UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERÍA

MAESTRIA EN GESTIÓN ESTRATÉGICA DE TECNOLOGÍAS DE INFORMACIÓN Y COMUNICACIÓN

PROYECTO DE TITULACIÓN

ANÁLISIS Y EVALUACIÓN DEL DESARROLLO DE SOFTWARE TRADICIONAL VS EL DESARROLLO DE SOFTWARE GUIADO POR PRUEBAS EN TÉRMINOS DE CALIDAD DE SOFTWARE Y PRODUCTIVIDAD DE LOS DESARROLLADORES PARA UNA EMPRESA DEL SECTOR PÚBLICO

PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN GESTIÓN ESTRATÉGICA DE TECNOLOGIAS DE INFORMACIÓN Y COMUNICACIÓN.

AUTOR: Ing. Carolina Elizabeth Palomeque Rodríguez
CI: 0502767155

DIRECTOR: Ing. Manuel Fernando Uyaguari Uyaguari
CI: 0103109799

CUENCA- ECUADOR
2017

Resumen

La experimentación es una disciplina que nos permite crear modelos con los cuales se pueden relacionar procesos con objetos, siguen un ciclo y dependiendo de los resultados que se obtienen se genera el conocimiento (Wohlin, Runeson, Höst, Ohlsson, Regnell y Wesslén, 2012).

TDD es una técnica de desarrollo de software que se ha utilizado esporádicamente durante décadas, pero ha emergido recientemente promoviendo buenos resultados, sin embargo, existe poca evidencia empírica en el contexto industrial que apoya o niega la utilidad de esta práctica. (Nagappan, Maximilien, Bhat y Williams, 2008)

Actualmente no contamos con un cuerpo de conocimiento de TDD, por lo que se revela la importancia de realizar replicaciones. Las réplicas experimentales nos ayudan a confirmar resultados de experimentos realizados y así ampliar las bases de conocimiento con las que cuentan los investigadores. (Mendonça, et al., 2008)

De acuerdo a la literatura científica, los experimentos realizados en el ámbito de la industria que comparan TDD con otras técnicas de programación, dan resultados contradictorios y no permiten asegurar que un método sea mejor que otro.

Durante el mes de marzo del 2016, como parte del proyecto FidiPro[1], ESEIL[2] y GrISE[3] realizaron un experimento en una Empresa de Finlandia (https://sites.google.com/site/fidiproeseil) en donde las hipótesis planteadas fueron similares a las que se plantean para éste experimento, por confidencialidad a esta Empresa se la nombrará como “Empresa F” a lo largo del documento.

Este proyecto abarca el estudio de TDD y la realización de una réplica experimental en una empresa pública de la ciudad de Cuenca.

Palabras claves: TDD, experimento, replicación, industria

[1] Finland Distinguished Professor
[2] Empirical Software Engineering Industrial Laboratory
[3] Grupo de Investigación de Ingeniería del Software Empírica de la Universidad Politécnica de Madrid

Abstract

Experimentation is a discipline that allows us to create models from which it is possible to relate processes with objects; these follow a cycle and, depending on the obtained results, knowledge is created (Wohlin, Runeson, Höst, Ohlsson, Regnell & Wesslén, 2012).

TDD is a software development technique that has been sporadically used for decades but has recently emerged promoting good results; nevertheless there is little empirical evidence in the industrial context that neither supports nor denies the utility of this practice (Nagappan, Maximilien, Bhat & Williams, 2008).

Nowadays there is not a knowledge body for TDD, thus the importance of performing replications. Experimental replications help us to confirm results from performed experiments and, in this manner, broaden the knowledge base available to researchers. (Mendonça, et al., 2008)

According with the scientific literature, experiments performed in the industrial context that compare TDD with other software development techniques provide contradicting results and do not determine whether a method outperforms another.

In March 2016, as part of the FidiPro project, ESEIL and GrlSE performed an experiment in a company in Finland where their hypothesis were similar to those of this experiment; due to confidentiality this company is referred as “Empresa F” throughout this document.

This project includes the study of TDD and the performance of an experimental replication in a public company in Cuenca.

Key Words: TDD, experiment, replication, industry

Índice General

Índice de Figuras

Índice de Tablas

Universidad de Cuenca
Cláusula de licencia y autorización para publicación en el Repositorio Institucional

CAROLINA ELIZABETH PALOMEQUE RODRIGUEZ en calidad de autora y titular de los derechos morales y patrimoniales del trabajo de titulación "ANÁLISIS Y EVALUACIÓN DEL DESARROLLO DE SOFTWARE TRADICIONAL VS EL DESARROLLO DE SOFTWARE GUIADO POR PRUEBAS EN TÉRMINOS DE CALIDAD DE SOFTWARE Y PRODUCTIVIDAD DE LOS DESARROLLADORES PARA UNA EMPRESA DEL SECTOR PÚBLICO", de conformidad con el Art. 114 del CÓDIGO ORGÁNICO DE LA ECONOMÍA SOCIAL DE LOS CONOCIMIENTOS, CREATIVIDAD E INNOVACIÓN reconozco a favor de la Universidad de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos.

Asimismo, autorizo a la Universidad de Cuenca para que realice la publicación de este trabajo de titulación en el repositorio institucional, de conformidad a lo dispuesto en el Art. 144 de la Ley Orgánica de Educación Superior.

Cuenca, 17 de octubre del 2017

Cláusula de Propiedad Intelectual

CAROLINA ELIZABETH PALOMEQUE RODRIGUEZ, autora del trabajo de titulación "ANÁLISIS Y EVALUACIÓN DEL DESARROLLO DE SOFTWARE TRADICIONAL VS EL DESARROLLO DE SOFTWARE GUIADO POR PRUEBAS EN TÉRMINOS DE CALIDAD DE SOFTWARE Y PRODUCTIVIDAD DE LOS DESARROLLADORES PARA UNA EMPRESA DEL SECTOR PÚBLICO", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autora.

Cuenca, 17 de octubre del 2017

Carolina Elizabeth Palomeque Rodríguez

C.I: 0502767155

AGRADECIMIENTOS

Un agradecimiento 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 este trabajo, así como también a los integrantes del equipo de desarrollo de software, quienes participación de manera activa y comprometida en el experimento descrito.

Agradezco de manera especial a mi esposo, Ing. Paúl Zhañay, que me brindó su ayuda personal y profesional en la ejecución de este proyecto, y a mis padres quienes me han enseñado a ser perseverante y siempre me han apoyado a cumplir todas las metas propuestas a pesar de los obstáculos que se presentan. ¡Gracias por confiar en mí!

Carolina Palomeque Rodríguez

DEDICATORIA

Dedico este trabajo a mi hija Paulina, quién con su corta edad tuvo la paciencia y comprensión que me permitió continuar con mis estudios, así como también a Adriana, quién desde mi vientre me dio la fortaleza y motivación para cumplir con esta meta.

Carolina Palomeque Rodríguez

CAPITULO 1

1.1. Introducción.

Con los avances tecnológicos que se generan dentro del área de Informática, la disciplina de la ingeniería de software ha incursionado también con nuevos métodos, procesos, técnicas y demás mecanismos relacionados al desarrollo y mantenimiento del software que ofrecen solventar las dificultades que se suelen presentar muy a menudo en las empresas dedicadas al desarrollo de software alrededor del mundo.

Existen metodologías tradicionales que controlan el proceso mediante actividades, artefactos y herramientas que se deben llevar a cabo de manera rigurosa; por lo contrario, existen hoy en día las metodologías ágiles que se centran en otras dimensiones, como el factor humano, incrementando el valor del personal de análisis y desarrollo así como también al cliente, entre otros. (Letelier, P., y Penadés, 2006). Existen estudios empíricos desde que apareció el manifiesto ágil, que demuestran el gran interés que existe a nivel de la industria en conocer cada vez más los métodos ágiles y adoptarlos a sus compañías. (Rodriguez, Markkula, Oivo y Turula, 2012)

Test Driven Development (TDD), un método ágil, que hace referencia al desarrollo de software guiado por pruebas (Beck, 2003). Es objeto central de estudio para la presente investigación, debido a que en la última década está siendo evaluada y adoptada al rededor del mundo tanto en el campo educativo como en la industria (Kollanus, 2010).

En el año 2016, en una Empresa Finlandesa, que será nombrada como Empresa F por temas de confidencialidad, se llevó a cabo un experimento con objetivos e hipótesis similares al que se plantea realizar en esta investigación para la Empresa Pública Municipal de Telecomunicaciones, Agua Potable y Saneamiento de Cuenca (ETAPA EP). El presente trabajo se lo describe como una réplica del experimento original, con el

propósito de generar nuevos datos, verificar los resultados y así brindar confiabilidad a los resultados. A continuación, en la Tabla 1, se presentan las principales características de los 2 experimentos, los detalles del mismo se presentarán más adelante en el desarrollo de este trabajo de investigación.

	EMPRESA F (experimento original)	**EMPRESA ETAPA EP (réplica)**
Unidad Experimental	Proceso para desarrollo de software	Proceso para desarrollo de software
Sujeto Experimental	Desarrolladores de la Empresa F	Desarrolladores de ETAPA EP
Tamaño muestral	13 sujetos pero solo 8 participaron durante todos las sesiones	13 sujetos, 12 entregaron su código fuente durante todas las sesiones
Variables Respuesta (variable dependiente)	Calidad del producto Productividad de los desarrolladores	Calidad del producto Productividad de los desarrolladores
Factor (variable independiente)	Técnica de desarrollo	Técnica de desarrollo
Tratamientos	Método propio (YW) Incremental Test-last (ITL) Test Drive Development (TDD)	Método propio (YW) Incremental Test-last (ITL) Test Drive Development (TDD)
Diseño	1 factor (técnica de desarrollo) 3 niveles (TDD, ITL, YW)	1 factor (técnica de desarrollo) 3 niveles (TDD, ITL, YW)
Instrumentos experimentales	Especificaciones de Software Suits de Casos de Prueba	Especificaciones de Software Suits de Casos de Prueba
Metodología	Proceso Experimental en Ingeniería del Software	Proceso Experimental en Ingeniería del Software
País	Finlandia	Ecuador
Lenguaje	Java	GeneXus
IDE	Eclipse	Evolution 1

Tabla 1: Características de los experimentos Empresa F y ETAPA EP

En la actualidad en ingeniería de software abundan los métodos y técnicas de desarrollo que prometen ayudar en los diferentes ciclos de vida del software; por la experiencia obtenida en algunas empresas de Cuenca y por la literatura estudiada (Juristo y Moreno, 2013), los desarrolladores seleccionan las técnicas en base a ideas,

creencias, sin contrastarlas con la realidad.

En ETAPA EP, empresa a ser estudiada, se utiliza la metodología SCRUM para desarrollar software. No se escriben casos de prueba para comprobar las funcionalidades desarrolladas, sino se ejecuta el programa y se realizan pruebas manuales con los usuarios en ambiente de desarrollo para demostrar el funcionamiento de lo solicitado.

La demanda de los requerimientos de software en ETAPA EP es alta y además se están presentando con frecuencia casos de reprogramación de funcionalidades entregadas a los usuarios debido básicamente a que no se detectaron falencias en fases tempranas del desarrollo, lo cual implica costos y tiempos, afectando además la confiabilidad del producto entregado y de los desarrolladores que participan en estas tareas. Por estos motivos ETAPA EP tiene interés en realizar este experimento en donde se tendrá a los desarrolladores de la Subgerencia de Tecnologías de la Información (STI) como sujetos experimentales para la investigación. Para el presente trabajo nos vamos a concentrar en los temas: ágil, TDD y experimentación.

ETAPA EP contará con los resultados que se obtengan del estudio de TDD y tendrán la posibilidad de tomar decisiones en base a ello.

Se realizará un experimento controlado para determinar el efecto de TDD en la calidad y la productividad, como recomendación y trabajo futuro se podrá sugerir cambios al modelo de trabajo de ETAPA EP adoptando la técnica que prometa incrementar la productividad y la calidad según este estudio.

Se van a desarrollar capacitaciones que fomenten la participación de los sujetos ya que TDD no se conocía en la Empresa hasta antes del experimento.

En términos generales el alcance del presente proyecto de tesis contempla la realización de una réplica del experimento de la Empresa F en ETAPA EP, se usará el lenguaje de programación de GeneXus en ambiente web para comparar TDD con ITL y

con el método propio (Your Way, YW) de ETAPA EP desde los aspectos Productividad y Calidad.

1.2.Objetivo General.

- Analizar las aproximaciones de desarrollo de software: TDD, ITL y el método propio de ETAPA EP con el propósito de compararlas con respecto a la calidad del software y productividad de los desarrolladores, desde el punto de vista de los investigadores.

1.3.Objetivos Específicos.

- Estudiar la problemática de ETAPA EP en cuanto al desarrollo de software.
- Estudiar la técnica de desarrollo de software Test Driven Development (TDD).
- Ejecutar un experimento controlado con los desarrolladores de ETAPA EP comparando TDD con ITL y el método propio (YW).
- Elaborar un informe para ETAPA EP con los resultados obtenidos, conclusiones y recomendaciones de mejoras en el proceso de desarrollo de software en ETAPA EP, así como lineamientos básicos para una posible adopción en caso que los resultados favorezcan a TDD.

1.4. Problemática de la Investigación.

En el medio existen varios estudios relacionados con estas técnicas de la ingeniería de software, específicamente de TDD existen algunos experimentos realizados en los que la mayoría tiene como objetivo descubrir si la calidad del software mejora o disminuye con la utilización de esta técnica, así como también el tiempo de

programación lo cual hace referencia a la productividad que demuestran los desarrolladores de software.

Existen varios estudios de TDD, sin embargo, solo 8 de ellos son experimentos en el campo de la industria, estos reportan resultados contradictorios que no permiten asegurar que un método sea más eficiente que otro en cuanto a la calidad y a la productividad, a continuación, se presentan una comparación de los resultados de algunos experimentos en la industria:

	Experimento 1	**Experimento 2**	**Experimento 3**	**Experimento 4**
AUTORES	G. Canfora, A. Cimitile, F. Garcia, M. Piattini, C.A. Visaggio	B. George, L.A. Williams	L. Madeyski, L. Szala	Hussan Munir, Krzysztof Wnuk, Kai Petersen, and Misagh Moayyed
ESTUDIO	Evaluating advantages of test driven development: a controlled experiment with professionals	A structured experiment of test-driven development.	The impact of test-driven development on software development productivity.	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.	TLD (Test Last) y se trabaja en el contexto XP (eXtreme Programming)	Test Last Development
AÑO	2006	2004	2007	2014
AMBITO DE INDUSTRIA	Empresa de Tecnología, manejo de software	Empresas de software y telecomunicaciones	Institución académica	No se indica

EMPRESA	Soluziona Software Factory	3 compañías: John Deep, RoleModel Software y Ericsson	No se indica	No se indica
LENGUAJE	Java (Eclipse y JUnit)	Java	Java (Eclipse) y Aspect J	Java (Eclipse y JUnit)
NÚMERO DE SUJETOS	28 sujetos	24 sujetos	No se indica	31 sujetos
LUGAR / PAIS	Ciudad Real, España	EEUU	No se indica	Londres, Inglaterra
RESULTADOS SOBRE CALIDAD	Con el experimento no se evidencia que TDD es mejor en calidad, sin embargo se concluye que mejoraría la calidad del producto final con la utilización más prolongada de esta técnica.	El autor afirma que TDD mejora la calidad del software.	No estudiado a profundidad.	No se encuentran diferencias significativas
RESULTADOS SOBRE PRODUCTIVIDAD	La implementación de TDD requiere más tiempo que TAC, sin embargo, esto no implica necesariamente que TDD pueda afectar a la productividad.	En cuanto a productividad se concluye que el tiempo que se invierte es mayor al usar TDD.	Se indica que TDD tiene un impacto positivo en cuando a productividad.	No se encuentran diferencias significativas

Tabla 2: Comparativo entre varios experimentos de TDD en la industria

Como se puede observar en la Tabla 2, que resume 4 experimentos significativos de

la industria sobre el uso de TDD, los resultados difieren para cada caso, por lo que no son concluyentes para emitir un único criterio. Inclusive, existen más estudios en el campo investigativo que demuestran que los resultados que se obtienen en los múltiples experimentos controlados con respecto a la calidad y productividad son diversos y no convergen en un único criterio (Kollanus, 2010) y (Turhan, Layman, Diep, Erdogmus y Shull, 2010).

En lo que respecta a la calidad, si bien el experimento 3 no ha realizado un estudio a profundidad y el experimento 4 no ha encontrado diferencias significativas, los experimentos 1 y 2 coinciden en que TDD mejoraría la calidad del producto final, incluso siendo el método de comparación TAC y Cascada respectivamente. El experimento 1 con una utilización más prolongada de TDD y el experimento 2 con una inclusión directa de TDD en el proceso de desarrollo de Software. Por lo tanto, se puede concluir que en términos de calidad TDD es optimista para el mejoramiento del producto final en el desarrollo de Software.

En lo referente a la productividad, los experimentos 1 y 2 coinciden en que el tiempo en la implementación de TDD es mayor, sin embargo, no son concluyentes en que la productividad sea necesariamente positiva. Por otro lado, solamente el experimento 3 es claro en indicar que existe un impacto positivo en cuanto a la productividad con el uso de TDD. Se puede concluir que no existe en ninguno de los criterios un aspecto negativo sobre el uso de TDD en la productividad, por tanto, no es precisamente desfavorable su utilización.

Finalmente, Juristo, Vegas, Solari, Abrahão, Ramos (2012), consideran que el uso de réplicas ayuda a confirmar las hipótesis, así como también a comprender mejor el fenómeno en estudio mediante la verificación de resultados o la identificación de variables contextuales que podrían influir o no en los resultados experimentales.

1.5. Problemática de la empresa a estudiar.

1.5.1. Preámbulo.

ETAPA EP es una Empresa Pública Municipal de la ciudad de Cuenca, constituida en el año 1968, ambiental y socialmente responsable que mejora la calidad de vida de las personas y contribuye al desarrollo de las organizaciones. Los servicios que brinda son:

- Telecomunicaciones:
    - Internet (Adsl, fibra óptica, evdo)
    - Telefonía fija
    - Telefonía fija inalámbrica (CDMA)
    - Televisión Satelital (DTH)
- Agua Potable:
    - Operación y mantenimiento de las plantas de Agua Potable
    - Instalaciones de domiciliarias de Agua Potable
    - Planes Maestros
- Saneamiento:
    - Servicio de alcantarillado
    - Plantas de tratamiento de Aguas residuales
    - Limpiezas de sistemas de alcantarillado interiores domésticos, comerciales, industriales – dentro o fuera del Cantón
    - Limpiezas de fosas sépticas domésticas, comerciales o industriales - dentro o fuera del Cantón.
- Gestión Ambiental:
    - Monitoreo y vigilancia de Recursos Hídricos y Clima

- Radar de medición de lluvias
- Conservación y manejo de Áreas protegidas municipales
- Desarrollo Sustentable
- Manejo de Cuencas Hidrográficas
- Programas de educación y capacitación ambiental
- Gestión de desechos y calidad ambiental

- Servicios Corporativos:
  - Data Center
  - Telefonía digital – Troncales SIP
  - Email empresarial
  - Enlaces de datos
  - Internet Corporativo

Desde hace 25 años, ETAPA EP cuenta con un departamento para el manejo de las TICS, actualmente nombrada como "Subgerencia de Tecnologías de Información" (STI). El trabajo de STI aporta principalmente a los objetivos de la perspectiva de Procesos Internos del mapa estratégico corporativo de ETAPA EP (Anexo 1), brindando soluciones tecnológicas para una constante mejora.

Para aportar a los procesos internos de la Estrategia Empresarial (Anexo 3), la Subgerencia de TI cuenta como una de sus principales funciones el desarrollo y mantenimiento de los sistemas informáticos de toda la Empresa.

El área de desarrollo está conformada por 9 funcionarios de planta, incluido el Administrador del Área y 4 funcionarios contratados, todos ingenieros de sistemas y la mitad de ellos cuentan con título de cuarto nivel en áreas de gestión empresarial más que en temas operativos, sin embargo, todos poseen una experiencia en el desarrollo de

software mínimo de 2 años, existiendo inclusive funcionarios certificados por la empresa Artech en la herramienta GeneXus.

1.5.2. Proceso de desarrollo de Software.

Las aplicaciones informáticas con las que inició la STI fueron escritas en RPG (Report Program Generator), que es un lenguaje de programación de IBM. Luego se desarrolló con herramientas de Microsoft .Net y actualmente los programas son implementados con el CASE GeneXus que genera código para .Net o para RPG según se requiera.

En cuanto a la metodología para el desarrollo de aplicaciones informáticas, en un principio, se lo llevaba a cabo a libre criterio de cada programador, paulatinamente fue surgiendo la necesidad de estandarizar la forma de realizar estas tareas y se crearon pautas y procesos que los desarrolladores debían seguir para llegar al objetivo final de un desarrollo que es la entrega del producto al usuario final.

A lo largo del tiempo, el mecanismo de programación ha ido evolucionando, dependiendo siempre de las opiniones y creencias del personal involucrado, experiencia de otras empresas similares o consejos de profesionales que viajaban al extranjero y transmitían visiones innovadoras. No se realizaba un análisis puntual para evaluar si un método era útil y ajustable a la realidad de la Empresa, simplemente se lo acogía.

Posteriormente se elaboró una metodología de desarrollo de software para programación estructurada que fue utilizada mayormente para las aplicaciones desarrolladas en .Net, y posteriormente en las primeras aplicaciones desarrolladas en GeneXus, con el objetivo de mantener de una manera estandarizada la gestión tanto de los proyectos de software como del desarrollo de las aplicaciones. Esta metodología se utilizó hasta antes del uso de la metodología SCRUM.

En la actualidad la metodología de desarrollo de software tradicional fue reemplazada por una nueva 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.

La calidad de los productos de software desarrollados en ETAPA EP se los evalúa a través de revisión y pruebas de requerimientos funcionales y no funcionales. Se utilizan plantillas para documentar lo acontecido, con listas de selección que sirven para certificar el cumplimiento total o parcial de los requerimientos solicitados, así como también el correcto accionar del software ante las funcionalidades requeridas.

Cabe indicar que la metodología actual, SCRUM, ha sido de gran utilidad para el departamento de desarrollo, debido a que los requerimientos de software se manejan de manera más ordenada y se cuenta con métricas actualizadas que permiten tener una visión general de los avances, la carga de trabajo del equipo y cumplimiento de las expectativas de los usuarios.

El equipo de software está dividido en 2 grupos, cada uno con su Supervisor (S*crum Master)*; el proceso que se sigue para el desarrollo es el siguiente:

1. Los usuarios de diferentes áreas de la Empresa envían sus requerimientos a la STI, sean estos cambios a aplicaciones o nuevas funcionalidades.
2. Se priorizan dichas solicitudes en un Lista de requerimientos priorizada (*Product Backlog)*, se analizan y de ser el caso se dividen en historias de usuario para ser atendidos en cada Iteración del desarrollo (*Sprint*).
3. Se realiza estimaciones basadas en la herramienta *Planning Poker* con el equipo de software y se asigna un desarrollador para implementar cada requerimiento.
4. Cada desarrollador implementa sus asignaciones utilizando la siguiente técnica:
    a. Análisis minucioso del requerimiento.

b. Codificación en la herramienta GeneXus / Pruebas manuales de lo implementado (este ciclo se realiza las veces que sea necesario hasta obtener el resultado esperado).
c. Pruebas técnicas con personal del área de Operaciones de Software.
d. Presentación y aceptación de los usuarios finales.
e. Implantación del producto en un ambiente de producción con las seguridades necesarias para su uso.

5. En las reuniones diarias (*daily meeting*) se reportan los avances y dificultades de las tareas asignadas para recibir apoyo del equipo y así cumplir con lo asignado en cada *Sprint*.
6. En la planificación semanal (*Sprint Planning Meeting*) se verifica el cumplimiento total o parcial de las actividades y se repite el proceso para el siguiente S*print* con nuevos requerimientos del *Product backlog*.

1.5.3. Problemática actual.

Debido al crecimiento de ETAPA EP y de la necesidad de automatización de sus procesos a través de soluciones informáticas, la demanda de requerimientos de software ha incrementado en los últimos años por lo cual los recursos que antes cubrían la demanda de una manera controlada actualmente esa demanda ya no es posible atenderla de manera satisfactoria. Esta situación ocasiona que existan permanentemente requerimientos rezagados como producto de la no atención oportuna de los requerimientos de software solicitados por el usuario. Se debe indicar que los requerimientos nos son atendidos como una fila sino en base a una priorización que solicita el negocio.

Desde que se inició con el proceso mencionado en la sección 1.5.2, por cada

requerimiento se crean “entregas” que no son más que repositorios en una base de datos de requerimientos para que los analistas documenten sus tareas realizadas por cada *Sprint*, controlen sus tiempos y adjunten actas que confirman la aceptación del usuario ante el producto entregado, por lo general se cuenta con una entrega por cada requerimiento, sin embargo, para requerimientos más complejos el tiempo estimado es mayor al total del *Sprint* (34 horas), por lo que se generan 2 o más entregas para cubrir 1 requerimiento, bajo este concepto en la Tabla 3 se presenta el total a mayo 2017 de las entregas abiertas (todo lo que la empresa han solicitado a la STI) vs las entregas cerradas (todo lo finalizado y entregado)

Total Entregas Abiertas	**Total Entregas Cerradas**
1595	1563

Tabla 3: Total entregas con corte a mayo del 2017

Un número exacto de requerimientos que se reciben mensualmente no se puede calcular, ya que es variante dependiendo de las necesidades del negocio y las solicitudes de las áreas de la Empresa, cada uno tiene distinta complejidad por tanto el tiempo estimado para su atención también varía.

Del mismo modo las entregas se procesan de manera variable, procurando cubrir el número de solicitudes entrantes. Muchas veces aparte de la priorización del negocio es necesario variar la atención entre los requerimientos de corto y largo alcance o entre los que implican una mayor o menor complejidad, de modo que se pueda equilibrar la carga para cubrir el número de solicitudes entrantes del período, así como también minimizar los requerimientos que se encuentran rezagados. El comportamiento resumen por cada año se presenta en la Figura 1 y la Tabla 4:

Figura 1: Comportamiento detallado mes a mes

	2015 (desde sept)		2016		2017 (hasta mayo)	
	Entregas Abiertas	**Entregas Cerradas**	**Entregas Abiertas**	**Entregas Cerradas**	**Entregas Abiertas**	**Entregas Cerradas**
Totales	282	259	974	971	339	333

Tabla 4: Variación entregas abiertas/cerradas

Como se indicó, los requerimientos de software que ingresan en un mes, no son necesariamente los que se atienden en ese mismo mes, por lo que el número de abiertos vs número de cerrados no tienen un comportamiento homogéneo, pero si se puede concluir que siempre se mantienen requerimientos rezagados que no se logran atender con la modalidad de trabajo actual.

Un punto importante de mencionar también es que en las estimaciones para cada requerimiento, se trata de minimizar el tiempo al máximo para responder a la demanda, los programadores pueden obviar pruebas a su código implementado y por tanto el producto se coloca en producción con errores, lo que da como resultados que el usuario final lo detecte y envíe un nuevo requerimiento para trabajar en ello, sumándole inclusive alguna funcionalidad adicional que la encuentran necesaria una vez que ya operan el software implementado.

Cabe indicar que por políticas y disposiciones de la Empresa no es posible incrementar de momento el personal para laborar en el área, por lo que lo deseable sería

encontrar un mecanismo que beneficie al tiempo de respuesta, para solventar los requerimientos y de esta manera responder de mejor manera a la demanda a la que afronta la STI.

ETAPA EP se encuentra interesada en la realización de un experimento en donde se evalúe el uso de la técnica TDD, ITL y YW desde el aspecto de la calidad y la productividad. Esto permitirá contar con un indicador de calidad del método que utilizamos actualmente y compararlo con la calidad utilizando otras técnicas distintas como TDD e ITL. Además, podremos comparar con los datos obtenidos en otra empresa distinta a ETAPA EP.

A partir de estos datos podremos realizar sugerencias sobre mejoras al método actual utilizado en la Empresa con el propósito de mejorar la calidad de nuestros productos que ofrecemos al cliente interno y mejorar la productividad de nuestros desarrolladores.

El conocimiento obtenido de esta experiencia podrá ser transmitido a las otras empresas municipales ya que son empresas que tienen problemáticas similares y usan metodologías de desarrollo similares a las utilizadas en ETAPA EP

El objetivo seguirá siendo mejorar algún aspecto de la metodología de desarrollo, adoptando técnicas que prometan beneficiar a la STI con los resultados que el cambio genere; pero en esta ocasión se lo realizará con indicadores y métricas que permitirá una toma de decisiones precisa en la empresa en base a hechos y no en base a supuestos como se lo ha venido realizando desde que inició la programación en la Empresa.

Cada experimento que se lleva a cabo impacta de alguna manera en el conocimiento de la disciplina de ingeniería de software y ayudará a predecir la validez de una técnica (Basili, Selby y Hutchens, 1986).

Muchas empresas atraviesan por los inconvenientes mencionados, sin embargo, ETAPA EP se beneficiará de los resultados obtenidos, llegando a ser una ventaja significativa, ya que podrá aplicar el método que le garantice mejores resultados en base a evidencia, incrementando así la calidad del trabajo, obteniendo el mejor provecho del personal que labora y pudiendo satisfacer de mejor manera las necesidades que tiene la Empresa.

Finalmente, como aporte a la problemática investigativa y la problemática empresarial, se ejecutará una réplica del experimento original de la Empresa F, de modo que se obtengan resultados en términos de calidad y productividad sobre el uso de TDD en ETAPA EP, con lo cual se aportará de 2 maneras: crecer el cuerpo de conocimiento que actualmente existe sobre esta rama y tomar decisiones en la empresa investigada a cerca de la adopción o no de esta nueva técnica.

CAPITULO 2

2.1. Marco Teórico.

2.1.1 Ingeniería de software experimental.

Las empresas que trabajan con ingeniería de software requieren y generan grandes cantidades de conocimiento, la gestión del conocimiento permite producir mejor software de una forma más rápida y económica, así como tomar mejores decisiones (Genero, Lemus, Velthuis, 2015) y (Lindvall y Rus, 2003)

Para construir software los equipos de desarrollo se apoyan en ideas no fundamentadas o afirmaciones hechas por colegas o comunidades de desarrollo de software, en la confianza hacia alguna técnica o proceso de desarrollo que ha funcionado en determinado lugar, mucha veces se pretende usar una herramienta de desarrollo que está en auge o que su documentación nos promete dar un resultado exitoso e invertir menos tiempo, sin embargo no siempre estas ideas son certeras; frente a este comportamiento, los ingenieros de software necesitamos evidencia de que una técnica o herramienta funcione, es aquí en donde nace la necesidad de realizar experimentación (Juristo y Moreno, 2013).

La norma IEEE 610.12 define a la Ingeniería de Software, al igual que otras disciplinas de ingeniería, como aplicar el conocimiento científico para el desarrollo, operación y mantenimiento de sistemas de software (Juristo y Moreno, 2013).

La experimentación se refiere a la confrontación que se realiza entre los hechos y las suposiciones, especulaciones y/o creencias que abundan en la construcción de software (Juristo y Moreno, 2013). Es una disciplina que nos permite crear modelos con los cuales se pueden relacionar procesos con objetos, siguen un ciclo y dependiendo de los resultados que se obtienen se genera el conocimiento. El ciclo que se utiliza en

ingeniería de software está compuesto por la construcción del modelo, experimentación y aprendizaje. La experimentación provee un método sistemático, disciplinado, cuantificable y controlado de evaluación basado en actividades y comportamiento humano (Wohlin et al, 2012).

Gran cantidad de variables participan en el desarrollo de software, y mientras más factores y variables se manipulen el tema se vuelve más complejo (Juristo y Moreno, 2013). El experimento es justamente la manipulación de una variable independiente midiendo el efecto que tiene sobre otra variable dependiente (Genero et al., 2015).

Los experimentos pueden ser orientados a personas o a la tecnología (Genero et al., 2015), para el presente estudio está orientado a la tecnología por lo que persigue probar una hipótesis para aceptarla o rechazarla, de modo que se pueda recomendar la utilidad o no del método que se investiga.

El proceso experimental en ingeniería de software incluye 4 grandes actividades: definición de objetivos, diseño del experimento, ejecución y el análisis de los resultados. Cada una de estas actividades tiene un producto de salida que alimenta a la siguiente actividad (Juristo y Moreno, 2013). Actividades y entregas se presentan en a Figura 2:

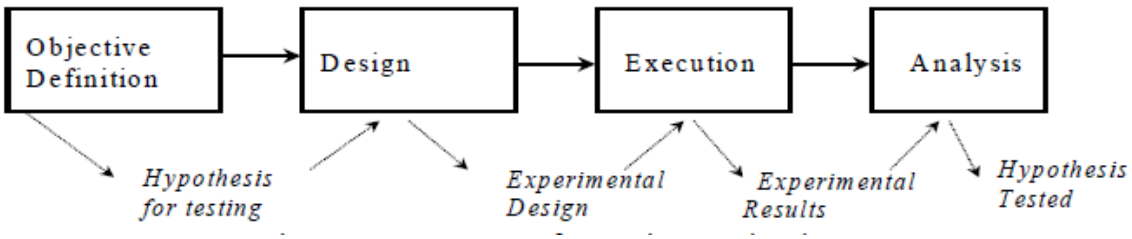


Figura 2: Proceso de experimentación en SE (Juristo y Moreno, 2013).

1. Definición de objetivo. - En esta fase se realiza definición del estudio, la

hipótesis se la describe en términos de variables que van a ser estudiadas

2. Diseño. - Es la planificación de las condiciones bajo las cuales se va a llevar a cabo el experimento, el esfuerzo se debe centrar en la obtención de la mayor cantidad de información sobre las variables que se involucran en el mismo.

3. Ejecución. - El experimento se llevará a cabo bajo el diseño establecido en la fase anterior.

4. Análisis. - Se estudia los datos recolectados en el experimento, de modo que se pueda determinar la relación o patrones entre las variables en estudio, buscando determinar si los datos obtenidos son o no estadísticamente significativos.

Todas estas etapas pueden ser desarrolladas de manera iterativa. Wohlin, Runeson, Höst, Ohlsson, Regnell y Wesslén, (2012) agrega una etapa más por considerarla de suma importancia que es la de Presentación y Difusión de los resultados, esta comunicación es clave para el uso de otros investigadores que desean replicar el experimento o basarse en el conocimiento empírico adquirido.

2.1.2 Replicaciones de experimentos.

Una replicación es la repetición de un experimento. El uso de réplicas a experimentos ayuda a confirmar la hipótesis, en ingeniería de software son numerosas las variables que participan en el experimento, por tanto, sus resultados también son variantes, las replicaciones exitosas o útiles ayuda a comprender mejor el fenómeno en estudio, mediante la verificación de resultados o la identificación de variables contextuales que podrían influir o no influir en los resultados experimentales (Juristo, Vegas, Solari, Abrahão y Ramos, 2012)

Llevar a cabo una réplica experimental se las cataloga como un esfuerzo difícil,

sobre todo las tareas de transferir los conocimientos y hallazgos experimentales no es un trabajo sencillo (Mendonça, et al., 2008). Sin embargo, replicar en Ingeniería de Software Empírico permite que la comunidad construir conocimiento acerca de qué resultados se obtiene dependiendo de las condiciones. Las replicaciones son válidas tanto si producen resultados similares como diferentes a su experimento original (Shull, Carver, Vegas y Juristo, 2008).

Existen varios tipos de replicaciones, las conceptuales evalúan la misma pregunta de investigación, pero utilizando un procedimiento experimental diferente, mientras que las exactas intenta seguir los procedimientos y mantener las condiciones lo más parecidas posible al experimento original (Shull, et al., 2008). Se dice que las exactas intentan mantener condiciones, ya que en ingeniería de software no es posible encontrar sujetos idénticos, unidades idénticas, etc. por lo que es importante resaltar las diferencias que existen entre el experimento original y la réplica (Juristo y Moreno, 2013).

Las replicaciones externas se dan cuando los experimentadores no son los mismos que en el experimento original, mientras que la réplica interna son realizadas por el mismo experimentador (Brooks, Daly, Miller, Roper y Wood, 1996).

Para el presente trabajo de investigación, según el tipo de procedimiento se clasifica como una replicación *exacta* y de acuerdo al investigador, se clasifica como *externa*.

2.1.3 Metodologías ágiles (ITL y TDD) y Your Way.

Desde la aparición del manifiesto ágil se han realizado numerosos estudios empíricos que demuestran el gran interés que existe a nivel de la industria en conocer cada vez más los métodos ágiles y adoptarlos a sus compañías; existen situaciones en las que el uso de metodologías tradicionales debe continuar, sin embargo, es adaptable a la mayoría de entornos, se puede indicar que en general es positivo el uso de

metodologías ágiles (Rodríguez, et al., 2012).

Otro estudio sobre el interés de metodologías ágiles en la industria, involucró la interacción con los representantes de empresas, realizando discusiones en grupos y reuniones cara a cara, esto aumentó la confianza entre los participantes, existiendo un interés mayor en aprender los puntos clave para su adopción y verificar el impacto que tiene sobre el proceso de desarrollo, lo cual se puede verificar mediante la experimentación (Misirli, Erdogmus, Juristo y Dieste, 2014).

Como se concluye entonces, este enfoque denominado métodos ágiles, están incursionando en el desarrollo de software. La planificación a corto plazo es beneficiosa para entregar pequeñas funcionalidades a los clientes y aprender del feedback en cada entrega, recalcando que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan, para muchos clientes esta flexibilidad será una ventaja competitiva además de reducir su coste (Figueroa, Solís, y Cabrera, 2008).

Una de las metodología ágiles es ITL (Incremental Test Last), conocida también como TLD (Test Last Development), en ITL se 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 es decir corregir y volver a ejecutar la prueba hasta que ésta sea satisfactoria (Erdogmus, Morisio, Torchiano, 2005). Se puede decir que en el caso de ITL las pruebas suelen estar un tanto condicionadas a lo implementado.

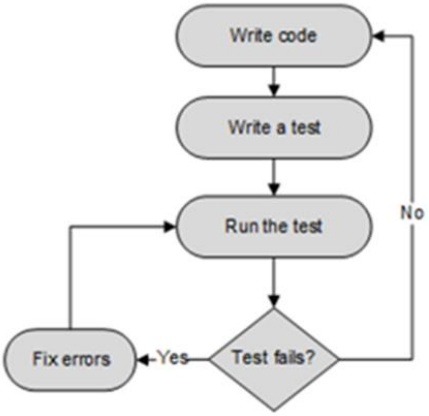


Figura 3: ITL (Hussan, et al., 2014)

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).

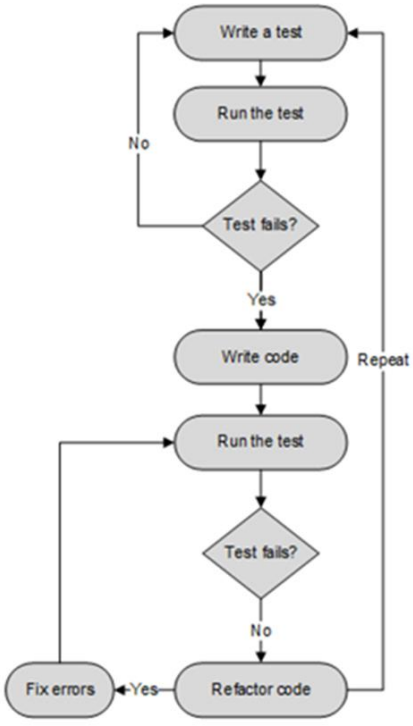


Figura 4: TDD (Hussan, et al., 2014)

La técnica Test Drive Development (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 (Beck, 2003).

La refactorización hace referencia a mejorar el código fuente existente, sin alterar su comportamiento externo, esto se logra con algunas recomendaciones como lo cita Araújo (2008): "(..) la eliminación de duplicación, la división en segmentos menores, la estructuración de manera que resulte más conveniente y la aplicación de las ventajas que brinde el lenguaje de programación."

Beck (2003) indica el concepto de TDD en 2 reglas simples:

1. No escriba una línea de código nuevo a menos que primero tenga una prueba automatizada fallida.

2. Elimine la duplicación.

Al usar TDD se puede analizar de manera previa los diversos escenarios que se desea solventar, así como también evitar la escritura de código innecesario. Se dice que la mayoría de personas quienes aprenden TDD encuentran que su práctica de programación cambia para bien (Beck, 2003).

En TDD existen 3 estados que caracterizan su proceso: Red/Green/Refactor. Siendo Red el estado de prueba fallida y Green el estado de una prueba ejecutada sin errores y Refactor la refactorización del código sin errores. (Figueroa, et al., 2008).

Erdogmus, Morisio, Torchiano (2005) presenta una comparación de las principales características entre ITL y TDD que se puede visualizar en la Tabla 5:

	Test-First	Test-Last
Who writes and runs tests?	Programmer	Programmer
When are tests written?	Before production code	After production code
When are tests run?	While writing production code, frequently	After writing production code, less frequently
Incremental development?	Yes	Yes
Regression testing?	Yes	Yes

Tabla 5: Comparación de ITL vs TDD

Como tercer método se describe a *Your Way*, que hace referencia al método propio que utilizan los desarrolladores de la STI que serán los sujetos del presente experimento. En ETAPA EP, se utiliza la metodología SCRUM, por lo que predomina el trabajo en equipo y la planificación de las tareas por *sprints*.

El detalle del proceso de desarrollo de software se lo presentó en el Capítulo 1, sección 1.5.2.

2.1.4 Calidad y Productividad.

La calidad es subjetiva, por lo que sus significados difieren dependiendo del contexto, la ISO 25010 define a la calidad como: El grado en que un producto software satisface necesidades declaradas e implícitas cuando se usa en condiciones específicas (ISO/IEC 25010, 2011).

Maximizar la calidad en el software es el planteamiento de muchas empresas, sin embargo, se deben conocer técnicas y métodos que permitan llegar a este objetivo. Las Normas ISO son procesos estándar los cuales sirven de guía para establecer controles y se dividen en característica y sub características para una mejor comprensión y guía para los usuarios (ISO/IEC 25010, 2011).

Figura 5: División Normas ISO

Una empresa con sus procesos, políticas y de acuerdo a sus líneas de negocio y necesidades puede tomar a las normas ISO como guía para garantizar la calidad y productividad esperada y de esta manera crecer en cuanto a calidad.

El modelo de calidad que se toma como referencia tiene 8 características principales, como se presenta en la Figura 6, la característica que se abordará para el presente trabajo es la Funcionalidad, que estudia el nivel de funciones que proporciona un producto, ya sean estas descritas textualmente o implícitas en el software.

Concretamente este trabajo se concentra en las sub-características completitud funcional (*Functional completeness*) y correcta funcionalidad (*functional correctness)*.

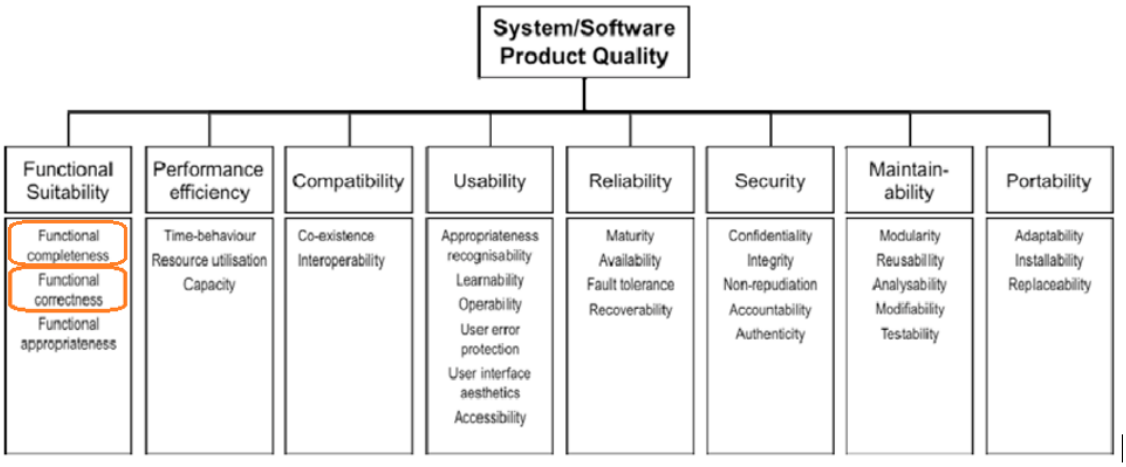


Figura 6: Características y subcaracterísticas (ISO/IEC 25010, 2011).

La Sub-característica *Functional* correctness estudia la correcta funcionalidad del software, su objetivo es establecer pautas de manera que las aplicaciones que se generan proporcionen los resultados esperados, así como también mecanismos de control para mantener la funcionalidad con el nivel de precisión requerido (ISO/IEC 25010, 2011).

La Sub-característica *Functional completeness* se encarga de controlar que las funcionalidades que se entrega al usuario se encuentren completas, es decir que cubra los diferentes aspectos que fueron especificados para un producto de software (ISO/IEC 25010, 2011).

En el presente trabajo se establecieron casos de pruebas unitarias como mecanismo de control para evaluar la calidad del software generado por los sujetos experimentales.

El objetivo para obtener un software con calidad es que sea **completo + correcto,** por tanto para evaluar *Functional completeness* debemos fijarnos en el número de requerimientos que los sujetos experimentales pudieron realizar y para evaluar *Functional* correctness revisaremos el grado de precisión de lo desarrollado, revisando la cantidad de pruebas satisfactorias que obtenga cada sujeto. Por lo expuesto la fórmula aplicada para la métrica de calidad será:

$$\frac{\#\text{ de casos de prueba acertados}}{\#\text{ de casos de prueba ejecutados}}$$

En la ejecución del experimento, el tiempo para desarrollar las tareas experimentales es el mismo en cada uno de los tratamientos, por tanto, sigue siendo la métrica válida la proporción entre el número de casos de prueba acertados y el total de pruebas ejecutadas, esto significa que la respuesta a la productividad estará embebida en los resultados obtenidos en la métrica de la calidad.

2.1.5 Experimento en la Empresa F, tomado para replicación.

El último experimento conocido que se ha realizado sobre TDD y que aún se encuentra en proceso de publicación, se ha ejecutado en la Empresa F de Finlandia (Tosun, 2016) como parte del proyecto FidiPro, ESEIL contó con el apoyo de GrISE-UPM, en el cual el objetivo planteado fue analizar los efectos de TDD sobre la calidad externa y la productividad de los desarrolladores en un entorno industrial.

Las hipótesis planteadas para dicho estudio fueron las siguientes:

En relación a la Calidad:

H_{Q0} = La calidad con ITLD es igual a la calidad con TDD (Hipótesis Nula)

H_{Q1} = La calidad con ITLD es diferente a la calidad con TDD (Hipótesis Alternativa)

$$H_{Q0}\colon\ \mu(QLTY)_{ITLD} = \mu(QLTY)_{TDD}\ \text{(Null Hypothesis)}$$
$$H_{Q1}\colon\ \mu(QLTY)_{ITLD} \neq \mu(QLTY)_{TDD}\ \text{(Alternative Hypothesis)}$$

En relación a la Productividad:

H_{P0} = La productividad con ITLD es igual a la productividad con TDD (Hipótesis Nula)

H_{P1} = La productividad con ITLD es diferente a la productividad con TDD (Hipótesis Alternativa)

$$H_{P0}\colon\ \mu(PROD)_{ITLD} = \mu(PROD)_{TDD}\ \text{(Null Hypothesis)}$$
$$H_{P1}\colon\ \mu(PROD)_{ITLD} \neq \mu(PROD)_{TDD}\ \text{(Alternative Hypothesis)}$$

Las características principales del experimento en la Empresa F se presentaron en la Tabla 1; los resultados que se obtuvieron en el experimento en mención se presentan en la Tabla 6.

	VARIABLES RESULTADOS DEL EXPERIMENTO DE LA EMPRESA F
Resultados en Calidad	- No existe una diferencia estadística en la calidad del trabajo entre YW y TDD, sin embargo, en valores obtenidos en calidad son mínimamente superiores usando YW.

	- Para ITL se cuenta con muy pocos datos para emitir criterios, sin embargo la Empresa F menciona que no se puede considerar a este método una mala opción.
Resultados en Productividad	- En cuanto a productividad se aprecia que al usar TDD se obtuvo un porcentaje superior en productividad comparado con YW e ITL, sin embargo, sus valores no son estadísticamente significativo. - Se concluye por tanto que TDD conduce a mejoras en la productividad, después de realizar este corto entrenamiento sin experiencia de los sujetos.

Tabla 6: Variables Resultado de la Empresa F

El experimento efectuado en la Empresa F, es el primero de la industria que utiliza al método de desarrollo YW para el análisis. En los resultados que presenta enfatiza que desafortunadamente se perdió un grupo experimental, por lo que no tuvieron datos completos para realizar las mediciones como se había planteado, obtuvieron por tanto resultados haciendo referencia a 1 de las 3 tareas implementadas en la que obtuvieron la mayor cantidad de información; este evento de cualquier manera disminuye la validez a los resultados que se obtienen. La empresa menciona que con una mayor participación de los sujetos se podría detectar más efectos en el experimento.

En función de que las variables dependientes e independientes que se plantean para la presente investigación tienen mucha similitud con las utilizadas en el experimento de la Empresa F, se realizará una réplica del mismo, de modo que los resultados obtenidos establezcan condiciones en las que los efectos sean válidos.

2.1.6 Herramienta de desarrollo.

La herramienta de desarrollo informático que utiliza ETAPA EP para crear software es GeneXus en la versión Evolution 1, herramienta que se utiliza desde hace más de 6

años y ha aportado significativamente para el mantenimiento de las funciones del core del negocio.

En la STI no se escriben casos de prueba para asegurar funcionalidades creadas, el modo de verificar que el código es correcto son las pruebas manuales que ejecutan los desarrolladores con los usuarios al finalizar la implementación; sin embargo, para esta herramienta de desarrollo, los componentes que se pueden utilizar para escribir pruebas unitarias son GxUnit y GxTest.

Para el presente estudio nos centraremos en GxUnit (Araújo, 2008). Tiene la funcionalidad similar a JUnit y NUnit pero adaptada a la plataforma GeneXus, permite escribir casos de prueba para verificar la correcta funcionalidad de procesos desarrollados en las aplicaciones. Una guía a cerca de GxUnit se lo puede encontrar en el Proyecto GxUnit de Araújo (Almeida y Araújo, 2008).

CAPITULO 3

3.1. Descripción del Experimento.

3.1.1 Antecedentes.

Conducir experimentos en el ámbito de la industria es mucho más costoso que realizarlo en el ámbito académico, no solo por el tiempo de los profesionales que laboran con diferentes responsabilidades y presión, sino también porque la logística demanda un trabajo intensivo y es propenso a errores, se requiere la recopilación de datos, especificación de tareas, información de fondo, etc. crece más los inconvenientes cuando el experimento se lo lleva a cabo fuera del lugar habitual del trabajo, por lo que es necesario contar con soporte tecnológico y herramientas electrónicas (Basili, et al., 1986).

Para llevar a cabo lo planteado como proyecto de tesis se obtuvo el patrocinio de la Subgerencia de Tecnologías de la Información (STI) de ETAPA EP

Se contó también con el apoyo del grupo de investigadores de la Universidad de Madrid, GrISE-UPM, entidad que también apoyó en el experimento de la Empresa F, del cual, en esta ocasión se ejecutará una réplica, por lo tanto, tiene varias similitudes en su diseño.

El presupuesto detallado para el proyecto se lo puede visualizar en el cuadro del Anexo 1.

ETAPA EP apoyó con el personal participante, auditorio para llevar a cabo el experimento, etc.; la maestrante con los gastos de logística y GrISE-UP aportó con el viaje y viáticos de una persona especializada en el tema para impartir la capacitación en la ciudad de Cuenca, se trata de un investigador español, Oscar Dieste, quién tiene una

gran experiencia en actividades asociadas a esta rama de la ingeniería y es reconocido en el mundo de la investigación por sus diversas publicaciones sobre el tema.

3.1.2 Objetivos.

3.1.2.1 Objetivo General.

- Analizar las aproximaciones del desarrollo de software: TDD, ITL y el método propio de ETAPA EP con el propósito de compararlas con respecto a la calidad del software y productividad de los desarrolladores, desde el punto de vista de los investigadores.

3.1.2.2 Objetivos Específicos

- Estudiar la problemática de ETAPA EP en cuanto al desarrollo de software.
- Estudiar la técnica de desarrollo de software Test Driven Development (TDD).
- Ejecutar un experimento controlado con los desarrolladores de ETAPA EP comparando TDD con ITL y el método propio.
- Elaborar un informe para ETAPA EP con los resultados obtenidos, conclusiones y recomendaciones de mejoras en el proceso de desarrollo de software en ETAPA EP, así como lineamientos básicos para una posible adopción en caso que los resultados favorezcan a TDD.

3.1.3 Planteamiento de Hipótesis.

3.1.3.1 Productividad

Hipótesis Nula:

$H_{P0} = TDD\ no\ representa\ una\ mejora\ significativa\ ante\ los\ otros\ métodos\ de\ desarrollo\ analizados\ en\ cuanto\ a\ la\ productividad\ de\ los\ desarrolladores$

Hipótesis Alternativa 1:

$H_{P1} =$ TDD maximiza la productividad de los desarrolladores frente a ITL y el método propio

Hipótesis Alternativa 2:

$H_{P2} =$ ITL maximiza la productividad de los desarrolladores frente a TDD y el método propio

Hipótesis Alternativa 3:

$H_{P3} =$ *El* método propio maximiza la productividad de los desarrolladores frente a ITL y TDD

3.1.3.2 Calidad

Hipótesis Nula:

$H_{C0} =$ TDD no representa una mejora significativa ante los otros métodos de desarrollo analizados en cuanto a la calidad del software

Hipótesis Alternativa 1:

$H_{C1} =$ TDD maximiza la calidad del software frente a ITL y el método propio

Hipótesis Alternativa 2:

$H_{C2} =$ ITL maximiza la la calidad del software frente a TDD y el método propio

Hipótesis Alternativa 3:

$H_{C3} =$ *El* método propio maximiza la calidad del software frente a ITL y TDD

3.1.4 Diseño.

El diseño del experimento se realizó en conjunto con los investigadores del grupo GRISE de la UPM de Madrid-España quienes diseñaron anteriormente el experimento original en la Empresa F.

3.1.4.1 Gestión.

La gestión con la STI de ETAPA EP se realizó con la debida anticipación para que acudan todos los desarrolladores a la capacitación y experimento.

Se invitó también a más empresas municipales de Cuenca que utilizan la herramienta de programación GeneXus para que participen en este taller, las empresas invitadas fueron:

- Empresa Pública Municipal de Movilización - EMOV EP
- GAD Municipal del Cantón Cuenca
- Empresa Pública Municipal de Aseo de Cuenca - EMAC EP

El lugar programado para el evento fue el "Auditorio de la Central Telefónica de Totoracocha" de ETAPA EP, fuera de sus puestos habituales de trabajo, de modo que los funcionarios no tengan interrupciones del día a día y puedan concentrarse en sus tareas.

Con el propósito que los sujetos tengan una comprensión básica de pruebas unitarias y conceptos de TDD se ofreció entrenamiento durante el taller. Las sesiones fueron diseñadas para una duración de 12 horas en total, incluyendo conferencias, ejercicios prácticos individuales y tareas experimentales, distribuidas como se presenta en la agenda de la Tabla 7.

El capacitador brindó ayuda en conceptos de las diferentes técnicas de programación que se utilizan y la autora de este trabajo brindó ayuda en el uso de GxUnit, componente para pruebas unitarias de la herramienta GeneXus.

	LUNES	**MARTES**	**MIÉRCOLES**
08h00 - 09h45	Introducción Metodologías Agiles Slicing Ejercicios	Pruebas unitarias GxUnit ITL Ejercicio Grupal Ejercicio Individual	TDD Ejercicio Grupal Ejercicio Individual
09h45 - 10h00	Pausa Activa	Pausa Activa	Pausa Activa
10h00 - 11h45	Desarrollo de ejercicios usando YW	Desarrollo de ejercicios usando ITL	Desarrollo de ejercicios usando TDD

11h45 - 12H00	*Recolección de código fuente trabajado*	*Recolección de código fuente trabajado*	*Recolección de código fuente trabajado*

Tabla 7: Agenda planificada

3.1.4.2 Encuestas demográficas.

Mientras se realizaba la planificación detallada del experimento, se solicitó a los sujetos experimentales llenar una encuesta demográfica para obtener información acerca de su perfil académico, carrera profesional, experiencia en programación, experiencia en lenguajes de programación y metodologías utilizadas en el experimento, entre otros. La encuesta se realizó con un cuestionario diseñado en Google Docs cuyo link fue compartido a los sujetos 3 días antes del experimento. En la Tabla 8 se presenta un resumen de los resultados de la encuesta en lo relativo a la experiencia.

# Años	Carrera Profesional	Experiencia en desarrollo	Experiencia en GeneXus	Experiencia en GxUnit	Experiencia en TDD
0-1	15.38	0.00	30.77	0.00	7.69
2-5	15.38	15.38	46.15	0.00	0.00
6-9	46.15	38.46	23.08	0.00	0.00
10 o más	23.08	46.15	0.00	0.00	0.00

Tabla 8: Experiencia de los sujetos experimentales

El 85% de los participantes se encuentran entre los 30 y 35 años de edad. Todos los sujetos son ingenieros de sistemas, el equivalente a ingenieros informáticos. La mitad tienen título de master en áreas afines a la gestión de TI.

El 92% de los participantes tienen experiencia en desarrollo orientado a objetos sin embargo actualmente todos desarrollan software en el case GeneXus. El 38% de los sujetos comparten sus actividades de desarrollo con tareas de analistas de software. El 7.69% son Supervisores de desarrollo que también realizan actividades de desarrollo, aunque en menor proporción, y el 7.69% realiza desarrollo en otras herramientas distintas a GeneXus.

El 15.38% de los sujetos han asistido a entrenamiento de pruebas unitarias. Ninguno de los sujetos tiene experiencia con el framework de pruebas GxUnit. El 23% tienen un conocimiento básico acerca de TDD pero no lo han utilizado en sus tareas, el 7.69% ha usado TDD durante 1 año.

En la Tabla 9 se presentan los resultados sobre formación académica, conocimientos de metodologías de desarrollo y lenguajes de programación que han sido utilizados por los sujetos.

Aproximadamente la mitad de los sujetos han utilizado el método cascada durante su formación académica o en su carrera profesional y la tercera parte de los sujetos han utilizado el método incremental. Todos los sujetos han utilizado el método Scrum y tienen formación en Scrum, este método se utiliza actualmente en ETAPA EP.

Educación	**Degree**	**Leng. Prog conocidos**	**Framewrok de pruebas**	**Metodologías**
Ing. de Sistemas (13)	BS (6)	GeneXu (13)	Junit, Cunit, NUnit (1)	Scrum (13)
	MS (7)	Java (5)		Waterfull (6)
		. NET (4)		Incremental (4)
		C# (3)		
		Visual Basic (2)		
		RPG (2)		
		PHP (2)		
		Oracle (2)		

Tabla 9: Conocimientos de los sujetos experimentales

3.1.4.3 Infraestructura para el experimento. (Instrumentación).

Paralelamente también se preparó la infraestructura tecnológica para los sujetos, incluye las herramientas que fueron usadas durante el entrenamiento y las sesiones experimentales. Esta infraestructura fue empotrada en una máquina virtual VMWare 10.0. La imagen incluye sistema operativo Windows 7, Web browser, CASE GeneXus Evolution 1 en ambiente .Net y GxUnit como framework de pruebas. GeneXus es un

Case y fue configurado para generar código en lenguaje C#. Se instaló el framework .Net 2.0. Se entregó esta máquina virtual a los asistentes y cada uno colocó ésta en su computadora portátil, esta práctica de desarrollo fue útil por las siguientes razones:

a) Para aislar el ambiente de desarrollo experimental del ambiente de desarrollo que los sujetos usan en su trabajo diario.

b) para controlar la tecnología que usan los sujetos y,

c) por la complejidad de instalación del case GeneXus con sus componentes.

Las máquinas virtuales fueron entregadas a los sujetos una semana antes del experimento para que lo instalen en sus equipos portátiles.

3.1.4.4 Tareas experimentales.

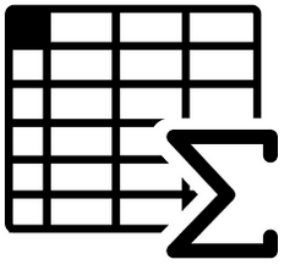
Las tareas experimentales tienen funcionalidades cortas que no requieren almacenamiento en BD ni condiciones especiales en los programas, a estos desarrollos lo denominamos software juguetes, en esta ocasión se trabajó con los descritos a continuación:

Bowling Score Keeper (BSK)

Juego de bolos.- El software calculará el puntaje de cada jugador según sus lanzamientos y basándose en las reglas dadas, informará sobre los resultados para determinar al ganador. Anexo4

Mars Rover (MR)

Hace referencia a un cuerpo que se desplaza por Marte según instrucciones dadas, por tanto, el algoritmo debe permitir a Mars Rover avanzar en su camino en las direcciones que se le indique a través de comandos de teclado. Anexo 5



Spreadsheet (SS)

Se trata de una hoja electrónica en la que el usuario coloca datos en las celdas y ejecuta fórmulas de las 4 operaciones básicas haciendo referencia a dichos datos o con números escritos directamente en las fórmulas. Anexo 6

Los ejercicios se entregaron a los sujetos de manera aleatoria cada día, pero cuidando que todos desarrollen los 3 ejercicios con una técnica diferente cada uno.

Al finalizar las tareas diarias, se recolectó el código fuente generado por los sujetos, se solicitó un archivo de extensión xpz con los objetos utilizados, de modo que los investigadores podían importar dichos archivos y revisar el código en otra máquina.

Los investigadores recolectamos el aporte de cada sujeto en un disco duro extraíble, así como también para mayor seguridad se solicitó a los participantes colocar su información en un espacio asignado del google drive, etiquetándolo con su nombre y colocándolo en la carpeta del día correspondiente.

En lo posterior al experimento, para comprobar la funcionalidad de estas tareas, se crearon casos de prueba, con el objetivo de ejecutarlos sobre el código escrito por los participantes y determinar si cumple o no cada prueba.

Los casos de prueba se implementaron en la herramienta GeneXus, usando GXUnit.

Los casos de prueba originales que se utilizaron para verificar el código fuente en la Empresa F están escritos en Java, por tanto, fue necesario transcribirlos a GeneXus.

Se utilizó revisión semántica en el código fuente entregado por los

desarrolladores, además fue necesario en la mayoría de los casos escribir código adicional para lograr un mapeo entre las diferentes formas de programar de cada desarrollador y los casos de prueba genéricos para probar las funcionalidades de modo que la conexión entre los casos de prueba y la programación realizada por los desarrolladores sea válida.

3.1.5 Factores y tratamientos.

Los factores, tratamientos y demás características que están presentes en el actual experimento, se presentan en la Tabla 1, se puede verificar también las similitudes y diferencias con relación al experimento original de la Empresa F:

Cabe indicar que los desarrolladores en su técnica de desarrollo *Your Way* no escriben pruebas unitarias para probar el código que escribieron, en lugar de ello ejecutan el sistema y observan la respuesta de las diferentes situaciones que se plantearon como requerimientos, este comportamiento se detalla en la sección 1.5.2.

3.1.6 Variables respuesta.

Las variables respuesta que se desean estudiar en la investigación es la calidad del software (QLTY) y la productividad de los desarrolladores (PROD).

La métrica que se utilizará para medir la calidad está enfocada a los casos de prueba que acierten los sujetos experimentales (TEST), su detalle se describe en la sección 2.1.4.

Para medir la productividad de los desarrolladores el principal factor es el tiempo invertido para realizar una actividad, sin embargo, está directamente anidado a la calidad con la que se genera el producto; dado que en este experimento se ha brindado el mismo número de horas para implementar las tareas con cada método de desarrollo,

se obtendrán las conclusiones en base a la calidad y a los resultados generados por el experimento. En la sección 5.1.1 se especifica esta particularidad como parte de validez del experimento.

3.1.7 Casos de prueba utilizados para la obtención de las métricas.

Se establecieron un total de 132 casos de prueba con 170 aserciones, para validar la correcta y completa funcionalidad de las aplicaciones desarrolladas por los sujetos. El número de casos de prueba y aserciones establecidos para cada tarea experimental (MR, BSK y SS), se detalla en la Tabla 10, con la distribución de casos de prueba para cada tarea experimental, en donde US hace referencia a la Historia de Usuario, Test a los Casos de Pruebas y Assert a las aserciones que puede tener cada Caso de Prueba:

Casos de Prueba								
Mars Rover (MR)			**Bowling Score Keeper (BSK)**			**Spreadsheet (SS)**		
US	**Tests**	**Assert**	**US**	**Tests**	**Assert**	**US**	**Tests**	**Assert**
1	1	1	1	0	0	0	1	1
2	7	11	2	0	0	1	5	5
3	4	8	3	0	0	2	5	5
4	4	7	4	1	2	3	0	0
5	4	8	5	5	5	4	2	2
6	8	8	6	6	6	5	7	7
7	6	15	7	8	8	5B	4	4
8	4	8	8	5	5	6	1	1
9	3	8	9	5	5	7	1	1
10	3	7	10	2	2	8	1	1
11	8	8	11	2	2	9	7	7
			12	2	2	9B	3	3
			13	2	2	10	2	2
						11	2	2
						12	1	1
Total	**52**	**89**		**38**	**39**		**42**	**42**

Tabla 10: Casos de prueba para ETAPA EP.

Cabe indicar que, aunque las tareas experimentales fueron las mismas que

realizaron los sujetos en el experimento original, existen casos de prueba que no son aplicables a esta réplica debido a la herramienta de programación. GeneXus no es un lenguaje orientado a objetos como lo es Java, por tanto, los casos de prueba de declaraciones, asignaciones o inicialización de variables no pueden ser reproducidas en GeneXus, la diferencia de cantidad de casos de prueba entre una empresa y otra se presenta en la Tabla 11.

	Empresa F	**ETAPA EP**
MR	81	89
BSK	62	39
SS	52	42

Tabla 11: Número de casos de prueba establecidos de la Empresa F y ETAPA EP.

De cualquier manera, el valor buscado está en función de los casos de pruebas que acierte cada sujeto sobre los casos de prueba establecidos, por tanto, el valor que se desea obtener será el resultado de la siguiente fórmula:

$$\frac{\#\text{ de casos de prueba acertados}}{\#\text{ de casos de prueba ejecutados}}$$

CAPITULO 4

4.1. Ejecución de la Replicación.

4.1.1 Asistencia de sujetos experimentales.

En cuanto a la asistencia de los sujetos experimentales se obtuvo excelentes resultados, asistieron los 13 funcionarios de ETAPA EP, es decir se logró tener la afluencia de todos los desarrolladores de la STI. La participación fue obligatoria los dos días laborables (jueves y viernes) y voluntaria el día no laborable (sábado).

Se contó además con la asistencia de los 5 desarrolladores invitados del Municipio de Cuenca y 2 desarrolladores invitados de la EMOV, tal como lo habían confirmado sus directivos.

Los sujetos participantes fueron motivados con el training que se dio en los tres días. Todos los sujetos implementaron todas las tareas cada uno contó con una computadora técnica, con acceso a internet y que disponía de una máquina virtual en la que se instalaron previamente las herramientas necesarias para el taller.

Cabe anotar que los desarrolladores de estas empresas participaron en el experimento, pero sus datos no fueron tomados en cuenta debido a que al ser pocos participantes por empresa es bajo el poder estadístico. Por ese motivo se tomó en cuenta únicamente los datos de ETAPA EP ya que se tiene el mayor número de participantes.

4.1.2 Desarrollo del taller.

El Taller teórico–práctico fue impartido en 3 sesiones por el Dieste O., investigador de GRISE-UPM, cada sesión tuvo una duración de 4 horas, de las cuales la mitad del tiempo se destina a capacitar al personal con teoría y ejemplos de cada técnica y la otra mitad se realizan los ejercicios individuales que son los insumos para el experimento.

El primer día se presentó una introducción, objetivos planteados y expectativas para el taller, se realizó una revisión de las metodologías ágiles y también se capacitó sobre Desarrollo Incremental, Slicing, la mayoría de personas presentes estaban familiarizadas con esta forma de programación por lo que hubo una participación activa de los asistentes. Para la realización del primer ejercicio del experimento se solicitó implementarlo tal como lo realizan a diario en sus empresas (sin pruebas unitarias), por lo que sería la técnica denominada Your Way (YW).

El segundo día el estudio central fueron las pruebas unitarias, conceptos y utilidad, se presentó como herramienta a GxUnit, componente de GeneXus que permite escribir pruebas unitarias; cabe indicar que para la mayoría de los presentes el uso de este componente fue totalmente nuevo por lo que se presentaron ejemplos y se solventaron ejercicios proyectados en pantalla para receptar la participación de todos los asistentes. Se presentó también a la técnica ITL en la que se hacía uso de las pruebas unitarias. Para el ejercicio del día se solicitó a los participantes solventar un requerimiento y escribir casos de prueba que demuestren que su implementación es la correcta, la técnica utilizada por tanto fue ITL.

Para el tercer día el tema de estudio fue TDD, en donde cambia totalmente el método tradicional al que los funcionarios estaban acostumbrados a utilizar, deben escribir primero pruebas unitarias y luego escribir el código fuente que sirva para aprobar dichas pruebas. El ejercicio del tercer día fue solicitado realizarlo con la técnica TDD.

Como se puede apreciar los temas y horarios se cumplieron de acuerdo al diseño del experimento, existiendo pequeñas variaciones en cuanto a los tiempos debido básicamente al alto interés de los participantes por los temas abordados, finalmente el taller se lo llevó según el cronograma que se describe en la Tabla 12:

	LUNES	MARTES	MIÉRCOLES
08h00 - 10h00	Introducción Metodologías Agiles Slicing Ejercicios	Pruebas unitarias GxUnit ITL Ejercicio Grupal Ejercicio Individual	TDD Ejercicio Grupal Ejercicio Individual
10h00 – 10h15	Pausa Activa	Pausa Activa	Pausa Activa
10h15 - 12h00	Desarrollo de ejercicios usando YW	Desarrollo de ejercicios usando ITL	Desarrollo de ejercicios usando TDD
12h00 - 12h20	*Recolección de código fuente trabajado*	*Recolección de código fuente trabajado*	*Recolección de código fuente trabajado*

Tabla 12: Cronograma real de actividades

Durante la resolución de ejercicios individuales los sujetos podían realizar consultas al capacitador sobre las tareas que debían realizar, existió algo de dificultad por el hecho de que los requerimientos estaban escritos en Ingles, por tanto, las preguntas más se enfocaron a ese tema. Consultaban también sobre el componente GxUnit, en donde se les brindó ayuda por parte de la autora del proyecto.

Conversaban entre ellos para entender mejor el requerimiento solicitado, sin embargo, luego de comprender el enunciado guardaron total silencio para resolver los ejercicios.

4.1.3 Distribución de tareas.

La distribución de las tareas al personal de la STI que son los sujetos para el experimento, se las detalla en la Tabla 13, pudiendo evidenciarse que se cumplió con el objetivo de que los sujetos intercalen la técnica de desarrollo y la tarea asignada en el transcurso de los 3 días del taller.

	Dia1 / YW	Dia2 / ITL	Dia3 / TDD
Sujeto 1	BSK	SS	MR
Sujeto 2	MR	BSK	SS
Sujeto 3	MR	BSK	SS
Sujeto 4	SS	MR	BSK
Sujeto 5	BSK	SS	MR
Sujeto 6	MR	BSK	SS
Sujeto 7	SS	MR	BSK
Sujeto 8	SS	BSK	MR
Sujeto 9	BSK	SS	MR
Sujeto 10	BSK	SS	MR
Sujeto 11	SS	MR	BSK
Sujeto 12	BSK	SS	MR
Sujeto 13	MR	BSk	SS

Tabla 13: Distribución de sujetos experimentales

4.1.4 Recolección de código fuente.

La recolección de código fuente que generaron los sujetos en cada ejercicio para el experimento no presentó ninguna novedad mayor, cada sujeto respaldó su código fuente utilizando la funcionalidad export en la herramienta GeneXus, lo cual genera un archivo .xpz con todos los objetos utilizados, de modo que los investigadores podamos realizar un import del archivo y revisar el código en otra máquina. Los investigadores recolectamos el aporte de cada sujeto en un disco duro extraíble, para mayor seguridad se solicitó a los participantes colocar también su información en un espacio asignado del google drive, etiquetándolo con su nombre y colocándolo en la carpeta del día correspondiente. El tiempo aproximado que tomó esta actividad fue de 20 minutos diarios.

4.1.5 Novedades presentadas.

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

- El primero día se presentaron algunos inconvenientes que los sujetos experimentales no mencionaron hasta el momento de realizar las taras experimentales, como falta de conexión a la red, necesidad de un mouse, claves de acceso etc. lo cual de cierta manera acortó el tiempo que tuvieron para la ejecución de la tarea experimental. Siendo esta una diferencia entre el primer día y los 2 siguientes en donde no tuvieron ningún inconveniente que les atrase la ejecución de la tarea.
- Uno de los sujetos experimentales tiene muy poca experiencia en la utilización de la herramienta de programación GeneXus, por lo que su aporte fue casi nulo, sin embargo, participó los 3 días del taller intentando realizar los ejercicios
- El segundo día un sujeto utilizó Java en lugar de GeneXus para desarrollar el software que se le asignó, por tanto, dicho código no puede ser evaluado como los demás.
- Como punto positivo se indica que se contó con la presencia de los ingenieros Fonseca R. y Raura G., investigadores de la Universidad de las Fuerzas Armadas, ESPE de Quito-Ecuador, quienes colaboraron con la logística del evento.

CAPITULO 5

5.1. Análisis e interpretación de resultados.

5.1.1. Evaluación de la validez del experimento.

En cuanto a la validez del experimento, la cual se refiere al análisis de amenazas que se pueden producir a lo largo del experimento, se lo ha venido realizando a lo largo del estudio, sin embargo, se lo explicará en este capítulo ya que demuestran que tan válidos son los resultados obtenidos y repercute directamente en la interpretación de los mismos. La evaluación de la validez de un experimento, comprende validez interna, externa, de constructo y de conclusión. A continuación, se describe las acciones tomadas en cuenta para asegurar la validez del experimento:

En el presente experimento los sujetos experimentales son todos los desarrolladores de la empresa ETAPA EP, por lo que no fue necesario realizar acciones para seleccionarlos, incluso, cabe indicar que 2 personas eran novatos en la herramienta, sin embargo, participaron del experimento porque son parte del grupo de desarrollo de la empresa.

El primer día del taller, en el cual la programación se realizó con el método propio, se les entregó las tareas experimentales en forma aleatoria, armándose de esta manera los grupos, en los 2 días siguientes se les entregó las tareas faltantes a cada uno con lo que se puede asegurar que cada sujeto desarrollo diferentes tareas y con distintos tratamientos cada sesión.

Con respecto a acontecimientos inesperados en la ejecución del experimento, se puede anotar que el primer día existió para algunos sujetos complicaciones en el acceso a internet o a la máquina virtual, lo cual retrasó unos minutos el inicio de su desarrollo.

Se mitigó el riesgo de tener invalidez en temas externos gestionando con la debida

anticipación la participación de los sujetos experimentales. Se contó con el apoyo de la Empresa para establecer como obligatorio la asistencia de los desarrolladores al experimento.

El taller se llevó a cabo en un auditorio distante a las oficinas de trabajo, de modo que no existan interrupciones en la concentración de cada sujeto, el auditorio contó con la comodidad para el desarrollo de las actividades planificadas y fue posible que todos los sujetos estuvieran físicamente en un solo lugar. Se destinaron 15 minutos diarios para una pausa activa lo cual ayudó a los sujetos a tomar un descanso adecuado y continuar con las tareas.

Los sujetos tenían conocimiento de que participarían en el taller 1 mes antes de su ejecución y se les entregó indicaciones e implementos a medida que se acercaba el día del evento, como por ejemplo manual de usuario para el componente GxUnit, la máquina virtual, entre otros. Todo esto con el objetivo de que puedan revisar el material y consultar cualquier duda para que el día del taller los sujetos se encuentren listos para su participación sin contratiempos.

La motivación que tuvieron los participantes fue la de contar con la presencia de un capacitador de Madrid - España y la de saber que del resultado del experimento se lograría mejorar el desempeño en sus tareas diarias, aumentando su confianza y productividad.

Es importante señalar, que, al finalizar el primer día del taller, los sujetos manifestaron que tuvieron dificultad en entender completamente las tareas, por el hecho de que estaban escritas en inglés. Conversaron entre los sujetos y se apoyaron para entender los enunciados ese momento de las 3 tareas de modo que no tuvieran el mismo inconveniente los siguientes días. Con lo cual, se debe descartar la idea de que los sujetos, conociendo previamente los enunciados de sus tareas, hayan podido realizar

adelantos en sus códigos, ya que luego del experimento los sujetos regresaban a sus oficinas a continuar con sus labores diarias, impidiéndoles que pudieran adelantar tareas del experimento.

Al inicio del diseño del experimento, se contaba con fórmulas separadas para medir las 2 variables: calidad y productividad, sin embargo, al realizar el análisis de datos, se evidenció que las 2 fórmulas generaban resultados muy semejantes. Haciendo referencia a la validez del constructo, se analizó este comportamiento y se tomó la decisión de utilizar solo a la calidad como métrica apropiada para manifestar los resultados de calidad, con un análisis estadístico para cada sujeto en el experimento, sin embargo, la productividad no será analizada de manera estadística, sino que se emitirán conclusiones en base a los resultados generados por el experimento.

Para que los resultados que se presentan sean válidos, se tomaron los análisis estadísticos apropiados para el tipo de datos que se manejan en este experimento.

5.1.2. Análisis de datos.

Una vez finalizada la etapa de ejecución del experimento, la cual concluyó con la recolección del código fuente generado por los sujetos experimentales, se realizó la validación de esta información, para ello se contó con una máquina virtual configurada con las mismas características que las máquinas utilizadas por los sujetos experimentales en los días que se ejecutó el experimento. Descripción detallada de infraestructura en la sección 3.1.7.

Los casos de prueba que constituyeron el material para realizar la medición de la calidad del software, fueron escritos en una Base de Conocimiento (Knowledge Base, KB) de GeneXus, con el componente GxUnit, ésta KB fue la utilizada para realizar la revisión de datos a lo largo del análisis.

Sobre la misma KB se destinaron secciones para colocar el código de cada sujeto experimental, por tanto, se realizó la importación del código fuente entregado por los sujetos en cada espacio correspondiente.

Al revisar el código fuente escrito por los participantes, fue necesario realizar una intervención semántica en el código de algunos sujetos, a continuación, se detallan algunas correcciones puntuales que se realizaron para ejecutar su código:

- Concatenar caracteres como "(", "," para que presente la variable de salida en el formato requerido.
- Conversión de minúsculas a mayúsculas para cumplir con la posible respuesta.
- Colocar por defecto el "=" al inicio de las sentencias para el caso de la hoja de cálculo.
- El uso de constantes en el código, se lo cambió por variables de entrada para que se ejecute la suit de casos de prueba genérica.
- Cambio en un bucle que inicie en 2 y no en 1.
- Ampliar los límites de la matriz, entre otros.

Además, surgió la necesidad de escribir código que conecte los casos de prueba con el código fuente origen, debido a que, por tratar de simular el comportamiento orientado a objetos para facilitar la solución a las tareas asignadas, los sujetos implementaron sus soluciones con mucha diversidad en los estilos de programación, entre los más significativos:

- Usaron dominios para simular clases
- Usaron varios procesos para desarrollar cada tarea
- Utilizaron constantes en lugar de variables, entre otros.

Adicionalmente, se implementó un aplicativo para optimizar la ejecución de los casos de prueba sobre el código fuente de cada sujeto de manera masiva, teniendo como respuesta la escritura de los resultados obtenidos por cada dificultad en un archivo de extensión txt. Por cada caso de prueba ejecutado, se podían obtener 3 posibles resultados:

- PASS Prueba correcta.
- FAIL: Prueba incorrecta.
- ERR: Error al ejecutar la prueba.

En los Anexos 7, 8 y 9 se presentan los resultados de las diferentes tareas experimentales obtenidas de manera individual. Por temas de privacidad, no se dan a conocer los nombres de cada sujeto experimental.

Para lograr la transformación de los datos presentados anteriormente a términos de calidad se ha clasificado los resultados de cada usuario como se presenta en la Tabla 14, es decir, se necesita contar con el número de aserciones totales y clasificar los resultados obtenidos en cada historia de usuario.

Para determinar la variable calidad por cada sujeto experimental se calcula la porción del número de casos de uso pasados satisfactoriamente y el número definido total de aserciones. Para representar este valor en formato de porcentaje, los resultados se multiplican por 100.

			ASSERTS
US0	PASS	0	1
	ERR	0	
	FAIL	1	
US1	PASS	0	5
	ERR	0	
	FAIL	5	
US2	PASS	0	5
	ERR	0	
	FAIL	5	
US4	PASS	0	2
	ERR	0	
	FAIL	2	
US5	PASS	4	7
	ERR	0	
	FAIL	3	
US5B	PASS	0	4
	ERR	0	
	FAIL	4	
US6	PASS	0	1
	ERR	0	
	FAIL	1	
US7	PASS	0	1
	ERR	0	
	FAIL	1	
US8	PASS	0	1
	ERR	0	
	FAIL	1	
US9	PASS	7	7
	ERR	0	
	FAIL	0	
US9B	PASS	2	3
	ERR	0	
	FAIL	1	
US10	PASS	0	2
	ERR	0	
	FAIL	2	
US11	PASS	0	2
	ERR	0	
	FAIL	2	
US12	PASS	1	1
	ERR	0	
	FAIL	0	

Tabla 14: Evaluaciones de un sujeto x

Para el ejemplo que se presenta en la Tabla 14, en la que se encuentran los datos de un sujeto x seleccionado de forma aleatoria, la fórmula de calidad que se aplicó es:

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

Una vez realizadas las evaluaciones detalladas anteriormente para cada sujeto y en cada tarea experimental, se transcribieron los resultados a la Tabla 15 en donde se puede apreciar la consolidación de todos los datos clasificados por tareas y sujetos experimentales:

ID	MARS ROVER				BOWLING SCOREKEEPER				SPREADSHEET			
	#TUS	PERTUS	QLTY	TRAT	#TUS	PERTUS	QLTY	TRAT	#TUS	PERTUS	QLTY	TRAT
Sujeto 1	#N/A	#N/A	#N/A	TDD	0.00	0.00	0.00	YW	1.00	7.14	2.38	ITL
Sujeto 2	0.00	0.00	0.00	YW	6.00	60.00	30.77	ITL	1.00	7.14	2.38	TDD
Sujeto 3	5.00	45.45	30.34	YW	8.00	80.00	38.46	ITL	2.00	14.29	9.52	TDD
Sujeto 4	1.00	9.09	2.25	ITL	0.00	0.00	0.00	TDD	3.00	21.43	19.05	YW
Sujeto 5	11.00	100.00	40.45	TDD	0.00	0.00	0.00	YW	4.00	28.57	33.33	ITL
Sujeto 6	11.00	100.00	78.65	YW	10.00	100.00	74.36	ITL	5.00	35.71	38.10	TDD
Sujeto 7	3.00	27.27	11.24	ITL	3.00	30.00	10.26	TDD	2.00	14.29	16.67	YW
Sujeto 8	0.00	0.00	0.00	TDD	3.00	30.00	12.82	ITL	3.00	21.43	16.67	YW
Sujeto 9	0.00	0.00	0.00	TDD	2.00	20.00	10.26	YW	4.00	28.57	11.90	ITL
Sujeto 10	3.00	27.27	8.99	TDD	0.00	0.00	0.00	YW	2.00	14.29	4.76	ITL
Sujeto 11	6.00	54.55	30.34	ITL	0.00	0.00	0.00	TDD	1.00	7.14	2.38	YW
Sujeto 12	3.00	27.27	7.87	TDD	0.00	0.00	0.00	YW	4.00	28.57	21.43	ITL
Sujeto 13	#N/A	#N/A	#N/A	YW	#N/A	#N/A	#N/A	ITL	#N/A	#N/A	#N/A	TDD

Tabla 15: Calidad por sujeto experimental y tarea, diferenciando tratamientos.

Con esta información consolidada se aplicaron análisis estadísticos de modo que se pudieran aceptar o rechazar las hipótesis planteadas en la investigación. Se utilizó R como software para realizar el análisis de datos.

En la Figura 7 se presenta un gráfico de caja y bigotes, generado sobre los datos de calidad (QLTY) de los sujetos experimentales en los 3 tratamientos utilizados, se puede observar la mediana, primer y tercer cuartil y los valores atípicos de cada tratamiento, en la Figura 8 se puede apreciar de manera detallada como varían los valores de calidad de cada sujeto según el tratamiento que utilizaron.

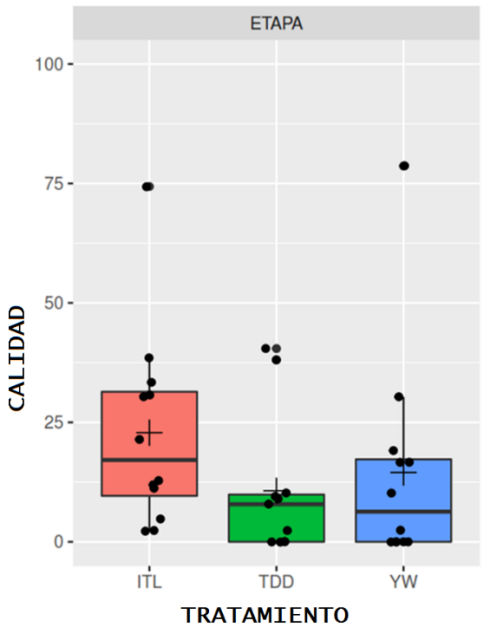


Figura 7: Diagrama de caja y bigotes para Calidad, segmentado por tratamientos

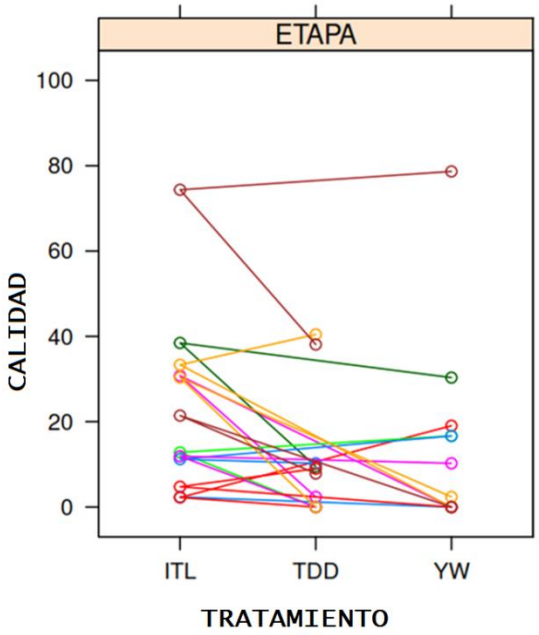


Figura 8: Detalle de la variación de calidad por cada sujeto

Analizando el comportamiento de datos e interpretando los gráficos estadísticos presentados anteriormente, se puede anotar lo siguiente:

- Los datos no tienen un comportamiento netamente normal, existe agrupación de sujetos que desarrollan en un rango similar de calidad, sin embargo, existen algunos valores atípicos, se puede decir que, con cualquiera de los 3 métodos, la mayoría de los sujetos se encuentran agrupados en un rango similar de calidad, mientras que la cuarta parte aproximadamente generan software de mejor calidad que el resto de sujetos experimentales.
- Estadísticamente no existen diferencias significativas entre los valores de calidad obtenidos de los métodos YW e ITL, ya que la caja de los valores de YW se solapan con los valores de la caja de ITL, se puede inclusive ver que la media de YW es igual al mínimo de ITL, por lo que no es posible asegurar que ITL sea definitivamente mejor que YW, al menos con los resultados de este experimento. Cuando los datos se solapan no se pude tomar una conclusión definitiva del comportamiento, para este caso, aunque despunte ITL, los 3 valores están en rangos muy similares de calidad.
- Si se analizan las medianas únicamente, si se podría decir que ITL es mejor que TDD y YW, considerando además que TDD y YW tienen varios valores de calidad en 0, no así ITL. Sin embargo, si se realiza un estudio paramétrico, se concluye que los 3 métodos obtienen resultados similares. La mayoría de sujetos tienden a superponerse dentro de los otros valores; inclusive, tomando en cuenta todo el rango de valores entre los mínimos y máximos señalados, sus resultados se solapan totalmente.

TRATAMIENTO YW
Data: subset(dETAPA, TREATMENT == 0)$PROD
W= 0.6858, p-value = 0.0006171

Tabla 16: Pruebas de Shapiro-Wilk para el tratamiento YW

TRATAMIENTO ITL
Data: subset(dETAPA, TREATMENT == 1)$PROD
W= 0.85955, p-value = 0.04826

Tabla 17: Pruebas de Shapiro-Wilk para el tratamiento ITL

TRATAMIENTO TDD
Data: subset(dETAPA, TREATMENT == 2)$PROD
W= 0.71455, p-value = 0.0007204

Tabla 18: Pruebas de Shapiro-Wilk para el tratamiento TDD

Aplicando pruebas de Shapiro-Wilk, método que permite contrastar la normalidad de un conjunto pequeño de datos, se puede evidenciar que los datos en el experimento de ETAPA EP no provienen de una distribución normal, ya que los resultados de las tablas 16, 17 y 18, presentan valores p (p-value) menores al índice de tolerancia 0,05 en los tratamientos TDD, YW e ITL

Con el comportamiento obtenido, no es aconsejable realizar análisis más profundos de las variables ya que no se obtendrían conclusiones que aporten significativamente al estudio.

Tanto el experimento de ETAPA EP como el de la Empresa F tienen efectos pequeños al analizar sus datos, pero, al tratarse esta investigación de una replicación de experimentos, se realizaron comparaciones de las 2 empresas y combinaciones de variables creando modelos lineales mixtos entre los experimentos, buscando encontrar diferencias más llamativas, sin embargo, se puede concluir con lo siguiente:

- Se analizó resultados tomando como variable a las tareas experimentales (MR, BSK, SS), sin embargo, no parece ser un tema a estudiar debido a que se trataba de tareas con fines netamente académicos, en el día a día no se realizan programaciones semejantes, por lo que no es procedente realizar un análisis profundo de esta variable.

- En la Empresa F no se puede evaluar la iteración de tratamiento con tarea ya que los sujetos experimentales no asistieron a todas las sesiones, por lo que no se cuenta con datos completos para realizar esta combinación de datos y por tanto no puede realizarse comparaciones con ETAPA EP.
- Rectificando lo anotado anteriormente con la Tabla 19, se concluye que comparar tratamientos, tareas, tratamientos con tareas o tratamiento con experimentos, no es estadísticamente significativo. La única variable que podría indicarse como estadísticamente significativa es EXPERIMENT, es decir, realizar comparaciones entre los 2 experimentos de manera global si generaría un resultado válido, sin embargo, se considera conveniente realizar una evaluación del efecto que causa la utilización de los diferentes tratamientos en cada una de las empresas para hacer algunas observaciones comparables que se describen más adelante.

	numDF	denDF	F-value	p-value
(Intercept)	1	27	87.13387	<.0001
TREATMENT	2	27	0.17466	0.8407
TASK	2	27	2.69661	0.0856
EXPERIMENT	1	18	40.62387	<.0001
TREATMENT:TASK	4	27	1.60589	0.2015
TREATMENT:EXPERIMENT	2	27	1.99006	0.1562

Tabla 19: Análisis de varianza

Analizando las desviaciones estándar de los promedios obtenidos al realizar la unión de 2 tratamientos, se presentan en las Figuras 9 (ITL con YW) y la Figura 10 (TDD con YW) tanto para ETAPA EP como para la Empresa F, con lo que se puede evidenciar que: tanto en la unión de YW con ITL así como la unión de YW con TDD, la desviación estándar de ETPA EP es menor que el de la Empresa F, pues, como se

observaba en gráficos anteriores la variabilidad de resultados que presenta la Empresa F es mayor a los de ETAPA EP, ocasionando por tanto que, cualquier estimado en los modelos estadísticos que produjeran los datos de la Empresa F, pueden ocasionar mayores errores.

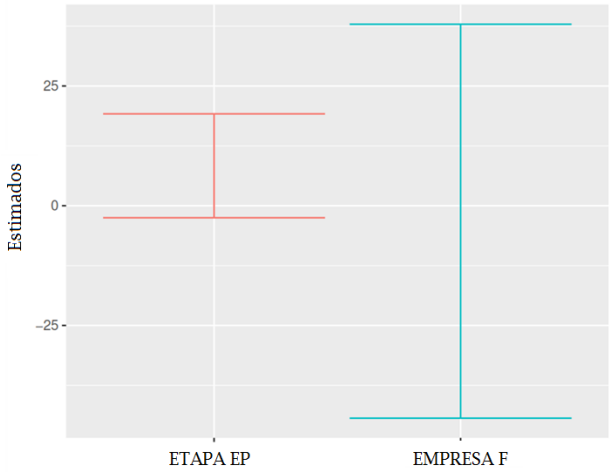


Figura 9: Diferencia estimada de tratamientos (ITL-YW) a través de experimentos

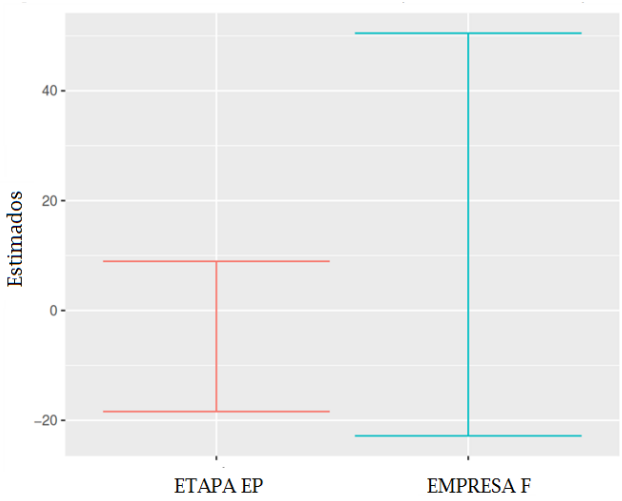


Figura 10: Diferencia estimada de tratamientos (TDD-YW) a través de experimentos

A continuación se presenta gráficos estadísticos de los datos de calidad obtenidos en ETAPA EP y la Empresa F seccionado por tratamientos, en la figura 11 se los presenta con gráfico de caja y bigotes de todos los sujetos, mientras que en la figura

12 se presenta una ilustración gráfica de cada sujeto en el rango de calidad y segmentado también en los 3 tratamientos.

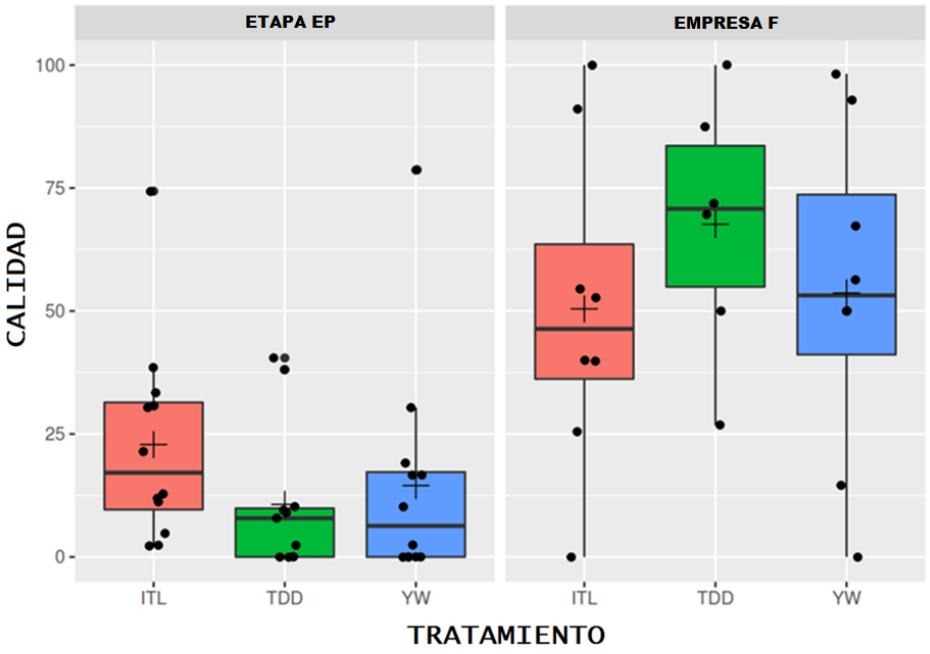


Figura 11: Gráfico de caja y bigotes por tratamientos para ETAPA EP y Empresa F

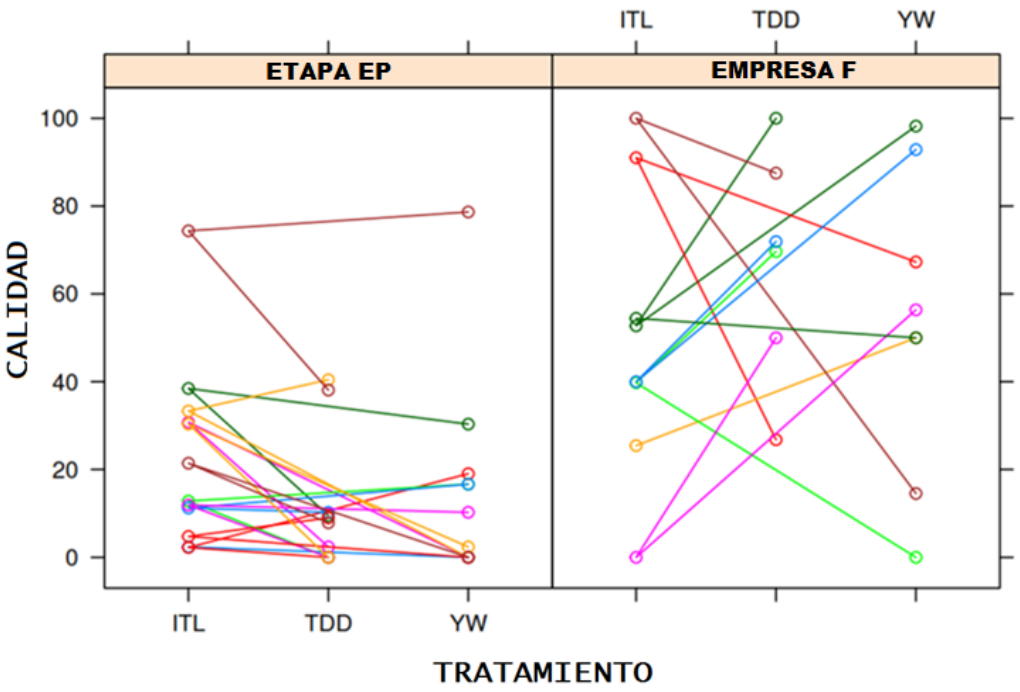


Figura 12: Detalle de la variación por sujetos entre ETAPA EP y Empresa F

Los datos de ETAPA EP no tienen un comportamiento normal, sin embargo, comparado con la Empresa F, ETAPA EP presenta menos variabilidad, la mayoría de sujetos se concentran en un rango determinado, salvo ciertos valores atípicos, mientras que en la Empresa F existen muchos valores en toda la escala de calidad, por lo que se concluye que aunque en promedio la calidad de los productos que desarrollan en la Empresa F es superior al promedio de ETAPA EP, en ETAPA EP se tiene una agrupación más concentrada de sujetos que desarrollan en el mismo rango de calidad y esta homogeneidad es positiva para ETAPA EP, ya que las conclusiones serán más aplicables a todo el grupo de desarrolladores.

	n	mean	sd	median	trimmed	min	max	range	ske	kurtosis	se
ETAPA EP	35	16.16	19.81	10.26	12.83	0	78.65	78.65	1.62	2.30	3.35
EMPRESA F	28	52.02	30.12	47.20	52.36	0	100	100	0.04	-1.14	5.69

Tabla 20: Estadística Descriptiva para Calidad entre los 2 tratamientos

Con la Tabla 20 se puede corroborar que la Empresa F tiene más dispersión que ETAPA EP, por los valores presentados como máximos y mínimos, además el valor de kurtosis indica que los datos de ETAPA EP se prestan más para un análisis paramétrico que los de la Empresa F.

Los gráficos de violín presentados en la Figura 13, son la versión continua de los gráficos de cajas y presentan la densidad en torno a resultados que no tiene una distribución normal.

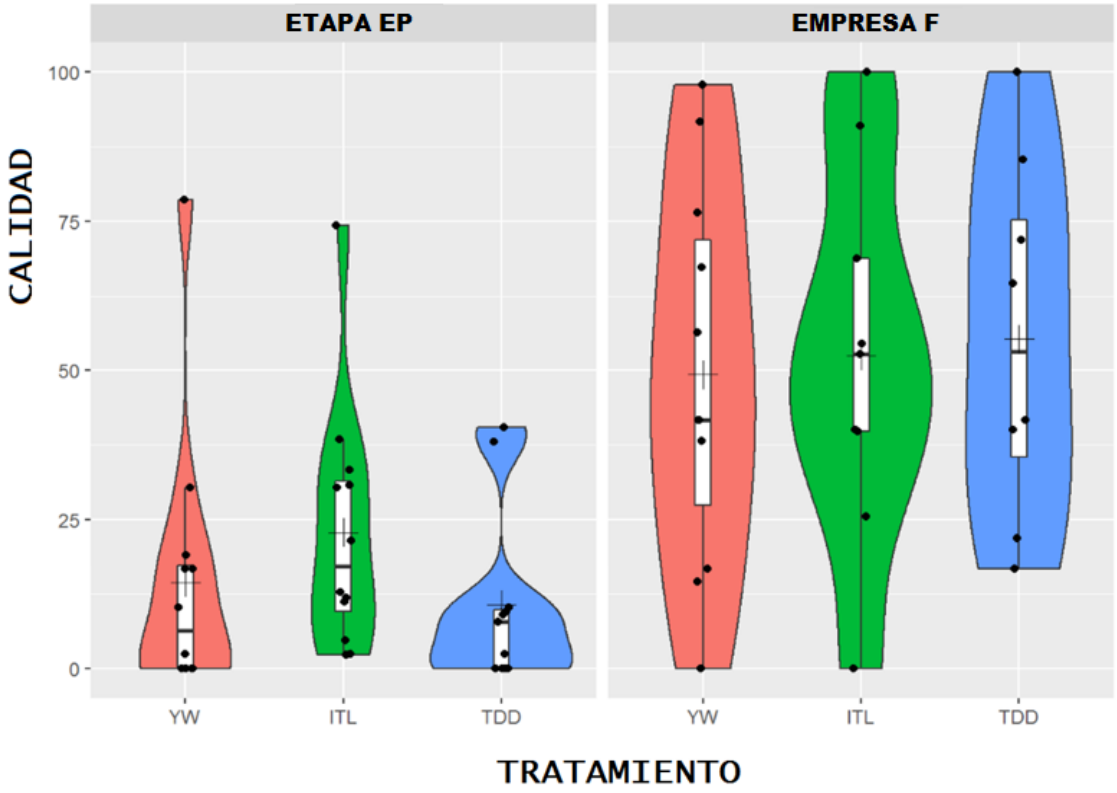


Figura 13: Gráficos de violín por tratamientos para ETAPA EP y Empresa F

De cualquier manera, se evidencia que en la Empresa F los sujetos se encuentran dispersos en el rango de calidad en los 3 tratamientos, mientras que en ETAPA EP los resultados obtenidos por los sujetos son más homogéneos y están más concentrados lo cual permite obtener conclusiones más precisas en términos de calidad.

CAPITULO 6

6.1. Conclusiones de la investigación.

Una vez finalizado todo el estudio del experimento y analizados los datos estadísticos del Capítulo 5 para la calidad del software, se puede afirmar que no se descarta la hipótesis nula.

Hipótesis Nula:

$H_{C0} =$ TDD no representa una mejora significativa ante los otros métodos de desarrollo analizados en cuanto a la calidad del software

Estadísticamente no existe una diferencia significativa entre un método y otro, inclusive, analizando los datos que se presentan TDD tiene el menor impacto positivo en cuanto a calidad frente a los otros 2 tratamientos: ITL y YW.

En cuanto a productividad y como se indicó en la sección 2.1.4 y 3.1.10, los resultados están ligados a la fórmula de calidad que se obtiene mediante métricas, ya que el tiempo asignado a cada desarrollador fue el mismo para los 3 tratamientos y por tanto el resultado de la calidad ya incluye la productividad de los desarrolladores. No generaría un efecto positivo, que un desarrollador finalice sus requerimientos en el menor tiempo posible, si los resultados de las pruebas son Fallidos (FAIL) o con Error (ERROR), por lo anotado, el resultado de la métrica de calidad es la que ayuda a predecir también el resultado de productividad para este experimento.

Con lo antes expuesto, se concluye que, de las hipótesis planteadas, se cumple con la hipótesis nula en cuanto a productividad de los desarrolladores.

Hipótesis Nula:

$H_{P0} =$ *TDD no representa una mejora significativa ante los otros métodos de desarrollo analizados en cuanto a la productividad de los desarrolladores*

Analizando los resultados obtenido en cuanto a las hipótesis acertadas de calidad y

productividad, se puede concluir que, aunque estadísticamente no existen diferencias significativas entre los tratamientos utilizados en el experimento de ETAPA EP, si existe una diferencia práctica que se explican a continuación:

Los métodos ITL y YW tienen la misma secuencia de pasos para el desarrollo de software, sin embargo, en ITL se escriben y ejecutan casos de prueba, mientas que en YW se lo realiza de forma manual, por este motivo se puede concluir que ambos tratamientos se adaptaron con mayor facilidad para los desarrolladores.

Mientras que, aplicar el método TDD, involucra un cambio de paradigma de desarrollo de software, ya que requiere en primer lugar, la escritura de las pruebas unitarias antes de la escritura del código fuente. Esto resultó más complicado para los desarrolladores, no obstante, no se puede descartar la posibilidad de que, con la práctica de la escritura de pruebas unitarias, se pueda hacer uso de este método en el futuro. Esto se puede corroborar con la opinión de Dieste, Fonseca, Raura y Rodríguez (2015), ya que en otros experimentos en los que el uso de TDD duraba más de 1 sesión, se evidenció que los resultados mejoraron, de tal manera que podría existir la posibilidad de que con mayor entrenamiento en TDD, los sujetos experimentales que participaron en esta investigación también obtengan mejores resultados en el desarrollo de sus productos.

Analizando el trabajo de replicación de experimentos en base a una interpretación general, se puede evidenciar que existe una diferencia sistemática entre los 2 experimentos, las conclusiones y recomendaciones de la investigación se detallan a continuación:

- En ETAPA EP el utilizar ITL o YW dio un resultado muy similar entre sí, mientras que en la Empresa F si se obtuvieran efectos significativos al usar una técnica u otra.

- Con el método TDD, el efecto es justamente lo contrario entre las 2 empresas, en ETAPA EP el uso de TDD indica una disminución de calidad frente a YW, mientras que en la Empresa F utilizar TDD tuvo mejores efectos que YW.
- En la Empresa F los desarrolladores obtuvieron código con mejor calidad utilizando el método TDD, seguido por YW y finalmente ITL, mientras que en ETAPA EP el método que mejores resultados generó fue ITL, seguido de YW y finalmente TDD.
- Evidentemente existe alguna variable moderadora que está causando este efecto invertido entre las 2 empresas, se podría asegurar que esta variable es el lenguaje de programación utilizado para el desarrollo. Las tareas experimentales que se desarrollaron fueron exactamente las mismas en las 2 empresas y hacían referencia al desarrollo de algoritmos que permitan el comportamiento de un software juguete. Los lenguajes de programación utilizados tienen enfoques diferentes; Java, utilizado en la Empresa F es orientada a objetos, mientras que GeneXus es un Case enfocado a brindar soluciones visuales y de rápido mantenimiento de datos.
- De manera general la calidad y productividad de la Empresa F es significativamente más alta que en ETAPA EP, sin embargo, no necesariamente significa que los sujetos de la Empresa F sean mucho más productivos que ETAPA EP, ya que el lenguaje de programación que se utilizó para el tipo de tareas experimentales asignadas, fue complicado implementar en un Case del tipo q es GeneXus. Por otro lado también se puede indicar que las tareas experimentales establecidos tuvieron fines netamente académicos, estos no es lo que se realiza en el día a día.
- Se sugiere que, para una próxima réplica, el lenguaje de programación debe ser

necesariamente el mismo en los 2 experimentos, de modo que los resultados sean más comparables, caso contrario las conclusiones son menos precisas.

- Si bien el diseño se lo realizó en conjunto con el personal de España y al ser una réplica debía asemejarse lo más posible a la Empresa F, se sugiere que en un próximo estudio se podría planificar tareas experimentales más cercanas a las reales que se desarrollan en cada empresa, si bien estos son software juguetes que se utilizan justamente para experimentar con sujetos, implementar tareas del día a día obtuviera resultados más reales.

6.2.Recomendaciones para la Empresa.

Al realizar la presente investigación, se pudo comprender con mayor claridad la realidad por la que atraviesa la STI de ETAPA EP, respecto de su proceso de desarrollo de software, con lo cual, es posible plantear varias recomendaciones que permitan mejorar la calidad de los productos de software y la productividad de sus desarrolladores, optimizando de esta manera, la atención de la demanda de los requerimientos de software empresariales.

En la ejecución de los talleres experimentales, se pudo observar que los sujetos demostraron gran interés en su participación, ya que por un lado, el hecho de salir de la rutina y aprender nuevas técnicas para su trabajo diario, fue un motivador para participar con entusiasmo de las capacitaciones y desarrollo de las actividades, y por otro lado la expectativa de encontrar mecanismos que ayuden a mejorar el tiempo de respuesta a sus actividades de desarrollo.

Como se recalca a lo largo de este trabajo de tesis, actualmente en la STI no se escriben pruebas unitarias para comprobar la funcionalidad de los requerimientos de software desarrollados, no obstante, en estas sesiones de trabajo, se tuvo la oportunidad

de aprender y aplicar por primera vez pruebas unitarias al código desarrollado. Estas pruebas unitarias se llevaron a cabo a través del componente GxUnit, que se adapta directamente a la herramienta de desarrollo GeneXus, siendo un valioso aporte para el conocimiento de los desarrolladores.

Luego de las respectivas capacitaciones técnicas y ejemplos desarrollados entre los sujetos, sobre el uso del componente GxUnit para GeneXus, se pudieron evidenciar distintos comportamientos con respecto a los resultados del experimento. En el segundo día de trabajo con las tareas experimentales, se utilizó por primera vez el componente GxUnit con el método ITL, con lo cual, los sujetos obtuvieron mejores calificaciones que utilizando el método YW (método propio).

En la siguiente sesión, los sujetos nuevamente utilizaron el componente GxUnit para solventar sus tareas experimentales, pero en esta ocasión se utilizó el método de programación TDD. Los resultados referentes a calidad disminuyeron notablemente con relación a los días anteriores (ITL y YW). Se puede afirmar que este comportamiento se produjo debido a que TDD es un método de programación muy distinto al proceso que se usa en ITL o en YW, no tiene la misma secuencia de actividad y, por tanto, su uso se considera inclusive como un cambio de paradigma en el desarrollo de software.

Bajo este contexto, se recomienda el uso de pruebas unitarias en la metodología actual de programación de la STI, posterior a ello incorporar el método ITL y finalmente, con la experiencia obtenida al usar pruebas unitarias, intentar la adopción de TDD reforzando con capacitaciones sobre la técnica. Considerando adicionalmente que con el uso de pruebas unitarias se pueden obtener las siguientes ventajas:

- ✓ Se obtienen mejores resultados en cuanto a calidad de SW, ya que aportan en la comprobación del resultado esperado sobre una funcionalidad desarrollada.

- ✓ Se otorga mayor seguridad a los desarrolladores, incrementando por tanto su confianza en el código que desarrollan.
- ✓ Se brinda mayor confianza al área de soporte, ya que pueden visualizar que las pruebas se ejecuten correctamente y no de manera superficial ni subjetiva.

Con respecto al método ITL, es importante señalar que a pesar de que tuvo una mayor variabilidad con respecto al resto de métodos, existieron sujetos a los cuales su aplicación les permitió mejorar la calidad de su desarrollo. Se podría considerar reforzar con capacitaciones y talleres a los sujetos de menor puntaje para lograr un incremento de calidad a todo el equipo de desarrollo.

En lo que respecta al método propio (YW), cuyo proceso se describió en la sección 1.5.2, se recomienda incluir algunas mejoras específicamente en el paso 4, el mismo que hace referencia a la técnica de implementación para los requerimientos asignados a cada programador. Se recomienda tomar en cuenta las siguientes consideraciones detalladas como pasos a seguir:

1. Visión general del requerimiento y segmentación para solventarlo
2. Análisis de un segmento del requerimiento
3. Codificación incremental del requerimiento en la herramienta GeneXus
4. Escritura de pruebas unitarias / ejecución de las mismas
    a) Si las pruebas son satisfactorias, se continua con el paso 2
    b) Si las pruebas no fueron satisfactorias, se repite el paso 3.

b. Presentación de lo implementado al área de Operaciones de Software a través de la ejecución satisfactoria de las pruebas unitarias.

c. Presentación y aceptación de los usuarios finales.
d. Implantación del producto en un ambiente de producción con las seguridades necesarias para su uso.

Se puede afirmar que tomando estas recomendaciones en el proceso de desarrollo de software en la STI de ETAPA EP, incrementará la calidad de los productos entregados a los usuarios finales, evitando de esta manera la reprogramación de funcionalidades, problemática que se viene generando en los últimos tiempos.

El tiempo que inviertan los desarrolladores al incluir la escritura y ejecución de pruebas unitarias puede verse afectado en un inicio, sin embargo, con la experiencia que se adquiera este tiempo se reducirá significativamente y el objetivo de contar con desarrolladores de alta productividad se verá cumplido.

Al mejorar la productividad de los desarrolladores crece la confianza en su trabajo, lo cual repercute en la generación de software de alta calidad que satisfaga a los usuarios externos, todo ello aporta para mejorar el cumplimiento de la demanda que existe en la empresa.

CAPITULO 7

7.1. Referencias Bibliográficas

Beck, K. (2003). Test-driven development: by example. Addison-Wesley Professional.

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.

Araújo Pérez, A. (2008). Especificación de un marco de pruebas asociado a Genexus. con adaptación de funcionalidades de FIT.

Genero Bocco M., Lemus, J. A. C., & Velthuis, M. G. P. (2015). Métodos de investigación en ingeniería del software. Ediciones de la U.

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

Natalia Juristo, Sira Vegas, Martín Solari, Silvia Abrahão, Isabel Ramos, (2012). A process for managing interaction between experimenters to get useful similar replications.

https://sites.google.com/site/fidiproeseil (Repositorio proyecto FiDiPro, experimento de la Empresa F)

Maximilien, E. M., & Williams, L. (2003, May). Assessing test-driven development at IBM. In *Software Engineering, 2003. Proceedings. 25th International Conference on* (pp. 564-569). IEEE.

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

Geras, A., Smith, M., & Miller, J. (2004, September). A prototype empirical evaluation of test driven development. In Software Metrics, 2004. Proceedings. 10th International Symposium on (pp. 405-416). IEEE.

Madeyski, L., & Szała, Ł. (2007). The impact of test-driven development on software development productivity—an empirical study. Software Process Improvement, 200-211.

Hussan Munir, K., Petersen, K., & Moayyed, M. (2014). An Experimental Evaluation of Test Driven Development vs. Test-Last Development with Industry Professionals.

Letelier, P., & Penadés, M. C. (2006). Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP).

Figueroa, R. G., Solís, C. J., & Cabrera, A. A. (2008). Metodologías tradicionales vs. Metodologías ágiles. Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

Basili, V. R., Selby, R. W., & Hutchens, D. H. (1986). Experimentation in software engineering. IEEE Transactions on software engineering, (7), 733-743.

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.

Tosun, A., Dieste, O., Fucci, D., Vegas, S., Turhan, B., Erdogmus, H., ... &

Juristo, N. (2016). An industry experiment on the effects of test-driven development on external quality and productivity. Empirical Software Engineering, 1-43.

Almeida, E., & Araújo, A. (2008). Proyecto GxUnit. In XIV Congreso Argentino de Ciencias de la Computación.

Lindvall, M., & Rus, I. (2003). Lessons learned from implementing experience factories in software organizations. In Proc. Workshop on Learning Software Organizations (pp. 59-63).

Rodriguez, P., Markkula, J., Oivo, M., and Turula, K. (2012). Survey on agile and lean usage in finnish software industry. In Six International Symposium on Empirical Software Engineering and Measurement.

Misirli, A. T., Erdogmus, H., Juristo, N., & Dieste, O. (2014, June). Topic selection in industry experiments. In Proceedings of the 2nd International Workshop on Conducting Empirical Studies in Industry (pp. 25-30). ACM.

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

Turhan , B., Layman, L., Diep, M., Erdogmus, H., & Shull, F. (2010). How effective is test-Driven Development. Making Software: What Really Works, and Why We Believe It, 207-217.

Rafique, Y., & Mišić, V. B. (2013). The effects of test-driven development on external quality and productivity: A meta-analysis. IEEE Transactions on Software Engineering, 39(6), 835-856.

Shull, F. J., Carver, J. C., Vegas, S., & Juristo, N. (2008). The role of

replications in empirical software engineering. Empirical Software Engineering, 13(2), 211-218.

Mendonça, M. G., Maldonado, J. C., De Oliveira, M. C., Carver, J., Fabbri, S. C., Shull, F., ... & Basili, V. R. (2008, March). A framework for software engineering experimental replications. In Engineering of Complex Computer Systems, 2008. ICECCS 2008. 13th IEEE International Conference on (pp. 203-212). IEEE.

Brooks, A., Daly, J., Miller, J., Roper, M., & Wood, M. (1996). Replication of experimental results in software engineering. International Software Engineering Research Network (ISERN) Technical Report ISERN-96-10, University of Strathclyde.

Juristo, N., & Moreno, A. M. (2013). Basics of software engineering experimentation. Springer Science & Business Media.

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

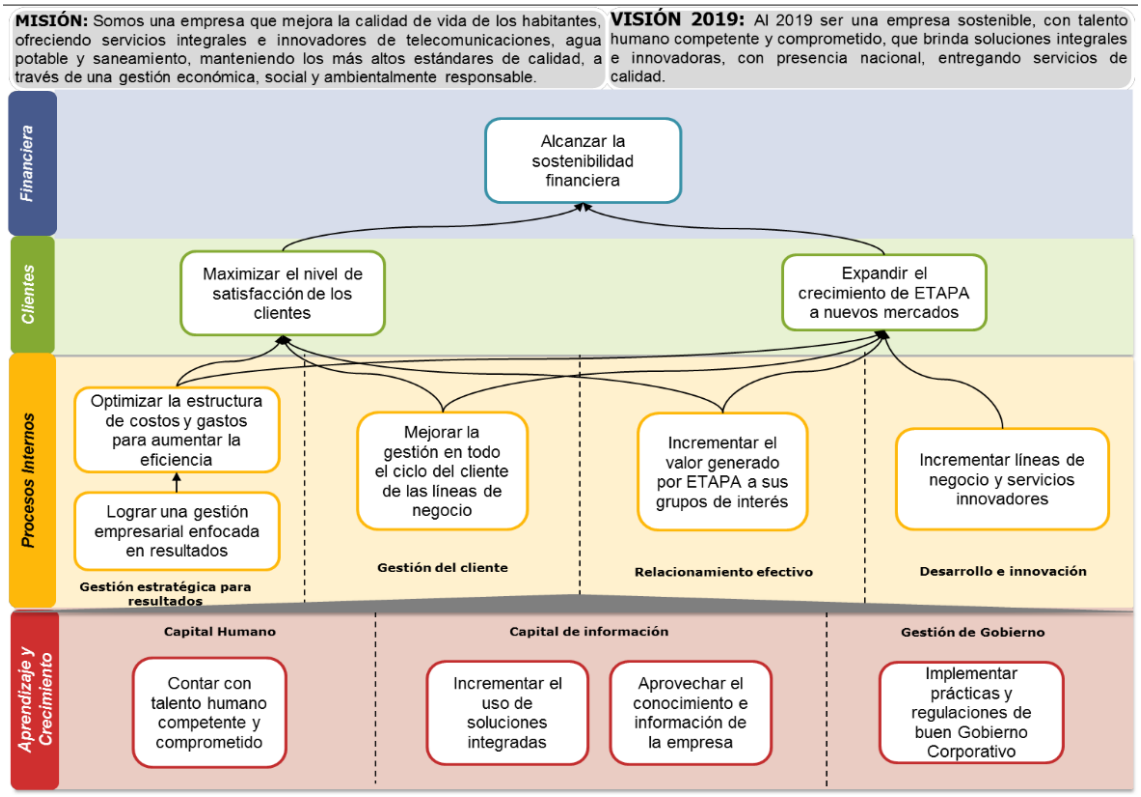
ISO/IEC 25010 (2011) Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models

ANEXOS

Anexo 1: Presupuesto del proyecto

Rubro	Unidad	Costo Unit.	Cantidad	Subtotal	Financiador
Computadora, internet, luz y demás servicios básicos.		-		45.00	Autora
Encuestas	Número	0.20	100	20.00	ETAPA EP
Laboratorio para realizar el experimento.	Laboratorio	-	1	-	ETAPA EP
Pasajes para los profesores de la Universidad de Madrid	Boletos aéreos	2.400.00	1	2.400.00	GRISE-UPM
Refrigerios para asistentes al experimento.	Refrigerios	2.50	70	175.00	Autora
Hojas de papel bond, copias, anillados y demás		-		100.00	Autora
Capacitación y Ejecución del Experimento	Horas	12.00	100	1.200.00	GRISE-UPM ETAPA EP
Horas laborables de los sujetos experimentales	Horas	12.00	117	1.400.00	ETAPA EP
TOTAL				**5.340.00**	

Anexo 2: Mapa Estratégico Corporativo de ETAPA EP

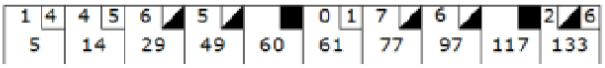


Anexo 3: Objetivos estratégicos de la perspectiva procesos internos

Procesos internos	P1	Optimizar la estructura de costos y gastos para aumentar la eficiencia.	Subgerente Financiero	Implementar una mejora continua en la estructura de costos y gastos de cada línea de negocio con información oportuna y veraz para la toma de decisiones.
	P2	Lograr una gestión empresarial enfocada en resultados.	Subgerente de Planificación	Implementar un nuevo modelo de gestión empresarial enfocado en resultados que facilite la ejecución de la estrategia a través de proyectos, procesos, talento humano, tecnología y cultura organizacional.
	P3	Mejorar la gestión en todo el ciclo del cliente en las líneas de negocio.	Gerente Comercial	Se refiere a la identificación, captación, atención y fidelización de los clientes, en base al soporte técnico que deben tener los servicios con el fin de mejorar su satisfacción.
	P4	Incrementar el valor generado por ETAPA a sus grupos de interés.	Subgerente de Comunicación	Aprovechar la identificación y priorización de los grupos de interés, con el fin de establecer acciones que entreguen valor tanto a la empresa como a los grupos de interés, sin que esto vaya en detrimento de la empresa.
	P5	Incrementar líneas de negocio y servicios innovadores.	- GAPASA - Gerente Telecomunicaciones - Subgerencia Ambiental	Potenciar los servicios actuales con nuevos desarrollos tecnológicos, generar nuevas líneas de negocio y/o servicios innovadores que creen mayor valor al cliente.

Anexo 4: Enunciado de la tarea Bowling Score Keeper

Bowling Score Keeper



The game consists of 10 frames as shown above. In each frame the player has two opportunities to knock down 10 pins. The score for the frame is the total number of pins knocked down, plus bonuses for strikes and spares.

A spare is when the player knocks down all 10 pins in two tries. The bonus for that frame is the number of pins knocked down by the next roll. So in frame 3 above, the score is 10 (the total number knocked down) plus a bonus of 5 (the number of pins knocked down on the next roll.)

A strike is when the player knocks down all 10 pins on his first try. The bonus for that frame is the value of the next two balls rolled.

In the tenth frame a player who rolls a spare or strike is allowed to roll the extra balls to complete the frame. However no more than three balls can be rolled in tenth frame.

Anexo 5: Enunciado de la tarea MarsRover API

MarsRover API

The API manages a rover that moves on a planet (/squared grid) of arbitrary size (x,y). The rover starts the movement at position (0,0). The direction of the movement can be N (north), S (south), E (east) and W (west). The rover starts facing North.

The rover receives one string of commands: l (left), r (right), f (forward) and b (backward). l and r change the rover's direction counter- and clockwise, respectively, but not the position. f and b move the rover 1 position on the grid towards the direction it is facing or away from it, respectively. The facing does not change. When the rover moves over the edges of the planet, it spawns on the opposite side.

The planet (/grid) may contain obstacles. Obstacles are defined as a list of coordinates `"(obs1X,obs1Y)(obs2X,obs2Y)…"` . When the rover finds an obstacle during a tour, it skips the current command (i.e.: does not move to the cell where the obstacle is located) and continue executing the remaining commands.

Upon processing the string of commands, the rover returns its position and facing in the format "(posX,posY,facing)". If obstacles are found, the output will be `"(posX,posY,facing)(obs1X,obs1Y)(obs2X,obs2Y)…"` The same obstacle shall be reported only once. Obstacles are reported in the order they are found.

Anexo 6: Enunciado de la tarea Spreadsheet

Spreadsheet

This exercise aims creating a basic spreadsheet. The goal **is not** to develop the GUI, but the code that implements the data structures, performs the operations and returns the results back to the GUI. The spreadsheet contains cells, organized in rows and columns (similar to MS Excel).

The cells store numbers, strings (enclosed by simple quotes '', e.g.: *'This is a string'*), or formulas (starting with the = sign). In this version, numbers shall be restricted to (positive or negative) integers.

The operations are: addition (*+*), subtraction (-), multiplication (*), integer division (*/*), module (*%*) and concatenation (*&*).

Formulas shall follow Excel conventions (e.g.: *"=1+2"*, *"=B3*(4+1)"*). To avoid tedious parsing, precedence of */% shall be explicitly specified using parentheses. Otherwise, evaluation proceeds strictly from left to right. For instance, *"=1+2*3"* is evaluated to 9 no matter the correct value is 7. Arbitrary spaces can be inserted between symbols (e.g.: *" = 1 + 2* 3 "* is valid)

Errors shall be caught during evaluation and an *#Error* or *#Circular* error message shall be returned. *#Error* is the general error message and shall be returned when operations yield wrong results (e.g.: division by zero, references to incorrect cells, empty cells, etc.). *#Circular* shall be returned when formulas make circular references (causing the recursion never end).

Anexo 7: Resultados de los desarrolladores de ETAPA EP en la tarea SS

US ID	#ASSERT	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5	Sujeto 6	Sujeto 7	Sujeto 8	Sujeto 9	Sujeto 10	Sujeto 11	Sujeto 12	Sujeto 13
0	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	-
1	1	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	-
1	2	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	-
1	3	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	-
1	5	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	-
1	6	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	-
2	1	FAIL	FAIL	FAIL	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
2	2	FAIL	FAIL	FAIL	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
2	3	FAIL	FAIL	FAIL	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
2	4	FAIL	FAIL	FAIL	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
2	5	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
3	1													-
4	1	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	PASS	FAIL	FAIL	FAIL	PASS	-
4	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
5	1	FAIL	PASS	PASS	PASS	PASS	PASS	PASS	FAIL	PASS	PASS	PASS	PASS	-
5	2	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	PASS	-
5	3	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	-
5	4	FAIL	FAIL	FAIL	ERR	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
5	5	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
5	6	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
5	7	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
5B	1	FAIL	FAIL	FAIL	ERR	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
5B	2	FAIL	FAIL	FAIL	ERR	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
5B	3	FAIL	FAIL	FAIL	ERR	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
5B	4	FAIL	FAIL	FAIL	ERR	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
6	1	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
8	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9	1	FAIL	FAIL	FAIL	ERR	PASS	PASS	PASS	FAIL	PASS	PASS	FAIL	PASS	-

9	2	FAIL	FAIL	PASS	PASS	PASS	PASS	PASS	FAIL	FAIL	FAIL	FAIL	PASS	-
9	3	FAIL	FAIL	FAIL	PASS	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	PASS	-
9	4	FAIL	ERR	FAIL	PASS	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	PASS	-
9	5	FAIL	FAIL	PASS	FAIL	PASS	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	-
9	6	FAIL	FAIL	PASS	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9	7	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	-
9B	1	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9B	2	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9B	3	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
10	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
10	2	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
11	1	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	PASS	-
11	2	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
12	1	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	-

Anexo 8: Resultados de los desarrolladores de ETAPA EP en la tarea MR

US ID	#ASSERT	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5	Sujeto 6	Sujeto 7	Sujeto 8	Sujeto 9	Sujeto 10	Sujeto 11	Sujeto 12	Sujeto 13
1	1	-	FAIL	FAIL	FAIL	PASS	PASS	PASS	FAIL	FAIL	PASS	PASS	PASS	-
2	1	-	FAIL	PASS	PASS	PASS	PASS	PASS	FAIL	FAIL	PASS	PASS	PASS	-
2	2	-	FAIL	PASS	FAIL	PASS	PASS	PASS	FAIL	FAIL	PASS	PASS	PASS	-
2	3	-	FAIL	PASS	FAIL	PASS	PASS	PASS	FAIL	FAIL	PASS	PASS	PASS	-
2	4	-	FAIL	PASS	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	-
2	5	-	FAIL	PASS	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	-
2	6	-	FAIL	PASS	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	-
2	7	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
3	1	-	FAIL	PASS	FAIL	PASS	PASS	PASS	FAIL	FAIL	FAIL	PASS	FAIL	-
3	2	-	FAIL	PASS	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	PASS	FAIL	-
3	3	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
3	4	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
4	1	-	FAIL	PASS	FAIL	PASS	PASS	FAIL	FAIL	FAIL	PASS	PASS	FAIL	-
4	2	-	FAIL	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	-
4	3	-	FAIL	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	-
4	4	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
5	1	-	FAIL	PASS	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	-
5	2	-	FAIL	PASS	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	-
5	3	-	FAIL	PASS	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	-
5	4	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
6	1	-	FAIL	PASS	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	PASS	PASS	-
6	2	-	FAIL	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	-
6	3	-	FAIL	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
6	4	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
6	5	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
6	6	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
6	7	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
6	8	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	1	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-

7	2	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	3	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	4	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	5	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	6	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	7	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	8	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	9	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	10	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	11	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	12	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	13	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	14	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
7	15	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
8	1	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
8	2	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
8	3	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
8	4	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
8	5	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
8	6	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
8	7	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
8	8	-	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9	1	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9	2	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9	3	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9	4	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9	5	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9	6	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9	7	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
9	8	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
10	1	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
10	2	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-

10	4	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
10	5	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
10	6	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
10	7	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
11	1	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
11	2	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
11	3	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
11	4	-	FAIL	FAIL	FAIL	PASS	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
11	5	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
11	6	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
11	7	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-
11	8	-	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	-

Anexo 9: Resultados de los desarrolladores de ETAPA EP en la tarea BSK

US ID	#ASSERT	Sujeto 1	Sujeto 2	Sujeto 3	Sujeto 4	Sujeto 5	Sujeto 6	Sujeto 7	Sujeto 8	Sujeto 9	Sujeto 10	Sujeto 11	Sujeto 12	Sujeto 13
1	1													
2	1													
3	1													
4	1	FAIL	PASS	PASS	FAIL	FAIL	PASS	PASS	FAIL	PASS	ERR	ERR	ERR	-
4	2	FAIL	PASS	PASS	FAIL	FAIL	PASS	PASS	FAIL	PASS	ERR	ERR	ERR	-
5	1	FAIL	PASS	PASS	FAIL	FAIL	PASS	FAIL	PASS	FAIL	ERR	ERR	ERR	-
5	2	FAIL	FAIL	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
5	3	FAIL	FAIL	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
5	4	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
5	5	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
6	1	FAIL	PASS	PASS	FAIL	FAIL	PASS	PASS	PASS	FAIL	ERR	ERR	ERR	-
6	2	FAIL	PASS	PASS	FAIL	FAIL	PASS	FAIL	PASS	FAIL	ERR	ERR	ERR	-
6	3	FAIL	PASS	PASS	FAIL	FAIL	PASS	FAIL	PASS	FAIL	ERR	ERR	ERR	-
6	4	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
6	5	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
6	6	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
7	1	FAIL	FAIL	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
7	2	FAIL	FAIL	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
7	3	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
7	4	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
7	5	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
7	6	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
7	7	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
7	8	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
8	1	FAIL	FAIL	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
8	2	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
8	3	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
8	4	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
8	5	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
9	1	FAIL	FAIL	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
9	2	FAIL	FAIL	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
9	3	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
9	4	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
9	5	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
10	1	FAIL	PASS	PASS	FAIL	FAIL	PASS	PASS	PASS	FAIL	ERR	ERR	ERR	-
10	2	FAIL	PASS	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
11	1	FAIL	PASS	PASS	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
11	2	FAIL	PASS	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	ERR	ERR	ERR	-

12	1	FAIL	PASS	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	PASS	ERR	ERR	ERR	-
12	2	FAIL	PASS	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	PASS	ERR	ERR	ERR	-
13	1	FAIL	FAIL	FAIL	FAIL	FAIL	PASS	FAIL	FAIL	FAIL	ERR	ERR	ERR	-
13	2	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	ERR	ERR	ERR	-