Abstraction & Application **8** (2013) 26 – 37 UADY

Aplicación del proceso de experimentación a la Ingeniería de Software

Omar S. Gómez*, Juan P. Ucán, Gerzon E. Gómez

Facultad de Matemáticas, Universidad Autónoma de Yucatán, México

*omar.gomez@uady.mx

Abstract

Contrary to other engineering disciplines, Software Engineering (SE) is a young discipline, where at the moment, there is not enough solid knowledge that can be applied effectively by a software engineer. A way of building and consolidating SE knowledge is through experimentation. Through experimentation it is possible to obtain objective observations, which allow to accumulate a body of knowledge in SE. This empirical knowledge can be codified and applied effectively by a software engineer. In this paper we present an experiment process applied to SE. The activities of this process are illustrated using an experiment that studies a couple of aspects of pair programming. This process can be applied either by a software engineer or by a SE researcher.

Resumen

A diferencia de otras disciplinas ingenieriles, la Ingeniería de Software (IS) es una disciplina joven, donde al día de hoy, se carece de suficiente conocimiento sólido que pueda ser aplicado de manera eficaz por el ingeniero de software. Una vía para construir y consolidar este conocimiento es a través de la experimentación. A través de la experimentación es posible obtener observaciones objetivas que permitan acumular un cuerpo de conocimientos en IS. Este conocimiento puede ser entonces codificado para ser aplicado de manera efectiva por el ingeniero de software. En este artículo se presenta un proceso de experimentación aplicado a la IS. Las actividades de este proceso se ejemplifican con un experimento que estudia algunos aspectos de la programación en pareja. Este proceso puede ser aplicado tanto por el ingeniero de software como por el investigador en IS.

Keywords and phrases : ingeniería de software, programación en pareja, proceso de experimentación, experimento.

2010 *Mathematics Subject Classification* : 68N30.

---

1. Introducción

La Ingeniería de Software (IS) es un disciplina joven cuyo objetivo principal es la aplicación de enfoques sistemáticos, disciplinados y cuantificables para diseñar, desarrollar, operar y mantener un producto software[1].

Básicamente, en cualquier ingeniería se codifica el conocimiento científico referente a un problema de dominio tecnológico de tal manera que éste sea directamente útil para el profesional, proporcionando así respuestas a preguntas que ocurren comúnmente en la práctica [24].

Por ejemplo, en una ingeniería tan antigua como la ingeniería civil, durante varios siglos se desarrollaron las bases científicas teóricas referentes al problema de la construcción de estructuras. Este conocimiento teórico se ha codificado de tal forma que en la actualidad el ingeniero civil lo aplica para la resolución de problemas referentes al diseño, construcción o mantenimiento de estructuras, obras hidráulicas o de transporte.

En el caso de la IS aún no se cuenta con suficiente evidencia que apoye la mayoría de las prácticas usadas para la construcción de software [14]. Esto se debe a la carencia de una base teórica sólida para la disciplina[2].

En IS, los resultados de la aplicación de una determinada tecnología para la construcción de software son, al día de hoy, impredecibles [34]. A menudo se adoptan tecnologías nuevas sin contar con evidencia convincente de que serán efectivas [35].

En IS, aún hace falta codificar el conocimiento científico de este disciplina de tal manera que pueda ser aplicado de manera efectiva por el ingeniero de software. Una vía para lograr tal fin es a través de la experimentación.

A través de la experimentación es posible obtener observaciones objetivas que permitan acumular un cuerpo de conocimientos en IS. Este conocimiento puede ser entonces codificado para que sea aplicado por el ingeniero de software. En la historia de 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 [25, 24].

En este artículo se presenta un proceso de experimentación para aplicarlo a la IS. Este proceso se ejemplifica con un experimento afín a la IS que estudia algunos efectos de la programación en pareja. El resto del documento se encuentra organizado de la siguiente manera: en la sección 2 se describe de manera general la experimentación en ingeniería de software. En la sección 3 se presenta un proceso de experimentación que puede aplicarse a la IS. En la sección 4 se describe la programación en pareja así como se discuten experimentos que han estudiado esta práctica. En la sección 5 se ejemplifica la aplicación del proceso de experimentación usando como referencia un experimento que estudia la programación en pareja. Por último, en la sección 6 se presentan las conclusiones.

2. Experimentación en Ingeniería de Software

La aplicación del enfoque experimental aplicado a la IS cuenta ya con varias décadas. Los primeros intentos de aplicar este enfoque se remontan a los años 60's y 70's [12, 16, 9, 30], donde distintos investigadores evaluaban tecnologías que apoyaban la construcción de software; esto lo realizaban a través de aproximaciones científicas basadas en la observación. No obstante, no fue sino hasta los 80's cuando se enfatizó la necesidad de aplicar la experimentación a la IS [3].

De manera general, el objetivo de la experimentación consiste en identificar las causas por las que se producen determinados resultados. Al aplicarla a la IS, la experimentación ayuda a identificar y comprender distintas variables así como conexiones que entran en juego en la construcción de software. Esta identificación de variables y conexiones permite validar o refutar con hechos las prácticas que basamos para construir

[1] Donde un producto software también puede entenderse como programa, sistema o aplicación software.

[2] Tal como se discute en los trabajos de Shull y Feldmann [26], Sjøberg et al. [27], Hannay et al. [11], Kitchenham et al. [15], así como Tichy [29].

software. Así pues, la meta de la experimentación en IS es poder prescribir soluciones que estén soportadas por un cuerpo de conocimientos validado por evidencia científica.

La unidad principal del enfoque experimental es el experimento. En un experimento se modelan las principales características de una realidad (en este caso, la construcción de software) bajo condiciones que pueden controlarse, permitiendo así estudiar y comprender mucho mejor esa realidad. De esta forma, los experimentos son útiles para ayudar a confirmar o refutar teorías, creencias (juicios convencionales), o explorar relaciones causales.

La estructura básica de un experimento se compone por dos tipos de variables que son: factores y variables de respuesta. Los factores son aquellas variables que pueden manipularse o controlar en el experimento, mientras que las variables de respuesta son variables que se analizan para observar el efecto que producen los cambios en los factores.

Por ejemplo, en un experimento donde se analiza la duración y el esfuerzo que conlleva programar en pareja respecto a programar de manera individual, el "tipo de programación" actúa como un factor que en este caso se conforma de dos niveles o tratamientos: programación en pareja y programación individual. Por otra parte, la duración y el esfuerzo actúan como variables de respuesta. A través de estas variables es posible observar la existencia de un posible efecto ocasionado por estos dos tipos de programación.

3. Proceso de Experimentación

Algunos investigadores en IS han definido procesos para apoyar la realización de experimentos. Cada uno de estos procesos consta de varias actividades. En la Tabla 1 se muestran las actividades principales de estos procesos. De manera general, el proceso de experimentación en IS puede dividirse en cuatro actividades principales que son: definición, diseño o planificación, ejecución y análisis.

Tabla 1: Actividades del proceso de experimentación en IS

	Definición	**Diseño**	**Ejecución**	**Análisis**	
Basili et al. [3]	definición	planificación	operación	interpretación	
Wohlin et al. [34]	definición	planificación	operación	análisis e interpretación	presentación y empaquetado
Juristo y Moreno [13]	objetivo	diseño	ejecución	análisis	
Wohlin et al. [33]	alcance	planificación	operación	análisis e interpretación	presentación y empaquetado

Empleando el lenguaje UML (Unified Modeling Language) [23], en la Figura 1 se representan las actividades principales del proceso de experimentación en IS, donde cada actividad se realiza de manera secuencial. A continuación se describe cada una de estas actividades.

3.1. Definición

Esta actividad consiste en especificar de manera general los aspectos principales del experimento a realizar así como en definir las hipótesis de trabajo. En esta fase es posible usar el método GQM (Goal-Question-Metric) [2] que facilita definir: el objeto de estudio del experimento, el propósito, el enfoque de calidad, la perspectiva así como el contexto donde se realiza el experimento.

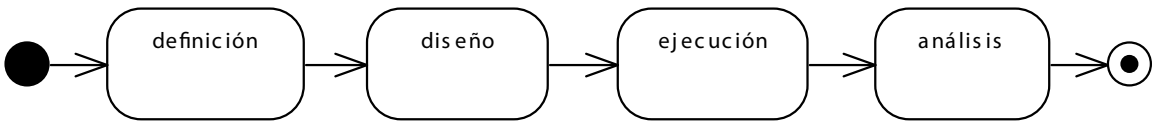

Figura 1: Actividades del proceso de experimentación en IS

En esta actividad Basili et al. [3] así como Wohlin et al. [33, 34] emplean el método GQM para especificar el objetivo y las metas del experimento. Por otra parte, Juristo y Moreno [13] definen las hipótesis a examinar a partir de la especificación del objetivo del experimento. Estos dos enfoques pueden complementarse para llevar a cabo esta actividad.

3.2. Diseño o Planificación

Una vez definido el experimento y sus hipótesis, la siguiente fase de este proceso consiste en diseñar cómo se llevará a cabo el experimento. En esta actividad se especifica cómo se asignan los tratamientos a los sujetos, el tipo de sujetos a emplear así como se preparan los instrumentos o materiales para realizar o ejecutar el experimento. Respecto a cómo asignar los tratamientos a los sujetos, en la literatura existen diferentes tipos de diseños experimentales que satisfacen propósitos particulares (revisar [13, 6] para profundizar en el tema).

Aunque Basili et al. [3] así como Wohlin et al. [33, 34] identifican esta actividad como "planificación", en ella se abordan principalmente cuestiones relacionadas con el tipo de diseño experimental a utilizar.

3.3. Ejecución

Una vez definido el experimento y seleccionado algún diseño, es momento de llevar a cabo el experimento. En esta fase se tienen listos los materiales a utilizar, los equipos, así como el espacio donde se llevará a cabo. Antes de comenzar la sesión del experimento, los sujetos reciben una serie de instrucciones, así como los materiales con los que trabajarán durante la sesión del experimento.

3.4. Análisis

Tras ejecutar el experimento, el investigador recolecta una serie de métricas generadas por los sujetos. En esta actividad del proceso, las métricas son analizadas y contrastadas con las hipótesis previamente definidas. En esta actividad se emplean diferentes técnicas estadísticas para analizar e interpretar las métricas obtenidas. Usualmente en el diseño de experimentos se emplea el análisis de la varianza [8, 7] (en Inglés, ANOVA). Básicamente el ANOVA consiste en examinar la variabilidad de las métricas obtenidas con respecto a los tratamientos aplicados, así como en examinar la variabilidad de otros tipos de variables (como pueden ser variables de bloqueo o covariables). El ANOVA proporciona una prueba estadística acerca de si las medias o promedios de varios grupos son o no equivalentes.

Wohlin et al. [33, 34] añaden otra actividad al proceso de experimentación identificada como "presentación y empaquetado". Esta actividad consiste en documentar y presentar los hallazgos encontrados en el experimento. No obstante esta actividad puede considerarse opcional y prescindir de ella como sucede en los procesos definidos por Basili et al. [3] y Juristo y Moreno [13].

En la siguiente sección se describe la programación en pareja así como se describen experimentos que han estudiado esta práctica. Tomando como referencia este contexto, se usará un experimento que estudia un par de aspectos de la programación en pareja para ejemplificar las actividades del proceso de experimentación antes descritas.

4. Programación en Pareja

Una de las doce prácticas principales de la programación extrema creada a finales de los 90's por Kent Beck [5, 4] consiste en programar en pareja. Básicamente, en esta práctica dos programadores trabajan en conjunto resolviendo una o varias tareas 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, esto 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 consultar referencias para ayudar a resolver alguna cuestión que se presente durante la realización de la tarea.

Se dice que el uso de 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 a la duración de realizarla de manera individual [19, 20, 32, 31, 22].

Se han realizado algunos experimentos que estudian los beneficios de esta práctica [1, 20, 18, 21, 31, 22]. A continuación se describen algunos resultados reportados en experimentos que examinan la programación en pareja con respecto al tiempo de compleción de tareas y al esfuerzo necesario para completar las tareas.

4.1. Resultados con respecto al tiempo para completar una o varias tareas

Nosek [22] usó 15 profesionales agrupados 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 de completar las tareas a favor de la programación en pareja.

En otro experimento, Williams et al. [31] usaron 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.

Por otra parte, Nawrocki and Wojciechowski [21] 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.

En otro experimento, Lui and Chan [18] emplearon 5 parejas y 5 individuos. Los autores reportan un decremento del 52 % de tiempo en favor de la programación en pareja.

Por su parte, Müller [20] 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. [1] 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.

4.2. Resultados con respecto al esfuerzo para completar una o varias tareas

De acuerdo a Arisholm et al. [1], el esfuerzo puede definirse como la cantidad de trabajo requerido para realizar una tarea, es decir, el esfuerzo total de programación en persona-minutos para codificar un programa. Una forma de obtener el esfuerzo total de las parejas es duplicar el tiempo requerido por la pareja para completar la tarea, esto debido a que se emplean dos recursos (la pareja).

En el caso de Nosek [22] se calculó esta métrica donde se observa un decremento del 29 % a favor de la programación individual. Por el contrario, los resultados de Lui y Chan [18] indican un decremento del 4 % a favor de la programación en pareja. Por último, Arisholm et al. [1] reportan un incremento del 84 % en el esfuerzo que conlleva la programación en pareja .

5. Aplicación del Proceso de Experimentación

Una vez presentado el contexto de experimentos donde se ha estudiado la programación en pareja, a continuación se ejemplifican las actividades del proceso de experimentación empleando como referencia un

experimento que estudia un par de aspectos de la programación en pareja realizado en la Facultad de Matemáticas de la Universidad Autónoma de Yucatán (UADY) [10].

5.1. Definición

Esta primera actividad consiste en especificar de manera general los aspectos principales del experimento a realizar, así como en definir las hipótesis de trabajo. Por ejemplo, usando el método GQM (Goal-Question-Metric) [2] para identificar el objeto de estudio, propósito, enfoque de calidad, perspectiva, así como el contexto, un experimento puede definirse de la siguiente manera:

"Estudiar la programación en pareja y la programación individual (objeto de estudio) con el propósito de evaluar posibles diferencias entre estas dos formas de programar con respecto al esfuerzo (enfoque de calidad) que conlleva cada tipo de programación. El estudio se realiza desde el punto de vista (perspectiva) del investigador en el contexto de un aula con computadoras donde alumnos de licenciatura realizarán en pareja o de manera individual dos programas".

La perspectiva y el contexto pueden variar dependiendo de dónde se realice el experimento. Por ejemplo, imaginemos que en el área de procesos de una empresa de desarrollo de software han previsto institucionalizar esta práctica en los equipos de desarrollo, no obstante, antes de implementarla se decide realizar un experimento para conocer de manera más confiable si esta práctica podría redituar beneficios en la empresa.

En este ejemplo, el experimento se realiza desde la perspectiva del personal del área de procesos de la empresa y el contexto se conforma por una muestra de programadores pertenecientes a distintos equipos de desarrollo.

Dos hipótesis que pueden definirse para contrastarlas con las observaciones (métricas) generadas por el experimento son:

- H_{0_a}: El tiempo requerido para codificar un programa utilizando programación en pareja es equivalente al tiempo requerido para codificarlo de manera individual o, programación en pareja = programación individual con respecto a la duración.
- H_{0_b}: El esfuerzo requerido para codificar un programa utilizando programación en pareja es equivalente al esfuerzo requerido para codificarlo de manera individual o, programación en pareja = programación individual con respecto al esfuerzo.

Al tipo de hipótesis antes planteada se le conoce como hipótesis nula (H_0) y se especifica de tal modo que no tenga valor o efecto. Esta hipótesis se acepta como verdadera hasta que el contraste con alguna evidencia estadística demuestre lo contrario, es decir la evidencia conformada por observaciones respalde una hipótesis alternativa (H_1). En este caso, la existencia de una diferencia entre la programación en pareja y la programación individual con respecto al tiempo de completar alguna tarea y/o al esfuerzo de realizarla.

5.2. Diseño

En esta actividad se especifica cómo se llevará a cabo el experimento, cómo se asignan los tratamientos a los sujetos, el tipo de sujetos a emplear así como la preparación de instrumentos o materiales para ejecutar el experimento.

Por ejemplo, en el experimento realizado en Gómez et al. [10], al tipo de diseño empleado se le conoce como Cuadrado Latino [17]. La característica de este diseño es que emplea dos tipos de variable conocidas como variables de bloqueo las cuales ayudan a incrementar la precisión de las observaciones obtenidas en el experimento. Es decir, ya que en un experimento existen diversos factores que pueden influir en los resultados, este diseño experimental ayuda a aislar fuentes de variación no deseada, por lo que se tienen resultados con mayor precisión. En la Tabla 2 se muestra la organización de los tratamientos acorde a este tipo de diseño.

Tabla 2: Diseño cuadrado Latino usado en el experimento

Programa / Herramienta	NetBeans IDE	Editor de texto
Programa 1	Individual	Par
Programa 2	Par	Individual

Como se observa en la Tabla 2, las dos variables de bloqueo son programa y herramienta, mientras que los tratamientos de interés son los referentes a la programación en pareja e individual. A continuación se describen más detalles sobre la asignación de los tratamientos y las variables de bloqueo usadas en este experimento.

En este experimento participaron 21 estudiantes universitarios (sujetos) inscritos en uno de los cursos de la carrera de IS de la UADY. La mayoría de ellos, en su tercer año de carrera. Los sujetos se asignaron de manera aleatoria a dos grupos que representan los tratamientos a examinar, el grupo de sujetos que trabajó en pareja (siete parejas) y aquellos que trabajaron individualmente (siete solos).

El experimento se dividió en dos sesiones, en cada sesión los sujetos codificaron un programa diferente (variable de bloqueo). En la primera sesión los sujetos que trabajaron individualmente usaron el NetBeans IDE[3] 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 (variable de bloqueo), los sujetos que antes trabajaron individualmente con NetBeans IDE ahora trabajaron con un editor de texto, mientras que los sujetos que trabajaron en pareja codificaron el segundo programa con NetBeans IDE así como intercambiaron los roles (controlador y observador). Para las sesiones del experimento se eligieron programas pequeños (aproximadamente 100 líneas de código) que los sujetos pudieran codificar en cada sesión.

En esta actividad del proceso de experimentación también se preparan los materiales y equipos a usar previo a la ejecución del experimento. Por ejemplo, se prepararon las especificaciones a usar para codificar los programas de manera individual y en pareja, se instaló el NetBeans IDE en los equipos de cómputo y se verificó el funcionamiento correcto de las computadoras a utilizar.

5.3. Ejecución

La siguiente actividad del proceso de experimentación consiste en aplicar los tratamientos a los sujetos bajo condiciones controladas. Continuando con el ejemplo, en esta actividad cada sesión tuvo una duración de 90 minutos. Las dos sesiones se llevaron a cabo en uno de los salones del centro de cómputo de la facultad. Al comienzo de cada sesión se les explicó a los sujetos la especificación del programa a codificar. Los sujetos registraron el tiempo de inicio y final que les llevó codificar el programa. La diferencia de estos tiempos (calculada en minutos) se usó como métrica para obtener la duración y el esfuerzo. Respecto a la métrica de esfuerzo, que mide la cantidad de trabajo gastado para realizar una tarea (en este caso, persona-minuto), se duplicó el tiempo usado por las parejas. Los tiempos de inicio y fin registrados por los sujetos se usaron para definir las métricas con respecto a la duración y el esfuerzo. En la Tabla 3 se muestran las métricas recolectadas con respecto a la duración mientras que en la Tabla 4 se muestran las métricas obtenidas con respecto al esfuerzo.

Tabla 3: Métricas obtenidas con respecto a la duración

	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

[3]Donde IDE es un acrónimo en Inglés de Integrated Development Environment. Consultar la url http://netbeans.org para más información de este entorno de desarrollo.

Tabla 4: Métricas obtenidas con respecto al esfuerzo

	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

5.4. Análisis

Una vez obtenidas las métricas tras ejecutar el experimento, la última actividad de este proceso consiste en realizar un análisis. En esta actividad se emplean diferentes técnicas estadísticas para analizar e interpretar las métricas recolectadas.

En este ejemplo, los grupos de medias obtenidas pertenecen a programadores en pareja y programadores individuales. En la Tabla 5 se muestra de manera abreviada los resultados tras aplicar dos ANOVAS, una por cada métrica.

Tabla 5: Valores p obtenidos tras el análisis de la varianza

Factor	Métrica	Prueba F	valor p
Tipo de programación	Duración	2.98	0.09
Tipo de programación	Esfuerzo	2.89	0.1

El ANOVA usa un tipo de prueba conocido como prueba F de Fisher [28] que ayuda al investigador a determinar si hay o no una diferencia significativa entre los tratamientos examinados en el experimento. El valor p es la probabilidad de obtener un resultado (en este caso F) al menos tan extremo como el observado. Usualmente el investigador define un valor crítico llamado alfa (α) para contrastarlo con el valor p observado. Si un valor p es menor o igual al valor alfa entonces la hipótesis nula es rechazada. Por ejemplo de acuerdo a los valores p observados, si se fija un valor $\alpha = 0.1$ en ambos casos se rechaza la hipótesis nula y entonces se acepta la alternativa. Esto significa que si este experimento se realizara 100 veces bajo las mismas condiciones, el 90 % de las veces se obtendría una diferencia significativa entre estos dos tipos de programación.

En la Tabla 6 se muestran las diferencias obtenidas (parejas y solos) en minutos con respecto a la duración y al esfuerzo. Como se observa en esta tabla, se obtuvo una diferencia de 36 minutos a favor de la programación en pareja (28 % decremento). Con un nivel de confianza del 95 % esta diferencia puede variar de 6 a 66 minutos[4] (decremento de 4 % a 51 %). Con respecto al esfuerzo, se observa una diferencia de 56 minutos a favor de la programación individual (30 % decremento). Este valor puede variar de 8 a 104 minutos (decremento de 4 % a 55 %).

Tabla 6: Diferencias entre parejas y solos con respecto a la duración y el esfuerzo

Métrica	Diferencia	valor p	LCI (95 %)	LCS (95 %)
Duración	36.5	0.09	6.15	66.98
Esfuerzo	56.5	0.1	8.79	104.2

En esta actividad también pueden emplearse gráficos. Por ejemplo, en la Figura 2 se muestra un histograma que indica el tiempo promedio que les llevó a los sujetos completar los programas. Como se observa en la Figura 2, el tiempo promedio de programación invertido por las parejas es de 93, mientras que los sujetos que trabajaron solos necesitaron 129 minutos para completar los programas.

Otros análisis que pueden realizarse en esta actividad son la prueba de normalidad para evaluar la normalidad del modelo estadístico, en este caso ANOVA, o estimar el tamaño de los efectos observados en los tratamientos.

En el ANOVA se asume que las observaciones obtenidas se encuentran normalmente distribuidas. En la

[4]De acuerdo a los límites de control inferior (LCI) y superior (LCS).

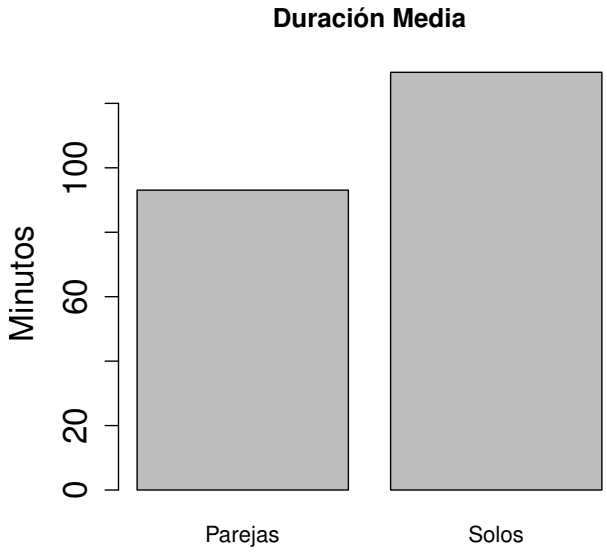

Figura 2: Duración promedio de las parejas y los programadores solos

Figura 3 se muestra un ejemplo de un gráfico conocido como cuantil-cuantil o "Gráfico Q-Q" que se emplea para evaluar la normalidad de un conjunto de observaciones.

En esta figura se ejemplifica la distribución con las métricas obtenidas ya normalizadas (residuos) del experimento. Estos residuos son entonces comparados con la distribución normal $N(0, 1)$. Cuanto más se asemeje la distribución de los puntos a una línea recta (bisectriz de los ejes de coordenadas), estos estarán aproximadamente normalizados, es decir, se asemejarán a los de una distribución normal.

En la Figura 3 se observa que la distribución de los puntos se asemeja a una línea recta por lo que puede decirse que las métricas obtenidas están aproximadamente normalizadas.

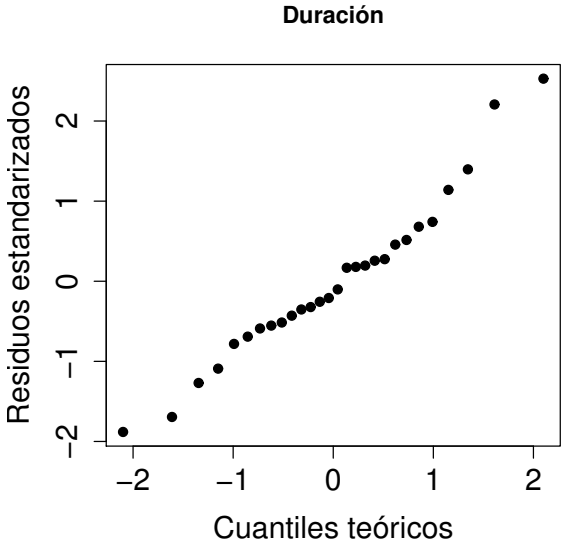

Figura 3: Gráfico Q-Q para diagnosticar la normalidad

6. Conclusiones

En este artículo se ha presentado un proceso de experimentación aplicado a la IS. Este proceso se ha ejemplificado con un experimento que estudia algunos efectos de la programación en pareja. El proceso de experimentación aquí descrito puede ser empleado tanto por el investigador como por el ingeniero de software. A continuación se describe un par de situaciones donde puede aplicarse este proceso.

Como investigador en IS, el proceso de experimentación permite estudiar de manera objetiva aspectos involucrados en la construcción de software. El conocimiento derivado de los experimentos puede entonces usarse para acumular un cuerpo de conocimientos y codificarse para que sea aplicado por el ingeniero de software. Como ingeniero de software, este proceso permite realizar experimentos para comparar de manera objetiva las diversas tecnologías que apoyan el proceso de construcción de software. Como compañía de software, el uso de este proceso puede ayudar a una compañía a validar tanto prácticas como tecnologías que se desean adoptar.

Agradecimientos. Reconocemos a los revisores anónimos los comentarios y sugerencias realizadas, que han permitido mejorar de manera sustancial el artículo aquí presentado.

Referencias

[1] E. Arisholm, H. Gallis, T. Dybå, and D.I.K. Sjøberg, *Evaluating pair programming with respect to system complexity and programmer expertise*, IEEE Transactions on Software Engineering **33** (2007), no. 2, 65–86, ISSN: 0098-5589.

[2] V.R. Basili, G. Caldiera, and H.D. Rombach, *Goal question metric paradigm*, Encyclopedia of Software Eng (1994), 528–532, John Wiley & Sons.

[3] V.R. Basili, R.W. Selby, and D.H. Hutchens, *Experimentation in software engineering*, IEEE Trans. Softw. Eng. **12** (1986), no. 7, 733–743, ISSN: 0098-5589.

[4] K. Beck, *Embracing change with extreme programming*, Computer **32** (1999), no. 10, 70–77, ISSN: 0018-9162.

[5] K. Beck, *Extreme programming explained: embrace change*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000, ISBN: 0-201-61641-6.

[6] G.E.P. Box, W.G. Hunter, J.S. Hunter, and W.G. Hunter, *Statistics for experimenters: An introduction to design, data analysis, and model building*, John Wiley & Sons, June 1978, ISBN: 0471093157.

[7] R.A. Fisher, *The correlation between relatives on the supposition of mendelian inheritance*, Transactions of the Royal Society Edinburgh **52** (1918), 399–433.

[8] R.A. Fisher, *The design of experiments*, Oliver & Boyd, Edimburgh, 1935.

[9] 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 **8** (1967), no. 1, 33–48.

[10] O.S. Gómez, J.L. Batún, and R.A. Aguilar, *Pair versus solo programming – an experience report from a course on design of experiments in software engineering*, Int. J. of Computer Science Issues **10** (2013), no. 1, 734–742, ISSN: 1694-0784.

[11] J.E. Hannay, D.I.K. Sjøberg, and T. Dyba, *A systematic review of theory use in software engineering experiments*, Software Engineering, IEEE Transactions on **33** (2007), no. 2, 87–107, ISSN: 0098-5589.

[12] D.R. Hanson, *Ratsno - an experiment in software adaptability.*, Software - Practice and Experience **7** (1977), no. 5, 625–630.

[13] N. Juristo and A.M. Moreno, *Basics of software engineering experimentation*, Kluwer Academic Publishers, 2001.

[14] N. Juristo, A.M. Moreno, and S. Vegas, *A survey on testing technique empirical studies: How limited is our knowledge*, Proceedings of the 2002 International Symposium on Empirical Software Engineering (Washington, DC, USA), IEEE Computer Society, 2002, ISBN: 0-7695-1796-X, pp. 161–.

[15] B.A. Kitchenham, S.L. Pfleeger, L.M. Pickard, P.W. Jones, D.C. Hoaglin, K. El Emam, and J. Rosenberg, *Preliminary guidelines for empirical research in software engineering*, IEEE Transactions on Software Engineering **28** (2002), no. 8, 721–734, ISSN: 0098-5589.

[16] K.W. Kolence, *Experiments & measures in computing.*, (1973), 69–72.

[17] R.O. Kuehl, *Design of experiments: Statistical principles of research design and analysis*, second ed., Duxbury Thomson Learning, California, USA., 2000.

[18] K.M Lui and K.C.C. Chan, *When does a pair outperform two individuals?*, Proceedings of the 4th international conference on Extreme programming and agile processes in software engineering (Berlin, Heidelberg), XP'03, Springer-Verlag, 2003, ISBN: 3-540-40215-2, pp. 225–233.

[19] K.M. Lui, K.C.C. Chan, and J. Nosek, *The effect of pairs in program design tasks*, IEEE Trans. Softw. Eng. **34** (2008), no. 2, 197–211, ISSN: 0098-5589.

[20] M.M. Müller, *Two controlled experiments concerning the comparison of pair programming to peer review*, Journal of Systems and Software **78** (2005), no. 2, 166–179, ISSN: 0164-1212.

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

[22] J.T. Nosek, *The case for collaborative programming*, Commun. ACM **41** (1998), no. 3, 105–108, ISSN: 0001-0782.

[23] J.E. Rumbaugh, I. Jacobson, and G. Booch, *The unified modeling language reference manual*, Addison-Wesley-Longman, 1999, ISBN: 978-0-201-30998-0.

[24] M. Shaw, *Prospects for an engineering discipline of software*, IEEE Softw. **7** (1990), no. 6, 15–24, ISSN: 0740-7459.

[25] M. Shaw, *Continuing prospects for an engineering discipline of software*, IEEE Software **26** (2009), no. 6, 64–67, ISSN: 0740-7459.

[26] F. Shull and R.L. Feldmann, *Building theories from multiple evidence sources*, Guide to Advanced Empirical Software Engineering (2008), 337–364.

[27] 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 (2008), 312–336.

[28] G.W. Snedecor, *Calculation and interpretation of analysis of variance and covariance*, Collegiate Press, Ames, Iowa, 1934.

[29] W.F. Tichy, *Should computer scientists experiment more?*, Computer **31** (1998), no. 5, 32–40, ISSN: 0018-9162.

[30] G.M. Weinberg and G.L. Gressett, *An experiment in automatic verification of programs*, Commun. ACM **6** (1963), 610–613, ISSN: 0001-0782.

[31] L. Williams, R.R. Kessler, W. Cunningham, and R. Jeffries, *Strengthening the case for pair programming*, Software, IEEE **17** (2000), no. 4, 19–25, ISSN: 0740-7459.

[32] L. Williams, Ch. McDowell, N. Nagappan, J. Fernald, and L. Werner, *Building pair programming knowledge through a family of experiments*, Proceedings of the 2003 International Symposium on Empirical Software Engineering (Washington, DC, USA), ISESE '03, IEEE Computer Society, 2003, ISBN: 0-7695-2002-2, pp. 143–.

[33] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, and B. Regnell, *Experimentation in software engineering*, Springer, 2012, ISBN: 978-3-642-29043-5.

[34] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering: An introduction*, Kluwer Academic Publishers, Boston, 1999.

[35] M.V. Zelkowitz, D.R. Wallace, and D.W. Binkley, *Experimental validation of new software technology*, pp. 229–263, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2003, ISBN: 981-02-4914-4.