programa y la hora final en que lo terminaron, esto

es, una vez verificado el funcionamiento del programa acorde a su especificación.

Para las sesiones del experimento se eligieron programas pequeños que los sujetos puedan codificar, compilar, ejecutar y probar durante la sesión. El primer programa identificado como "calculadora" consiste en que los sujetos codifiquen una calculadora que evalúe expresiones con números decimales así como los operadores +, -, ×, / e imprima el resultado sobre la pantalla. El segundo programa identificado como "codificador" consiste en que los sujetos escriban un programa sencillo que codifique y decodifique un mensaje de texto. Dado un cambio de letra (por ejemplo, la letra *a* representa la letra *p* codificada y viceversa), el programa codifica o decodifica una línea de texto.

C. Ejecución

Cada sesión se programó con una duración de 90 minutos. Las dos sesiones se llevaron a cabo en uno de los salones del centro de cómputo de la Facultad de Matemáticas de la UADY. La primera sesión del experimento comenzó 30 minutos tarde debido a que no se encontraban todos los sujetos en el salón. Cuando el grupo se completó se dio inicio a la sesión, se proyectó en la pantalla la especificación del programa a realizar (calculadora). Debido al retraso en esta sesión, varios sujetos no terminaran el programa en tiempo por lo que se les pidió registrar el tiempo para que finalizaran en casa.

Dada esta situación, a los sujetos que trabajaron individualmente se les pidió terminaran el programa en casa y que terminaran de registrar sus tiempos. Por otra parte, a los sujetos que trabajaron en pareja y no terminaron, se les programó una sesión especial al día siguiente en uno de los salones del centro de cómputo para que bajo supervisión terminaran el programa. La segunda sesión inicio en tiempo, se proyectó en pantalla la segunda especificación del programa a realizar (codificador). En esta sesión los sujetos terminaron satisfactoriamente el programa. En cada sesión, una vez que los sujetos terminaban su programa se les pedía ejecutarlo para verificar su correcto funcionamiento.

D. Mediciones Recolectadas

Los tiempos de inicio y fin registrados por los sujetos se usaron para definir las mediciones: duración y esfuerzo.

Duración. Es el tiempo transcurrido en minutos para codificar un programa. Antes de iniciar a codificar, los sujetos registraron la hora de inicio. Una vez que los sujetos completaron de manera satisfactoria el programa registraron la hora final. La duración se calculó como la diferencia en minutos entre la hora final y la hora inicial. En la Tabla III se muestran las mediciones obtenidas con respecto a la duración.

TABLA III. MEDICIONES OBTENIDAS RESPECTO A LA DURACIÓN

Programa / Herramienta	IDE	Editor Texto
Calculadora	Individual: 110, 136, 281, 239, 126, 69, 205	En pareja: 256, 184, 114, 59, 37, 89, 135
Codificador	En pareja: 70, 48, 88, 85, 43, 39, 56	Individual: 66, 102, 128, 107, 106, 76, 64

Esfuerzo. El esfuerzo mide la cantidad de trabajo requerido para realizar una tarea. En este caso es el esfuerzo total de programación en persona-minutos para codificar un programa. El esfuerzo total de las parejas es la duración de la pareja multiplicada por dos. En la Tabla IV se muestran las mediciones recolectadas respecto al esfuerzo.

TABLA IV. MEDICIONES OBTENIDAS RESPECTO AL ESFUERZO

Programa / Herramienta	IDE	Editor Texto
Calculadora	Individual: 110, 136, 281, 239, 126, 69, 205	En pareja: 512, 368, 228, 118, 74, 178, 270
Codificador	En pareja: 140, 96, 176, 170, 86, 78, 112	Individual: 66, 102, 128, 107, 106, 76, 64

IV. RESULTADOS

Con las mediciones obtenidas es posible contrastar las hipótesis previamente definidas a través de un análisis estadístico. El modelo estadístico asociado con un diseño cuadrado latino se muestra en la ecuación (1):

$$y_{ijk}=\mu+\alpha_i+\tau_j+\beta_k+\varepsilon_{ijk} \quad (1)$$

Donde μ es la media general, α_i es el efecto de bloqueo común a la fila *i*, β es el efecto de bloqueo común a la columna *k*, τ_j es el efecto del tratamiento *j*-esimo, y ε_{ijk} es el error aleatorio el cual se asume es $N(0,\sigma^2)$.

Este diseño emplea el análisis de la varianza (en inglés ANOVA) para analizar los componentes del modelo (media general, factores de bloqueo, tratamientos y error aleatorio). El ANOVA se basa en examinar la variabilidad total de las mediciones recolectadas así como la variabilidad de los distintos componentes estadísticos. El ANOVA proporciona una prueba estadística sobre si las medias de varios grupos de mediciones son o no todas iguales. La hipótesis nula indica que todas las medias de los grupos de mediciones son simplemente muestras aleatorias de la misma población. Esto implica que todos los tratamientos tienen el mismo efecto (quizás ninguno). Rechazar la hipótesis nula implica que los distintos tratamientos producen diferentes efectos. En este experimento se cuenta con dos grupos de medias (programación en pareja y programación individual) donde cada grupo es bloqueado por los factores de bloqueo programa y herramienta, obteniendo así cuatro grupos de medias.

A. Validación de Supuestos

Antes de llegar a una conclusión, se deben verificar los siguientes supuestos en el modelo estadístico:

1) *Todas las mediciones son independientes (independencia)*
2) *La varianza es igual para todas las mediciones (homocedasticidad)*
3) *Las mediciones dentro de cada grupo presentan una distribución normal (normalidad)*

El primer supuesto se aborda a través de la distribución aleatoria de sujetos a tratamientos usada en este diseño experimental. Además, las mediciones de una muestra no se relacionan con las mediciones de otras muestras. El segundo y tercer supuesto se evalúa a través de la prueba de Levene [22] para evaluar la homocedasticidad de varianzas así como la prueba de Kolmogorov-Smirnov [23,24] para evaluar la normalidad.

Aplicando la prueba de Levene [22] para evaluar el segundo supuesto, se obtiene un valor *p* de 0.05947. Estableciendo un valor α de 0.05 esta prueba es significativa (tomando sólo dos decimales del valor *p* sin redondeo) lo que implica que el segundo supuesto no se valida de manera satisfactoria.

Analizando con más detalle esta situación, se observó que el tiempo requerido para codificar el segundo programa (Codificador) fue menor. En la Fig. 1 se muestra la duración media en minutos que les llevó a los sujetos codificar ambos programas. Como se muestra en esta figura, al no proporcionar a los sujetos programas similares, la homocedasticidad de varianzas se ve afectada. Para paliar esta situación, en experimentos posteriores se tiene previsto emplear programas con una complejidad similar.

Continuando con la evaluación del tercer supuesto, tras aplicar la prueba de Kolmogorov-Smirnov [23,24] se obtiene un valor *p* de 0.8806, lo que significa que se acepta la hipótesis nula en favor de la normalidad.

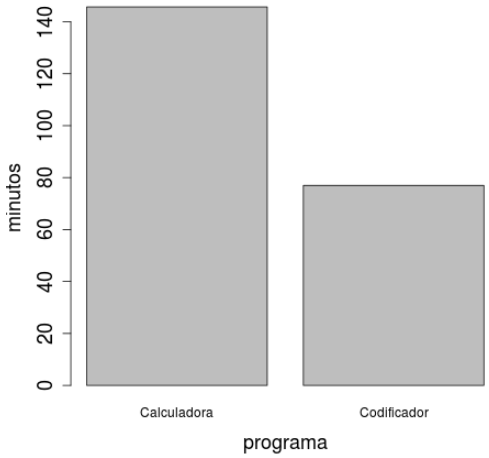

Fig. 1. Esfuerzo promedio en cada programa

Dado el tipo de diseño experimental usado, se recomienda verificar otro supuesto que es la aditividad. Los diseños experimentales que hacen uso del bloqueo como el caso de un diseño cuadrado latino asumen que no hay interacción entre el tratamiento y los factores de bloqueo. Bajo esta situación se dice que los efectos del tratamiento y de los factores de bloqueo son aditivos [25]. Este supuesto puede ser verificado empleando la prueba de Tukey de no aditividad [26]. En la Tabla V se muestran los resultados de la prueba de Tukey [26] para evaluar la no aditividad en el diseño cuadrado latino empleado.

TABLA V. RESULTADOS DE LA PRUEBA DE NO ADITIVIDAD

Medición	Bloque	Valor *F*	Valor *p*
Duración	Programa	0.0084	0.9277
Duración	Herramienta	1.0936	0.3061
Esfuerzo	Programa	0.0899	0.7669
Esfuerzo	Herramienta	0.9861	0.3306

Como se observa en los resultados de la tabla, los valores *p* no son significativos, es decir, los resultados cumplen de manera satisfactoria el supuesto de aditividad dada la falta de interacción entre el tratamiento y los factores de bloqueo.

B. Análisis de la Varianza

Una vez que los supuestos son evaluados, a continuación se describen los análisis de varianza (ANOVA) realizados. En la Tabla VI se muestra el análisis de la varianza con respecto a la duración mientras que en la Tabla VII se muestra el análisis de la varianza con respecto al esfuerzo.

TABLA VI. ANOVA RESPECTO A LA DURACIÓN

Componente	DF	SS	MS	Valor *F*	Valor *p*
Bloque prog.	1	33,052	33,052		
Bloque herram.	1	185	0.185		
Tratamiento	1	9,362	9,362	2.9843	0.0969
Residuos	24	75,293	3,137		

Si se establece un valor α de 0.05 ninguno de los tratamientos en ambas pruebas ANOVA son significativos. No obstante, si se establece un valor α de 0.1 que representa un nivel de confianza del 90% se obtienen diferencias significativas en ambos tratamientos. Para el caso del primer ANOVA (Tabla VI) se obtiene un valor *p*=0.0969 con respecto a la duración. En el segundo ANOVA (Tabla VII) se obtiene un valor *p*= 0.1017.

Aunque este segundo valor *p* es ligeramente mayor a 0.1, para este experimento se ha considerado también como significante (En la práctica suelen establecerse valores α de 0.1, 0.5 y 0.01; dependiendo el contexto del experimento y la precisión que se requiera).

TABLA VII. ANOVA RESPECTO AL ESFUERZO

Componente	DF	SS	MS	Valor *F*	Valor *p*
Bloque prog.	1	70,702	70,702		
Bloque herram.	1	4,969	4,969		
Tratamiento	1	22,346	22,346	2.8953	0.1017
Residuos	24	185,232	7,718		

Tomando en cuenta este valor α de 0.1, a continuación se describen las comparaciones entre tratamientos (también conocidas como pruebas de contraste) con respecto a cada medición (duración y esfuerzo). En la Tabla VIII se muestran las medias de los tratamientos, el error estándar así como las réplicas con respecto a la duración y al esfuerzo.

TABLA VIII. MEDIAS, ERROR ESTÁNDAR Y RÉPLICAS RESPECTO A LA DURACIÓN

Medición	Trat.	Duración media (mins)	Error estándar	Réplicas
Duración	Ind.	129.6428	17.8114	14
Duración	Pareja	93.0714	16.7054	14
Esfuerzo	Ind.	129.6428	17.8114	14
Esfuerzo	Pareja	186.1429	33.4108	14

Existen diferentes pruebas para realizar las comparaciones entre los tratamientos. Estas pruebas analizan pares de medias con el fin de detectar posibles diferencias significativas.

Empleando la prueba de Scheffé [27] para comparar tratamientos, en la Tabla IX se presentan los resultados de las comparaciones con respecto a la duración.

TABLA IX. COMPARACIÓN CON RESPECTO A LA DURACIÓN

Comparación	Diferencia	Valor *p*	LCI (95%)	LCS (95%)
Prog. ind. – Prog. en pareja	36.5714	0.0969	6.1578	66.9850

Como se observa en la Tabla IX, existe una diferencia significativa (de acuerdo a α=0.1) de 36 minutos en favor de la programación en pareja (28% de decremento en duración). Con un intervalo de confianza del 95% esta diferencia varía entre 6 y 66 minutos (entre un 4% y 51% de decremento en duración).

La Tabla X muestra la comparación de los tratamientos con respecto al esfuerzo. Como se observa, existe una diferencia significativa (de acuerdo a α=0.1) de 56.5 minutos en favor de la programación individual (30% de decremento en el esfuerzo). Con un intervalo de confianza del 95%, esta diferencia varía ente 8.7 y 104.2 minutos (entre 4% y 55% de decremento en esfuerzo).

TABLA X. COMPARACIÓN CON RESPECTO AL ESFUERZO

Comparación	Diferencia	Valor *p*	LCI (95%)	LCS (95%)
Prog. en pareja – Prog. ind.	56.5	0.1017	8.7967	104.2032

V. DISCUSIÓN

En mayor o menor medida, los experimentos están sujetos a diferentes tipos de amenazas que pueden incidir de manera negativa en sus resultados. Cook y Campbell [28] agrupan diferentes tipos de amenazas en cuatro categorías. De acuerdo a estas categorías, a continuación se describen tanto aquellas amenazas que pueden impactar en los resultados de este experimento así como se describen sugerencias de mejora para tomar en cuenta en futuras versiones de este experimento.

A. Amenazas a la Validez de Conclusión Estadística

Estas amenazas se refieren a cuestiones que pueden afectar la habilidad de obtener una conclusión correcta acerca de la existencia de una relación entre el tratamiento y la variable de respuesta. Algunas amenazas en esta categoría que pudieron afectar los resultados de este experimento son las siguientes:

- Aunque el modelo estadístico usado satisface de manera satisfactoria la independencia y normalidad, el modelo pudo verse afectado por una falta de homocedasticidad. Se han identificado los programas como posibles causantes de esta situación, por lo que en versiones posteriores de este experimento se pretende emplear programas con un nivel de complejidad similar.
- Otra amenaza que pudo haber impactado en la validez de conclusión es la referente a la confiabilidad de las mediciones. Aunque todas las mediciones se obtuvieron en la segunda sesión del experimento, algunas mediciones no se recolectaron durante la primera sesión por falta de tiempo. En esta sesión los sujetos que trabajaron de forma individual se les pidió terminaran el ejercicio en casa y registraran sus tiempos.

B. Amenazas a la Validez Interna

Las amenazas en esta categoría se refieren a si los resultados observados se debieron no a los tratamientos sino a otros factores. Para abordar estas amenazas, los sujetos se asignaron de manera aleatoria a los tratamientos. El diseño cuadrado latino empleado mitiga problemas con el efecto de aprendizaje, fatiga o cansancio ya que los sujetos codificaron diferentes programas empleando diferentes herramientas. Los sujetos participaron en el experimento bajo mismas condiciones y de forma separada (parejas e individuos) para que no hubiese interacción entre ellos.

C. Amenazas a la Validez de Constructo

Las amenazas en esta categoría se refieren a aspectos que pueden influir negativamente en la relación que hay entre la teoría y la observación. Una posible amenaza que pudo afectar esta validez en el experimento es que los sujetos no contaban con experiencia previa sobre programación en pareja así como los sujetos que trabajaron en pareja no habían programado antes con su compañero. Los resultados presentados en este experimento pueden considerarse como conservadores. En versiones futuras de este experimento se reforzará esta validez asignando más programas a las parejas para que cuenten con mayor experiencia.

D. Amenazas a la Validez Externa

Estas amenazas se refieren a aspectos que pueden limitar la habilidad de generalizar los resultados del experimento a otros contextos, por ejemplo generalizar los resultados a la industria. El uso de estudiantes como sujetos en lugar de profesionales tal vez pudo afectar este tipo de validez. No obstante, como se menciona en [29], el uso de estudiantes como sujetos permite al investigador en IS obtener evidencia preliminar para confirmar o refutar hipótesis que pueden ser contrastadas posteriormente en contextos industriales.

Una vez presentadas las limitaciones del experimento, a continuación se discuten algunos resultados obtenidos en experimentos similares y se contrastan (respecto a la duración y esfuerzo) con los resultados arrojados en este experimento.

E. Duración

En el experimento de Nosek [19] se emplearon 15 profesionales donde se agruparon en 5 parejas y 5 individuos. Los sujetos codificaron un script afín al área de bases de datos. Los resultados indican un decremento del 29% en la duración a favor de la programación en pareja.

Williams et al. [15] emplearon 41 estudiantes agrupados en 14 parejas y 13 individuos. Durante el experimento los sujetos completaron cuatro ejercicios. Los autores reportan una reducción de tiempo en la compleción de ejercicios entre un 40% y 50% con respecto a los programadores individuales.

Nawrocki and Wojciechowski [16] emplearon 16 estudiantes como sujetos (5 parejas y 6 individuos) donde codificaron cuatro programas. Los autores no encontraron diferencias significativas entre la programación en pareja y la programación individual.

Lui and Chan [17] emplearon 5 parejas y 5 individuos. Los autores reportan un decremento del 52% de tiempo en favor de la programación en pareja.

Müller [18] empleó 38 estudiantes (14 parejas y 13 individuos). Los estudiantes trabajaron sobre cuatro ejercicios de programación donde los ejercicios se descompusieron en fases de implementación y aseguramiento de la calidad, así como la

fase completa (implementación más aseguramiento de la calidad). El autor reporta un incremento de tiempo de 11% para la programación individual con respecto a la fase de implementación.

Arisholm et al. [14] emplearon 295 profesionales agrupados en 98 parejas y 99 individuos. Los sujetos realizaron diferentes tareas de mantenimiento sobre dos sistemas. Los autores reportan un decremento de tiempo del 8% en favor de la programación en pareja.

En contraste, los resultados reportados en el presente experimento sugieren un decremento del 28% en la duración de la compleción de los ejercicios a favor de la programación en pareja. Estos resultados refuerzan aquellos observados en [19].

F. Esfuerzo

No en todos los experimentos se reporta esta medición, en algunos experimentos fue posible calcular esta medición de acuerdo a los datos disponibles (se multiplicó por dos el tiempo empleado por las parejas).

En el caso de Nosek [19] se calculó esta medición que indica un decremento del 29% a favor de la programación individual. Por el contrario, los resultados de Lui y Chan [17] indican un decremento del 4% a favor de la programación en pareja. Por último, Arisholm et al. [14] Reportan un incremento en el esfuerzo que conlleva programar en pareja del 84%.

En contraste, los resultados aquí reportados sugieren un decremento en el esfuerzo de la programación individual del 30%. De nuevo, estos resultados refuerzan aquellos observados en [19].

VI. Conclusiones

Actualmente, la comunidad científica en IS está de acuerdo en la carencia de una base teórica sólida para la disciplina [30]-[34]. Esto significa que no existe conocimiento suficiente para realizar una evaluación y selección adecuada de la gama de tecnologías que dispone el ingeniero de software.

A través del paradigma experimental es posible obtener observaciones de manera objetiva logrando así acumular un cuerpo de conocimientos en IS. Este cuerpo de conocimientos puede entonces ser usado para comenzar a construir teorías que puedan ser aplicadas en beneficio de los ingenieros de software.

En este artículo se presentó el paradigma experimental aplicado a la IS para estudiar los efectos de la programación en pareja. El objetivo de este experimento se centró en estudiar la duración y el esfuerzo que conlleva la programación en pareja.

El grupo de parejas de sujetos que codificaron los ejercicios de programación les tomó menos tiempo (28% menos de tiempo) que aquellos quienes trabajaron individualmente. Por el contrario, al grupo de parejas de sujetos les llevó mayor esfuerzo (30% mayor) que aquellos quienes trabajaron individualmente. Los resultados aquí reportados son muy similares a los reportados en [19].

Con el fin de mejorar la práctica de investigación en IS, en este artículo se reportan todas las mediciones obtenidas del experimento. Esta información puede ser de utilidad para que otros investigadores corroboren o re-analicen [35] los resultados aquí arrojados. Esta información también puede emplearse para sintetizarla con otros experimentos [36].

Referencias

[1] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in Software Engineering: An Introduction.Boston: Kluwer Academic Publishers, 1999.

[2] N. Juristo, A. M. Moreno, and S. Vegas, "A survey on testing technique empirical studies: How limited is our knowledge," in Proceedings of the 2002 International Symposium on Empirical Software Engineering. Washington, DC, USA: IEEE Computer Society, 2002, pp. 161.

[3] M. V. Zelkowitz, D. R. Wallace, and D. W. Binkley, Experimental validation of new software technology. River Edge, NJ, USA: World Scientific Publishing Co., Inc., 2003, pp. 229–263.

[4] M. Shaw, "Prospects for an engineering discipline of software," IEEE Softw., vol. 7, no. 6, pp. 15–24, 1990.

[5] ——, "Continuing prospects for an engineering discipline of software," IEEE Software, vol. 26, no. 6, pp. 64–67, 2009.

[6] G. M. Weinberg and G. L. Gressett, "An experiment in automatic verification of programs," Commun. ACM, vol. 6, pp. 610–613, October 1963.

[7] E. E. Grant and H. Sackman, "An exploratory investigation of programmer performance under on-line and off-line conditions," IEEE Transactions on Human Factors in Electronics, vol. 8, no. 1, pp. 33–48, 1967.

[8] K. Kolence, "The software empiricist experimental disciplines & computer measurements," SIGMETRICS Perform. Eval. Rev., vol. 2, no. 3, pp. 21–23, Sep. 1973.

[9] D. R. Hanson, "Ratsno - an experiment in software adaptability." Software - Practice and Experience, vol. 7, no. 5, pp. 625–630, 1977.

[10] V. Basili, R. Selby, and D. Hutchens, "Experimentation in software engineering," IEEE Trans. Softw. Eng., vol. 12, no. 7, pp. 733–743, 1986.

[11] S. L. Pfleeger, "Albert Einstein and empirical software engineering," Computer, vol. 32, no. 10, pp. 32–38, 1999.

[12] K. Beck, "Embracing change with extreme programming," Computer, vol. 32, no. 10, pp. 70–77, 1999.

[13] ——, Extreme programming explained: embrace change. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2000.

[14] E. Arisholm, H. Gallis, T. Dybå, and D. I. Sjøberg, "Evaluating pair programming with respect to system complexity and programmer expertise," IEEE Transactions on Software Engineering, vol. 33, no. 2, pp. 65–86, 2007.

[15] L. Williams, R. Kessler, W. Cunningham, and R. Jeffries, "Strengthening the case for pair programming," Software, IEEE, vol. 17, no. 4, pp. 19 –25, jul/aug 2000.

[16] J. Nawrocki and A. Wojciechowski, "Experimental evaluation of pair programming," in Proceedings of the 12th European Software Control and Metrics Conference, London, April 2001, pp. 269–276.

[17] K. M. Lui and K. C. C. Chan, "When does a pair outperform two individuals?" in Proceedings of the 4th international conference on Extreme programming and agile processes in software engineering, XP'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 225–233.

[18] M. M. Müller, "Two controlled experiments concerning the comparison of pair programming to peer review," Journal of Systems and Software, vol. 78, no. 2, pp. 166 – 179, 2005.

[19] J. T. Nosek, "The case for collaborative programming," Commun. ACM, vol. 41, no. 3, pp. 105–108, Mar. 1998.

[20] V. Basili, G. Caldiera, and H. Rombach, "Goal question metric paradigm," Encyclopedia of Software Eng, pp. 528–532, 1994, john Wiley & Sons.

[21] H. L. Dreyfus and S. Dreyfus, Mind over Machine. The Power of Human Intuition and Expertise in the Era of the Computer. New York: Basil Blackwell, 1986.

[22] H. Levene, "Robust tests for equality of variances," in Contributions to probability and statistics, I. Olkin, Ed. Palo Alto, CA: Stanford Univ. Press., 1960.

[23] A. N. Kolmogorov, "Sulla determinazione empirica di una legge di distribuzione," Giornale dell'Istituto Italiano degli Attuari, vol. 4, pp. 83–91, 1933.

[24] N. V. Smirnov, "Table for estimating the goodness of fit of empirical distributions," Ann. Math. Stat., vol. 19, pp. 279–281, 1948.

[25] R. Kuehl, Design of Experiments: Statistical Principles of Research Design and Analysis, 2nd ed. California, USA.: Duxbury Thomson Learning, 2000.

[26] J. W. Tukey, "One degree of freedom for non-additivity," Biometrics, vol. 5, no. 3, pp. pp. 232–242, 1949.

[27] H. Scheffé, "A method for judging all contrasts in the analysis of variance," Biometrika, vol. 40, no. 1/2, pp. 87–104, 1953.

[28] T. Cook and D. Campbell, The design and conduct of quasi-experiments and true experiments in field settings. Chicago: Rand McNally, 1976.

[29] J. Carver, L. Jaccheri, S. Morasca, and F. Shull, "Issues in using students in empirical studies in software engineering education," in METRICS '03: Proceedings of the 9th International Symposium on Software Metrics. Washington, DC, USA: IEEE Computer Society, 2003, p. 239.

[30] W. Tichy, "Should computer scientists experiment more?" Computer, vol. 31, no. 5, pp. 32–40, 1998.

[31] B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," IEEE Transactions on Software Engineering, vol. 28, no. 8, pp. 721–734, 2002.

[32] J. Hannay, D. Sjøberg, and T. Dybå, "A systematic review of theory use in software engineering experiments," Software Engineering, IEEE Transactions on, vol. 33, no. 2, pp. 87 –107, feb. 2007.

[33] D. I. K. Sjøberg, T. Dybå, B. C. D. Anda, and J. E. Hannay, "Building theories in software engineering," Guide to Advanced Empirical Software Engineering, pp. 312–336, 2008.

[34] F. Shull and R. L. Feldmann, "Building theories from multiple evidence sources," Guide to Advanced Empirical Software Engineering, pp. 337–364, 2008.

[35] O. S. Gómez, N. Juristo, and S. Vegas, "Replication, reproduction and re-analysis: Three ways for verifying experimental findings," in International Workshop on Replication in Empirical Software Engineering Research (RESER'2010), Cape Town, South Africa, May 2010.

[36] L. V. Hedges and I. Olkin, Statistical methods for meta-analysis. Academic Press, Orlando, 1985.

Omar S. Gómez es Ingeniero en Computación por la Universidad de Guadalajara (México, 2001), Maestro en Ingeniería de Software por el Centro de Investigación en Matemáticas (México, 2005), y Doctor en Software y Sistemas por la Universidad Politécnica de Madrid (España, 2012). Actualmente es Profesor Asociado en la Facultad de Matemáticas de la Universidad Autónoma de Yucatán (México). Su trabajo de investigación se centra en Experimentación en Ingeniería Software así como en temas relacionados con la Calidad del Software y el Diseño de Software.

Julio C. Díaz es Ingeniero Industrial en Producción por el Instituto Tecnológico de Mérida (ITM). Candidato al grado de Maestro en Ciencias de la Computación, por el Centro Nacional de Cálculo del Instituto Politécnico Nacional. Especialista en Docencia de la Universidad Autónoma de Yucatán. Actualmente imparte asignaturas en las Licenciaturas en Ingeniería de Software, y en Ciencias de la Computación de la Facultad de Matemáticas, relacionadas con las áreas de Ingeniería de Software. Su interés se enfoca en Procesos de Software e Ingeniería de Software Educativa.

Raúl A. Aguilar es Licenciado en Ciencias de la Computación por la Universidad Autónoma de Yucatán, Doctor en Informática por la Universidad Politécnica de Madrid, España. Actualmente es profesor de tiempo completo en la Facultad de Matemáticas de la Universidad Autónoma de Yucatán. Su trabajo de investigación incluye las siguientes áreas: Ingeniería de Software e Informática Educativa.