Experimento sobre calidad externa con Test-Driven Development en la industria de telecomunicaciones

Carolina Palomeque*, *Fernando Uyaguari

Programa de Maestría en Gestión Estratégica de Tecnologías de la Información, Universidad de Cuenca, Facultad de Ingeniería, Cuenca, Ecuador.

Autores para correspondencia: caro_palomeque@hotmail.com, fuyaguari01@gmail.com

Fecha de recepción: 30 de julio de 2017 - Fecha de aceptación: 15 de agosto de 2017

RESUMEN

Test-Driven Development (TDD) es una técnica de desarrollo de software cuyo estudio se ha incrementado en los últimos quince años, promete efectos positivos en el proceso de desarrollo, sin embargo, existen pocos estudios en el ámbito industrial. Con el propósito de comparar la técnica TDD con ITL (Iterative Test Last) con la técnica propia de desarrollo de una empresa de telecomunicaciones ETAPA EP[1], se realizó un experimento controlado para determinar si la técnica de programación TDD incrementa la calidad del software. El experimento fue realizado siguiendo las actividades que se indican en el proceso experimental en ingeniería del software. En lo relativo a la calidad, ITL presenta una diferencia estadísticamente significativa frente a TDD, el tratamiento ITL incrementa la calidad del código fuente del sujeto. No existe diferencia significativa entre TDD y la técnica propia de desarrollo de ETAPA EP. Desde el punto de vista práctico, ITL produce mejores resultados de calidad que la técnica propia de ETAPA EP, la técnica propia incrementa la calidad en comparación con TDD. Los resultados obtenidos en el experimento han permitido realizar recomendaciones prácticas respecto al proceso de desarrollo utilizado en ETAPA EP.

Palabras clave: Calidad, CC, computación, experimento, ITL, software, TDD.

ABSTRACT

Test-Driven Development (TDD) is a software development technique and studies in TDD increased in the last fifteen years. TDD promises positive effects in the development process, however there are few studies in the industrial field. To verify is the TDD programming technique increases the quality of software a controlled experiment was conducted comparing the software development technique TDD (Test-Driven Development) with ITL (Iterative Test Last) and the technique of the ETAPA EP[1] Telecommunications Company. The experiment was implemented according to the software engineering experimental process. The analysis clearly shows that quality-wise ITL is statistically significant superior to TDD. ITL increases the quality of the source code of the subject. From the practical point of view, ITL produces better code quality than the ETAPA EP development technique, however in comparison with the TDD approach increases the ETAPA EP technique the quality, but not to the extent as the ITL technique. The experiment allowed to define practical recommendations regarding the development process used in ETAPA EP.

Keywords: Quality, CC, computation**,** experiment, ITL, software, TDD.

[1] Empresa Pública Municipal de Telecomunicaciones, Agua Potable y Saneamiento de Cuenca

1. INTRODUCCIÓN

Las empresas que trabajan con ingeniería de software requieren y generan grandes cantidades de conocimiento y la gestión de ese conocimiento permite producir software de una forma rápida y económica, así como tomar mejores decisiones (Genero Bocco, Lemus, & Velthuis, 2015). Para construir software, los equipos de desarrollo se apoyan en ideas no fundamentadas o afirmaciones hechas por expertos o comunidades de desarrollo de software, en la confianza hacia alguna técnica o proceso de desarrollo que ha funcionado de manera empírica; muchas veces se pretende usar una herramienta de desarrollo que está en auge o que su documentación promete dar un resultado exitoso e invertir menos tiempo. No siempre estas ideas de partida son certeras. Justamente por eso, los ingenieros de software necesitan evidencia de que una técnica o herramienta funciona en un entorno o infraestructura determinada, es aquí en donde nace la necesidad de realizar experimentación (Juristo & Moreno, 2001).

Incremental Test Last (ITL) es una técnica de programación que inicia con la escritura de código fuente para resolver alguna funcionalidad y luego se crean casos de prueba para comprobar dicha funcionalidad (Erdogmus, Morisio, & Torchiano, 2005). TDD consiste en una técnica de programación que se basa en la escritura de casos de prueba sobre un requerimiento dado, antes de escribir el código fuente que lo resuelva (Beck, 2003). Ha sido utilizada esporádicamente durante décadas, pero recientemente ha emergido promoviendo buenos resultados, si bien existe poca evidencia empírica en el contexto industrial que apoya o niega su utilidad. (Nagappan, Maximilien, Bhat, & Williams, 2008). En la Tabla 1 se presentan los experimentos que han sido llevados a cabo por Canfora, Cimitile, Garcia, Piattini, & Visaggio (2006), George & Williams (2004) y Munir, Wnuk, Petersen, & Moayyed (2014) sobre el uso de la técnica. Los resultados en términos de calidad difieren según el caso, por lo que no son concluyentes para emitir un criterio definitivo sobre su validez.

Tabla 1. Comparativo entre varios experimentos de TDD en la industria.

	Experimento 1	Experimento 2	Experimento 3
Autores	Canfora, Cimitile, Garcia, Piattini, & Visaggio (2006)	George & Williams (2004)	Munir, Wnuk, Petersen, Moayyed (2014)
Estudio	Evaluating advantages of test driven development: a controlled experiment with professionals	A structured experiment of test-driven development.	An experimental evaluation of test driven development vs. test-last development with industry professionals.
Control	TAC (Testing After Coding)	Cascada. Usaron programación en pares para los 2 casos.	Test Last Development
Año	2006	2004	2014
Lenguaje	Java (Eclipse y JUnit)	Java	Java (Eclipse y JUnit)
Número de sujetos	28 sujetos	24 sujetos	31 sujetos
Lugar/País	Ciudad Real, España	EE.UU.	Londres, Inglaterra
Resultados sobre calidad	No se evidencia que TDD sea mejor en calidad, pero se concluye que la calidad del producto final mejoraría con la utilización más prolongada de esta técnica.	El autor afirma que TDD mejora la calidad del software.	No se encuentran diferencias significativas cuando se aplica o no la técnica del TDD.

Inclusive existen más estudios que demuestran que los resultados que se obtienen en los múltiples experimentos controlados con respecto a la calidad del software son diversos y no convergen en un único criterio (Kollanus, 2010; Turhan, Layman, Diep, Erdogmus, & Shull, 2010).

ETAPA EP es una organización con 1,380 funcionarios que ofrece servicios de telecomunicaciones, agua potable y gestión ambiental a la ciudad de Cuenca, Ecuador. Dentro de su estructura orgánica se encuentra la Subgerencia de Tecnologías de la Información (STI), la que cuenta a su vez con un Departamento de desarrollo informático conformado por 13 funcionarios que se encargan de desarrollar y mantener los sistemas informáticos de la empresa que se ocupan de asuntos comerciales, de gestión, ambientales, de georreferenciación, entre otros. No obstante, la STI también ha adoptado técnicas de programación con base en supuestos. Actualmente emplean una metodología ágil basada en SCRUM, con el objetivo de agilitar el desarrollo de software a través de entregas incrementales y de esta manera generar valor al negocio de manera oportuna.

Debido al constante crecimiento de la empresa ETAPA EP y a las necesidades de automatización de procesos, la demanda de requerimientos de software se ha incrementado notoriamente en los dos últimos años. El resumen del comportamiento se presenta en la Figura 1. Se puede apreciar que el número de requerimientos abiertos (solicitados por los usuarios) frente al número de requerimientos cerrados (atendidos y entregados a los usuarios) no tienen un comportamiento homogéneo ya que existe dependencia de las necesidades del negocio, pero sí se puede afirmar que siempre se mantienen requerimientos rezagados que no se logran atender con la modalidad de trabajo actual.

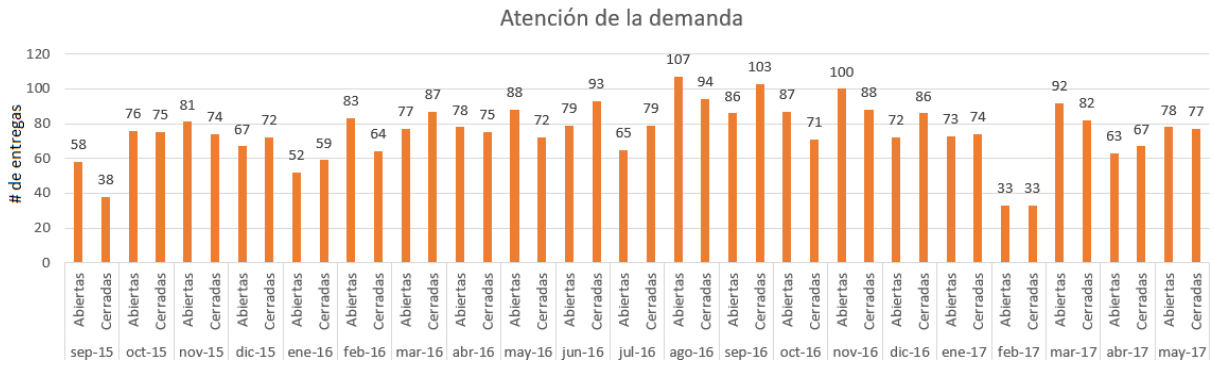

Figura 1. Atención de la demanda de requerimientos de software en la empresa ETAPA EP durante el periodo septiembre 2015-mayo 2017.

A este problema se suma el hecho de que los recursos humanos que antes cubrían la demanda de una manera controlada actualmente ya no puedan hacerlo puesto que las políticas y disposiciones de la empresa, por el momento, no permiten incrementar el recurso humano del área. Los programadores, por su parte, intentan minimizar el tiempo para cada desarrollo, pero esto causa que, en repetidas ocasiones, el producto entregado contenga errores y requiera reprogramación de funcionalidades, lo que implica costos y tiempos para la empresa y afecta la confiabilidad del producto entregado y de los desarrolladores que participan en estas tareas. Para responder de mejor manera a la demanda que afronta la STI, los funcionarios buscan mecanismos o metodologías de desarrollo que puedan reducir los tiempos y costos, a la vez que cubran la demanda sin alterar los limitados recursos.

Tabla 2. Hipótesis planteadas para la aplicación de la técnica TDD en la empresa ETAPA EP

Hipótesis	Valor
Hipótesis nula	HCO1 = TDD no representa una diferencia significativa en calidad del código respecto a ITL HCO2 = TDD no representa una diferencia significativa en calidad del código respecto a la técnica propia
Hipótesis alternativa	HC11 = TDD incrementa la calidad del código frente a ITL HC12 = TDD disminuye la calidad del código frente a ITL HC21 = TDD incrementa la calidad del código frente a la técnica propia HC22 = TDD disminuye la calidad del código frente a la técnica TDD en la empresa ETAPA EP

Es en este punto donde se exhibe la contribución de esta investigación. Entregará resultados del experimento mediante la aplicación de la técnica TDD, de tal modo que la evidencia entregue información para definir si se adopta o rechaza la nueva técnica de programación. Las hipótesis planteadas para la investigación se citan en la Tabla 2.

2. MARCO TEÓRICO

En esta sección se brinda información sobre las diferentes técnicas de desarrollo ágiles que se utilizaron en el experimento de ETAPA EP, así como también conceptos y métricas para determinar la calidad del software, es decir los valores numéricos empleados para la variable respuesta del experimento.

2.1. Metodologías ágiles

Al comparar las metodologías ágiles con las metodologías tradicionales se aprecia que estas últimas se centran especialmente en el control del proceso, pues establecen rigurosamente las actividades involucradas, mientras que las metodologías ágiles se centran en otros aspectos como el factor humano, la colaboración con el cliente, entre otros aspectos (Letelier & Penadés, 2006). Además, los métodos ágiles incursionan en el desarrollo de software porque planifican a corto plazo, lo cual es beneficioso ya que entrega pequeñas funcionalidades a los clientes y aprende de la retroalimentación en cada entrega. Ciertamente, la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan; para muchos clientes esta flexibilidad constituye una ventaja competitiva, además de reducir su coste (Figueroa, Solís, & Cabrera, 2008).

ITL (Incremental Test Last)

Otra de las metodologías ágiles existentes y que ha sido estudiada en esta investigación es ITL, conocida también como TLD (Test Last Development). El proceso que sigue ITL se presenta en la Figura 2. Inicia con la escritura de código fuente para generar alguna funcionalidad solicitada, luego se crea un caso de prueba, se lo ejecuta y si la prueba es fallida se debe depurar el código y volver a ejecutar la prueba hasta que sea satisfactoria (Erdogmus *et al*., 2005).

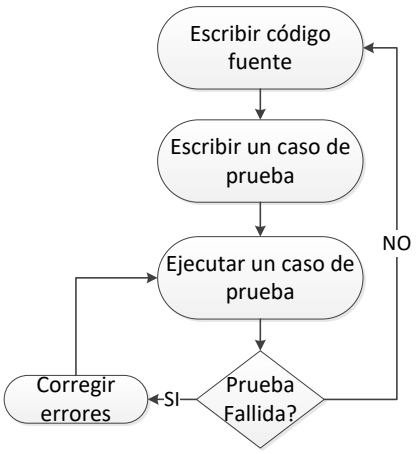

Figura 2. Proceso de aplicación de la técnica ITL (Hussan *et al*., 2014).

TDD (Test-Driven Development)

Kent Beck, ingeniero de software estadounidense inmerso en la creación de metodologías ágiles, ha redescubierto la técnica TDD, Desarrollo Guiado por Pruebas, en el año 2002. Beck afirma que TDD es un modo predictivo de desarrollo, se enfoca en el desarrollo por incrementos y propone la definición de casos de pruebas antes de la escritura del código fuente (Beck, 2003). Su proceso se presenta en la Figura 3.

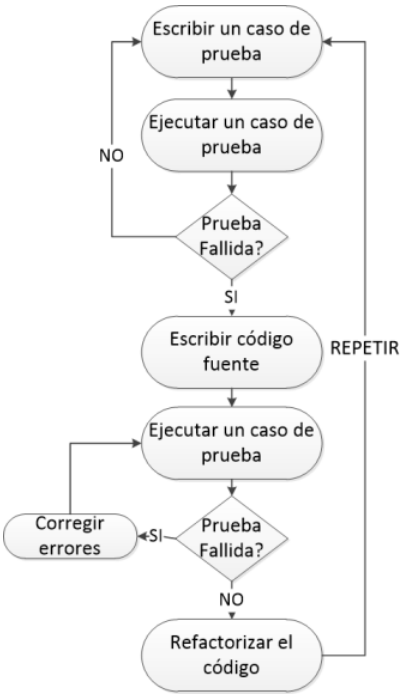

Figura 3. Proceso de aplicación de la técnica TDD (Hussan *et al*., 2014).

La técnica TDD se basa en escribir casos de prueba sobre un requerimiento dado antes de escribir el código fuente que lo resuelva, de modo que las pruebas exitosas confirmen la funcionalidad del requisito. Posterior a ello se realiza la refactorización para optimizar el código fuente escrito sin perder su correcta funcionalidad. (Beck, 2003). El uso de TDD faculta analizar de manera previa los diversos escenarios que se desea solventar, así como también evita la escritura de código innecesario. La mayoría de las personas quienes aprenden TDD encuentran que su programación cambia para bien (Beck, 2003).

Técnica propia de desarrollo de software en ETAPA EP

A la técnica que utilizan los desarrolladores de la STI se la ha denominado 'Your Way' o su acrónimo 'YW' que en español significa 'A su modo'. En ETAPA EP se utiliza la metodología SCRUM, por lo que predomina el trabajo en equipo y la planificación de las tareas por *Sprints*[2]. La herramienta de programación utilizada es GeneXus, por tanto, se podría utilizar el componente GxUnit para escribir pruebas unitarias (Araújo Pérez, 2008). La técnica de desarrollo consiste en no escribir casos de prueba para comprobar las funcionalidades desarrolladas, sino en ejecutar el programa y realizar pruebas manuales con los usuarios en ambientes de desarrollo para demostrar el funcionamiento de lo solicitado.

2.2. Calidad del Software

La calidad se define según el grado en que un producto de software satisfaga las necesidades declaradas e implícitas cuando se usa en condiciones específicas (ISO/IEC 25010, 2011). Para el caso, se ha tomado a la calidad como la variable respuesta que se desea determinar a través de este experimento y se han ejecutado variaciones en la metodología de desarrollo de software. En el presente trabajo se establecieron *pruebas unitarias con casos de prueba pasados* como mecanismo para medir la calidad del software generado por los sujetos experimentales.

Para obtener un software con calidad es necesario que sus funcionalidades sean completas y correctas, por tanto, del modelo de calidad ISO/IEC 25010 (2011), se han tomado 2 sub-características:

[2] Iteraciones, con duración de una semana.

a) *Functional completeness* (completitud funcional) - Para su evaluación se tomó el número de requerimientos que los sujetos experimentales pudieron cumplir.
b) *Functional correctness* (funcionalidad correcta) - Se revisó el grado de precisión de lo desarrollado y la cantidad de pruebas satisfactorias que obtenga cada sujeto.

Por lo expuesto, y tomando a los casos de prueba como mecanismo de control, la fórmula aplicada para la métrica de calidad será:

$$QLTY = \frac{\#\text{ de casos de prueba acertados}^{3}}{\#\text{ de casos de prueba ejecutados}^{4}}$$

2.3. Ingeniería de software experimental

La experimentación se refiere a la confrontación que se realiza entre los hechos y las suposiciones, especulaciones o creencias que abundan en la construcción de software (Juristo & Moreno, 2001). Gran cantidad de variables participan en el desarrollo de software: mientras más factores y variables se manipulen el tema se vuelve más complejo (Juristo & Moreno, 2001). El experimento consiste justamente en la manipulación de una variable independiente para medir su efecto sobre otra variable dependiente (Genero Bocco *et al.*, 2015).

El proceso experimental en ingeniería de software incluye cuatro grandes actividades: definición de objetivos, diseño del experimento, ejecución y análisis de los resultados. Cada una de estas actividades entrega un producto de salida que alimenta a la siguiente actividad (Juristo & Moreno, 2001). Actividades y entregas se presentan en la Figura 4.

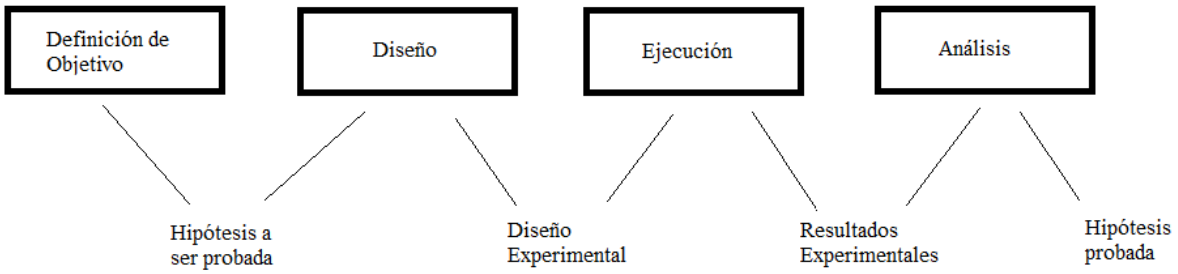

Figura 4. Proceso de experimentación en SE (Juristo & Moreno, 2001).

Todas estas etapas pueden ser desarrolladas de manera iterativa. Wohlin *et al.* (2012) consideran importante agregar una etapa más al proceso de experimentación, la presentación y difusión de los resultados. La comunicación es clave para que otros investigadores que puedan replicar el experimento o basarse en el conocimiento empírico adquirido.

3. EXPERIMENTACIÓN

3.1. Descripción del experimento

El experimento fue realizado durante el mes de abril de 2016 en las instalaciones de ETAPA EP en la ciudad de Cuenca, Ecuador. Los sujetos experimentales fueron todos los desarrolladores de la STI, por lo que no fue necesario realizar selección de sujetos; los resultados que se obtuvieron se consideran significativos para aprobar o rechazar la hipótesis. Para realizar el diseño y ejecución del experimento se contó con el apoyo del investigador, Oscar Dieste de GrISE-UPM[5]. Los factores, tratamientos y demás características que determinaron el experimento se exhiben en la Tabla 3.

[3] Se contabilizará el número de casos de prueba que cada sujeto experimental pase.
[4] Hace referencia al total de número de casos de prueba de la suite.
[5] Grupo de Investigación en Ingeniería del Software Empírica de la Universidad Politécnica de Madrid.

Tabla 3. Características del experimento realizado en la empresa ETAPA EP, abril 2016.

Característica	Valor
Unidad Experimental	Proceso para desarrollo de software
Sujeto Experimental	Desarrolladores de ETAPA EP
Tamaño muestral	13 sujetos
Variable respuesta (variable dependiente)	Calidad del código fuente
Factor (variable independiente)	Técnica de desarrollo
Tratamientos	YW, ITL, TDD
Diseño	1 factor (técnica de desarrollo) 3 niveles (TDD, ITL, YW)
Instrumentos experimentales	Especificaciones de software (tareas experimentales) Suite de casos de prueba
Metodología	Proceso experimental en ingeniería de software
Lugar	Cuenca, Ecuador
Lenguaje de programación	GeneXus Evolution 1
Componente para pruebas unitarias	GxUnit

Con el propósito de que los sujetos dispongan de una comprensión básica de pruebas unitarias y conceptos de TDD e ITL, se brindó entrenamiento de manera alternada con la realización de tareas experimentales. El entrenamiento, además de conferencias, incluyó ejercicios prácticos grupales e individuales para un mejor aprendizaje. La agenda se especifica en la Tabla 4.

Tabla 4. Agenda del experimento realizado en la Empresa ETAPA EP, abril 2016.

Horario	Jueves	Viernes	Sábado
08h00-10h00	Metodologías ágiles Entrenamiento en Slicing	Pruebas unitarias Entrenamiento en GxUnit e ITL	Entrenamiento en TDD
10h00-10h15	Break	Break	Break
10h15-12h00	Tarea experimental usando YW	Tarea experimental usando ITL	Tarea experimental usando TDD
12h00-12h20	*Recolección de código fuente*	*Recolección de código fuente*	*Recolección de código fuente*

Cada sujeto experimental contó con una computadora técnica con acceso a internet y una máquina virtual (VMWare 10.0.) en la que se instalaron previamente las herramientas necesarias para el taller:

a) Sistema operativo Windows 7
b) Framework .Net 2.0
c) Navegador Web
d) CASE GeneXus Evolution 1 en ambiente.Net (configurado para generar código en lenguaje C#)
e) GxUnit (componente de GeneXus para escribir casos de prueba)

Adicionalmente, se entregó a los sujetos una guía de quince páginas con ejemplos sobre la implementación de casos de prueba con GxUnit.

Las tareas experimentales cumplían funcionalidades cortas que no requerían almacenamiento en bases de datos y cumplía fines netamente académicos:

a) Bowling Score Keeper (BSK). Juego de bolos, calcula puntajes y determina el ganador.
b) Mars Rover (MR). Robot que se desplaza con instrucciones de teclado.
c) Spreadsheet (SS). hoja electrónica para ejecutar operaciones matemáticas.

Se aplicó un diseño de medidas repetidas con un factor y tres niveles. Como variable de bloque se utilizó la tarea experimental que fue asignada a los sujetos de manera aleatoria. Los tratamientos fueron aplicados a los sujetos experimentales durante tres sesiones realizadas en tres días. El primer día aplicaron el tratamiento YW; el segundo día, el ITL y el tercer día, el TDD. La Tabla 5 evidencia la tarea desarrollada por cada sujeto experimental (S_x) con la técnica utilizada.

Tabla 5. Distribución de sujetos experimentales por tratamiento y tarea empleados en la empresa ETAPA EP, abril de 2016.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}
YW	BSK	MR	MR	SS	BSK	MR	SS	SS	BSK	BSK	SS	BSK	MR
ITL	SS	BSK	BSK	MR	SS	BSK	MR	BSK	SS	SS	MR	SS	BSk
TDD	MR	SS	SS	BSK	MR	SS	BSK	MR	MR	MR	BSK	MR	SS

Durante el experimento, se presentaron algunas novedades que se anotan a continuación:

- El primer día se presentaron inconvenientes que los sujetos experimentales no mencionaron hasta el momento de realizar las tareas experimentales (falta de conexión a la red, claves de acceso etc.) lo cual, de cierta manera, acortó el tiempo que disponían para la ejecución de la tarea experimental. En los dos siguientes días no existió ningún inconveniente que atrase la ejecución de la tarea.
- Uno de los sujetos experimentales poseía muy poca experiencia en la utilización de la herramienta de programación GeneXus, por lo que su aporte fue bastante limitado, sin embargo, participó los tres días del taller intentando realizar los ejercicios.
- A pesar de todo el entrenamiento brindado, el segundo día uno de los sujetos utilizó Java en lugar de GeneXus para desarrollar el software que se le asignó, por tanto, dicho código no pudo ser evaluado en este experimento.

Al finalizar cada tarea, se recolectó el código fuente generado por los sujetos para revisión. Los instrumentos utilizados para la extracción de la métrica de calidad (QLTY) consisten en tres suites de casos de prueba, por lo que se escribió una suite de 89 casos para MR, 39 casos para BSK y 42 casos para SS.

En la mayoría de los casos fue necesario que el investigador escriba el código para lograr una conexión exitosa entre los casos de prueba genéricos y las diferentes formas de programar de cada desarrollador (estas diferencias en las formas de programar corresponden al uso de dominios para simular clases, varios procesos para desarrollar una tarea, constantes en lugar de variables, entre otros.). Finalmente, con una conexión correcta entre los casos de prueba que se ejecutaban y la programación realizada por cada sujeto, se ejecutó la suite de casos establecida para comprobar las funcionalidades de cada tarea experimental. Cada prueba ejecutada podía tener uno de los siguientes estados: a) Prueba pasada satisfactoriamente, b) Prueba incorrecta, y c) Error al ejecutar la prueba.

3.2. Validez del experimento

Para asegurar la validez del experimento se tomaron algunas medidas respecto a la validez interna, externa, de constructo y de conclusión:

- Se diseñó el experimento de tal forma que los tratamientos no se confundan con la tarea experimental. La variable de bloque fue la tarea experimental.
- Se trabajó con sujetos experimentales representativos de la industria, en este caso, la industria de las telecomunicaciones, con experiencia en desarrollo de software.
- En lo relativo a la participación de los sujetos en el experimento, para evitar la deserción, se motivó la participación con entrenamiento gratuito en TDD y testing.
- Las tareas experimentales no eran conocidas previamente por los sujetos para evitar su influencia en el resultado experimental.
- Para la operacionalización del constructo teórico de la variable respuesta calidad se tomó en cuenta el estándar correspondiente a la calidad del software definido por ISO 2510.
- La medición se basó en el uso de suites de casos de prueba que fueron utilizados en otros experimentos industriales reportados en la literatura científica.
- Debido a que Ecuador carece de la experiencia en experimentación en software para el diseño y ejecución del experimento, se contó con el apoyo de investigadores con trayectoria en esta área experimental.

4. RESULTADOS Y DISCUSIÓN

Para transformar los resultados de cada sujeto en términos de calidad (QLTY) se obtuvo la división entre el número de casos de prueba pasados satisfactoriamente y el número total de casos de prueba definidos. Para representar este valor en formato de porcentaje, el resultado se multiplica por 100. Ejemplo: la fórmula de calidad que se aplicó para un sujeto *x* que tuvo 14 casos de prueba pasados, de un total de 42 de la tarea SS es:

$$QLTY_X = \left(\frac{14}{42}\right) * 100$$

Una vez realizadas las evaluaciones detalladas anteriormente para cada sujeto y en cada tarea experimental, se obtuvieron los resultados descritos en la Tabla 6, en donde se incluyen los valores estadísticos descriptivos.

Tabla 6. Valores estadísticos descriptivos del proceso de experimentación en la empresa ETAPA EP, abril de 2016.

	Media	Error estándar	Mediana	Varianza	Desviación estándar	Mínimo	Máximo	Rango	Rango intercuartil
ITL	22.83	5.93	17.12	423.05	20.56	2.25	74.36	72.11	26.31
TDD	10.68	4.44	7.87	217.55	14.74	0	40.45	40.45	10.26
YW	14.50	6.52	6.32	510.21	22.58	0	78.65	78.65	18.46

El valor medio para ITL fue de 22.83 puntos porcentuales, 10.68 para TDD y 14.50 para YW. Estos valores fueron obtenidos con la herramienta de software estadística SPSS[6] para realizar el análisis descriptivo y la prueba de hipótesis.

En la Figura 5 se presenta un diagrama de caja y bigotes generado sobre los resultados de calidad de código de los sujetos experimentales. En los tres tratamientos utilizados se puede observar la mediana, primer y tercer cuartil y los valores atípicos de cada tratamiento. En la Figura 6 se puede apreciar de manera detallada cómo varían los valores de calidad de cada sujeto según el tratamiento que utilizaron.

Al analizar el comportamiento de la información estadística y con base en la Figura 5 y 6, se concluye:

i. La media con valor superior es la correspondiente a ITL, seguido de YW luego de TDD. Existen algunos valores atípicos que han sido incluidos en el análisis. En el caso de los tratamientos ITL y YW se presenta un rango y varianza mayor comparado con TDD en donde se encuentran más compactos los valores. Al aplicar los tratamientos de YW e ITL se han alcanzado valores máximos más altos, aproximadamente 78, mientras que con TDD alcanza a 40.

ii. Según el diagrama de variación de calidad de código por sujeto, se observa un comportamiento en donde los sujetos puntúan mejor con ITL y disminuye con los otros tratamientos.

Se ha utilizado el Modelo lineal general (GLM) para la prueba de hipótesis ya que tenemos un diseño de medidas repetidas de un factor con tres niveles. El requerimiento para aplicar esta prueba es que se cumpla con el requerimiento de esfericidad, para lo cual hemos obtenido el W de Mauchly correspondiente a 0.847, un valor cercano a 1 por lo que se puede aplicar el modelo GLM. Según la prueba de Lambda de Wilks, existe un valor de significación estadística de 0.035, menor a 0.05, lo cual indica que el factor técnico de desarrollo posee un efecto significativo en la variable respuesta calidad.

[6] Software estadístico, versión 22, https://www.ibm.com/analytics/ec/es/technology/spss/

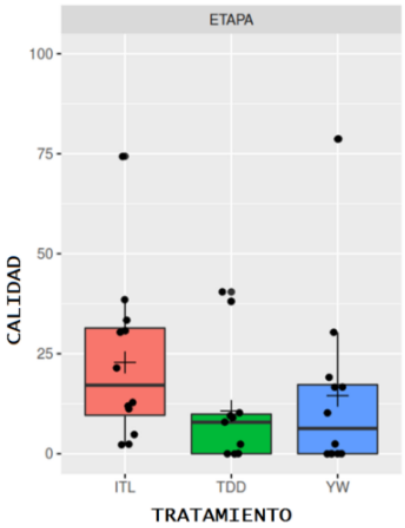

Figura 5. Diagrama de caja y bigotes por tratamientos.

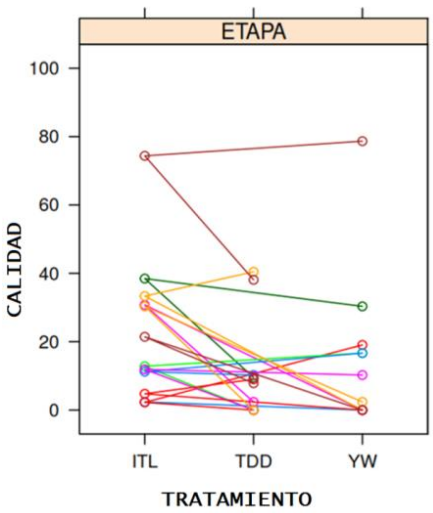

Figura 6. Detalle de la variación de calidad por cada sujeto, experimento ETAPA EP, abril de 2016.

Para la comparación entre los tratamientos se ha utilizado Bonferroni, que compara las diferencias de medias. A continuación, se anotan los resultados:

i. Estadísticamente existe diferencia significativa entre los efectos de los tratamientos ITL y TDD. El valor de significación estadística obtenido 0.035 es menor a 0.05. Por lo tanto, se rechaza la hipótesis nula H01 y se acepta la hipótesis alternativa HC12. TDD disminuye la calidad de código frente a ITL.

ii. No existe diferencia significativa entre los tratamientos TDD y YW. El valor de significancia es 1.000, es mayor que 0.05, por lo tanto, no se puede rechazar la hipótesis nula HC02. Tampoco existe diferencia significativa entre los tratamientos ITL y YW, el valor de significación estadística es 0.338, mayor a 0.05.

iii. Hay diferencia práctica entre YW y TDD. El tratamiento YW incrementa la calidad respecto a TDD. También existe diferencia práctica a favor de ITL comparado con YW.

El tratamiento ITL incrementa la calidad respecto a TDD, posiblemente porque ITL sigue un procedimiento similar al método tradicional propio de la empresa estudiada, mientras que TDD fue una técnica totalmente nueva para los sujetos experimentales.

Aplicar la técnica TDD involucra un cambio de paradigma de desarrollo de software, ya que requiere la escritura de pruebas unitarias antes de la escritura del código fuente, lo cual resultó bastante complicado para los desarrolladores en el corto tiempo de entrenamiento, no obstante, no se puede descartar la posibilidad de que, incrementando la experiencia en la escritura de casos de pruebas, se pueda hacer uso de esta técnica en el futuro. Esto se puede corroborar con la opinión de Dieste, Fonseca, Raura, & Rodríguez (2015), ya que en otros experimentos en los que el uso de TDD duraba más de una sesión, se evidenció que los resultados mejoraron para TDD, de tal manera que podría existir la posibilidad de que con mayor entrenamiento los sujetos experimentales que participaron en esta investigación también obtengan mejores resultados en el desarrollo de sus productos.

Es necesario indicar que al utilizar el case GeneXus para implementar las tareas experimentales asignadas, los desarrolladores tuvieron que hacer ciertos artificios para simular clases, hecho que pudo influenciar en los bajos resultados porcentuales obtenidos con todas las técnicas (TDD, ITL y YW). El valor de calidad de código del tratamiento YW es, en la práctica, superior a TDD, lo que se debe posiblemente a la familiaridad que tienen los sujetos con su técnica propia de desarrollo.

Los valores superiores de calidad que ofrece el tratamiento ITL constituyen una oportunidad para mejorar el proceso de desarrollo de ETAPA EP, ya que al incluir casos de prueba en su proceso de desarrollo se podría obtener código de mejor calidad. Asimismo, se observa que la varianza es amplia

en los sujetos que aplican ITL. Se sugiere que para mejorar el proceso en ETAPA EP, exista capacitación en *testing* de modo que proporcione homogeneidad entre los resultados de los sujetos experimentales por tratamiento, en especial cuando se aplica ITL. Posiblemente en un futuro, cuando los sujetos estén más entrenados, se pueda realizar un nuevo experimento en TDD y reforzar los resultados obtenidos en este experimento.

Comparando los resultados obtenidos en el experimento en ETAPA EP con los resultados de otros experimentos industriales reportados en la literatura científica, no hay coincidencia sino diferencias, posiblemente debido a otros factores instrumentales o del ambiente de desarrollo que se requerirá investigar con profundidad en un futuro.

5. CONCLUSIONES

El experimento presenta diferencias estadísticamente significativas entre los tratamientos ITL y TDD, el tratamiento ITL incrementa la calidad de código respecto a TDD. No se presentan diferencias estadísticamente significativas respecto a la comparativa entre los tratamientos YW y TDD, YW e ITL. Sin embargo, existe una diferencia práctica: YW incrementa la calidad en comparación con TDD, e ITL incrementa la calidad en comparación con YW. El tratamiento ITL ofrece mejores resultados de calidad que los tratamientos YW y TDD. No obstante, es el tratamiento en donde la varianza entre los valores de calidad que dan los sujetos es grande.

AGRADECIMIENTOS

Un agradecimiento al Dr. Oscar Dieste, investigador de la Universidad Politécnica de Madrid y a los ingenieros Juan Guamán, Bolívar Piedra y Pablo Cazorla, funcionarios de la Subgerencia de Tecnologías de la Información de ETAPA EP debido al apoyo brindado para la ejecución de esta investigación. De igual modo, agradecemos a los integrantes del equipo de desarrollo de software, quienes participación de manera activa y comprometida en el experimento descrito.

REFERENCIAS

Araújo Pérez, A. (2008). *Especificación de un marco de pruebas asociado a Genexus. con adaptación de funcionalidades de FIT*. Tesis de Maestría en Ingeniería en Computación. Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, 267 p. Disponible en https://www.colibri.udelar.edu.uy/jspui/bitstream/123456789/2949/1/tesis-araujo.pdf

Beck, K. (2003). *Test-driven development: By example* (1ª ed.). Boston, Massachusetts: Addison-Wesley Professional.

Canfora, G., Cimitile, A., García, F., Piattini, M., Visaggio, C. A. (2006). *Evaluating advantages of test driven development: a controlled experiment with professionals*. Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering, Rio de Janeiro, Brazil, pp. 364-371.

Dieste, O., Fonseca, E. R., Raura, G., Rodríguez, P. (2015). Efectividad del test-driven development: un experimento replicado. *Revista Latinoamericana de Ingeniería de Software*, *3*(3), 141-147.

Erdogmus, H., Morisio, M., Torchiano, M. (2005). On the effectiveness of the test-first approach to programming. *IEEE Transactions on software Engineering*, *31*(3), 226-237.

Figueroa, R. G., Solís, C. J., Cabrera, A. A. (2008). *Metodologías tradicionales vs. Metodologías ágiles*. Working paper, Escuela de Ciencias en Computación, Universidad Técnica Particular de Loja, Loja, Ecuador, pp. 1-10.

Genero Bocco, M., Lemus, J. A. C., Velthuis, M. G. P. (2015). *Métodos de investigación en ingeniería del software*. Madrid, España: Ra-Ma S.a. Editorial Y Publicaciones.

George, B., Williams, L. (2004). A structured experiment of test-driven development. *Information and Software Technology*, *46*(5), 337-342.

ISO/IEC 25010. (2011). *Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models*. Disponible en https://www.iso.org/obp/ui/#iso:std:iso-iec:25010:ed-1:v1:en

Juristo, N., Moreno, A. M. (2001). *Basics of software engineering experimentation*. Dordrecht, The Netherlands: Springer Science & Business Media.

Kollanus, S. (2010). *Test-driven development-still a promising approach?* IEEE Seventh International Conference on the Quality of Information and Communications Technology (QUATIC), pp. 403-408.

Letelier, P., Penadés, M. C. (2006). *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. Disponible en http://www.cyta.com.ar/ta0502/v5n2a1.htm

Munir, H., Wnuk, K., Petersen, K., Moayyed, M. (2014). *An experimental evaluation of Test Driven Development vs. Test-Last Development with industry professionals*. EASE '14 Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, Article No. 50.

Nagappan, N., Maximilien, E. M., Bhat, T., Williams, L. (2008). Realizing quality improvement through test driven development: results and experiences of four industrial teams. *Empirical Software Engineering*, *13*(3), 289-302.

Turhan, B., Layman, L., Diep, M., Erdogmus, H., Shull, F. (2010). *How effective is test-Driven Development?* Chapter 12, pp. 207-219. Sebastopol, CA: Publisher: O'Reilly.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A. (2012). *Experimentation in software engineering*. Dordrecht, The Netherlands: Springer Science & Business Media.