VICERRECTORADO DE INVESTIGACIÓN
INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DE SOFTWARE
II PROMOCIÓN

TESIS PREVIO A LA OBTENCIÓN DEL TÍTULO DE
MAGISTER EN INGENIERÍA DE SOFTWARE

TEMA: "EVALUACIÓN DE LA EFECTIVIDAD DE LAS TÉCNICAS DE PRUEBAS DE SOFTWARE ESTRUCTURALES Y FUNCIONALES MEDIANTE REPLICACIÓN EXPERIMENTAL, CASO PRÁCTICO ESPE SEDE LATACUNGA"

AUTOR: ING. EFRAÍN R. FONSECA C. MSc.

DIRECTOR: ING. GEOVANNY RAURA MIS.

LATACUNGA

2016

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

MAESTRÍA EN INGENIERÍA DE SOFTWARE

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"EVALUACIÓN DE LA EFECTIVIDAD DE LAS TÉCNICAS DE PRUEBAS DE SOFTWARE ESTRUCTURALES Y FUNCIONALES MEDIANTE REPLICACIÓN EXPERIMENTAL, CASO PRÁCTICO ESPE SEDE LATACUNGA."**, realizado por el Ing. EFRAÍN RODRIGO FONSECA CARRERA, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo que cumple con los requisitos teóricos científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor EFRAÍN RODRIGO FONSECA CARRERA para que lo sustente públicamente.

Latacunga, 23 de Mayo de 2016

Ing. Geovanny Raura MIS.
Director

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

MAESTRÍA EN INGENIERÍA DE SOFTWARE

AUTORÍA DE RESPONSABILIDAD

Yo, EFRAÍN RODRIGO FONSECA CARRERA, con cédula de identidad Nº 1710979574, declaro que este trabajo de titulación "EVALUACIÓN DE LA EFECTIVIDAD DE LAS TÉCNICAS DE PRUEBAS DE SOFTWARE ESTRUCTURALES Y FUNCIONALES MEDIANTE REPLICACIÓN EXPERIMENTAL, CASO PRÁCTICO ESPE SEDE LATACUNGA."ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 23 de Mayo de 2016

Ing. Efraín Rodrigo Fonseca Carrera MSc.
C.C.: 1710979574

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

MAESTRÍA EN INGENIERÍA DE SOFTWARE

AUTORIZACIÓN

Yo, EFRAÍN RODRIGO FONSECA CARRERA, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación "EVALUACIÓN DE LA EFECTIVIDAD DE LAS TÉCNICAS DE PRUEBAS DE SOFTWARE ESTRUCTURALES Y FUNCIONALES MEDIANTE REPLICACIÓN EXPERIMENTAL, CASO PRÁCTICO ESPE SEDE LATACUNGA.", cuyo contenido, ideas y criterios son de mi autoría y responsabilidad

Latacunga, 23 de Mayo de 2016

Ing. Efraín Rodrigo Fonseca Carrera MSc.
C.C.: 1710979574

Dedicatoria

El presente trabajo y todo el esfuerzo que representa, se lo dedico a mis hijos Rodrigo Daniel y Martín Alejandro como hito de ejemplo y superación.

Agradecimientos

Mi mayor agradecimiento para el creador, Dios nuestro Señor, que con su inmensa gloria me ha bendecido y guiado durante todo el camino, hasta llegar a la consecución de este objetivo.

A mis padres, que se han constituido como el pilar fundamental para este logro. Su amor, apoyo incondicional, ejemplo y moral han; forjado en mi los valores de humildad, responsabilidad y superación.

A mi esposa y a mis hijos, que con mucha paciencia, amor y comprensión, fueron mi principal soporte durante todo el desarrollo de este trabajo.

Un especial agradecimiento a Geovanny Raura, mi Director de tesis, amigo y compañero, cuyo apoyo coadyuvó a la consecución de este trabajo.

A la Universidad de las Fuerzas Armadas ESPE, por brindarme su confianza y financiar estos estudios.

Finalmente, a todas y cada una de las personas que de una u otra manera han hecho posible este logro.

A todos, ¡Muchas Gracias!

Índice de Contenidos

Índice de Tablas

Índice de Figuras

Resumen

La actividad de verificación y validación juega un papel fundamental en el mejoramiento de la calidad del software. La determinación de las técnicas más efectivas para llevar a cabo esta actividad han atraído desde hace ya varios años el interés de la investigación experimental en Ingeniería de software. Esta tesis reporta un experimento controlado donde se evalúa la efectividad de dos técnicas de prueba de unidad (técnica de pruebas funcional por particiones de equivalencia (EP) y la técnica de pruebas estructural de control de flujo de rama (BT)). Este experimento es una replicación literal de [Juristo et al., 2013]. Ambos experimentos tienen como propósito determinar si la efectividad de BT y EP varía dependiendo de si las faltas son visibles o no, por la técnica (InScope u OutScope, respectivamente). Se ha empleado los materiales, diseño y procedimientos del experimento original, realizando los siguientes cambios por necesidades de adaptación al contexto: (1) reducción de 3 a 2 el número de técnicas estudiadas; (2) asignación de sujetos a grupos experimentales mediante aleatorización estratificada para balancear la influencia de la experiencia en programación; (3) localización de los materiales experimentales y (4) adaptación de la duración del entrenamiento. La replicación fue realizada en la Universidad de las Fuerzas Armadas - ESPE Sede Latacunga (ESPEL), como parte del curso de Evaluación de Sistemas Software. Los sujetos experimentales fueron 23 estudiantes de maestría. EP es más efectiva que BT en la detección de faltas InScope. Se observan efectos significativos en las variables sesión/programa y grupo. BT es más efectiva que EP en la detección de faltas OutScope. No se observan efectos significativos para las variables sesión/programa y grupo en este caso. La replicación confirma los resultados del experimento original en lo referente a las técnicas de testing, pero difiere en lo referente al factor grupo. Se cree que dichas diferencias se deben al efecto de tamaño de muestra. Los resultados para el factor sesión/programa son inconsistentes para faltas InScope. Se cree que esas diferencias se deban a una combinación del efecto de fatiga y a la interacción técnica x programa. A pesar que los efectos principales han podido reproducirse, los cambios en el diseño del experimento original hacen imposible identificar con certeza las causas de las discrepancias. Se cree necesario realizar más replicaciones, similares al experimento original, para profundizar la comprensión de los fenómenos bajo estudio.

PALABRAS CLAVES:

- VERIFICACIÓN Y VALIDACIÓN DE SOFTWARE
- CALIDAD DE SOFTWARE
- EFECTIVIDAD DE LAS TÉCNICAS DE PRUEBAS DE SOFTWARE
- EXPERIMENTACIÓN EN INGENIERÍA DE SOFTWARE
- EXPERIMENTO CONTROLADO
- REPLICACIÓN

Abstract

The verification and validation activities play a fundamental role in improving software quality. Determining which is the most effective techniques for carrying out this activity has been an aspiration of experimental software engineering researchers for years. This thesis reports a controlled experiment evaluating the effectiveness of two unit testing techniques (the functional testing technique known as *equivalence partitioning* (EP) and the control-flow structural testing technique known as *branch testing* (BT)). This experiment is a literal replication of [Juristo et al., 2013]. Both experiments serve the purpose of determining whether the effectiveness of BT and EP varies depending on whether or not the faults are visible for the technique (InScope or OutScope, respectively). We have used the materials, design and procedures of the original experiment, but have made the following changes in order to adapt the experiment to the context: (1) reduced the number of studied techniques from 3 to 2; (2) assigned subjects to experimental groups by means of stratified randomization to balance the influence of programming experience; (3) localized the experimental materials and (4) adapted the training duration. We ran the replication at Universidad de las Fuerzas Armadas - ESPE Sede Latacunga (ESPEL) as part of a software verification & validation course. The experimental subjects were 23 master's degree students. EP is more effective than BT at detecting InScope faults. The session/program and group variables are found to have significant effects. BT is more effective than EP at detecting OutScope faults. The session/program and group variables have no effect in this case. The results of the replication and the original experiment are similar with respect to testing techniques. There are some inconsistencies with respect to the group factor. They can be explained by small sample effects. The results for the session/program factor are inconsistent for InScope faults. We believe that these differences are due to a combination of the fatigue effect and a technique x program interaction. Although we were able to reproduce the main effects, the changes to the design of the original experiment make it impossible to identify the causes of the discrepancies for sure. We believe that further replications closely resembling the original experiment should be conducted to improve our understanding of the phenomena under study.

KEYWORDS:

- SOFTWARE VERIFICATION AND VALIDACIÓN
- SOFTWARE QUALITY
- EFFECTIVENESS OF SOFTWARE TESTING TECHNIQUES
- EXPERIMENTAL SOFTWARE ENGINEERING
- CONTROLLED EXPERIMENT
- REPLICATION

Capítulo I

EL PROBLEMA

1.1. Antecedentes

La globalización se ha constituido como uno de los grandes fenómenos socio-económicos, caracterizada por la predominante y legendaria imposición cultural, la revolución industrial y el rápido crecimiento actual de las innovaciones tecnológicas [Camejo R, 2008]. Paralelamente, parecería ser que la globalización mantiene una estrecha relación con el consumismo como medio para sustentar tal sistema. La alta competitividad que parecería demandar la globalización en el mercado actual, al parecer incentiva la tendencia consumista del hombre, dando lugar a la adopción de un sin número de nuevas tecnologías que no constituyen en sí una necesidad vital y que además, no han sido rigurosamente probadas previa su utilización.

La revolución de la tecnología de la información, como parte de la revolución industrial, ha significado entre otras cosas, que el software llegue a se parte de más y más productos; lo que supone que una vasta cantidad de software ha sido y está siendo desarrollada [Wohlin et al., 2012].

Sin embargo, para la construcción de los productos software, la Ingeniería de Software no aplica el método científico, como en las demás ingenierías, lo que al parecer ha causado que haya un gran porcentaje de proyectos fracasados, por lo que se torna necesario un enfoque para elegir entre alternativas viables y predecir o simular el comportamiento de los procesos/productos [Juristo and Moreno, 2001]. Las propuestas sobre nuevos métodos, tecnologías, etc. que hacen las demás ramas de la ingeniería, son validados generalmente mediante la realización de *estudios empíricos*, lo que parecería ser el camino más adecuado que debería seguir la Ingeniería del Software para alcanzar la madurez de sus productos, como en las demás disciplinas.

Dado el hecho de que la cotidianidad de las personas esta tendiendo a la dependencia tecnológica, especialmente de los productos software, en algún momento todas las personas han sufrido algún error informático; ya sea en el Internet, haciendo fila en un banco o en la pérdida de todo un día del trabajo, debido a un fallo

de software. Dichos problemas nacen de la complejidad del software; la extrema dificultad para construir sistemas software, multiplica la probabilidad de persistencia de errores aún después de haberse finalizado y entregado el sistema, manifestándose cuando éste es utilizado por el cliente. Por ello parecería ser que recurrir al método científico, como se mencionó en la sección I, es la mejor alternativa para incrementar notoriamente la estadística de los proyectos exitosos de software [Juristo et al., 2006].

1.2. Tema

Los antecedentes mencionados, motivaron a realizar una ***"Evaluación de La Efectividad de Las Técnicas de Pruebas de Software Estructurales y Funcionales Mediante Replicación Experimental, Caso Práctico ESPE Sede Latacunga"***, para validar los hallazgos de un experimento perteneciente a una familia de experimentos de pruebas de software que tiene tradición, con el objetivo de coadyuvar el esfuerzo por mejorar el producto software.

1.3. Planteamiento del Problema

1.3.1. Contextualización del Problema

Para comprender el alcance del problema sobre la efectividad de las técnicas de pruebas de software, es necesario contextualizar y describirlo, en sus ámbitos de aplicación.

El avance incesante de la tecnología, coadyuvado por el consumismo del hombre, ambos muy evidentes, entre otras cosas, han dado lugar a que la mayoría de los dispositivos de uso cotidiano, vengan embebidos de elementos actualizables de software, provocando que los requerimientos por software ágil, haya crecido en forma exponencial en los últimos tiempos; como efecto de lo descrito han aparecido Metodologías Ágiles bien estructuradas de Desarrollo de Software, pero también han aparecido "técnicas ágiles de desarrollo"[1], en su mayoría basadas en la experiencia de los desarrolladores, con el propósito de liberar software en tiempos record, para poder satisfacer los requerimientos con la celeridad del caso. Este desenfrenado desarrollo "prematuro"[2] ha generado muchos problemas, que principalmente han influenciado en la calidad del producto software.

La aplicación de malas prácticas, en este caso las actividades de pruebas mal estructuradas desde las fases iniciales; sin lugar a dudas, incrementa considerablemente

[1]Las técnicas ágiles de desarrollo, se podría consideras a aquellas metodologías sin una estructura formal, fundamentalmente basadas en la experiencia.

[2]Desarrollo desenfrenado es referido a un desarrollo falto de estándares y pruebas.

el coste del producto software, ya que al corregir un error cuando se ha avanzado en el desarrollo del sistema, o peor aún cuando ya se ha implantado, cuesta mucho más que corregirlo en las primeras etapas del desarrollo; tomando en cuenta que, un alto porcentaje de los errores del software se generan en las primeras fases del ciclo de vida de desarrollo.

Uno de los factores más importantes que aseguran la calidad de un proyecto de desarrollo de sistemas es la ejecución de las pruebas adecuadas de software, mismas que se realizan una vez codificado el sistema o sus módulos, y tienen como objetivo descubrir tantos defectos como sea posible, antes de ser liberado. El presente trabajo pretende validar y profundizar hallazgos de estudios previos de efectividad de técnicas de pruebas de software más utilizadas.

1.3.2. Análisis Crítico

Una vez que se ha contextualizado es necesario adentrarse más en la esencia misma del problema, cuestionar su origen, sus causas, sus efectos y sus consecuencias; analizando críticamente la situación, a partir del árbol de problemas en base a cuestionamientos referentes al problema.

¿A qué se debe el hecho de que la efectividad de las técnicas de pruebas de software influya en la calidad final del software?

Entre las razones empíricas, se puede señalar:

- Inexistencia de criterios científicos para una selectividad adecuada de las técnicas de pruebas de software dependiendo de la situación.
- Inexistencia de las suficientes replicaciones experimentales orientadas a técnicas de pruebas de software que estructuren el marco científico para su adecuada selectividad.
- Incremento en la demanda con premura de software, mismo que es liberado sin un régimen estricto de pruebas.
- El avance incesante de la tecnología, coadyuvado por el consumismo del hombre, han dado lugar a que la mayoría de los servicios de hoy en día, vengan embebidos de elementos actualizables de software.

Este análisis crítico lleva a plantear los siguientes interrogantes:

- ¿Existen criterios científicos para una adecuada selectividad de las técnicas de pruebas de software?
- ¿La inexistencia de las suficientes replicaciones experimentales orientadas a técnicas de pruebas ha perjudicado la estructuración del marco científico para su adecuada selectividad?

- ¿Porqué se ha incrementado la demanda con premura de los productos software?
- ¿Cuál es la causa para la liberación del software sin un régimen adecuado y estricto de pruebas?
- ¿Porqué el avance incesante de la tecnología, coadyuvado por el consumismo del hombre, han dado lugar a que la mayoría de los servicios de hoy en día, vengan embebidos de elementos actualizables de software?

1.3.3. Prognosis

Para tener una visión más clara del problema, no solo es conveniente contextualizar y analizarlo en su realidad actual, sino que es necesario describir lo que podría suceder en el futuro, al no solucionar el problema de la aplicación sin mayor criterio o la falta de aplicación de pruebas de software antes de la liberación de los productos software.

El presente y futuro de los productos software, en caso de continuar las malas prácticas en la aplicación de pruebas a los productos software o peor aún la no aplicación de las mismas, degeneraría aún más la calidad y lo que es peor se perdería la credibilidad en la efectividad de la aplicación de las técnicas que demanda la Ingeniería de Software.

1.3.4. Formulación del Problema

¿Cómo Evaluar la Efectividad de las Técnicas de Pruebas de Software Estructurales y Funcionales?

1.3.5. Preguntas Directrices

- ¿Cuál es el fundamento teórico para la experimentación en Ingeniería de Software?
- ¿Cuál es la estrategia para realizar un diseño de una replicación de un experimento en Ingeniería de Software?
- ¿Cuáles son los resultados de la aplicación de una replicación experimental en Ingeniería de Software?

1.3.6. Delimitación del Problema

Delimitación de Contenido del Problema

Como se precisa de una *"replicación experimental"* para evaluar la efectividad de uno de los mecanismos más representativos para determinar la calidad del software, es un planteamiento que sin problema se posiciona en el plano de la investigación experimental en Ingeniería de Software (IS), ya que al recrear un experimento en un entorno diferente, ayudará a validar o a refutar los resultados y mejorará sin duda la aplicación de las técnicas.

Delimitación temporal del Problema

El período en el que se ubica esta investigación está circunscrito a los referentes documentales, datos e informaciones que se producirán durante el periodo académico 2010 - 2011.

Delimitación espacial del Problema

La presente investigación se realizará en la asignatura de Evaluación de Sistemas Software de la Maestría de Ingeniería de Software, que se dicta en la Universidad de las Fuerzas Armadas - ESPE Sede Latacunga (ESPEL), que se encuentra ubicada en:

Provincia	:	Cotopaxi
Cantón	:	Latacunga
Parroquia	:	Ignacio Flores
Barrio	:	La Laguna

1.4. Justificación del Proyecto

Ente las razones que justifican el proyecto, están las siguientes:

- Las buenas prácticas de la Ingeniería de Software deben sustentarse sobre la base del conocimiento científicamente demostrado, por ello se justifica el estudio del marco teórico de la Experimentación en IS y la *replicación experimental*, ya que su aplicación y resultados, sustentan el conocimiento.
- La experimentación en IS y en todas las ciencias, así como la *replicación experimental*, incrementan la veracidad del conocimiento científico, por lo que la realización del presente trabajo está plenamente justificada.
- Luego de un adecuado diseño, ejecución y análisis experimentales de una replicación, los datos obtenidos reflejarán transparentemente la validez o no de

resultados de experimentos previos; lo que enriquecerá los conocimientos y posiblemente proporcione nuevos indicios para futuras experimentaciones.

1.5. Objetivos del Proyecto

1.5.1. Objetivo General

Evaluar la *efectividad* de las *técnicas de pruebas* de software *estructurales* y *funcionales* mediante una *replicación experimental*, llevada a cabo en la Universidad de las Fuerzas Armadas - ESPE Sede Latacunga.

1.5.2. Objetivos Específicos

- Estudiar el fundamento teórico de la experimentación en IS, para sustentar la replicación de un experimento.
- Diseñar, planear y aplicar la *replicación experimental* de un experimento de técnicas de pruebas de software para evaluar la efectividad de las mismas.
- Evaluar los resultados de la aplicación de la replica del experimento de técnicas de pruebas de software y en función de ello plantear conclusiones y futuras líneas de investigación.

Una vez definida la problemática a ser abordada, a continuación se da una visión general de los capítulos restantes de la tesis.

El segundo capítulo describe la metodología de investigación aplicada y el Marco Teórico que sustenta la investigación.

A continuación, el tercer capítulo se centra en el diseño de la replicación experimental, sobre la base del experimento original. Fundamentalmente aquí se explica las diferencias entre los dos experimentos.

El cuarto capítulo detalla la ejecución de la replicación y da una idea de la complejidad que representa llegar a la ejecución de una replicación.

Luego, el quinto capítulo se centra en la parte estadística, de donde se obtienen los principales indicios para la comparación con los resultados del experimento original.

Finalmente, el sexto capítulo enumera las conclusiones a las que se ha llegado, tanto por la investigación realizada sobre las técnicas de pruebas software, como por los resultados obtenidos por la ejecución de la replicación. Po otro lado, este capítulo también plantea algunas directrices para líneas de trabajo futuras que se podrían derivar de esta investigación.

Capítulo II

METODOLOGÍA Y MARCO TEÓRICO

En el presente Capítulo se describe la metodología a ser a aplicada, analizando la aplicabilidad de métodos y técnicas; y luego, se lleva a cabo la descripción del marco teórico, en base a la red de categorías que sustenta la investigación.

2.1. Metodología

En la presente investigación, debido a que se enmarca dentro del enfoque crítico, se adoptará una metodología cuantitativa, debido a que requiere sustentar su comprobación a través de la interpretación de las diferentes fuentes y factores que intervienen en la recolección de datos e información, entre ellos los datos estadísticos. Así también, debido a que el objeto de investigación se inserta en las ciencias sociales e informáticas, como es el caso de La *efectividad de las técnicas de pruebas de software estructurales y funcionales mediante replicación experimental, aplicada a un caso práctico en la ESPEL.* A continuación, se analiza la aplicabilidad de métodos y técnicas.

2.1.1. Métodos de la Investigación

Método Deductivo

Se partirá de un argumento o de la formulación de una hipótesis, donde la conclusión se la obtiene necesariamente de las premisas y se compara con datos o información existente.

Método Inductivo

El proyecto se estructura sobre la base del conocimiento de datos de experimentos y replicaciones particulares previas, para obtener conclusiones generales.

Método Analítico

Se realizará un análisis de las caracterizaciones estadísticas, generadas a través de la aplicación de técnicas de análisis orientados al diseño de experimentos y replicaciones.

2.1.2. Técnicas e Instrumentos para Recolección de Datos

Técnica Documental

Con esta técnica se pretende investigar la base conceptual de la experimentación en IS, orientado a experimentos y replicaciones referentes a técnicas de pruebas de software.

Técnica de Encuestas

Esta técnica será utilizada para obtener datos previos a la realización del experimento, y así estructurar adecuadamente la inducción del marco de conocimiento para la aplicación de las técnicas; así como al final del experimento, para indagar a los sujetos experimentales sobre información valiosa para hacer análisis estadísticos complementarios.

Técnicas Empíricas

Existen dos tipos de paradigmas de investigación que tienen diferentes enfoques para estudios empíricos. Por un lado, la *investigación exploratoria* estudia objetos en su configuración natural, permitiendo que los hallazgos emerjan de las observaciones. Esto implica la necesidad de un diseño de investigación flexible, para adaptar los cambios observados en el fenómeno. Un diseño flexible de investigación, también es referido como una *investigación cualitativa*, ya que principalmente está alimentada por datos cualitativos. La investigación inductiva trata de interpretar un fenómeno, basada en las explicaciones de la gente que lo propone. Esto tiene que ver con el descubrimiento de las causas percibidas por los sujetos en el estudio y la comprensión de su visión del problema en cuestión [Wohlin et al., 2012].

Por otro lado, la *investigación explicativa* principalmente tiene que ver con cuantificar una relación, o comparar dos o más grupos con el objetivo de identificar una

relación causa-efecto. La investigación es a menudo llevada a cabo a través de la conformación o creación de un experimento controlado. Este tipo de estudio es una investigación de diseño fijo, lo que implica que los factores son fijados antes de que el estudio sea puesto en marcha. Una investigación de diseño fijo es también referida como una *investigación cuantitativa*, ya que en este caso su principal sustento son los datos cuantitativos. Los estudios cuantitativos son adecuados cuando se prueba el efecto de alguna manipulación o actividad. Una ventaja es que los datos cuantitativos permiten comparaciones y análisis estadísticos. Esta será la principal técnica del presente trabajo, ya que a través de ella se efectuará una *replicación experimental* con el objetivo de afianzar o refutar los resultados de un experimento base [Wohlin et al., 2012].

2.1.3. Técnicas e Instrumentos para el Procesamiento y Análisis

Para estar dentro de las exigencias de un Postgrado, la presente investigación tratará de alcanzar el tercer nivel de procesamiento, ya que utilizará:

Nivel Exploratorio

Cuando se tenga que diagnosticar la situación de la aplicación de las técnicas de pruebas en los sistemas software.

Nivel Descriptivo

Ya que se evaluará tipos específicos de pruebas, para de entre ellas y sobre la base del análisis estadístico, escoger la más efectiva, o en su defecto, exponer y concluir los resultados obtenidos.

Nivel de Asociación de Variables

Porque esta investigación tiene que comprobar una hipótesis a través de la incidencia de la variable independiente en la variable dependiente.

Nivel Explicativo

Los datos obtenidos de las encuestas y de la aplicación de la *replicación experimental*, se cotejarán con los datos referentes de la bibliografía y con los de otras replicaciones del experimento base; esto se relacionará con las causas que generan el problema; para de esta manera llegar a determinar las conclusiones y futuras líneas de investigación.

2.2. Marco Teórico

2.2.1. Antecedentes Investigativos

La alta competitividad en el mercado actual y la tendencia consumista del hombre han dado lugar a la adopción de nuevas tecnologías sin probar su utilidad práctica.

En la Ingeniería del Software no se aplica el método científico como en las demás ingenierías, esto ha causado que exista un gran porcentaje de proyectos software fracasados, por lo que se torna necesario un enfoque para elegir entre alternativas viables y predecir o simular el comportamiento de los procesos/productos, lo que se conoce como "Evidence-Based Software Engineering".

Las propuestas sobre nuevos métodos, tecnologías, etc. son validadas mediante la realización de ESTUDIOS EMPÍRICOS, de igual manera debería proceder la Ingeniería del Software para alcanzar su Madurez [Genero et al., 2012].

Considerando que la vida cotidiana de las personas esta tendiendo a la dependencia tecnológica, especialmente de los productos software; en algún momento todas las personas, han sufrido algún error informático, ya sea en el Internet, haciendo fila en un banco o en la pérdida de todo un día del trabajo, por culpa de un fallo en el software. Dichos problemas nacen de la complejidad del software. La extrema dificultad para construir sistemas software multiplica la probabilidad de que persistan errores aún después de haberse finalizado y entregado el sistema, manifestándose cuando éste es utilizado por el cliente [Juristo et al., 2006].

En definitiva, la construcción de un sistema software tiene como objetivo principal, satisfacer una necesidad planteada por un cliente. ¿Cómo se puede saber si el producto construido corresponde exactamente a lo que el cliente deseaba? y ¿Cuál es la certeza de que el producto que se ha construido funcionará correctamente? Desgraciadamente, la capacidad para medir la fiabilidad del software es muy inferior a lo que sería necesario. Lo plausible sería que los informáticos pudieran demostrar matemáticamente la corrección de sus programas, al estilo de los otros ingenieros. Los otros ingenieros recurren a análisis matemáticos para predecir cuál será el comportamiento de sus creaciones en el mundo real. Esa predicción permite descubrir defectos antes de que el producto esté operativo. Por desdicha, las matemáticas tradicionales, aptas para la descripción de sistemas físicos (los tipos de sistemas tratados por las otras ingenierías), no son aplicables al universo binario de un programa de ordenador [Juristo et al., 2006].

Es la matemática discreta, una especialidad mucho menos madura, y casi no estudiada hasta la aparición de las computadoras, la que gobierna el campo de los sistemas software. Dada la imposibilidad de aplicar métodos matemáticos rigurosos, el modo que tienen los informáticos para respaldar la confianza de los programas es la verificación empírica [Juristo et al., 2006].

2.2.2. Fundamentación Filosófica

El presente trabajo de investigación se ubica en el paradigma crítico-propositivo, ya que trata una realidad en donde se analiza la situación actual, para entonces buscar la solución al problema realizando la *replicación experimental* para reforzar el conocimiento científico y se fundamenta sobre todo en el Pensamiento Complejo, porque tiene en cuenta el criterio de la totalidad dentro de la teoría sistémica de la realidad [Fonseca C., 2009].

Este paradigma se fundamenta *ontológicamente* en la concepción objetiva de la realidad independiente de la conciencia, sujeto a leyes y en permanente cambio y movimiento; una realidad socialmente construida e interrelacionada en sistemas, dentro de una visión de relativismo científico, que conceptualiza a la ciencia en devenir, nunca acabada, en espiral ascendente abierta y progresiva, que no refleja, sino que interpreta la realidad, a través de una pluricausalidad dialéctica.

Epistemológicamente defiende que el conocimiento no es una simple información, sino una interrelación entre sujeto y objeto para lograr transformaciones, y que los conocimientos científicos van más allá de la comprobación experimental y formulación matemática, para llegar a una comprensión crítica de ciencia, como un conjunto de conocimientos destinados a la transformación social y al mejoramiento de la calidad de vida del ser humano. La investigación se realizará sobre un extracto de estudiantes de la Maestría en Ingeniería del Software de la Universidad de las Fuerzas Armadas - ESPE Sede Latacunga (ESPEL), quienes generarán la información; misma que será procesada y tabulada; y los resultados obtenidos serán utilizados para estar al tanto de la realidad, para dar a conocer a la comunidad científica, y para la investigación en sí.

Axiológicamente, esta investigación se sustenta en el compromiso por el bien común de la humanidad, en la práctica de los valores más trascendentes de la sociedad, como el de la solidaridad, la tolerancia, el respeto a las diferencias y la defensa por la identidad cultural del pueblo. Para que se lleve a cabo la investigación, existe el compromiso del investigador con la institución educativa auspiciante así como consigo mismo, sin dejar a un lado sus valores de profesionalismo y su ética. Existe además el compromiso de la universidad a través del profesor director de tesis para guiar, sugerir y apoyar al maestrante en la consecución de su trabajo, garantizando que sea de calidad.

2.2.3. Red de Categorías

Con el fin de buscar la pertinencia en la fundamentación teórica de la presente investigación, conviene estructurar una red de las principales categorías que intervienen en la explicación y comprensión científica del tema objeto de estudio; dicha red se muestra en la Figura 1.

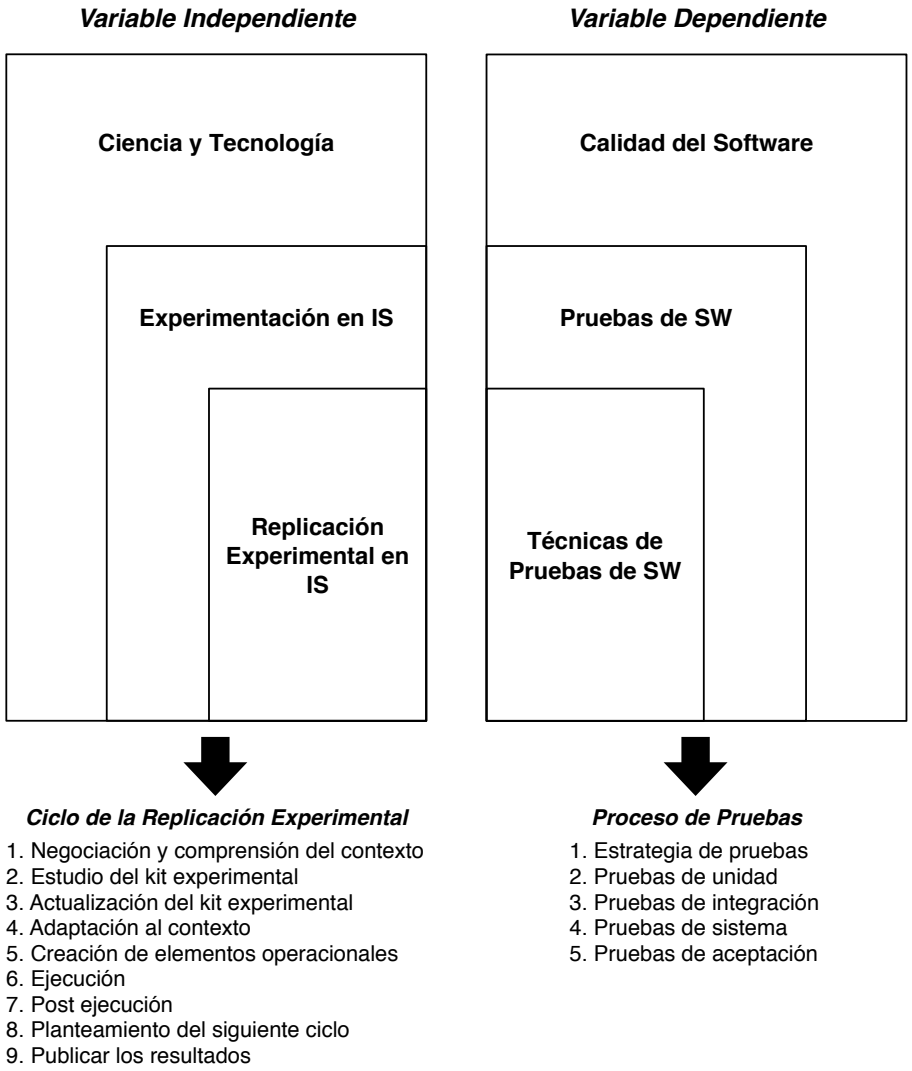

Figura 1: Red de Categorías de las Variables de Investigación

2.2.4. Fundamentación Científica de la Variable Independiente

Ciencia y Tecnología

Sin lugar a dudas el desarrollo de la humanidad ha estado condicionado al avance del conocimiento científico y tecnológico, el progreso de los pueblos se debe en gran medida al impulso que las máquinas han dado al trabajo del hombre, lo cual se ha podido apreciar en diferentes periodos de la historia [Fonseca C., 2009].

La evolución de la ciencia y la tecnología se ha caracterizado por la creciente renovación de sus conceptos; en la antigüedad y en la Edad Media la tecnología se manifestaba en la construcción de enormes y eficientes obras militares y civiles como las fortalezas y los acueductos romanos, las catedrales góticas, como es el caso de

Notre Dame en París y Colonia en Alemania, las materias primas utilizadas eran la arcilla, la piedra, madera y el hierro, esta era una tecnología artesanal [Fonseca C., 2009].

Con la Revolución Industrial, la tecnología tuvo expresiones mayores al hacer uso de la energía con las máquinas de vapor y las máquinas eléctricas, esta etapa remarcó una tecnología dinámica que tuvo énfasis en los modos de producción y en la transportación terrestre y marítima [Fonseca C., 2009].

A mediados del siglo XX, la tecnología tuvo un impresionante desarrollo inesperado, impulsado lastimosamente por uno de los sucesos más impactantes que ha sufrido la humanidad, la Segunda Guerra Mundial, es en ese momento en que la tecnología alcanza máximas expresiones en instrumentos para la Guerra, como fueron los aviones de combate y el surgimiento de la electrónica, que inició con los tubos o lámparas de vacío e impulsada luego con los transistores (1948) y los chips (1959). La electrónica es una fina técnica que permite generar y conmutar señales eléctricas viabilizadas por débiles corrientes de electrones [Fonseca C., 2009].

Con el advenimiento de la electrónica y el desarrollo acelerado de las computadoras y las comunicaciones en la segunda mitad del siglo XX, la tecnología ingresa en la era de la información, ya que la sociedad puede acceder a casi todo tipo de información a través de la red, los diversos grupos humanos pueden comunicarse entre sí y conocer qué está ocurriendo en localidades tan apartadas y con culturas diferentes a la suya; es la tecnología y su utilización la que lleva a considerar a la sociedad actual como una sociedad globalizada [Fonseca C., 2009].

Experimentación en Ingeniería de Software

La investigación es una actividad realizada de manera voluntaria y consciente por la humanidad, en busca del conocimiento indiscutible sobre una cuestión determinada, es decir, sacar a la luz un lote de conocimientos que se desconocía. Sin embargo, la meta de un investigador no siempre es sólo ampliar los conocimientos. A menudo, los investigadores tratarán de obtener ciertos conocimientos para cumplir un fin particular, prácticas de interés tecnológico, social o económico [Juristo and Moreno, 2001].

En un campo más específico, como en "La naturaleza de la Ingeniería"Rogers (1983) describe al objetivo de la investigación tecnológica como: La esencia de la investigación tecnológica, se orienta hacia el servicio del proceso de diseño y construcción de cosas particulares, cuya finalidad ha sido claramente definida. Es posible que se desee diseñar un puente que use menos material, construir una presa más segura, mejorar la eficiencia en una central eléctrica, viajar más rápido en los ferrocarriles, y así sucesivamente. La investigación tecnológica es, en este sentido, más prescrita que una investigación científica. También es más limitada, ya que puede determinar cuando se ha llegado a una solución adecuada de un problema técnico [Juristo and Moreno, 2001].

Para un grupo de conocimientos que se consideran científicos, su verdad y validez debe ser probada. Un elemento particular de conocimiento es considerado como científicamente válido si se ha comprobado con la realidad. El progreso científico se basa en el estudio y solución de las discrepancias entre el conocimiento y la realidad. La investigación científica es la antítesis de la opinión. Idealmente, los investigadores no opinan, ellos explican los resultados obtenidos; sus estudios no se basan en factores subjetivos, como emociones, opiniones o gustos. Las investigaciones científicas son estudios objetivos, basados en observaciones de la experimentación o del mundo real y sus cambios mensurables [Juristo and Moreno, 2001].

Tradicionalmente, la investigación científica fue definida como la investigación en busca de conocimiento sobre el universo físico, a diferencia de la investigación filosófica, histórica y literaria. En la actualidad, el método científico es omnipresente en todas las disciplinas. En un principio, la investigación científica se basaba en la observación de la naturaleza, el mundo y el universo, sin modificar nada de la naturaleza. Ahora, una investigación más científica se basa en la experimentación, es decir, en la observación de los fenómenos provocados con fines de investigación (mediante la modificación de la realidad) y en la medición de las variables que intervienen en los fenómenos. La antigua visión asumida por el método científico que era aplicable sólo a las ciencias naturales (física, química, biología); ahora ya es obsoleta [Juristo and Moreno, 2001].

La investigación es un proceso de aprendizaje dirigido, y consta de tres modos de razonamiento para llegar a una hipótesis, que son: deducción, inducción y abducción. La deducción demuestra que algo debe ser, la inducción muestra que algo es realmente operativo y la abducción se limita a lo que sugiere que algo podría ser [Juristo and Moreno, 2001]. Así, el aprendizaje se desarrolla por iteración. Una hipótesis inicial (idea, modelo ó conjetura) conduce por un modelo de *deducción*, a ciertas consecuencias necesarias que se pueden comparar con los datos. Cuando las conclusiones y los datos no coinciden, las discrepancias pueden llevar por un proceso de *inducción* a la variación del modelo. Se inicia, por lo tanto, un segundo ciclo de iteración. Se deducen las consecuencias del modelo modificado y nuevamente se comparan con los datos previos ó, con los nuevos, que a su vez pueden llevar a nuevas modificaciones y por lo tanto a un aumento del conocimiento. El proceso de adquisición de datos puede ser un experimento científico, pero también puede ser el producto de una visita a la biblioteca o al Internet o por medio de la observación [Box et al., 2008].

Desafortunadamente, se debe admitir que las ideas no se comparan con la realidad en al ámbito de la IS, con la frecuencia que sería necesaria para asegurar la validez de los modelos, procesos, métodos y técnicas que constantemente han sido propuestos y utilizados en la construcción de software. En IS aún se trabaja en el campo de la subjetividad, la opinión y la especulación, o como mucho, en el ámbito de los estados en disputa [Juristo and Moreno, 2001].

Como en cualquier área madura, en la IS hay la necesidad de entender sus compo-

nentes y sus relaciones. Un proceso experimental proporciona la base necesaria para mejorar el conocimiento y la comprensión. Dado que la IS está en su adolescencia, es ciertamente un candidato para el método experimental de análisis. La experimentación es llevada a cabo con el propósito de evaluar, predecir, entender, controlar y mejorar el proceso de desarrollo del producto software [Basili et al., 1985].

La experimentación en ingeniería de software, al igual que cualquier otro procedimiento experimental, involucra una iteración de una hipótesis y un proceso de pruebas. Los modelos del proceso o producto software son construidos, las hipótesis acerca de estos modelos son probadas y la información aprendida es utilizada para refinar las hipótesis antiguas y desarrollar otras nuevas. En una área como la IS, este enfoque toma una especial importancia porque es necesario en gran medida mejorar el conocimiento de cómo se desarrolla el software, el efecto de diversas tecnologías, y qué áreas necesitan mejorar más. Hay mucho que aprender y la intuición no es siempre la mejor guía [Basili et al., 1985].

La evolución de la tecnología informática ha provocado, entre otras cosas, que el software llegue a estar embebido, cada vez más, en los productos de uso cotidiano. Este ímpetu en la producción ha ocasionado paralelamente que los productos software presenten problemas de: funcionalidad, costos, tiempos de entrega y calidad. La Ingeniería de Software (IS) fue concebida justamente para mejorar el desarrollo de los sistemas software [Wohlin et al., 2000].

Sin embargo, y a diferencia de las otras disciplinas de ingeniería que basan la construcción de sus artefactos en el uso de un conocimiento maduro, por medio del cual pueden obtener resultados previsibles, el tipo de conocimiento con el que opera la IS puede ser considerado inmaduro [Juristo and Moreno, 2001] [Wohlin et al., 2000]. En IS, los desarrolladores son guiados por un razonamiento basado en su intuición, la moda o en el rumor del mercado, en lugar de por hechos o afirmaciones indiscutibles propios de una disciplina de ingeniería [Juristo et al., 2002]. No existe evidencia alguna que apoye la mayoría de las creencias sobre las que se basa la construcción de software [Juristo and Moreno, 2002]. Actualmente, se desconoce la adecuación, límites, cualidades, costes y riesgos de las tecnologías que se emplean en el desarrollo de software [Juristo et al., 2002]. El conocimiento inmaduro utilizado en la IS pudiera ser que haya influenciado en el fracaso de algunos proyectos, por lo que se torna necesario un enfoque para elegir entre alternativas viables y predecir o simular su comportamiento [Juristo and Moreno, 2001].

En la historia de todas las disciplinas ingenieriles se ha producido una transición desde las creencias, especulaciones y aciertos casuales hasta el conocimiento científico mediante el cual una ingeniería alcanza resultados predecibles [Shaw, 1990]. El proceso experimental proporciona la base necesaria para contrastar las creencias y las opiniones con la realidad, convirtiéndolas de este modo en conocimiento científico o desechándolas definitivamente [Latour et al., 1986]. Dado el bajo grado de madurez de la IS, ésta es sin duda una candidata ideal para la aplicación del método experimental. La Ingeniería del Software Experimental (ISE) traslada a la IS el paradigma

experimental que ha sido aplicado con éxito no ya en disciplinas científicas clásicas (Física, Química, Medicina, etc.) sino que más recientemente ha permitido avanzar en disciplinas tradicionalmente no experimentales, tales como la Economía [Davis and Holt, 1992] o la Sociología [Greenwood, 1976]. En la IS, los estudios empíricos y experimentales deberían guiar la construcción de nuevos métodos, técnicas, lenguajes y herramientas; para ser sugeridos, publicados o publicitados. Además, es crucial para evaluar nuevas propuestas y compararlas con las existentes [Wohlin et al., 2000].

Un experimento es una investigación formal, rigurosa y controlada donde un conjunto de variables bajo estudio (variables independientes o factores) toman distintos valores o niveles, investigándose los efectos (variables dependientes o respuesta) que produce cada nivel [Juristo and Moreno, 2001]. Ejemplos de factores en experimentos de IS son: métodos de diseño, técnicas de pruebas, experiencia de los desarrolladores, etc. Ejemplos de efectos estudiados en experimentos en IS son: calidad, eficacia, eficiencia, productividad.

Antes que los resultados obtenidos mediante un experimento se consideren como hechos, es necesario proporcionar a la comunidad todos los detalles del experimento realizado, lo que permitirá a otros investigadores repetir los experimentos y comprobar que los resultados se reproducen de forma consistente [Juristo and Moreno, 2001]. La repetición de un experimento por otros investigadores se denomina replicación experimental.

Replicación Experimental en Ingeniería de Software

Otro aspecto importante a ser considerado, es la posibilidad de replicar una investigación. El objetivo de una *replicación experimental* es mostrar que los resultados de un experimento original son válidos para una población más grande. Una replicación puede llegar a ser válida, si repite tanto el diseño, como los resultados del experimento original. No es raro que el objetivo de una investigación sea llevar a cabo una replicación, pero los resultados, de cierta manera, podrían resultar diferentes que los resultados del estudio original [Wohlin et al., 2012].

La realización de replicaciones en IS es complicada debido al estado de inmadurez en que todavía se encuentra el paradigma experimental aplicado al desarrollo de software. En la actualidad, cualquier replicación puede producir resultados experimentales distintos y difíciles de conciliar con el experimento original. Esto se debe a que en IS se desconoce todavía qué aspectos (como, por ejemplo: tipología de sujetos y tareas, características de los problemas, etc.) afectan a las distintas tecnologías [Juristo and Moreno, 2001]. Estos aspectos son lo que se conoce como variables moderadoras ó contextuales. Por ello, cualquier cambio que, a propósito o inadvertidamente, se introduzca en una replicación, puede llevar a que los resultados del experimento original sean imposibles de replicar [Fonseca C., 2012].

Dado que las replicaciones experimentales pueden arrojar resultados contradicto-

rios, es evidente que los experimentos aislados proporcionan únicamente resultados parciales. No es posible sacar conclusiones generales a partir de uno o unos pocos experimentos; es necesario consolidarlo en base a múltiples replicaciones y, posteriormente, combinar los resultados obtenidos [Fonseca C., 2012].

La replicación es un mecanismo clave en el paradigma experimental, ya que permite la verificación y validación de resultados encontrados en experimentos previos [Gómez et al., 2010].

No existe una clasificación estándar de los tipos de replicaciones [Gómez et al., 2010], distintos autores proponen varias clasificaciones, de las cuales se adoptó la propuesta hecha por [Gómez, 2012], la que se indica a continuación.

1. **Replicaciones por semejanza**

   Dependiendo de la semejanza que hay en los elementos de la situación experimental y del experimento base, proponen los siguientes tipos de replicación:

   a) **Replicación Exacta:** Se realiza una copia idéntica del experimento base. Sin embargo esto no es posible en el estado actual de la Ingeniería de Software Experimental (ISE) pues debido al desconocimiento de las variables relevantes del contexto siempre existen condiciones que cambian intencionada o inadvertidamente entre la replicación y el experimento base. En algunas áreas de las ciencias naturales es posible llegar a una aproximación bastante cercana de este tipo de replicación ya que se tiene un mayor control sobre las condiciones. Por ejemplo, en estas áreas se trabaja en laboratorios estériles, libres de variables extrañas que puedan afectar los resultados, es posible utilizar cantidades exactas de compuestos o materiales, así como utilizar ingredientes o componentes que no varían con el tiempo.

   b) **Replicación Literal:** Esta replicación es el equivalente en Ingeniería de Software Experimental a la replicación exacta. Se intenta realizar la replicación lo más exacta posible al experimento base. Los elementos de la situación experimental se mantienen igual, es decir, la replicación se realiza siguiendo el mismo protocolo, con las mismas operacionalizaciones y poblaciones equivalentes a las del experimento base. En esta replicación no se realizan cambios deliberados en los elementos de la situación experimental.

   c) **Replicación Artificial:** Se varían elementos del protocolo experimental con el fin de verificar que los resultados observados se reproducen con protocolos experimentales equivalentes. Es decir, los resultados no son artificiales.

   d) **Replicación Operacional:** Se varían las operacionalizaciones de causa y/o efecto con el fin de verificar hasta qué límites de las operacionalizaciones de los constructos de causa y/o efecto se mantienen los resultados.

En otras palabras, se estudia la sensibilidad de los resultados con respecto a las operacionalizaciones.

e) **Replicación Poblacional:** Se varían las poblaciones para verificar los límites de las poblaciones usadas en el experimento base. En otras palabras, se estudian las propiedades de las poblaciones que influyen en los resultados.

f) **Replicación Conceptual:** En esta replicación el investigador elabora su propio protocolo experimental y operacionalizaciones para verificar los resultados observados en el experimento base.

Las replicaciones artificiales, operacionales, poblaciones y conceptuales pueden identificarse como replicaciones **diferenciadas** con la finalidad de usar un nombre contrapuesto a la replicación literal para denotar una replicación que ha sufrido cambios respecto al experimento base.

2. **Replicaciones por Conexión entre Experimentadores**

Según el nivel de conexión entre los investigadores que realizaron el experimento base y los que realizan la replicación, propone los siguientes tipos:

a) **Replicación Nativa:** Si se realiza por los mismos investigadores que participaron en el experi- mento base.

b) **Replicación Conjunta:** Parte de los investigadores que participaron en el experimento base participan en la replicación.

c) **Replicación Ajena:** Se realiza por investigadores ajenos a quienes participaron en el experimento base.

3. **Replicaciones por lugar**

De acuerdo al lugar donde se realiza, los tipos de replicación que propone son:

a) **Replicación Interna:** Si se realiza en el mismo sitio donde se efectuó el experimento base.

b) **Replicación Externa:** Si se realiza en otro sitio distinto al del experimento base.

La replicación literal-nativa-interna puede denominarse también como **repetición**. De manera similar la replicación conceptual-ajena-externa puede denominarse también como **reproducción**.

Previo al inicio de la descripción del ciclo de una *replicación experimental*, es imprescindible definir algunos términos para un mejor entendimiento del ciclo experimental que atañe a la replicación.

1. **Unidad Experimental:** Los objetos en los que un experimento se ejecuta se llaman unidades experimentales u objetos de experimentación. Por ejemplo, los pacientes son las unidades experimentales en experimentos médicos,

al igual que cada pedazo de tierra en los experimentos agrícolas. Los experimentos en la IS implicarían someter el desarrollo del proyecto o de una parte específica del proceso de desarrollo sobre la base de determinadas condiciones y, a continuación recoger un conjunto de datos particular para el análisis. Ahora, suponiendo que el objetivo es comparar la precisión de tres técnicas de estimación, la unidad experimental serían los requisitos a los que las técnicas se aplican. Si se desearía comparar dos técnicas de prueba, la unidad experimental sería el trozo de código en el que las técnicas se aplican. Así, la unidad experimental sería un proceso o subproceso en el primer ejemplo, mientras que sería un producto en los dos últimos [Box et al., 2008].

2. **Sujetos Experimentales:** Los sujetos experimentales, son las personas que aplican los métodos o técnicas a las unidades experimentales. En el ejemplo de mejora de procesos citado en la definición de unidad experimental, el sujeto experimental sería todo el equipo de desarrolladores. En el ejemplo de estimación, los sujetos serían los estimadores que se aplican las técnicas de estimación; y, en el ejemplo de las pruebas de software, sería la gente que aplica las técnicas de prueba. A diferencia de otras disciplinas, en la IS el sujeto experimental tiene un efecto muy importante sobre los resultados de los experimentos y, por tanto, esta variable tiene que ser cuidadosamente considerada durante el diseño del experimento. Suponer, por ejemplo, que se tiene un experimento destinado a determinar en la agronomía, cual es el mejor fertilizante para el crecimiento de una semilla. Los sujetos experimentales de este experimento sería la gente que aplican los diferentes fertilizantes (variables experimento) en la misma semilla sembrada en un pedazo de tierra (unidad experimental). La acción de los diferentes sujetos no se espera que afecte tanto en el crecimiento de la semilla en este experimento; es decir la manera en que cada sujeto aplica el abono, es poco probable que difiera mucho. En el experimento sobre las técnicas de estimación en la IS, los sujetos del experimento serían ingenieros de software que aplican las tres técnicas de estimación de las necesidades particulares (unidad experimental). Como las técnicas de estimación no son independientes de las características del estimador que las aplica, el resultado puede variar mucho dependiendo del sujetos que aplica las técnicas. Del mismo modo, el resultado de la aplicación de la mayoría de las técnicas y procedimientos aplicados en la IS pasa a depender de quién las aplica. Por lo tanto, el papel de los sujetos en los experimentos en IS debe ser cuidadosamente tratado en el diseño del experimento [Box et al., 2008, Wohlin et al., 2012, Juristo and Moreno, 2001].

   En particular, si se desea llevar a cabo experimentos en los que no se tiene la intención de estudiar la influencia de los sujetos, será una buena idea seleccionar un diseño que anula la variabilidad implícita en el uso de diferentes desarrolladores.

3. **Perfiles involucrados en la Replicación Experimental:** Dada la relación de colaboración con un grupo de investigación, se ha encontrado que en un proceso experimental en IS pueden intervenir varios experimentadores, cada

uno de los cuales cumple uno o varios roles; entendiéndose por rol, a la función o grupo de funciones que una persona desempeña en un lugar o en una situación. La asignación de los roles son generalmente en base a las habilidades más representativas de los experimentadores, afines a la experimentación. Los perfiles más representativos identificados y que están involucrados con la *replicación experimental* son: Gestor de la investigación (GI), gestor del experimento (GE) y experimentador senior (ExSe).

a) **Gestor de la Investigación (GI):** Experimentador que conoce los objetivos de la investigación, versiones, numero de replicaciones y puede decidir sobre nuevas adaptaciones del experimento.

b) **Gestor del Experimento (GE):** Experimentador que es el encargado de administrar el experimento en todo el proceso experimental, la documentación previa, documentación de ejecución y documentación posterior al experimento.

c) **Experimentador Senior (ExSe):** Es la persona que ha ejecutado más de una vez el experimento, empieza a tener otros requerimientos y realiza modificaciones en el diseño del experimento.

4. **Variable Respuesta:** El resultado de un experimento se conoce como variable respuesta. Este resultado debe ser cuantitativo. La variable respuesta de un experimento en IS es el proyecto, la fase, el producto o el recurso típico que se mide para probar los efectos de las variaciones provocadas de un experimento a otro. Por ejemplo, suponer que un investigador propone una técnica de estimación de proyectos nuevos y sostiene que la técnica proporciona una mejor estimación de las técnicas existentes. El investigador debe realizar un experimento con varios proyectos, algunos utilizando la nueva técnica y otros con las técnicas existentes (el diseño experimental sería una ayuda para decidir cuántos proyectos se requiere para cada técnica). Una variable respuesta posible en estos experimentos sería la precisión de la estimación. Esta variable respuesta del ejemplo, la precisión, se puede medir usando diferentes métricas. Por ejemplo, se podría decidir medir la precisión en este experimento como la diferencia entre la estimación y el valor real. Sin embargo, si el investigador afirma que el nuevo método reduce los tiempos de desarrollo, la variable respuesta del experimento sería el tiempo de desarrollo. Por lo tanto, la variable respuesta es la característica del proyecto de software bajo análisis y que suele ser mejorada. Cada valor de la variable de respuesta reunido en un experimento se denomina observación y el análisis de todas las observaciones debería decidir si la hipótesis a probar puede ser validada [Box et al., 2008, Wohlin et al., 2012, Juristo and Moreno, 2001].

   La variable respuesta a en ocasiones es llamada variable dependiente. Este término no proviene del campo del diseño experimental, sino de otra rama de las matemáticas. El objetivo de la experimentación suele ser encontrar una función que relaciona la variable de respuesta con los factores que influyen

en ella. Por lo tanto, aunque el termino variable dependiente no es apropiado para diseño experimental, a veces se utiliza [Box et al., 2008, Wohlin et al., 2012, Juristo and Moreno, 2001].

5. **Parámetros:** Cualquier característica (cualitativa o cuantitativa) del proyecto software que será una constante, en la experimentación se denomina parámetro. Estos son, por tanto, características que no se desean investigar o que no se desea que influyan en el resultado del experimento o, alternativamente, la variable respuesta. Hay otras características del proyecto en el ejemplo de la técnica de estimación que pueden influir en la precisión de la estimación: la experiencia del director del proyecto que hace la estimación, la complejidad del sistema de software en fase de desarrollo, etc. Si se tiene la intención de analizar sólo la influencia de la técnica de la precisión de la estimación, las demás características tendrán que permanecer sin cambios de un experimento a otro (el mismo nivel de experiencia, la complejidad misma del desarrollo, etc.). Los parámetros tienen que ser fijados por la similitud y no por la identidad. Por lo tanto, los resultados de la experimentación serán particulares a las condiciones definidas por los parámetros. En otras palabras, los hechos o los conocimientos generados por la experimentación serán ciertos a nivel local de las condiciones reflejadas en los parámetros. La producción de conocimiento sólo puede ser generalizada al considerar los parámetros como variables en los experimentos sucesivos y estudiar su impacto en la variable de respuesta [Box et al., 2008, Wohlin et al., 2012, Juristo and Moreno, 2001].

6. **Factores:** Cada desarrollo típico de software que se estudiará y que afecta a la variable de respuesta se llama factor. Cada factor tiene varias alternativas posibles. La experimentación tiene como objetivo examinar la influencia de estas alternativas en el valor de la variable de respuesta. Por lo tanto, los factores de un experimento son todas las características del proyecto que son intencionalmente variadas durante la experimentación y que afectan el resultado del experimento. Tomando el ejemplo de la técnica de estimación, la técnica es en realidad el factor y sus posibles alternativas: la nueva técnica, COCOMO, el método de Putnam, etc [Box et al., 2008, Wohlin et al., 2012, Juristo and Moreno, 2001].

   Los factores también se llaman predictores, ya que son las características del experimento para predecir lo que sucedería con la variable respuesta. Otro término, tomado de las matemáticas y que se utiliza para los factores, son las variables independientes.

7. **Niveles:** Los valores posibles de los factores en cada experimento primario se llaman niveles. Esto significa que cada nivel de un factor es una alternativa para ese factor. En el ejemplo, las alternativas serían: la nueva técnica, el método COCOMO y Putnam, es decir, las alternativas utilizadas para la comparación.

   El termino tratamiento a menudo se utiliza para este concepto de alternativas de un factor en el diseño experimental. Este término se remonta a los orígenes

del diseño experimental, que fue concebido principalmente pensando en la experimentación agrícola. Los factores en este tipo de estudios usados para ser los insecticidas para las plantas o fertilizantes para la tierra, para los cuales el término tratamiento a largo plazo es muy apropiado. El tratamiento también es correcto en experimentos médicos y farmacológicos. Algo similar puede decirse para el nivel de expresión, que es muy apropiado para referirse al examen de las diferentes concentraciones de productos químicos, por ejemplo. El tratamiento de los términos y el nivel en la IS, sin embargo, puede ser apropiado en algunas ocasiones y no en otras. Las alternativas de los factores de los experimentos abordados, como el método COCOMO o Putnam, por ejemplo, son de orden cualitativo, como se mencionó anteriormente; sin embargo, hay que recordar que las variables respuesta reunidas en estos experimentos son cuantitativas. El objetivo de estos experimentos es determinar el efecto cuantitativo de una alternativa [Juristo and Moreno, 2001].

8. **Diseño Experimental:** El diseño de un experimento describe como está organizada la aplicación de los tratamientos por parte de los sujetos experimentales, distribuidos aleatoriamente (o no) en grupos y en periodos de tiempo (sesiones), dependiendo del número de tratamientos por grupo y por periodo.

Una vez definidos los términos más relevantes que se enmarcan dentro del ámbito de un experimento, se describe a continuación parte del ciclo de investigación experimental en IS; más específicamente, lo tocante a las replicaciones experimentales. En la Figura 2 se muestra el ciclo de una *replicación experimental*.

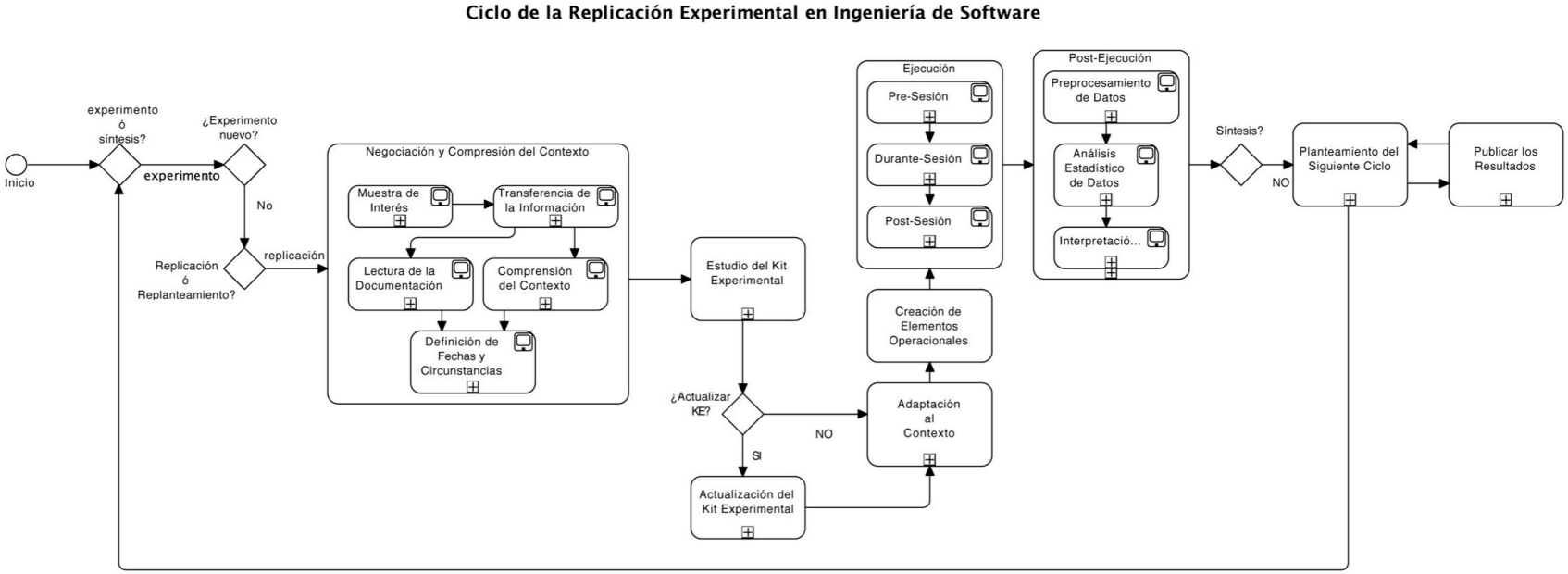

Figura 2: Workflow del Proceso de Replicación Experimental en ISo

1. **Negociación y Comprensión del Contexto**
   La fase de negociación y compresión del contexto, permite la coordinación de todos los aspectos y sujetos participantes, para la puesta en marcha de una replicación experimental; dentro de esta fase, se pueden especificar las siguientes subfases:

   a) **Muestra de Interés:** La muestra de interés inicia cuando un investigador está interesado en colaborar con la Investigación Experimental y a hecho contacto con un grupo de investigación, dando a notar su interés por replicar.

   b) **Transferencia de la Información:** Una vez validado el interés del investigador, el grupo de investigación entrega el material inicial necesario, para que el investigador interesado en replicar comprenda el experimento a replicar.

   c) **Comprensión del Contexto:** La comprensión del contexto es una subfase conjunta entre el investigador y el grupo de investigación, llevada a cabo para entender el contexto donde se ejecutará el experimento, con el fin de adaptarlo.

   d) **Lectura de la Documentación:** La lectura de la documentación es una subfase en la cual el investigador procede con la lectura del material proporcionado por el grupo de investigación para entender el experimento, en especial todo lo que requiera demanda de recursos.

   e) **Definición de Fechas y Circunstancias :**La subfase de definición de fechas y circunstancias es en donde se decide sobre que circunstancias, lugar, fechas, etc. sé llevará a cabo el experimento.

2. **Estudio del Kit Experimental**
   En la fase de estudio del kit experimental, el investigador interesado en replicar estudia el contenido del kit, para comprender a profundidad el experimento así como su protocolo de operación; entendiéndose por kit experimental, al conjunto de artefactos a través de los cuales es viable la ejecución experimental. Más en detalle, el kit experimental está compuesto por el kit de ejecución, el material guía del experimentador y el material de entrenamiento. A su vez, El kit de ejecución está compuesto por: instrumentos, objetos y guías experimentales utilizadas en el proceso de ejecución propiamente dicho.

   Dentro de la fase de estudio del kit experimental, se pueden identificar las siguientes subfases:

   a) **Estudio del Material Guía del Experimentador:** El Estudio del Material Guía del Experimentador es fundamental y consiste en examinar a detalle todo el material guía de ejecución experimental, como por ejemplo: Guía para crear el kit de materiales, guía para entregar el kit de materiales, guía para recolección y limpieza de datos, etc.

b) **Estudio del Material de Entrenamiento:** El estudio del material de entrenamiento consiste en examinar a detalle todas las guías teóricas (es decir, el material del capacitador para dictar la teoría) y prácticas (es decir: los ejercicios para ser desarrollados) necesarias para el entrenamiento en los tratamientos planificados para la ejecución experimental y su aplicación sobre los objetos experimentales.

c) **Estudio de los Instrumentos Experimentales:** El estudio de los instrumentos experimentales consiste en examinar a detalle los instrumentos destinados a la recolección de los resultados de la aplicación de los tratamientos sobre los objetos experimentales.

d) **Estudio de los Objetos Experimentales:** El estudio de los objetos experimentales consiste en examinar a detalle los artefactos que han sido utilizados como objetos experimentales, en la ejecución del experimento base.

e) **Estudio de los Materiales guía:** Finalmente y no menos importante, es el estudio a detalle de los materiales que contienen las instrucciones para aplicar los tratamientos al objeto experimental, durante la ejecución del experimento, con el objetivo de estandarizar el protocolo de aplicación y homogeneizar la forma de obtención de los resultados de la aplicación de los tratamientos.

3. **Actualización del Kit Experimental**
   La fase de actualización del kit experimental consiste en realizar alguna modificación en uno o varios de los artefactos que componen el kit experimental, debido a que posiblemente no sean adecuados para el contexto donde se va a llevar a cabo la replicación; y por lo tanto, requieren ser actualizados. Por ejemplo, en contextos donde el idioma o el léxico sean diferentes.

4. **Adaptación al Contexto**
   La adaptación al contexto es una fase en la que un experto del grupo de investigación presta su contingente para dar soporte al investigador interesado en replicar, en la adaptación del experimento de acuerdo al contexto, tratando de mantener en lo posible idénticas las condiciones a las del experimento original o de acuerdo a la hipótesis planteada. Este soporte puede ir desde que el experto haga toda la adaptación; hasta que, únicamente responda las inquietudes planteadas por el investigador interesado.

5. **Creación de Elementos Operacionales**
   En la fase de creación de elementos operacionales, se crean todos los elementos necesarios para la ejecución de un experimento en un contexto específico.

6. **Ejecución Experimental**
   La ejecución experimental es la parte operativa del experimento, es decir es hacerlo (aplicar el tratamiento sobre la objeto experimental).

Este proceso se compone de tres fases que son: Pre Sesión, Durante Sesión y Post Sesión.

a) **Pre Sesión:** La pre sesión es una fase que comprende el entrenamiento para la ejecución del experimento, asignación de sujetos a grupos, impresión del material experimental y una sesión de introducción al experimento.

- **Entrenamiento de los Sujetos:** Previa la ejecución del experimento, los sujetos experimentales son capacitados en los tratamientos que aplicarán a los objetos experimentales, para equilibrar el nivel de conocimientos. Esta capacitación debe tener un contenido teórico y uno práctico, para familiarizar a los sujetos con el experimento y enmarcarlos intrínsecamente con el objetivo del experimento.
- **Asignación de Sujetos a Tratamientos:** Es el proceso por medio del cual se estructuran los grupos de sujetos que aplicarán los tratamientos a los objetos experimentales, dependiendo de las decisiones de ejecución tomadas para el experimento. Para garantizar la validez de las observaciones obtenidas, el experimento precisa aleatorización para intentar evitar de la influencia de características inherentes al sujeto en el resultado del experimento.
- **Impresión del Material Experimental:** Impresión de los objetos experimentales para que el sujeto pueda realizar la tarea y de los instrumentos para recolectar datos; a continuación se generan paquetes personalizados para cada sujeto por sesión y tarea dentro de la sesión.
- **Sesión de Información a los Sujetos sobre su Papel en el Experimento:** La Sesión de introducción, es una reunión en la cual, el experimentador, explica a detalle a los sujetos experimentales, el protocolo de aplicación de los tratamientos a los objetos experimentales y también describe los formularios, orden en que los reciben y sus contenidos.

b) **Durante Sesión:** La fase durante sesión, es en si la ejecución in situ del experimento, en donde el sujeto experimental aplica el tratamiento sobre el objeto experimental. En esta fase se llevan a cabo las siguientes actividades: Explicación del objeto experimental, aplicación del tratamiento, entrega de los collected items y limpieza del material experimental.

- **Explicación del Objeto Experimental:** Previo a la aplicación del tratamiento, el Experimentador explica al sujeto experimental las características del objeto experimental. Las mencionadas características también podrían estar disponibles en el material suministrado.
- **Aplicación del Tratamiento:** La aplicación del tratamiento, es la actividad fundamental del experimento, ya que luego de esta acción,

de cada sujeto experimental se obtienen observaciones, luego de aplicar los tratamientos a cada objeto experimental.

- **Entrega de los Collected Items:** Una vez aplicados todos los tratamientos diseñados, el sujeto experimental entrega las observaciones obtenidas al encargado de controlar o monitorear el desarrollo del experimento.
- **Limpieza de los Collected Items:** Esta tarea, generalmente es realizada por el encargado de controlar o monitorear el desarrollo del experimento y consiste en eliminar cualquier material relacionado con los objetos experimentales, con el objetivo de separar los instrumentos experimentales cumplimentados.

c) **Post Sesión:** En esta fase se evalúa la aplicación de los tratamientos a los objetos experimentales. Las tareas que se llevan a cabo en esta fase son: Corrección de los collected items, generación de la raw data, y entrega de los collected items y raw data al GE del grupo de investigación.

- **Corrección de los Collected Items:** La corrección de los formularios de recolección de datos cumplimentados por los sujetos experimentales, o denominados collected items, es el proceso mediante el cual un experimentador examina los collected items y compara los casos de prueba generados versus plantillas de casos de prueba previamente elaboradas. Dicho estudio revela el nivel de descubrimiento en los collected items respecto a la aplicación de los tratamientos y en conformidad o no con la hipótesis planteada.
- **Generación de la Raw Data:** La generación de la plantilla de datos brutos o crudos, o mejor conocida como raw data, es una consecuencia del proceso de corrección anterior; y representa el nivel de coincidencia entre los casos de prueba generados con respecto a los casos de prueba previamente generados con las definiciones correctas.
- **Entrega de los Collected Items y Raw Data al Gestor del Experimento:** El gestor del experimento es el albacea de la información del grupo de investigación y por lo tanto debe consolidar la información de las replicaciones llevadas a cabo de cada experimento perteneciente al grupo de investigación; en especial lo correspondiente a los collected items y a la raw data, ya que otras de las funciones del GE es la agregación informal de replicaciones.

7. **Post Ejecución Experimental**
   La post ejecución experimental es la fase del proceso experimental en la que se tratan los datos, para su respectivo análisis e interpretación. Las actividades llevadas a cabo en la post ejecución experimental se describen a continuación.

   a) **Pre Procesamiento de Datos:** El pre pocesamiento de datos es una

actividad a través de la cual, la raw data es procesada y transformada en medidas de acuerdo a lo descrito en las métricas, para que a través de técnicas estadísticas pueda ser medida la variable respuesta.

b) **Análisis Estadístico de Datos:** El análisis estadístico de datos, es el proceso por el cual las medidas son analizadas por las técnicas estadísticas determinadas por la variable respuesta y por el tipo de diseño experimental, para validar o rechazar la hipótesis planteada en el experimento y para tener los datos necesarios para la interpretación.

c) **Interpretación de Datos:** La interpretación de datos consiste en una actividad por medio de la cual los resultados estadísticos son interpretados y convertidos en resultados, los cuales serán comparables o no con los del experimento original. Esta interpretación de datos, podría generar hallazgos nuevos que a su vez podrían resultar en una variación al experimento original o al protocolo de las replicaciones.

8. **Planteamiento del Siguiente Ciclo**
   En la fase de planteamiento del siguiente ciclo, se evalúa los resultados de la replicación y se los contrasta con los resultados de otras replicaciones o directamente con los resultados del experimento original, para decidir los siguientes paso en la investigación, pudiéndose decidir realizar una replicación similar a la anterior, alguna modificación de los elementos experimentales; o incluso quizás, el planteamiento de un nuevo experimento base.

9. **Publicación de los Resultados Experimentales**
   La fase de publicación de los resultados experimentales, consiste en difundir en la comunidad de ISE, los hallazgos que han resultado de una o varias replicaciones; esto dependerá, del ciclo de experimentación en el que se encuentre un experimento y de los resultados conjuntos o eventualmente individuales de las replicaciones de un experimento.

2.2.5. Fundamentación Científica de la Variable Dependiente

Calidad de Software

La tendencia tecnológica actual, determina que la mayoría de servicios que se prestan a las personas, sea por medio de un producto software o a través de un dispositivo que contenga software, por lo que el interés por su calidad, crece continuamente, a medida que los clientes se involucran más y más a sus tareas cotidianas con la tecnología, por lo que se vuelven más selectivos y comienzan a rechazar los productos poco fiables o que no dan respuesta a sus requerimientos [Juristo et al., 2006].

Es importante diferenciar, en este punto, entre la calidad del **producto** software y la calidad del **proceso** de desarrollo software. Las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso de desarrollo, ya que la calidad del producto va a estar en función de la calidad del proceso de desarrollo. Sin un buen proceso de desarrollo es casi imposible obtener un buen producto [Juristo et al., 2006].

No basta con tener en cuenta la calidad del producto una vez finalizado, cuando los problemas de mala calidad ya no tienen solución o la solución es muy costosa; la calidad de un producto software debe ser considerada en todos sus estados de evolución a medida que avanza el desarrollo según el ciclo de vida seleccionado para su construcción (especificaciones, diseño, código, etc.) [Juristo et al., 2006].

Definir el término calidad en un producto software y su correspondiente comprobación, son tareas en extremo complejas; por lo tanto ¿Es realmente posible encontrar un conjunto de propiedades en un producto software que denoten su calidad? Una posibilidad para responder a esta pregunta son los Modelos de Calidad, en los cuales se define la calidad de forma jerárquica y resuelven la complejidad mediante la descomposición [Juristo et al., 2006].

En el caso de la calidad del software, igualmente el término es difícil de definir, para intentar hacerlo, al software se lo caracteriza con atributos:

- Funcionalidad: Habilidad para realizar el trabajo deseado.
- Fiabilidad: Habilidad para mantenerse operativo (funcionando).
- Eficiencia: Habilidad para responder a una petición de usuario con la velocidad apropiada.
- Usabilidad: Habilidad para satisfacer al usuario.
- Mantenibilidad: Habilidad del software para poder realizar cambios en él fácilmente y con una adecuada proporción cambio/costo.
- Portabilidad: Habilidad para operar en diferentes entornos informáticos.

A su vez, cada una de estas características del software puede subdividirse en atributos aún más concretos. La tabla 1 muestra una subdivisión ampliamente difundida por la IEEE.

Tabla 1
Calidad de Software ISO 9126-1898

Características y Sub Características		**Descripción**
Funcionalidad		**Capacidad del producto software para proporcionar las funcionalidades que satisfacen las necesidades explícitas e implícitas cuando el software se usa bajo unas ciertas condiciones**
	Adecuación	Capacidad del producto software para proporcionar un conjunto de funciones apropiado para unas ciertas tareas y objetivos de usuario
	Exactitud	Capacidad del producto software para proporcionar los resultados o efectos correctos o acordados, con el grado necesario de precisión
	Interoperabilidad	Capacidad del producto software para interactuar con uno o más sistemas
	Seguridad	Capacidad del producto software para proteger información y datos de manera que las personas o sistemas no autorizados no puedan leerlos o modificarlos, al tiempo que no se deniega el acceso a las personas o sistemas autorizados
	Cumplimiento funcional	Capacidad del producto software para adherirse a normas, convenciones o regulaciones en leyes y prescripciones similares relacionadas con la funcionalidad
Fiabilidad		**Capacidad del producto software para mantener un nivel especificado de prestaciones cuando se usa bajo unas cierta condiciones**
	Madurez	Capacidad del producto software para evitar fallar como resultado de fallos en el software
	Tolerancia a fallos	Capacidad del software para mantener un nivel especificado de prestaciones en caso de fallos software o de infringir sus interfaces
	Capacidad de recuperación	Capacidad del producto software para restablecer un cierto nivel de prestaciones y de recuperar los datos directamente afectados en caso de fallo
	Conformidad	Capacidad del producto software para adherirse a normas, convenciones o regulaciones relacionadas con la fiabilidad
Usabilidad		**Capacidad del producto software para ser entendido, aprendido, usado y ser atractivo para el usuario, cuando se usa bajo condiciones especificadas**
	Comprensibilidad	Capacidad del producto software que permite al usuario entender si el software es adecuado y cómo puede ser usado para unas tareas o condiciones de uso particulares
	Facilidad de aprendizaje	Capacidad del producto software que permite al usuario aprender sobre su aplicación
	Operatividad	Capacidad del producto software que permite al usuario administrarlo y controlarlo
	Capacidad de ser atractivo	Capacidad del producto software para ser atractivo al usuario
	Cumplimiento de la usabilidad	Capacidad del producto software para adherirse a normas, convenciones, guías de estilo o regulaciones relacionadas con la usabilidad
Eficiencia		**Capacidad del producto software para proporcionar prestaciones apropiadas, relativas a la cantidad de recursos usados, bajo condiciones determinadas**
	Comportamiento temporal	Capacidad del producto software para proporcionar tiempos de respuesta y de proceso y índices de respuesta al realizar sus funciones bajo unas ciertas condiciones
	Utilización de recursos	Capacidad del producto software para usar las cantidades y tipos de recursos adecuados cuando el software lleva a cabo su función bajo condiciones determinadas
	Cumplimiento de la eficiencia	Capacidad del producto software para adherirse a normas o convenciones relacionadas con la eficiencia
Mantenibilidad		**Capacidad del producto software para ser modificado. Las modificaciones podrían incluir correcciones, mejoras o adaptación del software a cambios en el entorno, y requisitos y especificaciones funcionales**
	Analizabilidad	Capacidad del producto software para serle diagnosticadas deficiencias o causas de los fallos en el software, o para identificar las partes que han de ser modificadas
	Variabilidad	Capacidad del producto software que permite que una determinada modificación sea implementada
	Estabilidad	Capacidad del software para evitar efectos inesperados debidos a modificaciones del software
	Capacidad de prueba	Capacidad del producto software que permite que el software modificado sea validado
	Cumplimiento mantenibilidad	Capacidad del producto software para adherirse a normas o convenciones relacionadas con la mantenibilidad
Portabilidad		**Capacidad del producto software para ser migrado de un entorno a otro**
	Adaptabilidad	Capacidad del producto software para ser adaptado a diferentes entornos, sin aplicar acciones o mecanismos distintos de aquellos proporcionados para este propósito por el propio software
	Instalabilidad	Capacidad del producto software para ser instalado en un cierto entorno
	Coexistencia	Capacidad del producto software para coexistir con otro software independiente, en un entorno común, compartiendo recursos comunes
	reemplazabilidad	Capacidad del producto software para ser usado en lugar de otro producto software, para el mismo propósito, en el mismo entorno
	Cumplimiento portabilidad	Capacidad del producto software para adherirse a normas o convenciones relacionadas con la portabilidad

Pruebas de Software

Las pruebas de software o comúnmente conocidas como testing, son muy antiguas en la historia de las computadoras digitales; sin embargo siguen siendo fundamentales hasta la actualidad; ya que, son un importante medio de evaluación del software para determinar su calidad. Dado que el testing típicamente consumen entre un 40∼50 % del esfuerzo del desarrollo, y más en sistemas que requieren altos niveles de fiabilidad; se constituyen como una parte fundamental de la ingeniería de software . Con el desarrollo de lenguajes de cuarta generación (4GL), que aceleran el proceso de implementación, la proporción de tiempo dedicado a las pruebas ha aumentado. Como la cantidad de mantenimiento y actualización de los sistemas existentes crece, una cantidad significativa de pruebas también serán necesarios para verificar los sistemas después de realizar cambios. A pesar de los avances en los métodos y técnicas formales de verificación, un sistema aún tiene que ser probado antes de ser usado. El testing siguen siendo realmente el medio más eficaz para asegurar la calidad de un sistema software de complejidad no trivial, así como una de las áreas más complejas y menos entendidas en la ingeniería de software. El testing, es un área de investigación importante dentro de la informática, que probablemente sea cada vez más importante en el futuro [Luo, 2001].

El testing de software es un área muy amplia que involucra a muchas otras áreas técnicas y no técnicas como por ejemplo: la especificación diseño e implementación, mantenimiento, proceso y gestión de problemas en la IS. A continuación se hace referencia a ciertos conceptos técnicos, que ayudarán al entendimiento de las pruebas de software. A continuación se citan algunos conceptos técnicos que ayudan a comprender el testing.

1. **El Objetivo de las Pruebas de Software:** En diferentes Publicaciones, la definición de testing varía de acuerdo al propósito, al proceso y al nivel de descripción del testing. Miller da una buena descripción de testing en [Miller, 1980]:

   "The general aim of testing is to affirm the quality of software systems by systematically exercising the software in carefully controlled circumstances."

   Desde la perspectiva de la descripción de Miller sobre testing, la mayor garantía de calidad de software, la constituyen las actividades de testing. Miller sostiene que el objetivo principal del testing es encontrar errores. Un buen testing es aquel que tiene una alta probabilidad de encontrar un error que aún no ha sido detectado, y un testing exitoso es aquel que descubre un error, aún sin descubrir. Esta categoría general de las actividades de prueba de software puede ser aún más amplia. Para los propósitos de este documento, la prueba es el análisis dinámico de una pieza de software, que requiere la ejecución del sistema para producir resultados, que luego se comparan a los resultados esperados [Luo, 2001].

2. **Análisis Estático y Análisis Dinámico de las Pruebas de Software:**

En base a la interrogante de si la ejecución real de software bajo pruebas es necesaria o no, hay dos categorías de actividades principales de aseguramiento de la calidad [Luo, 2001]:

- **Análisis Estático:** El análisis estático se centra en la gama de métodos que se utilizan para determinar o estimar la calidad del software sin referencia a ejecuciones reales. Las técnicas en esta área incluyen la inspección de código, análisis de programas, análisis simbólico, y la verificación de modelos [Luo, 2001].
- **Análisis Dinámico:** El análisis dinámico trata con métodos específicos para comprobar y/o aproximar la calidad de software a través de ejecuciones reales; es decir, con datos reales y sobre circunstancias reales (o simuladas). Las técnicas en esta área incluyen la síntesis de entradas, el uso de procedimientos de prueba guiados estructuralmente y la automatización de la generación del ambiente de pruebas [Luo, 2001].

Técnicas de Pruebas de Software

El flujo de información del testing se muestra en la Figura 3. Como se puede notar, el testing supone la configuración de las entradas adecuadas, la ejecución del software sobre la entrada, y el análisis de la salida. La configuración de software incluye la especificación de requisitos, el código fuente, y demás. La configuración de la prueba incluye casos de prueba, plan de pruebas, protocolo, y herramientas de la prueba.

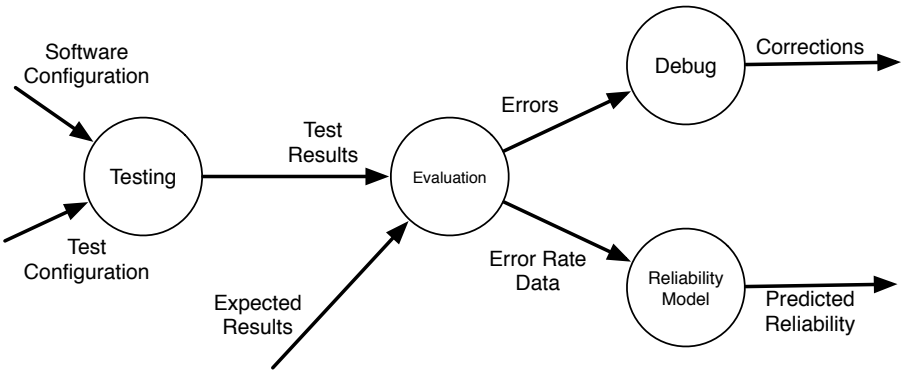

Figura 3: Flujo de Información de las Pruebas de Software
Fuente: ([Luo, 2001])

En base al flujo de información de las pruebas de software, una técnica de testing especifica la estrategia usada para seleccionar los casos de prueba de entrada y analizar los resultados. Diferentes técnicas de testing revelan diferentes aspectos de calidad de un sistema software.

Como se ha indicado anteriormente, es necesario evaluar el sistema software a medida que se va avanzando en el proceso de desarrollo de dicho sistema. De esta

forma se intenta que la detección de defectos se haga lo antes posible y tenga menor impacto en el tiempo y esfuerzo de desarrollo. Para realizar esta evaluación, las técnicas de evaluación estática se aplican en el mismo orden en que se van generando los distintos productos del desarrollo siguiendo una filosofía top-down. Esto es, la evaluación estática acompaña a las actividades de desarrollo, a diferencia de la evaluación dinámica que únicamente puede dar comienzo cuando finaliza la actividad de codificación, siguiendo así una estrategia botom-up. La evaluación estática es el único modo disponible de evaluación de artefactos para las primeras fases del proceso de desarrollo (análisis y diseño), cuando no existe código. Esta idea se muestra en la Figura 4 en la que como se observa la evaluación estática se realiza en el mismo sentido en que se van generando los productos del desarrollo de software, mientras que la dinámica se realiza en sentido inverso [Juristo et al., 2006].

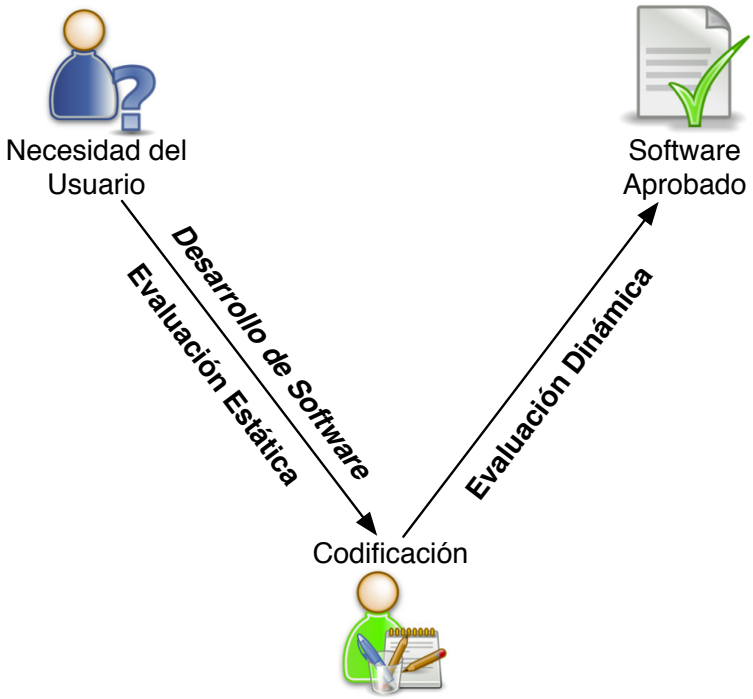

Figura 4: Relación entre Evaluación y Proceso Software

A detalle, la Figura 5 muestra la aplicación de las técnicas estáticas y dinámicas para evaluar software. La evaluación estática (también conocida como revisiones) se lleva a cabo en paralelo al proceso de desarrollo. Cada actividad de desarrollo está emparejada con una actividad de evaluación; es decir por ejemplo, la actividad de *definición de requisitos de usuario* va acompañada de una actividad de *revisión de requisitos de usuario*, la actividad de *definición de requisitos software* va emparejada con su correspondiente actividad de revisión, etc.

Las actividades de revisión marcan el punto de decisión para el paso a la siguiente actividad de desarrollo. Es decir, la actividad de requisitos interactúa con la actividad de revisión de requisitos en un bucle de mejora iterativa hasta el momento en

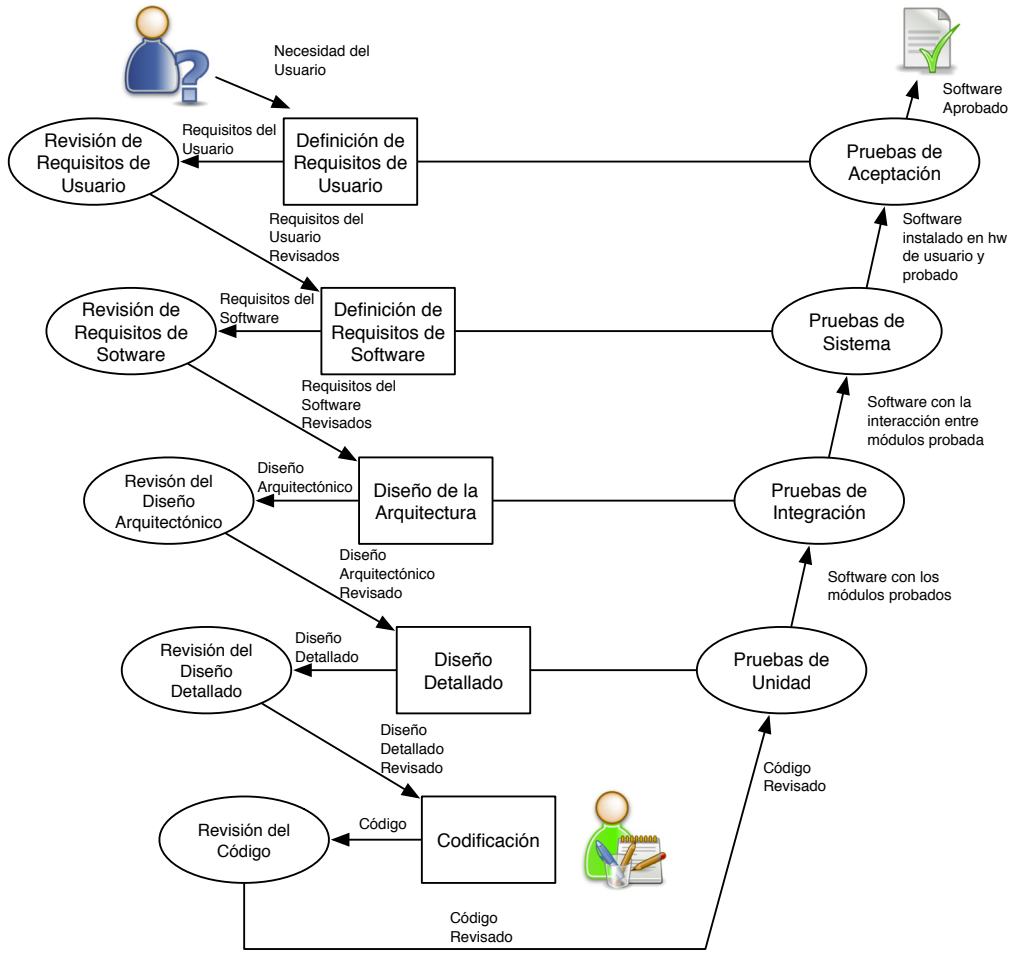

Figura 5: Modelo en V de evaluación de Software
Fuente: ([Juristo et al., 2006])

que la calidad de los requisitos permite abordar la subsiguiente fase de desarrollo. Lo mismo ocurre con el diseño arquitectónico: sufrirá una mejora iterativa hasta que su nivel de calidad permita pasar al diseño detallado y así, sucesivamente. Nótese que esto también ocurre en la fase de codificación. La actividad siguiente a la de implementación es la fase de pruebas unitarias. No obstante, antes de pasar a ella, los programas deberán evaluarse estáticamente. Del mismo modo que se ha hecho con los otros productos.

Una vez realizadas las revisiones se procede con la evaluación dinámica, que se realiza sobre el código. Más adelante se estudiarán los distintos tipos de pruebas dinámicas; sin embargo, se puede indicar que la primera prueba a realizar es la denominada Prueba de Unidad en la que se buscan errores en los componentes más pequeños del programa (módulos).

1. **Técnicas de Evaluación Estática** La búsqueda de defectos en las primeras fases de desarrollo software, es debida a que éstos defectos trascienden en cadena al producto final; es decir, defectos en los requisitos se traducirán en

defectos en el sistema final. Haciendo analogía con la arquitectura de edificios, si en un plano el color de una línea indica su significado, una confusión en el color se traducirá en un error en el edificio. El plano de un edificio es el artefacto equivalente al diseño de un producto software. Si un diseño contiene defectos, seguramente estos defectos se trasmitirán al código cuando los programadores usen ese diseño como guía para su trabajo [Juristo et al., 2006][Luo, 2001].

La detección de errores en las primeras etapas de desarrollo de software implica grandes beneficios, ya si las revisiones únicamente se aplican al código, mejoran la calidad y producen ahorros en los costos del proyecto; sin embargo, los ahorros son mayores si se inspeccionan artefactos tempranos del desarrollo. Estudiando los resultados publicados sobre ahorros con las revisiones, puede afirmarse que la utilización de inspecciones de código produce un ahorro del 39 % [Juristo et al., 2006] sobre el coste de detectar y corregir defectos, frente a únicamente utilizar la evaluación dinámica. Sin embargo, el ahorro es del 44 % [Juristo et al., 2006] si se inspecciona también el diseño.

Las técnicas de Evaluación estática de elementos del desarrollo se las conoce de modo genérico como *revisiones*; las cuales, pretenden detectar manualmente defectos en cualquier producto del desarrollo. Entiéndase por manualmente, a que el producto en cuestión (ya sea requisito, diseño, código, etc.) está impreso en papel y los revisores están analizando ese producto mediante la lectura del mismo, sin ejecutarlo.

Existen varios tipos de revisiones, dependiendo de qué se busca y cómo se analiza el producto en cuestión. Para citar en este trabajo, se ha adoptado la propuesta de [Juristo et al., 2006].

- **Revisiones Informales:** Las revisiones informales o también llamadas de forma inadecuada tan sólo como *revisiones* (lo cual genera confusión con el nombre genérico de todas estas técnicas), no dejan de ser un intercambio de opiniones entre los participantes. Esta técnica es muy inusual y caduca.
- **Revisiones Formales o Inspecciones:** Las Revisiones formales o inspecciones, representan una actividad en la cual los participantes son responsables de la fiabilidad de la evaluación, y generan un informe que refleja el acto de la revisión.
- **Walkthrough:** Es una revisión que consiste en simular la ejecución de casos de prueba para el programa que se está evaluando. No existe traducción exacta en español y a menudo se usa el término en inglés. Quizás la mejor traducción porque ilustra muy bien la idea es *recorrido*; de hecho, con los walkthrough se recorre el programa imitando lo que realizaría la computadora.
- **Auditorias:** Las auditorias contrastan los productos generados durante el desarrollo con estándares, generales o de la organización. Las auditorías

típicamente pretenden comprobar formatos de documentos, inclusión de toda la información necesaria, etc. Es decir, no se trata de comprobaciones técnicas, sino de gestión o administración del proyecto.

2. **Técnicas de Evaluación Dinámica** Las técnicas de evaluación dinámica proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas. La Figura 6 representa gráficamente la filosofía de las pruebas de evaluación dinámicas, más conocidas como pruebas de caja blanca y caja negra.

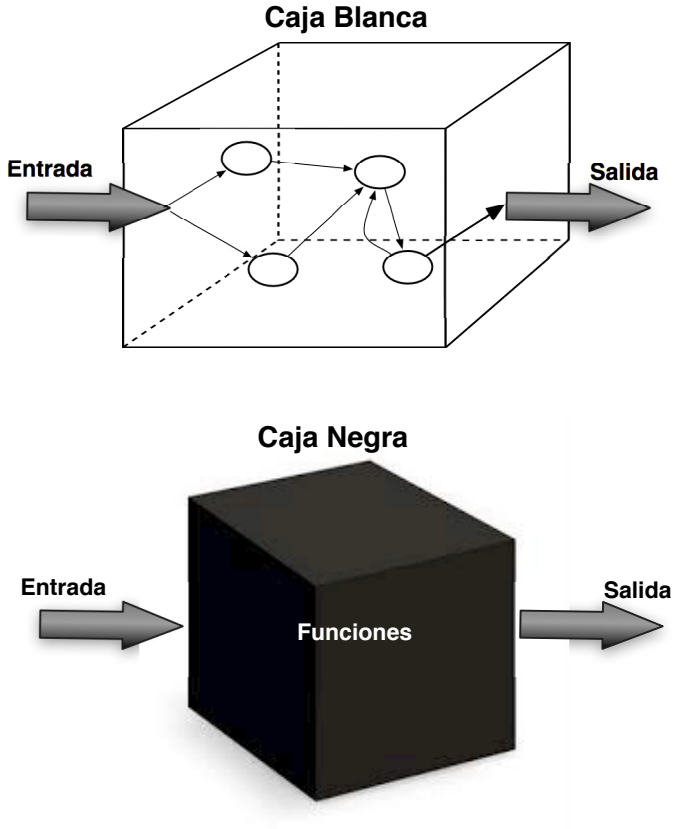

Figura 6: Representación de Pruebas Dinámicas

A primera vista parecería que una prueba de caja blanca completa llevaría a disponer de un código perfectamente correcto. De hecho esto ocurriría si se han probado todos los posibles caminos por los que puede pasar el flujo de control de un programa. Sin embargo, para programas de cierta envergadura, el número de casos de prueba que habría que generar sería excesivo, nótese que el número de caminos incrementa exponencialmente a medida que el número de sentencias condicionales y bucles aumenta. Sin embargo, este tipo de prueba no se desecha como impracticable. Se pueden elegir y ejercitar ciertos caminos representativos de un programa [Juristo et al., 2006].

Por su parte, tampoco sería factible en una prueba de caja negra probar todas

y cada una de las posibles entradas a un programa, por lo que análogamente a como ocurría con las técnicas de caja blanca, se seleccionan un conjunto representativo de entradas y se generan los correspondientes casos de prueba, con el fin de provocar fallos en los programas [Juristo et al., 2006].

Más en detalle, estas técnicas son explicadas a continuación:

a) **Técnica de Pruebas de Software de Caja Blanca o Estructural:** En la técnica de pruebas de software estructural, la entidad software es vista como una "**caja blanca**", y también se le conoce como *técnica de caja transparente o de cristal*. Este método se centra en cómo diseñar los casos de prueba, atendiendo al comportamiento interno y a la estructura del programa. Se examina así la lógica interna del programa sin considerar los aspectos de rendimiento. El objetivo de seleccionar tales casos de prueba consiste en provocar la ejecución de puntos específicos en la entidad software, tales como declaraciones específicas, ramas del programa o rutas. Los resultados esperados son evaluados en un conjunto de criterios de cobertura.

   De acuerdo a lo indicado anteriormente, puede ser impracticable realizar una prueba exhaustiva de todos los caminos de un programa; por ello, se han definido distintos criterios de cobertura lógica, que permiten decidir qué sentencias o caminos se deben examinar con los casos de prueba. Estos criterios son:

   - **Cobertura de Sentencias:** De acuerdo al criterio de cobertura de sentencias, se escriben casos de prueba suficientes para que cada sentencia en el programa se ejecute, al menos, una vez.
   - **Cobertura de Decisión:** De acuerdo al criterio de cobertura de decisión, se escriben casos de prueba suficientes para que cada decisión en el programa se ejecute una vez con resultado verdadero y otra con el falso.
   - **Cobertura de Condiciones:** Según el criterio de cobertura de condiciones, se escriben casos de prueba suficientes para que cada condición en una decisión tenga una vez resultado verdadero y otra falso.
   - **Cobertura Decisión/Condición:** Según del criterio de cobertura decisión/condición, se escriben casos de prueba suficientes para que cada condición en una decisión tome todas las posibles salidas, al menos una vez, y cada decisión tome todas las posibles salidas, al menos una vez.
   - **Cobertura de Condición Múltiple:** De acuerdo al criterio de cobertura de condición múltiple, se escriben casos de prueba suficientes para que todas las combinaciones posibles de resultados de cada condición se invoquen al menos una vez.

- **Cobertura de Caminos:** Según el criterio de cobertura de caminos, se escriben casos de prueba suficientes para que se ejecuten todos los caminos de un programa. Entendiendo camino como una secuencia de sentencias encadenadas desde la entrada del programa hasta su salida.

  Una de las técnicas empleadas para aplicar este criterio de cobertura es la *prueba del camino básico*. Esta técnica se basa en obtener una medida de la complejidad del diseño procedimental de un programa (o de la lógica del programa). Esta medida es la complejidad ciclomática de McCabe, y representa un límite superior para el número de casos de prueba que se deben realizar para asegurar que se ejecuta cada camino del programa.

  Los pasos a realizar para aplicar esta técnica son:

  - Representar el programa en un grafo de flujo
  - Calcular la complejidad ciclomática
  - Determinar el conjunto básico de caminos independientes
  - Derivar los casos de prueba que fuerzan la ejecución de cada camino

b) **Técnica de Pruebas de Software de Caja Negra o Funcional:** Las técnica de pruebas de software funcional, también conocidas como *pruebas de comportamiento*, se basan en la *especificación* del programa o componente a ser probado para elaborar los casos de prueba. El programa o el sistema bajo pruebas es visto como una "**caja negra**", cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas. Sin embargo, como el estudio de todas las posibles entradas y salidas de un programa sería impracticable, entonces se selecciona un subconjunto de ellas, sobre las que se realizan las pruebas. Para seleccionar el subconjunto de entradas y salidas sobre las que trabajar, hay que suponer que en todo programa existe un conjunto de entradas que causan un comportamiento erróneo en el sistema bajo pruebas, y como consecuencia producen una serie de salidas que revelan la presencia de defectos. Entonces, dado que la prueba exhaustiva es imposible, el objetivo final es pues, encontrar una serie de datos de entrada cuya probabilidad de pertenecer al conjunto de entradas que causan dicho comportamiento erróneo sea lo más alto posible [Juristo et al., 2006].

Tal como las técnicas de caja blanca, para confeccionar los casos de prueba de caja negra existen distintos criterios. Algunos de ellos son: Particiones de Equivalencia, análisis de valores límite, métodos basados en grafos, pruebas de comparación y análisis causa-efecto. De los criterios citados, a continuación se detallan algunos de ellos, debido a que son los utilizados

en la replicación llevada a cabo.

- **Criterio de Particiones de Equivalencia:** El criterio de particiones de clases de equivalencia, es un método de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente se dirige a una definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar.

  El criterio de particiones de equivalencia, intenta dividir el dominio de entrada de un programa en un número finito de clases de equivalencia. De tal modo que se pueda asumir razonablemente que una prueba realizada con un valor representativo de cada clase es equivalente a una prueba realizada con cualquier otro valor de dicha clase; Lo que quiere decir, que si el caso de prueba correspondiente a una clase de equivalencia detecta un error, el resto de los casos de prueba de dicha clase de equivalencia deben detectar el mismo error. Y viceversa.

  El diseño de casos de prueba según esta técnica consta de dos pasos: (1) Identificar las clases de equivalencia y (2) Identificar los casos de prueba.

  Una clase de equivalencia representa un conjunto de estados válidos y no válidos para las condiciones de entrada de un programa. Las clases de equivalencia se identifican examinando cada condición de entrada (normalmente una frase en la especificación) y dividiéndola en dos o más grupos. Se definen dos tipos de clases de equivalencia, las clases de equivalencia válidas, que representan entradas válidas al programa, y las clases de equivalencia no válidas, que representan valores de entrada erróneos.

  Finalmente, el objetivo es minimizar el número de casos de prueba, así cada caso de prueba debe considerar tantas condiciones de entrada como sea posible. No obstante, es necesario realizar con cierto cuidado los casos de prueba de manera que no se enmascaren faltas.

- Criterio de Análisis de Valor Límite: La experiencia muestra que los casos de prueba que exploran las condiciones límite producen mejor resultado que aquellos que no lo hacen [Juristo et al., 2006]. Las condiciones límite son aquellas que se hayan en los márgenes de la clase de equivalencia, tanto de entrada como de salida. Por ello, se ha desarrollado el análisis de valores límite como técnica de prueba. Esta técnica permite a elegir los casos de prueba que ejerciten los valores límite. Por lo tanto, el análisis de valores límite complementa la técnica de partición de equivalencia de manera que: (1) En lugar de seleccionar cualquier caso de prueba de las clases válidas e inválidas,

se eligen los casos de prueba en los extremos; y (2) En lugar de centrase sólo en el dominio de entrada, los casos de prueba se diseñan también considerando el dominio de salida.

Proceso de Pruebas

El Espectro de las Pruebas de software es amplio, ya que está involucrado en cada etapa del ciclo de vida software, pero el testing hecho en cada nivel de desarrollo de software es en esencia diferente; es más, tiene objetivos diferentes. A continuación se detalla las diferentes instancias del proceso de pruebas, que va de la mano con el proceso software, tal como se indica en la Figura 5.

1. **Pruebas de Unidad:** Las pruebas de unidad son realizadas a bajo nivel y prueban la unidad básica de software, la cual es la pieza más pequeña testeable de software, y es a menudo llamada "unidad", "modulo", o "componente"intercambiable.
2. **Pruebas de Integración:** Las pruebas de integración son llevadas a cabo cuando dos o más unidades testeadas son combinadas en una estructura más grande. La prueba es a menudo realizada tanto en las interfaces entre los componentes y en la estructura más grande que está siendo construida, si su propiedad de calidad no puede ser evaluada a partir de sus componentes.
3. **Pruebas de Sistema:** Las pruebas de sistema tienden a validar la calidad de todo el sistema, de principio a fin. Estas pruebas, se basan a menudo en la especificación requisito/funcional del sistema. Los atributos de calidad no funcionales, tales como: Fiabilidad, seguridad, y mantenibilidad, son también revisados.
4. **Pruebas de Aceptación:** Las pruebas de aceptación son realizadas cuando los desarrolladores entregan el sistema completo a los clientes o usuarios. El propósito de las pruebas de aceptación es más bien dar confianza que el sistema está trabajando, que encontrar errores.

2.2.6. Hipótesis

La aplicación de una Replicación Experimental, permitirá evaluar la efectividad de las Técnicas de Pruebas de Software Estructurales y Funcionales.

2.2.7. Señalamiento de Variables

Variable Independiente

Replicación Experimental.

Variable Dependiente

Técnicas de Pruebas de Software Estructurales y Funcionales.

Capítulo III

INFORMACIÓN SOBRE EL EXPERIMENTO ORIGINAL

En este apartado se pretende dar una visión global acerca del experimento original, su origen, motivación, objetivos, operación y resultados.

3.1. Introducción

El experimento original fue llevado a cabo en Diciembre de 2005 en la Universidad Politécnica de Madrid (UPM) por Juristo et al. [Juristo et al., 2013]. Este experimento tiene como objetivo estudiar la efectividad de las técnicas de pruebas estructurales y funcionales en lo que respecta a su capacidad de detección de faltas. El experimento de Juristo et al. [Juristo et al., 2013], es a su vez, una replicación diferenciada de la familia de experimentos iniciada por Basili and Selby [Basili and Selby, 1985] [Basili and Selby, 1987] en 1982, luego replicada por diversos investigadores tales como Kamsties y Lott [Kamsties and Lott, 1995] y Roper et al. [Wood et al., 1997]. La diferencia fundamental entre el experimento original y la familia de experimentos de Basili reside en el modo en que las faltas fueron sembradas en los programas.

Basili, Selby y otros refieren al tipo de faltas utilizadas en sus experimentos como bastante representativas de aquellas que tienden a ocurrir habitualmente en el desarrollo de software [Basili and Selby, 1987]. Por lo tanto, esos experimentos evaluaron cuan efectivo es el proceso de testing que se espera en la práctica. Sin embargo, los tipos de faltas usados en el experimento de Juristo et al. [Juristo et al., 2013], fueron más simples. En concreto, las faltas pueden dividirse en dos grandes tipos: faltas que pueden ser detectadas por las estrategias estructural o funcional de generación de casos de prueba (en lo que sigue faltas InScope) y, aquellas que, al menos en teoría, no pueden detectarse (faltas OutScope).

A modo de ejemplo, una falta detectable (InScope) para la técnica de pruebas de software funcional de clases de equivalencia (EP (de las siglas en inglés correspondientes a Equivalence Partitioning)) podría ser: "*Partes no implementadas de la especificación*". EP es capaz de revelar este tipo de faltas ya que genera casos de prueba para todas las especificaciones del programa, incluidas las no implementadas. Para el caso de la técnica de pruebas estructural por cobertura de decisión (BT (de las siglas del inglés correspondientes a Branch Testing)), una falta detectable sería la existencia de: "*Código para funcionalidad que no están en la especificación*" (por ejemplo: Cuando un programador por error ha escrito o copiado código que implementa funciones no tomadas en cuenta en la especificación). La estrategia de BT prescribe que los casos de prueba son generados para cubrir todas las alternativas de un 100 % de código de decisión, lo que permitiría descubrir el código sobrante.

Tal y como puede apreciarse en los ejemplos, una falta detectable para una técnica es no detectable (OutScope) para la otra técnica, y viceversa. La diferenciación de los dos tipos de faltas permite estudiar la efectividad de las técnicas de testing de forma más precisa que en experimentos anteriores ya que es posible diferenciar el efecto propio de la técnica (esto es, la localización de las faltas InScope), de los efectos positivos que la técnica induce en el tester pero que no pueden adscribirse a la técnica per se (esto es, las faltas OutScope). Ambos efectos fueron confundidos en experimentos previos.

3.2. Pregunta de Investigación

El reporte original no incluye una pregunta de investigación explícita. No obstante, dicha pregunta puede fácilmente inferirse del diseño experimental, pudiendo enunciarse en formato GQM [Basili, 1992] tal como sigue:

Analizar:	La aplicación de las técnicas de pruebas de software
con el propósito de:	conocer su efectividad a nivel unitario
con respecto a:	distintos tipos de faltas (InScope, OutScope)
desde el punto de vista del:	tester
en el contexto de:	un experimento controlado en la academia

La pregunta de investigación nos permite identificar los elementos esenciales del experimento original.

3.2.1. Factor Principal

En el experimento original se estudiaron dos técnicas de pruebas: Técnica de pruebas funcional por particiones de equivalencia (EP) y la técnica de pruebas estructural

de control de flujo por cobertura de decisión (BT). Como grupo de control, utilizaron la técnica estática de lectura de código (CR (de las siglas en inglés correspondientes a Code Reading)) por abstracciones sucesivas, como una estrategia capaz, al menos en teoría, de detectar todas las faltas (tanto InScope como OutScope).

3.2.2. Variable Respuesta

La efectividad de las técnicas se ha medido como el porcentaje de faltas localizadas por las mismas, sobre el total de faltas sembradas. Dado que la pregunta de investigación se interesa por las faltas **InScope** y **OutScope**, la efectividad de las técnicas se calculará independientemente para cada tipo de falta.

3.2.3. Hipótesis

Las hipótesis nula (H0) y alternativa (H1) tanto para las faltas que están dentro del ámbito de las técnicas, así como para las faltas que están fuera del ámbito de las técnicas, respectivamente, se indican a continuación:

H_{10}**:** No hay diferencia en la efectividad de EP, BT y lectura de código (CR) con respecto a la detección de faltas dentro de su ámbito.

H_{11}**:** EP, BT y CR se diferencian en su efectividad respecto a las faltas dentro de su ámbito.

H_{20}**:** No hay diferencia en le efectividad de EP, BT y lectura de código (CR) con respecto a la detección de faltas fuera de su ámbito.

H_{21}**:** EP, BT y CR se diferencian en su efectividad respecto a las faltas fuera de su ámbito.

3.3. Participantes

Los sujetos participantes en el experimento fueron 46 estudiantes de pregrado de Ingeniería en computación de la Universidad Politécnica de Madrid, que estaban tomando el curso de Verificación y Validación de Software, que se imparte en el 4º curso de la carrera (de 5 años de duración). Los estudiantes tienen poca o ninguna experiencia profesional en desarrollo de software.

3.4. Diseño

Debido a que con un número de 46 sujetos experimentales en un diseño inter sujetos (between-subjects), resultaría en muy pocos sujetos por grupo ($46/3 \simeq 15$) y un reducido poder estadístico, los autores del experimento original utilizaron un diseño intra sujetos (within-subjects), donde cada sujeto aplicó en tres momentos distintos (sesiones), cada una de las técnicas ensayadas. Cada sesión fue llevada a cabo sin restricción de tiempo, con una duración de cuatro horas promedio. La tabla 2 resume el diseño empleado.

Tabla 2
Diseño Experimental del Experimento Original

Programa	**Cmdline**			**Ntree**			**Nametbl**		
Sesión	**Sesión 1**			**Sesión 2**			**Sesión 3**		
Técnica	**CR**	**BT**	**EP**	**CR**	**BT**	**EP**	**CR**	**BT**	**EP**
Grupo 1	**X**	**-**	**-**	**-**	**X**	**-**	**-**	**-**	**X**
Grupo 2	**X**	**-**	**-**	**-**	**-**	**X**	**-**	**X**	**-**
Grupo 3	**-**	**X**	**-**	**-**	**-**	**X**	**X**	**-**	**-**
Grupo 4	**-**	**X**	**-**	**X**	**-**	**-**	**-**	**-**	**X**
Grupo 5	**-**	**-**	**X**	**X**	**-**	**-**	**-**	**X**	**-**
Grupo 6	**-**	**-**	**X**	**-**	**X**	**-**	**X**	**-**	**-**

El diseño intra sujetos, si bien aumenta el poder estadístico, introduce varias amenazas a la validez. Por ejemplo, pueden producirse efectos de aprendizaje (maduración), cansancio y carryover[1]. Para minimizar en lo posible estas amenazas, las sesiones se realizaron con una semana de diferencia entre ellas. Aún así, para determinar si dichas amenazas se han producido, se han introducido las variables de sesión y grupo en el diseño, tal y como muestra la tabla 2. Más específicamente, de acuerdo a los experimentadores originales, la variable sesión puede identificar la presencia de efectos de maduración o cansancio y la variable grupo puede detectar efectos de carryover.

3.5. Artefactos

Para operacionalizar el diseño del experimento original se utilizaron varios artefactos:

- Material de entrenamiento, para formar o reforzar el conocimiento de los sujetos en las técnicas bajo estudio.

[1]Carryover: El efecto residual que la administración de un tratamiento a un sujeto tiene en otro tratamiento administrado más tarde en el mismo sujeto, donde el efecto residual aumenta o disminuye la eficacia del tratamiento posterior, se conoce como carryover [Brown, 1980].

- Objeto experimental, para permitir la aplicación de las técnicas.
- Materiales experimentales, para soportar la realización de las tareas experimentales.

Los artefactos mencionados se describen a continuación. Todos ellos están disponibles en [Juristo et al., 2013], excepto los programas y descripción de faltas, para evitar que puedan ser conocidos de antemano por los estudiantes participantes en los experimentos. No obstante, los mismos pueden obtenerse contactando directamente a los autores, a través de correo electrónico.

3.5.1. Material de Entrenamiento

Dentro del material utilizado para el entrenamiento de los sujetos en la aplicación de las técnicas de testing, objeto de estudio, están:

Guía de referencia

Documento que contiene los fundamentos teóricos y ejercicios prácticos para el entrenamiento de los sujetos experimentales en la aplicación de las técnicas de pruebas software.

Diapositivas de resumen

Material de soporte para el capacitador, que contiene un resumen de la guía de referencia.

Programas de entrenamiento

Material utilizado para complementar el fundamento teórico, enfocado en familiarizar al sujeto experimental con el ambiente de ejecución del experimento.

3.5.2. Objeto Experimental

Para la aplicación de las técnicas de testing, debe haber al menos tantos programas como sesiones, sembrados con las faltas adecuadas, ya que si el sujeto verifica el mismo programa varias veces, se corre el riesgo de que aprende sobre las faltas que este contiene. Los programas y faltas utilizados en el estudio original han sido:

Programas

Para el experimento original fueron utilizados tres programas escritos en lenguaje C: **cmdline**, **nametbl** and **ntree**; los cuales son razonablemente similares en número de líneas de código y complejidad ciclomática. Estos programas son los mismos que se han utilizado en experimentos como Kamsties y Lott [Kamsties and Lott, 1995] o Roper et al. [Roper et al., 1997]. Dado que es esperable que los programas afecten, incrementando o decrementando la efectividad de las técnicas, los programas fueron consideradas por los experimentadores originales, a efectos de análisis, como variables de bloque. Estos programas realizan las siguientes funciones:

(a) **Cmdline:** Realiza un parsing de una línea de comandos para determinar si es válida. En dicho caso, muestra una descripción de la línea de comandos introducida o una indicación de por qué es incorrecta. Es importante notar que no lleva a cabo ninguno de los comandos, sino que únicamente valida la entrada.

(b) **Nametbl:** Lee comandos de un fichero y los procesa para probar una serie de funciones, las cuales permiten manejar un tipo abstracto de datos correspondiente a una tabla de símbolos de un lenguaje determinado.

(c) **Ntree:** Lee comandos de un fichero y los procesa para probar una serie de funciones, las cuales permiten manejar un tipo abstracto de datos correspondiente a un árbol n-ario.

Faltas:

Como ya se ha indicado anteriormente, el tipo de faltas utilizado es distintivo del experimento original y la principal razón que lo diferencia de los experimentos previos (por ejemplo: [Kamsties and Lott, 1995] y [Roper et al., 1997]). Con el fin de sembrar sistemáticamente las faltas en los programas, las faltas tendrían que haber sido clasificadas por sensibilidad de la técnica. Sin embargo, los experimentadores originales no encontraron una clasificación con las características necesarias y, por lo tanto, ellos mismos generaron el tipo de faltas requerido, lo que se muestra en la tabla 3. Los programas fueron sembrados con 6 faltas, 3 de ellas (denominadas F1-F3) InScope para EP, y otras 3 (F4-F6) de tipo InScope para BT. Recuérdese que la técnica CR en teoría puede detectar todos los tipos de faltas. La Tabla 4 detalla las faltas sembradas en cada programa. La descripción de los tipos concretos de faltas sembradas en los programas está disponible en el reporte del experimento original [Juristo et al., 2013]. Recuérdese que las faltas InScope para una técnica son de tipo OutScope para la otra, y viceversa.

Tabla 3
Tipos de Faltas

Faul type (T)	**Description**
T1	Especificación sin implementar
T2	Datos de prueba para lograr la cobertura
T3	Combinación de clases de equivalencia no válidas
T4	Combinación de clases de equivalencia válidas elegidas
T5	Datos de prueba para la combinación de clases
T6	Detalle de implementación
T7	Implementación de una funcionalidad no especificada

Tabla 4
Distribución de las Faltas en los Programas

	Técnica	FT	F1	F2	F3	F4	F5	F6
Cmdline	EP	1	X		X			
		2		X				
	BT	3				X		
		4						X
		5					X	
	Técnica	FT	F1	F2	F3	F4	F5	F6
Nametbl	EP	1	X					
		2		X	X			
	BT	3					X	
		5						X
		7				X		
	Técnica	FT	F1	F2	F3	F4	F5	F6
Ntree	EP	2	X	X	X			
	BT	3					X	
		5				X		
		6						X

3.5.3. Material Experimental

El material utilizado para la ejecución del experimento incluye: formulario de recogida de datos de los sujetos experimentales, especificaciones de los programas, formularios de recolección de datos experimentales, impresos del código fuente con faltas sembradas, código objeto de los programas con faltas sembradas y documento guías para la ejecución de la replicación.

3.6. Variables de Contexto

En el experimento original se consideraron de forma explícita las siguientes variables contextuales.

- **Entorno:** Academia.
- **Tipo de Sujetos:** Estudiantes de pregrado.
- **Experiencia:** Los estudiantes tienen poca o ninguna experiencia profesional en desarrollo de software.
- **Tipo de Programa:** Programas de tamaño pequeño (150-220 LOC) y complejidad ciclomática en un rango de 21 a 61.
- **Lenguaje del Programa:** En lenguaje C.

3.7. Procedimiento de Ejecución

En la ejecución del experimento original, se distinguen tres etapas claramente definidas, que los experimentadores originales han denominado pre sesión, durante sesión y post sesión.

3.7.1. Pre Sesión

La pre sesión es una etapa del experimento en la cual los sujetos experimentales reciben el entrenamiento sobre la aplicación de las técnicas de testing. Paralelamente son preparados los materiales para la ejecución del experimento. Estos materiales, disponibles en [Juristo et al., 2013], son los siguientes:

- Objetos experimentales
- Formas
- Guías

3.7.2. Durante Sesión

Esta etapa es la ejecución del experimento propiamente dicha. Los sujetos ocupan una sala habilitada al efecto, suficientemente separados para evitar que puedan copiarse entre si. Con anterioridad a la primera sesión, se realiza la aleatorización de sujetos a grupos experimentales. Ello permite que durante la sesión los experimentadores puedan entregar a los sujetos los materiales experimentales adecuados. Para la técnica EP los sujetos experimentales son provistos de:

- La especificación del objeto experimental (especificación del programa cmdline, nametbl ó ntree).
- Forma de colección de datos para el registro de los casos de prueba.

Una vez que los casos de prueba han sido generados, se proporciona el código ejecutable con las faltas sembradas, para probar los casos generados.

Para la técnica BT el sujeto experimental es provisto de:

- Listado impreso del código fuente con faltas sembradas, para la generación de los casos de prueba.
- Formas de colección de datos, para el registro de los casos de prueba.

Una vez que los casos de prueba han sido generados, se proporcionan:

- El código ejecutable con faltas sembradas.
- Las especificaciones del programa, para probar los casos generados.

3.7.3. Post Sesión

En esta face, se recoge todo el material utilizado por los sujetos experimentales, del cual son separadas las formas de colección de datos rellenadas con los casos de prueba diseñados por los sujetos experimentales. Dichas formas son sometidas a un proceso de análisis, el cual consiste en comparar los casos de prueba generados por los sujetos con plantillas de casos de prueba previamente elaboradas. Dicho análisis revela las faltas descubiertas por cada sujeto al aplicar cada una de las técnicas.

3.8. Resumen de Resultados del Experimento Original

El análisis fue realizado mediante una ANOVA de medidas repetidas (rANOVA), siendo considerados como factores: la técnica (BT, EP, CR), el programa (ntree, cmdline, nametbl) y el grupo (en el experimento original se definieron 6 grupos, denominados G1-G6). Las sesiones no fueron consideradas explícitamente como factores, al estar confundidas con los programas. Los factores sesión/programa y técnica son considerados como factores intra-sujetos. El modelo ajustado fue puramente aditivo, de la forma:

$$Y = TECNICA + PROGRAMA + GRUPO + e$$

Los resultados obtenidos por los experimentadores originales pueden ser sumarizados del modo siguiente:

3.8.1. Variable Respuesta InScope

La rANOVA proporcionó resultados significativos en todos los factores: técnica, programa/sesión y grupo:

Técnica

El resultado significativo en la rANOVA hace que se rechace la hipótesis nula. La comparación por pares mostró que la técnica de code reading es menos efectiva que las técnicas de *equivalence partitioning* y *branch testing*. Esto es, la técnica de code reading detecta aproximadamente el 54 % de las faltas, mientras que las técnicas de branch testing y equivalence partitioning detectan un 67.67 % y el 78.70 % de las faltas sembradas respectivamente.

Programa/Sesión

El resultado significativo en la rANOVA sugiere que o bien el programa, o bien la sesión influyen en la efectividad, sin que pueda decidirse de forma fehaciente cuál de ellos afecta realmente al hallarse confundidos. La interacción programa/sesión, si bien teóricamente puede afectar, no parece razonable que lo haga, ya que no es posible identificar ninguna razón por la cual un programa puede aumentar la efectividad de una sesión, o viceversa.

La comparación por pares muestra que cmdline<nametbl<ntree (se utiliza el símbolo "<"para indicar que la efectividad de los sujetos es menor para cmdline que para nametbl, y lo mismo para las restantes variables). Alternativamente, S1<S3<S2. Las diferencias solo son significativas para cmdline/S1 y ntree/S2. Dado que las variables sesión y programa están confundidas en el tipo de diseño utilizado, existen tres motivos que pueden justificar los efectos observados: maduración (aprendizaje), fatiga o que un programa sea más fácil de testear que otro. En presencia de un efecto de aprendizaje, se debería observar que S1<S2<S3. En caso de un efecto de fatiga (poco probable, por otra parte, al estar las sesiones separadas por una semana), sería esperable que S3<S2<S1. En consecuencia, es el programa el que tiene efecto sobre la efectividad, más no la sesión.

Grupo

Como ya se indicó anteriormente, el factor grupo fue introducido para detectar la presencia de carryover, al tratarse el experimento original de un diseño within-subjects.

Se conoce como carryover al efecto residual que la aplicación de un tratamiento sobre un sujeto ejerce en otro tratamiento aplicado posteriormente sobre el mismo

sujeto, de modo que dicho efecto residual incrementa o decrementa la efectividad del tratamiento posterior [Brown, 1980]. El carryover es un riesgo importante en los experimentos en medicina, ya que los residuos de un fármaco pueden permanecer en el cuerpo bastante tiempo, interaccionando con tratamientos posteriores [Senn, 2002]. En IS, una causa subyacente que produzca un efecto de carryover es más difícil de imaginar. No obstante, la existencia de carryover ha sido explícitamente planteada en la literatura de ISE [Kitchenham et al., 2003], por lo que es razonable tenerlo en consideración.

La comparación por pares señala diferencias significativas de efectividad entre las secuencias EP-CR-BT, EP-BT-CR y BT-CR-EP con respecto al orden en el cual fueron aplicadas las técnicas. La secuencia EP-CR-BT es más efectiva en la detección de faltas con un 82.41 %, seguida de las secuencias EP-BT-CR y BT-CR-EP con un 55.55 % y 57.41 %, respectivamente. Dado que existe algunas diferencias significativas entre algunos niveles de este factor pero no entre otros, los experimentadores originales afirmaron que no parece existir ningún efecto de carryover de una técnica a otra.

3.8.2. Variable Respuesta OutScope:

El rANOVA proporcionó resultados significativos para todos los factores: técnica, programa/sesión y grupo.

Técnica

El resultado significativo en la ANOVA hace que se rechace la hipótesis nula. La comparación por pares mostró que la técnica de equivalence partitioning es menos efectiva (14.12 %) que la técnica branch testing (29.09 %).

Programa/Sesión

Igualmente en este caso, el resultado en la ANOVA es significativo, lo que sugiere que o bien la sesión, o bien el programa influyen en la efectividad. La confusión entre el programa y la sesión impide decidir de forma fidedigna cual influyen en la efectividad.

La comparación por pares muestra que cmdline<nametble (o, en términos de sesiones S1<S3). Ntree (o S2) no presenta diferencias significativas respecto a los otros dos programas (sesiones). Esta diferencia en la efectividad entre S1 y S3, podría atribuirse a la sesión y no al programa. Sin embargo, los resultados obtenidos para la efectividad de las técnicas para faltas InScope y el hecho de que S3 y S2 se comportan de forma similar, sugiere a los experimentadores originales que es el programa y no

la sesión que hace la diferencia. En este caso, no habría efecto de aprendizaje, al igual que en las faltas InScope.

Grupo

La comparación por pares señala diferencias significativas de efectividad entre las secuencias EP-CR-BT (8.33 %), CR-BT-EP y EP-BT-CR (35.42 % y 36.11 %, respectivamente) solamente con respecto al orden en el cual fueron aplicadas las técnicas. Al igual que en la variable InScope, los experimentadores originales afirmaron que no parece existir ningún efecto de aprendizaje o de carryover que mejore la efectividad de las técnicas para faltas fuera de su alcance debido al orden de aplicación de las técnicas.

Capítulo IV

REPLICACIÓN EXPERIMENTAL

En este apartado se detalla la *replicación experimental*, poniendo énfasis en: Su origen, objetivos, motivación, soporte, cambios respecto al experimento original, operación y resultados.

4.1. Introducción

Las actividades de verificación y validación (V&V) juegan un papel fundamental en el mejoramiento de la calidad del software. Existe una gran variedad de aproximaciones para llevar a cabo la V&V pero a ciencia cierta se desconoce qué técnica o combinación de técnicas resulta ser más efectiva para la validación del software en cualquiera de sus modalidades: unidad, integración o sistema.

La determinación de la efectividad de las técnicas para la realización de la prueba de unidad atrajo pronto la atención de la investigación experimental en Ingeniería de Software (IS). En fechas tan tempranas como 1978, Myers comparó la efectividad de las técnicas funcionales y estructurales de prueba [Myers, 1978]. Poco tiempo después, Basili y Selby estudiaron la efectividad de las técnicas funcional, estructural y code reading [Basili and Selby, 1985], iniciando una familia de experimentos que se ha replicado en múltiples ocasiones.

En este trabajo, se presenta la *replicación experimental* de un experimento controlado perteneciente a la familia de Basili, que, como se indicó en el capítulo III, se realizó en Diciembre de 2005 en la Universidad Politécnica de Madrid (UPM). De acuerdo a Gómez et al. [Gómez et al., 2010], en comparación con el experimento original, la presente replicación puede caracterizarse como: literal (esto es, la replicación es lo más exacta posible al experimento original), conjunta (parte de los experimentadores originales participaron en la replicación) y externa (la replicación

se ha llevado a cabo en un sitio distinto).

La replicación fue llevada a cabo en la Universidad de las Fuerzas Armadas - ESPE Sede Latacunga (ESPEL) de Ecuador, con estudiantes de la tercera promoción de la Maestría en Ingeniería de Software que cursaban la asignatura de Evaluación de Sistemas Software. Esta asignatura tiene una duración de 80 horas lectivas divididas en 7 sesiones presenciales (de 10 horas cada una) y un periodo no presencial (de 10 horas), dedicado para tareas extras y temas administrativos académicos. Las sesiones presenciales se dividieron en 3 etapas: Una primera etapa de 3 sesiones consecutivas, donde se impartió la primera parte del marco teórico del entrenamiento. A continuación se tuvo 3 días de descanso. Seguidamente, la segunda etapa (de dos sesiones consecutivas) cubrió la segunda parte teórica y el desarrollo de los ejercicios de entrenamiento. A día seguido, se ejecutó la replicación en dos sesiones consecutivas, correspondientes a la tercera etapa. La replicación constituyó una de las pruebas de evaluación de mayor ponderación del curso, lo cual permite asegurar la motivación de los sujetos.

4.2. Motivación para Realizar la Replicación

La razón principal que motivó la realización de una replicación fue confirmar los resultados obtenidos en el experimento original ó, en caso de discrepancias, identificar cómo los factores o parámetros pueden afectar a la aplicación de las técnicas de testing. Secundariamente, se espera que la replicación independiente de un experimento permita mejorar las habilidades en la aplicación de métodos empíricos en la investigación en IS.

4.3. Nivel de Interacción con los Experimentadores Originales

La conducción de la replicación tuvo un alto nivel de interacción con los experimentadores originales, durante las fases previas a la ejecución de la replicación experimental. La ejecución de la replicación se desarrolló de forma independiente de los experimentadores originales. Finalmente, la obtención y análisis de los datos contó con una mínima participación de los experimentadores originales.

(a) El inicio de la replicación experimental estuvo marcado con el discernimiento profundo del experimento original, para lo cual se mantuvo varias reuniones con los experimentadores originales. Una vez que el experimento fue bien comprendido, el siguiente paso fue la adaptación del diseño del experimento original al contexto donde la replicación iba a realizarse. Como resultado de esta adaptación, se generó un documento de diseño, el cual fue validado por los experimen-

tadores originales. Finalmente, fueron facilitados los artefactos del experimento original.

(b) Durante la preparación de los materiales para el entrenamiento, la realización del entrenamiento propiamente dicho y la ejecución de la replicación, no se tuvo interacción alguna con los experimentadores originales.

(c) Una vez finalizada la ejecución de la replicación, los experimentadores originales dieron indicaciones del procedimiento para identificar si un caso de prueba ejecuta una falta o no a partir de los cuestionarios cumplimentados por cada sujeto experimental.

4.4. Cambios con Respecto al Experimento Original

En términos generales, la replicación es bastante fiel al experimento original en todos sus aspectos. Se conservó: las hipótesis, factores, faltas, variable respuesta, materiales y procedimiento experimental; con las siguientes excepciones:

- Se ha eliminado uno de los niveles del factor principal. En concreto, no se ensayó la técnica de code reading, esencialmente por la imposibilidad de realizar tres sesiones experimentales en el tiempo asignado a la realización de la replicación.
- Debido a la eliminación de una de las técnicas, una de las tres sesiones, así como uno de los programas, son innecesarios. En concreto, se ha eliminado el programa cmdline, y se ha reducido el número de sesiones de tres a dos.
- Se ha alterado el orden de uso de los programas para estudiar si es el programa o la sesión el que influye en la efectividad de las técnicas.
- Se ha aplicado una aleatorización estratificada [Kernan et al., 1999] para asignar los sujetos a los grupos experimentales, con el objeto de conseguir grupos balanceados.
- Se ha adaptado el entrenamiento a las restricciones temporales del curso en el marco del cual la replicación fue realizada.
- Se ha localizado los materiales experimentales a las particularidades dialectales del español de Ecuador.

Un resumen de los cambios realizados puede observarse en la tabla 5. A continuación se describe dichos cambios en mayor detalle.

Tabla 5
Diferencias entre UPM & ESPEL

Fase	Característica	UPM	ESPEL
Diseño	Numero de Sujetos	46	23
	Aleatorización	Normal	Estratificada
	Sesiones	3	2
	Factor Principal	CR, BT, EP	BT, EP
	Programas	cmdline, ntree, nametbl	nametbl, ntree
	Grupos	6	2
	Diferencias dialécticas en el material	Castellano	Ecuatoriano
Entrenamiento	Tipo de entrenamiento	Presencial	Semipresencial
	Duración	12 horas	50 horas
Ejecución	Duración	3 sesiones, tiempo ilimitado	2 sesiones, tiempo ilimitado

4.4.1. Cambios en los Niveles del Factor Principal

La planificación del curso de Evaluación de Sistemas Software permitía dedicar únicamente 2 días a la realización del experimento, mientras que en el experimento original se habían dedicado 3 días (uno por sesión). En esta situación, cabían dos alternativas:

1. Concentrar dos sesiones experimentales el mismo día.
2. Eliminar uno de los niveles del factor técnica.

Una tercera opción, que consistía en dividir una sesión entre 2 días, fue inmediatamente descartada al existir el riesgo de que la interrupción afectara el rendimiento de los estudiantes ó simplemente, a que éstos se copiaran entre sí.

De las dos opciones posibles, se eligió finalmente la segunda. En contra de la primera opción, estaba el hecho de que los días asignados a las sesiones experimentales estaban contiguos (Sábado y Domingo), por lo que realizar 3 sesiones implicaba un esfuerzo considerable de los alumnos y, por lo tanto, existía el riesgo de que el consiguiente cansancio influyera negativamente en los resultados. A favor de la segunda opción estaba, adicionalmente, el hecho de que las técnicas funcional y estructural eran realmente los tratamientos ensayados en el experimento original, mientras que la técnica de lectura de código se trataba fundamentalmente de un control y, por lo tanto, su aplicación resultaba opcional.

Por lo tanto, los niveles seleccionados finalmente fueron la técnica funcional de

pruebas por particiones de equivalencia (EP) y la técnica estructural de pruebas de control de flujo por cobertura de decisión (BT), aplicados uno en cada sesión, por dos grupos de sujetos experimentales (Grupo 1 y Grupo 2). La tabla 6 muestra el diseño experimental utilizado.

Tabla 6
Diseño Experimental - Replicación ESPEL

Session	**Session 1**		**Session 2**	
Program	**Nametbl**		**Ntree**	
Technique	**BT**	**EP**	**BT**	**EP**
Group 1	**X**	**-**	**-**	**X**
Group 2	**-**	**X**	**X**	**-**

4.4.2. Cambios en los Niveles de los Factores Secundarios

La eliminación de una de las técnicas de prueba (code reading) produce que una de las tres sesiones sea innecesaria. En consecuencia, ha sido necesario eliminar uno de los programas usados en el experimento original. El programa descartado ha sido cmdline, por dos razones esenciales:

1. En el experimento original, así como en experimentos de la misma familia [Juristo and Vegas, 2003], los sujetos experimentales manifestaron que *cmdline* era un programa más difícil de comprender y testear que *ntree* y *nametbl*. La mayor dificultad de cmdline queda también de manifiesto en la escasa efectividad del testing sobre este programa, tal como y se ha indicado en la sección 3.8.

2. Los programas ntree y nametbl son similares entre si (LOC 146-172 y complejidad ciclomática 21-29 respectivamente), y a su vez distintos de cmdline (LOC 209, complejidad ciclomática 61). La alta complejidad ciclomática de cmdline puede explicar la mayor dificultad en su testeo.

En el experimento original (ver tabla 2), el programa cmdline fué utilizado en la sesión 1, mientras que ntree y nametbl fueron utilizados en las sesiones 2 y 3, respectivamente. Esto implica a primera vista que las sesiones en el experimento original y en la replicación no son comparables, ya que están asociadas a programas distintos. Sin embargo, esto no debería suponer un problema ya que no existe ninguna relación íntima, más allá de lo puramente coincidente, entre sesión y programa, como ya se ha indicado anteriormente.

4.4.3. Cambio en el Orden de Uso de los Programas

Tal y como se indicó en la sección 3.8, la confusión programa/sesión no permite distinguir, durante el análisis, si los efectos se deben a uno u otro factor. Esta es una particularidad del diseño crossover[1] utilizado en el experimento original. En la presente replicación, dicha confusión sigue existiendo ya que fue respetado el tipo de diseño original. Por este motivo, se decidió cambiar el orden de uso de los programas desde ntree-nametbl del experimento original a nametbl-ntree. Comparando los experimentos original y replicado, se espera poder identificar si los posibles efectos se deben realmente al programa o a la sesión.

4.4.4. Balanceo de los Grupos Experimentales

El diseño de la replicación precisó la conformación de dos grupos de sujetos experimentales a los que se denominó Grupo 1 y Grupo 2. Para la conformación de estos grupos se contó con 23 sujetos experimentales que eran estudiantes de maestría, a diferencia del experimento original, donde los sujetos fueron estudiantes de pregrado. Dado que los estudiantes de maestría pueden potencialmente haber desempeñado actividades profesionales, se realizó una encuesta previa a 21 de ellos (2 no asistieron el día de la encuesta), para conocer mejor su experiencia profesional, ya que se hipotetizó que los sujetos expertos en programación y/o testing aplicarían las técnicas de forma más efectiva que los sujetos con menor experiencia.

En la encuesta, los sujetos experimentales fueron indagados puntualmente sobre: Nivel de experiencia en programación en general, experiencia en programación en el lenguaje C y experiencia en testing de software. Los resultados se muestran gráficamente en las Figuras 7, 8 y 9, y en forma tabular en la tabla 7.

[1]Crossover: Es un diseño en el que los sujetos reciben una secuencia de diferentes tratamientos de forma aleatoria, donde los resultados o medidas resultantes pueden ser recogidas varias veces de un mismo sujeto [Senn, 2002].

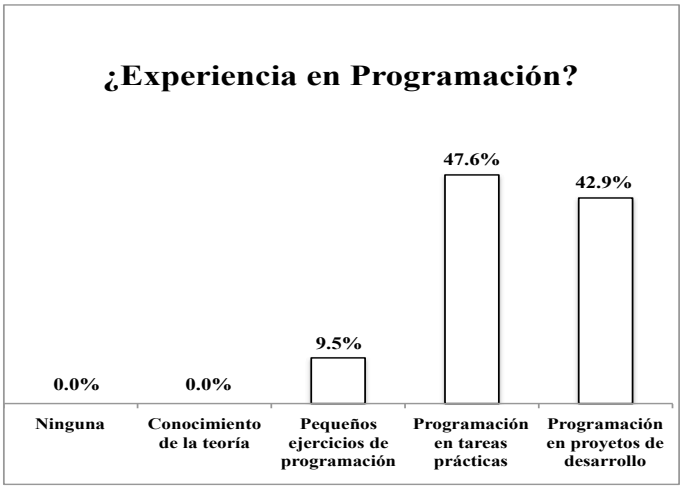

Figura 7: Resultados de la Encuesta - Conocimientos en Programación

Figura 8: Resultados de la Encuesta - Conocimientos en Programación en C

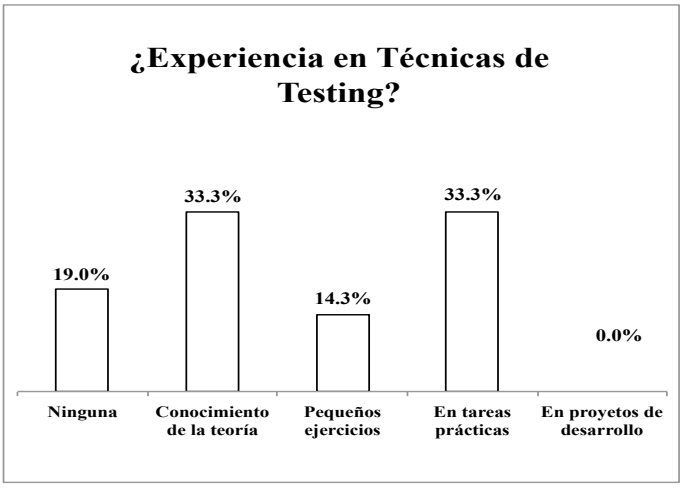

Figura 9: Resultados de la Encuesta - Conocimientos en Testing

Puede observarse que casi la mitad de sujetos experimentales son expertos en programación y la otra mitad solo han programado a nivel de prácticas. El lenguaje de programación C no es utilizado mayormente por los sujetos para desarrollo formal, pero todos lo conocen al menos en teoría. Las software testing techniques son desconocidas para un porcentaje considerable de sujetos experimentales (20 %), de hecho no existe ninguno que sea experto en ellas y en general su uso ha sido solo a nivel de prácticas.

Dado que los sujetos no eran testeadores con experiencia (en el mejor de los casos, la experiencia de todos ellos se limita a ejercicios de clase) ni programadores en C (el 81 % de los sujetos solo usa C en prácticas de clase en el mejor de los casos), se considera que dichas variables no deberían ejercer ningún efecto en los resultados del experimento. Por el contrario, la experiencia en programación varió considerablemente (aproximadamente un 40 % de los sujetos son programadores con experiencia profesional, frente a un 60 % de programadores que no tienen experiencia profesional), y es razonable asumir que dicha experiencia si puede ejercer un efecto. Por este motivo, se realizó una aleatorización estratificada [Kernan et al., 1999] basada en la experiencia de programación de los sujetos. Los dos sujetos que no fueron encuestados, cada uno fue ubicado aleatoriamente en un grupo diferente (G1 (BT-EP) o G2 (EP-BT)).

De esta manera, se conformaron dos grupos de sujetos experimentales que fueron equilibrados con respecto a los conocimientos de programación. En lo que respecta a la experiencia en programación en C, ambos grupos resultaron balanceados en el número de sujetos que utilizaron C en la industria o academia (al rededor del 20 % y 80 %, respectivamente), tal y como puede observarse en la Figura 10. Se aprecia una leve ventaja de los sujetos del grupo BT-EP en términos del tipo de experiencia en C que adquirieron en la academia. Un 54 % de sujetos del grupo BT-EP han usado C en tareas de prácticas, mientras que el 27.3 % restante sólo han utilizado C a nivel teórico o en pequeños ejercicios. En el grupo EP-BT los porcentajes son del 30 % y 50 % respectivamente.

La experiencia en software testing techniques parece ser también ligeramente superior en el grupo BT-EP, tal y como se indica en la Figura 11. Un 45.5 % de los sujetos tienen experiencia en la realización de tareas de prácticas, mientras que el 54.5 % restante tiene solo experiencia a nivel teórico o de pequeños ejercicios. En el grupo EP-BT, estos porcentajes son del 20 % y 80 %, respectivamente.

Esta falta de balanceo entre grupos podría originar que los sujetos del grupo BT-EP fueran en general más efectivos que los sujetos del grupo EP-BT. Esta posibilidad deberá tenerse en consideración durante la discusión de los resultados de la replicación, aunque en nuestra opinión tal posibilidad es reducida. Las diferencias entre grupos se restringen a que algunos sujetos en el grupo BT-EP han realizado más tareas prácticas que los miembros del grupo EP-BT. Esta diferencia es escasa. Adicionalmente, todos los sujetos han recibido, antes del experimento, entrenamiento específico en técnicas de testing, el cual incluye ejercicios prácticos. Este entrena-

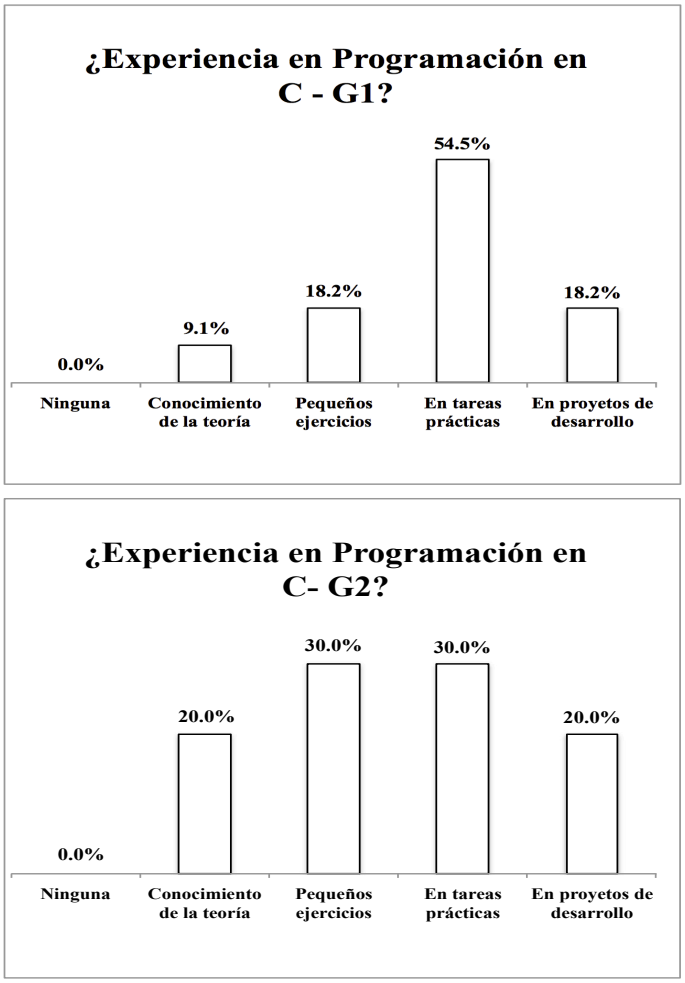

Figura 10: Resultados de la Aleatorización Estratificada para Programación en C

miento debería haber reducido aún más las diferencias entre grupos.

4.4.5. Adaptación del Entrenamiento

A diferencia de la modalidad de entrenamiento aplicada en el experimento original, organizada en tres sesiones de 4 horas cada una más tarea, con diferencia de una semana entre sesiones, el entrenamiento de la replicación tuvo que adaptarse a la modalidad de estudios de ESPEL, que es semi presencial, con 5 sesiones consecutivas de 10 horas cada una, durante las cuales se realizaron todos los ejercicios prácticos. Dado el modo de entrenamiento, el factor cansancio constituye una amenaza a la validez, que puede haber influido en el desenvolvimiento de los sujetos.

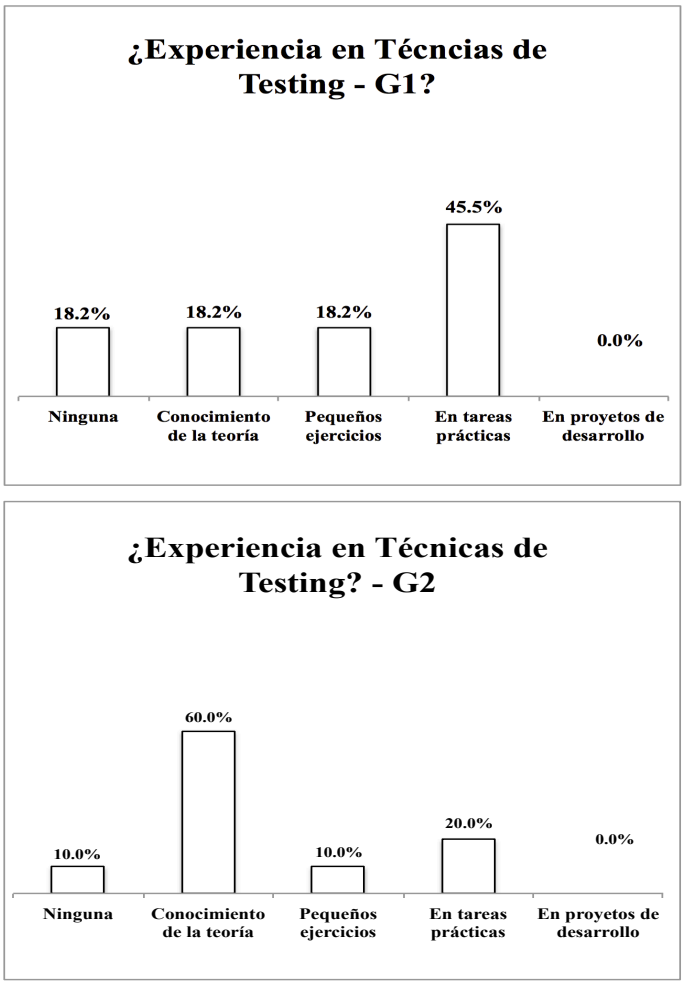

Figura 11: Resultados de la Aleatorización Estratificada para la Experiencia en Testing

4.4.6. Localización

El español de España (Castillian) y el español de Latinoamérica (en concreto, el hablado en Ecuador), si bien son similares sintácticamente, se diferencian ligeramente en el léxico y, sobre todo, en los modismos utilizados localmente. Por este motivo, y para facilitar la comprensión, se modificaron detalles como términos y frases que no eran comunes en el dialecto del entorno donde se realizó la replicación. Así mismo, se corrigieron algunas ambigüedades menores encontradas en el material original. El material utilizado para la replicación se encuentra disponible en `http://www.grise.upm.es/sites/extras/12/`.

4.5. Ejecución de la Replicación

La ejecución de la replicación fue similar a la del experimento original. Los sujetos recibieron los mismos materiales experimentales (objetos experimentales, especificaciones de los programas, listados impresos del código fuente de los programas con faltas sembradas, formas, guías y los ejecutables de los programas con faltas sembradas), generaron casos de prueba y procedieron a ejecutarlos, para verificar que las faltas identificadas producían fallos. No se produjeron eventos relevantes (e.g.: deserciones, errores en la entrega de materiales, etc.) durante la ejecución experimental. El análisis posterior de los datos se realizó siguiendo el mismo procedimiento que en el experimento original.

4.6. Resultados de la Replicación

En esta sección se describe el resultado de la *replicación experimental*. Específicamente, se reporta los test de hipótesis y comparaciones múltiples para cada variable respuesta (InScope u OutScope). Las estadísticas descriptivas y la raw data son reportadas en las tablas 18, 19 y 20.

El diseño experimental utilizado, al igual que el experimento original, sugiere la realización de un *Análisis de Varianza de Medidas Repetidas* (rANOVA) y el mismo modelo aditivo. Todos los cálculos han sido realizados utilizando SPSS V.20.

4.6.1. Variable Respuesta: Efectividad para Faltas InScope (Dentro del Alcance de la Técnica)

Previo al análisis estadístico, se ha comprobado que la muestra posea las propiedades de esfericidad y homocedasticidad, necesarios para realizar un rANOVA. Los estadísticos de Mauchly's W y Box's M, confirmaron dichas propiedades (W= 1.000, Approx. Chi-Square=0.000, df=0, Sig.=0.000; M=3.755, F=1.122, df1=3, df2=111064.484, Sig.=0.338).

Dado que el análisis contiene factores intra sujetos e inter sujetos, en lugar de la típica tabla de ANOVA, se obtienen dos tablas distintas, una para cada tipo de factor. Los resultados que indican las Tablas 8 y 9, muestran que tanto la técnica, como session/program y el grupo influencian la efectividad (en las dos tablas el valor de la prueba de hipótesis F es mayor al F para $\alpha = 0{,}05$) sobre faltas dentro del ámbito.

Por lo tanto, se rechaza la hipótesis nula (es decir se rechaza que: no hay diferencia en la efectividad de particiones de equivalencia y branch testing con respecto a la detección de faltas dentro de su ámbito) para esta variable respuesta.

Respecto del factor **technique**, los resultados de bonferroni pairwise comparision que se muestran en la tabla 10, indican que *branch testing* sería menos efectiva (diferencia entre medias (Mean Dif.) = -13.197) que *equivalence partitioning* (con efectividad de 31.943 % y 45.140 % respectivamente).

Respecto de **session/programa**, las comparaciones por pares indican que **S1/nametbl** resulta más efectiva (diferencia entre medias (Mean Dif.) = 20.140) que **S2/ntree** (con efectividad de 48.612 % y 28.471 % respectivamente) como muestra la tabla 11. Debido a que en una sesión se utiliza un solo programa, no es posible distinguir en este momento si el efecto es producido por la sesión, por el programa o por ambos. Se intentará dilucidar si la sesión o el programa producen el efecto observado cuando se estudie las faltas OutScope y, en especial, se compare los resultados de la replicación con el experimento original.

Los resultados de la comparación por pares para el factor del **Grupo** sugieren que el grupo BT-EP (que aplica *branch testing* en la 1º sesión seguido de *equivalence partitioning* en la 2º sesión) resulta menos efectivo (diferencia entre medias (Mean Dif.) = -22.918) que el grupo EP-BT (con efectividad de 27.082 % y 50.00 %, respectivamente) como se muestra en la tabla 12.

Hay dos explicaciones posibles para este resultado:

- Podría existir un efecto de carryover que influya en la efectividad de las técnicas dependiendo si son aplicadas en primer o en segundo lugar. El carryover representa el incremento (o reducción) en la efectividad de la técnica aplicada en segundo lugar por un sujeto.
- Tal y como se ha indicado anteriormente en la sección 4.4.4, el grupo EP-BT es ligeramente superior al grupo BT-EP en términos de experiencia en programación en C y software testing. Esta

superioridad podría explicar la mayor efectividad del grupo EP-BT.

Se intentará determinar si el efecto observado se debe a las diferencias entre grupos ó al carryover cuando se estudien las faltas OutScope y, posteriormente, se compare los resultados de la replicación con el experimento original.

4.6.2. Variable Respuesta: Efectividad para Faltas OutScope

(Fuera del Alcance de la Técnica)

Previo al análisis estadístico se ha comprobado si la muestra posee las propiedades de esfericidad y homocedasticidad. Los estadísticos de Mauchly's W y Box's M, confirmaron dichas propiedades (W= 1.000, Approx. Chi-Square=0.000 , df=0, Sig.=0.000; M=11.947, F=0.429, df1=3, df2=111064.484, Sig.=0.732).

Los resultados del análisis que se muestran en las tablas 13 y 14, indican que existen diferencias significativas para el factor technique, pero no así para los factores session/program y group. Por lo tanto, y al igual que en el caso de la variable respuesta InScope, se rechaza la hipótesis nula (en las dos tablas el valor de la prueba de hipótesis F es mayor al F para $\alpha = 0{,}05$).

Con respecto al factor **technique**, la tabla 15 muestra que *branch testing* resulta más efectiva (diferencia entre medias (Mean Dif.) = 15.910) que *equivalence partitioning* para las faltas fuera del alcance de la técnica (efectividad de 31.943 % y 11.489 % respectivamente). En la sección anterior, donde se mostró el análisis de las faults InScope, los resultados muestran la relación contraria (esto es, *branch testing* es menos efectiva que *equivalence partitioning* para las faltas within their scope). Esto indica que *equivalence partitioning* es más sensible a las faltas que están dentro de su alcance, pero *branch testing* tiene más habilidad para encontrar faltas fuera de su alcance.

Con respecto al factor **session/program**, las comparaciones múlti-

ples que se muestran en la tabla 16 indican que no existen diferencias significativas (diferencia entre medias (Mean Dif.) = 0.757) entre los niveles de este factor. Sin embargo, los resultados de la variable respuesta InScope que se mostraron en la sección 4.6.1 sí indicaban diferencias significativas. A simple vista, no parece haber ningún motivo por el cual el factor session/program se comporte de modo distinto dependiendo de los tipos de falta, por lo que no se puede aventurar ninguna hipótesis que permita explicar esta discrepancia, teniendo en cuenta solamente los resultados de la replicación.

Las comparaciones múltiples para el factor **group**, mostradas en la tabla 17, indican que no existen diferencias (diferencia entre medias (Mean Dif.) = 5.554) significativas entre BT-EP y EP-BT, al contrario de lo ocurrido para la variable respuesta InScope.

En la sección 4.6.1, se aventurában dos hipótesis que podrían explicar los resultados para las faltas InScope: la existencia de un efecto de carryover o, alternativamente, la mayor experiencia del grupo EP-BT en C y técnicas de testing. En cualquiera de estos dos casos, seria esperable que en este efecto fuera el mismo independientemente del tipo de falta. Considerando, de nuevo únicamente los resultados obtenidos en la replicación, no es posible aventurar una razón para la existencia de efectos significativos para la variable respuesta InScope.

Tabla 7
Datos Obtenidos de la Encuesta

Sujeto	Grupo	P1	P2	P3
S1	G2	3	3	2
S2	G1	NA	NA	NA
S3	G1	5	4	4
S4	G2	5	3	4
S5	G2	4	2	2
S6	G2	5	2	2
S7	G1	4	2	2
S8	G1	5	4	3
S9	G1	5	5	4
S10	G2	5	4	2
S11	G2	4	4	1
S12	G1	4	4	4
S13	G2	NA	NA	NA
S14	G1	4	5	4
S15	G1	3	4	1
S16	G1	4	3	3
S17	G1	4	4	1
S18	G2	4	3	3
S19	G2	5	5	4
S20	G2	4	5	2
S21	G1	5	3	2
S22	G1	5	4	4
S23	G2	4	4	2

P1 = Pregunta 1
P2 = Pregunta 2
P3 = Pregunta 3
Valor 1 = None
Valor 2 = Conozco la teoría
Valor 3 = Ejercicios pequeños
Valor 4 = Practica en asignaturas
Valor 5 = Proyectos de desarrollo

Tabla 8
Test de Efectos Intra Sujetos

Source	**Type III Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
Technique	1999.054	1	1999.054	4.517	0.046
Session/ Program	4655.948	1	4655.948	10.520	0.004
Error (Technique)	9294.241	21	442.583		

Tabla 9
Test de Efectos Inter Sujetos

Source	**Type III Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
Intercept	68201.322	1	68201.322	100.846	0.000
Group	6028.757	1	6028.757	8.914	0.007
Error	14202.195	21	676.295		

Tabla 10
Test de Comparaciones por Pares de Bonferroni para la Técnica

Tech1	**Tech2**	**Mean Dif.**	**Std. Dev.**	**Sig.**
BT	EP	-13.197	6.210	0.046

Tabla 11
Test de Comparaciones por Pares de Bonferroni para el Programa

Prog1	**Prog2**	**Mean Dif.**	**Std. Dev.**	**Sig.**
S1/nametbl	S2/ntree	20.140	6.210	0.004

Tabla 12
Test de Comparación por Pares de Bonferroni para el Grupo

Group1	**Group2**	**Mean Dif.**	**Std. Dev.**	**Sig.**
BT-EP	EP-BT	-22.918	7.676	0.007

Tabla 13
Test de Efectos Intra Sujetos

Source	**Type III Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
Technique	2905.36	1	2905.36	5.567	0.028
Session/ Program	6.577	1	6.577	0.013	0.912
Error (Technique)	10959.95	21	521.902		

Tabla 14
Test de Efectos Inter Sujetos

Source	**Type III Sum of Squares**	**df**	**Mean Square**	**F**	**Sig.**
Intercept	17357.588	1	17357.588	20.086	0.000
Group	354.129	1	354.129	0.41	0.529
Error	18147.296	21	864.157		

Tabla 15
Test de Comparaciones por Pares de Bonferroni para la Técnica

Tech1	**Tech2**	**Mean Dif.**	**Std. Dev.**	**Sig.**
BT	EP	15.910	6.743	0.028

Tabla 16
Test de comparaciones por Pares de Bonferroni para el Programa

Prog1	**Prog2**	**Mean Dif.**	**Std. Dev.**	**Sig.**
Session 1 / nametbl	Session 2 / ntree	0.757	6.743	0.912

Tabla 17
Bonferroni Pairwise Comparisons Test For Group

Group1	**Group2**	**Mean Dif.**	**Std. Dev.**	**Sig.**
BT-EP	EP-BT	5.554	8.677	0.529

Tabla 18
Descriptive Statistics InScope Variable

Technique	**N**	**Mean**	**Std. Dev.**
Branch Testing	23	31.883	25.581
Equivalence Partitioning	23	44.204	29.988

Tabla 19
Descriptive Statistics OutScope Variable

Technique	**N**	**Mean**	**Std. Dev.**
Branch Testing	23	31.883	25.581
Equivalence Partitioning	23	11.593	16.231

Tabla 20
Raw-Data de la Replicación Experimental

S	G	T	P	Se	Vis. for EP			Vis. for BT			In	Out
					F1	F2	F3	F4	F5	F6		
1	2	EP	Na	1	1		1				66,67 %	0,00 %
1	2	BT	Nt	2					1		33,33 %	0,00 %
2	1	BT	Na	1		1					0,00 %	33,33 %
2	1	EP	Nt	2	1						33,33 %	0,00 %
3	1	BT	Na	1							0,00 %	0,00 %
3	1	EP	Nt	2	1						33,33 %	0,00 %
4	2	EP	Na	1			1				33,33 %	0,00 %
4	2	BT	Nt	2							0,00 %	0,00 %
5	2	EP	Na	1	1		1				66,67 %	0,00 %
5	2	BT	Nt	2							0,00 %	0,00 %
6	2	EP	Na	1			1		1		33,33 %	33,33 %
6	2	BT	Nt	2							0,00 %	0,00 %
7	1	BT	Na	1							0,00 %	0,00 %
7	1	EP	Nt	2	1	1			1		66,67 %	33,33 %
8	1	BT	Na	1							0,00 %	0,00 %
8	1	EP	Nt	2							0,00 %	0,00 %
9	2	EP	Na	1		1	1				66,67 %	0,00 %
9	2	BT	Nt	2					1	1	66,67 %	0,00 %
10	1	BT	Na	1							0,00 %	0,00 %
10	1	EP	Nt	2							0,00 %	0,00 %
11	2	EP	Na	1		1	1				66,67 %	0,00 %
11	2	BT	Nt	2						1	33,33 %	0,00 %
12	1	BT	Na	1	1	1	1				0,00 %	100,00 %
12	1	EP	Nt	2	1					1	33,33 %	33,33 %
13	2	EP	Na	1							0,00 %	0,00 %
13	2	BT	Nt	2		1	1		1		33,33 %	66,67 %
14	1	BT	Na	1							0,00 %	0,00 %
14	1	EP	Nt	2							0,00 %	0,00 %
15	1	BT	Na	1		1					0,00 %	33,33 %
15	1	EP	Nt	2						1	0,00 %	33,33 %
16	1	BT	Na	1		1		1			33,33 %	33,33 %
16	1	EP	Nt	2			1	1			33,33 %	33,33 %
17	1	BT	Na	1		1					0,00 %	33,33 %
17	1	EP	Nt	2							0,00 %	0,00 %
18	2	EP	Na	1							0,00 %	0,00 %
18	2	BT	Nt	2				1		1	66,67 %	0,00 %
19	2	EP	Na	1	1	1					66,67 %	0,00 %
19	2	BT	Nt	2							0,00 %	0,00 %
20	1	BT	Na	1							0,00 %	0,00 %
20	1	EP	Nt	2					1		0,00 %	33,33 %
21	2	EP	Na	1	1	1	1		1		100,00 %	33,33 %
21	2	BT	Nt	2	1	1			1		33,33 %	66,67 %
22	1	BT	Na	1							0,00 %	0,00 %
22	1	EP	Nt	2							0,00 %	0,00 %
23	2	EP	Na	1	1	1					66,67 %	0,00 %
23	2	BT	Nt	2					1		33,33 %	0,00 %

S=Subject, G=Grupo, T=Técnica (EP=*Equivalence Partitioning*;
BT=*Branch Testing*)
P=Programa (Na=Nametbl; Nt=Ntree), Se=Sesión, F1-F6=Faltas
Vis.=Visible
In= Variable Respuesta InScope, Out= Variable Respuesta OutScope

Capítulo V

COMPARACIÓN DE RESULTADOS REPLICACIÓN - EXPERIMENTO ORIGINAL

En este Capítulo se hace una comparación minuciosa entre los resultados del experimento original y la *replicación*, poniendo énfasis en las coincidencias, así como en las diferencias encontradas entre los experimentos.

5.1. Introducción

Debido a que la replicación utiliza un subconjunto de los niveles de los factores del experimento original, algunos resultados de éste no podrán ser contrastados con la replicación y otros podrán ser contrastados únicamente de modo parcial. Respecto del factor técnica, los niveles *branch testing* y *equivalence partitioning* son comparables en ambos experimentos, dejando fuera de la comparación a la técnica *stepwise abstraction* que se utiliza únicamente en el experimento original.

El factor Sesión/Programa es parcialmente comparable, ya que los programas que se utilizan en ambos experimentos (nametbl y ntree) corresponden a sesiones diferentes en cada experimento (sesión 1º y 2º en ESPEL y sesión 3º y 2º en UPM respectivamente). Se definirán

claramente en cada caso si se habla del programa o de la sesión.

En el caso del factor grupo, son los niveles del factor técnica y las sesiones del experimento los que definen los niveles de cada grupo. Se cree que este factor no es comparable porque los niveles son distintos (en el experimento original se forman 6 grupos distintos y en la replicación 2). Los dos grupos pertenecientes a la replicación son subconjuntos de dos de los seis grupos del experimento original. Por lo tanto, la comparación solo puede realizarse a muy alto nivel.

Las secciones 5.2 y 5.3 contienen respectivamente las similaridades y las diferencias, entre el experimento original y la replicación.

5.2. Resultados Consistentes

Los resultados consistentes entre el experimento original y la replicación refieren al factor técnica, tanto para la variable respuesta InScope como para OutScope, tal y como se observa en la tabla 21. La tabla muestra un resumen de los resultados obtenidos en las comparaciones múltiples para el experimento original (UPM) y replicación (ESPEL), indicando si se rechaza o si se acepta la hipótesis nula (no hay diferencia en la efectividad de equivalence partitioning y branch testing con respecto a la detección de faltas) y la tendencia observada (relación de orden) entre los niveles del factor.

Tabla 21
Comparación UPM-ESPEL - Factor Técnica

Variable	**H0**		**Tendencia**	
Respuesta	**Upm**	**Espel**	**Upm**	**Espel**
InScope	Acepta	Rechaza	$BT < EP$	$BT < EP$
OutScope	Rechaza	Rechaza	$BT > EP$	$BT > EP$

5.2.1. Variable respuesta InScope

En ambos experimentos, *branch testing* resulta menos efectiva que *equivalence partitioning*, aunque con ciertos matices. En primer lugar,

y tal como se indica en la tabla 21, en ESPEL se han obtenido diferencias significativas entre los niveles de la técnica, mientras que en UPM no se ha observado tal diferencia.

En segundo lugar, los valores medios de efectividad (también conocidas como medias marginales) obtenidos para el factor técnica, en ESPEL son menores que en UPM (branch testing con 31.943 % vs. 67.670 % y equivalence partitioning con 45.140 % y 78.704 % respectivamente). Además de la diferencia entre las medias marginales, la dispersión[1] de la técnica de *branch testing* es mucho menor en ESPEL que en UPM, tal y como puede observarse en la Figura 12[2].

La menor dispersión de *branch testing* en ESPEL puede explicar por qué la diferencia entre esta técnica y equivalence partitioning es significativa en ESPEL. Otra cuestión es dilucidar porqué en ESPEL la efectividad de los sujetos es menor que en UPM. La explicación más plausible radica en el hecho de que el curso donde se ha desarrollado el experimento fuera intensivo (muchas horas lectivas concentradas en pocos días). Esto puede también haber influenciado de forma negativa en la efectividad de las técnicas, ya que los sujetos pueden no haber tenido tiempo suficiente para realizar prácticas y consolidar el uso de las técnicas, lo que perjudica a *branch testing*, al ser una técnica más compleja de entender y laboriosa de aplicar. Otro posible factor, además del método de enseñanza, es la inexperiencia del capacitador en dictar el curso de V&V, especialmente en tales circunstancias.

5.2.2. Variable Respuesta OutScope

Respecto de esta variable respuesta, en ESPEL se confirman los resultados obtenidos en UPM, ya que en ambos casos se rechaza la hipótesis nula, y adicionalmente, la tendencia es la misma, como se muestra en la tabla 21, en donde branch testing resulta ser más efectiva que equivalence partitioning para las faltas que están fuera de su

[1] La dispersión se refiere al intervalo de valores donde se concentran mayormente los correspondientes a los resultados de efectividad de las técnicas de testing.

[2] Los Boxplot son diagramas que representan la distribución de los resultados de efectividad de cada técnica de testing, destacando los valores: Primer cuartil, mediana, tercer cuartil, los valores mínimos y, los valores máximos de los datos.

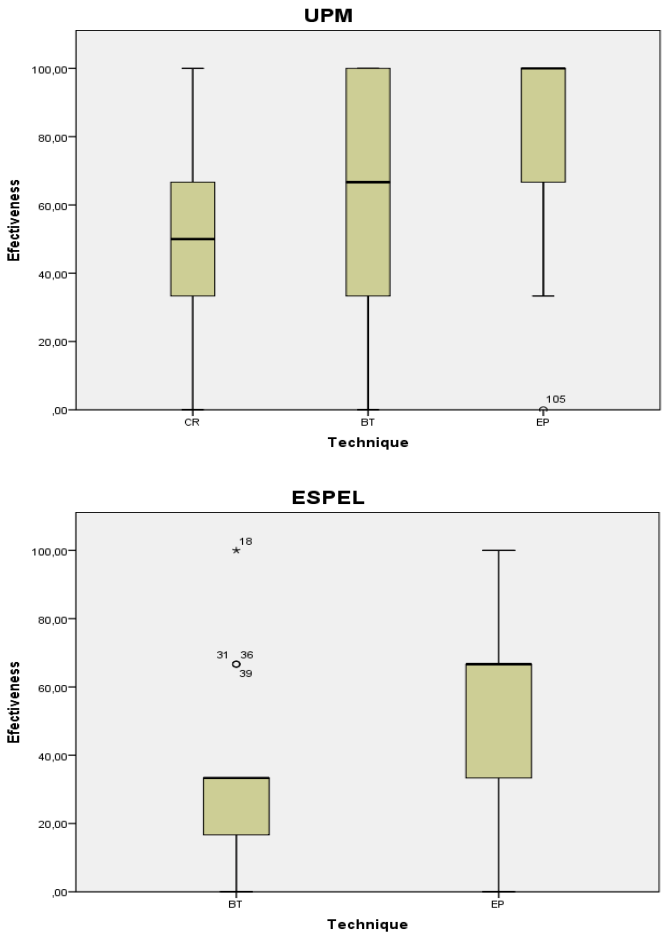

Figura 12: Boxplot para la Técnica en UPM y ESPEL - InScope

alcance.

Respecto de las medias marginales, los valores son ligeramente inferiores en ESPEL (*branch testing* con 27.398 % y *equivalence partitioning* con 11.489 %) que en UPM (branch testing con 29.089 % y equivalence partitioning con 14.119 %). Las dispersiones son también bastante parecidas en ambos casos, tal como se observa en la Figura 13. Resulta curioso que la baja efectividad de las técnicas en ESPEL sea mucho más acusada por las faltas InScope que para las faltas OutScope, ya que parece razonable suponer que las diferencias en el training (tanto el carácter intensivo del curso como una posible falta de experiencia del capacitador), afectase más o menos igual a ambas variables respuesta.

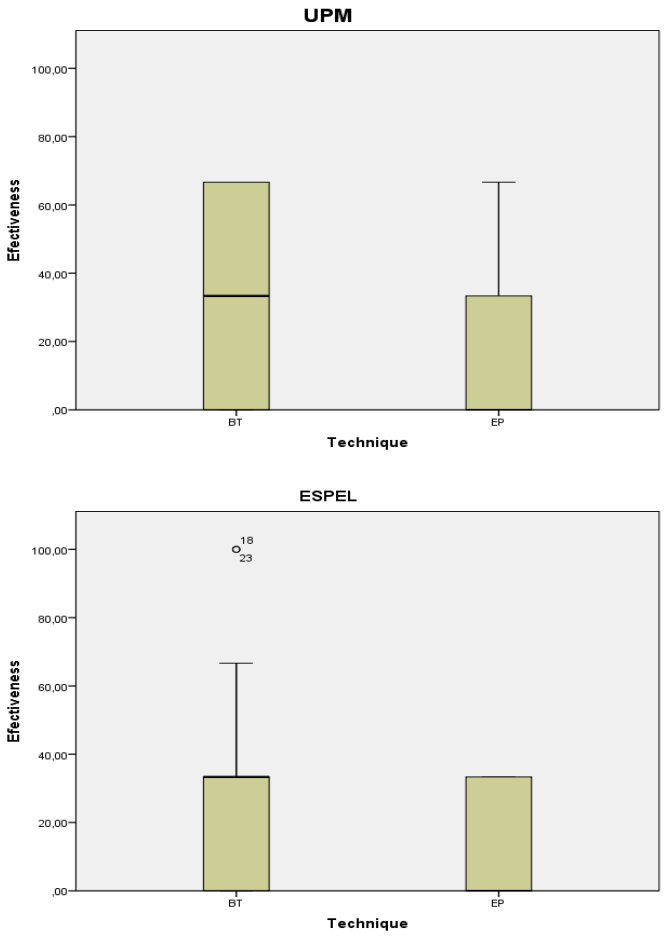

Figura 13: Boxplot para la Técnica en UPM y ESPEL - OutScope

Una posible explicación para el mejor desempeño de *branch testing* para faltas OutScope, es que su estrategia de generación de casos de prueba requiere del análisis del código fuente, punto en el que los sujetos podrían aplicar una revisión de código informal, y así complementar la técnica. Esta explicación es atractiva por dos motivos:

- Los sujetos que aplican la técnica *equivalence partitioning* no disponen del código fuente del programa que están probando (solo el ejecutable), tal y como se ha indicado en la sección 3.7.2. Por consiguiente, su habilidad para detectar faltas debería ser muy reducida. Este es, precisamente, el efecto observado, ya que la efectividad media de los sujetos experimentales es del 11.5 % o, lo que es lo mismo, cada sujeto identifica 0.3 faltas en promedio.

- Sin la ayuda de las técnicas de testing, sujetos que poseen características comparables (experiencia en programación, años de experiencia en la industria como es el caso de los estudiantes, etc.) deberían localizar aproximadamente las mismas faltas. De nuevo, éste es el efecto observado, ya que los sujetos UPM y ESPEL son parecidos y su efectividad es aproximadamente igual.

A pesar de que esta es una hipótesis razonable, tiene que ser corroborada en futuras replicaciones.

5.3. Diferencias en los Resultados

Las diferencias encontradas entre los resultados del experimento original y de la replicación se refieren a los factores sesión/programa y grupo. Ambos son tratados separadamente en lo que sigue.

5.3.1. Diferencias Relativas al Factor Sesión/Programa

La tabla 22 resume la comparación para el factor sesión/programa.

Tabla 22
Comparación UPM-ESPEL - Factor Sesión/Programa

Variable	**H0**		**Tendencia**	
Respuesta	**Upm**	**Espel**	**Upm**	**Espel**
InScope	Acepta	Rechaza	$S3 < S2$ $ntbl < ntree$	$S1 > S2$ $ntbl > ntree$
OutScope	Acepta	Acepta	$S3 > S2$ $ntbl > ntree$	$S1 < S2$ $ntbl < ntree$

Variable respuesta InScope

En lo que respecta al factor sesión/programa, el testing fue más efectivo en UPM cuando fue aplicado al programa ntree; mientras que justo lo contrario sucedió en ESPEL, en el caso de nametbl, como se

muestra en la Figura 14[3]. Adicionalmente, las diferencias son significativas en ESPEL, mientras que en UPM no lo son.

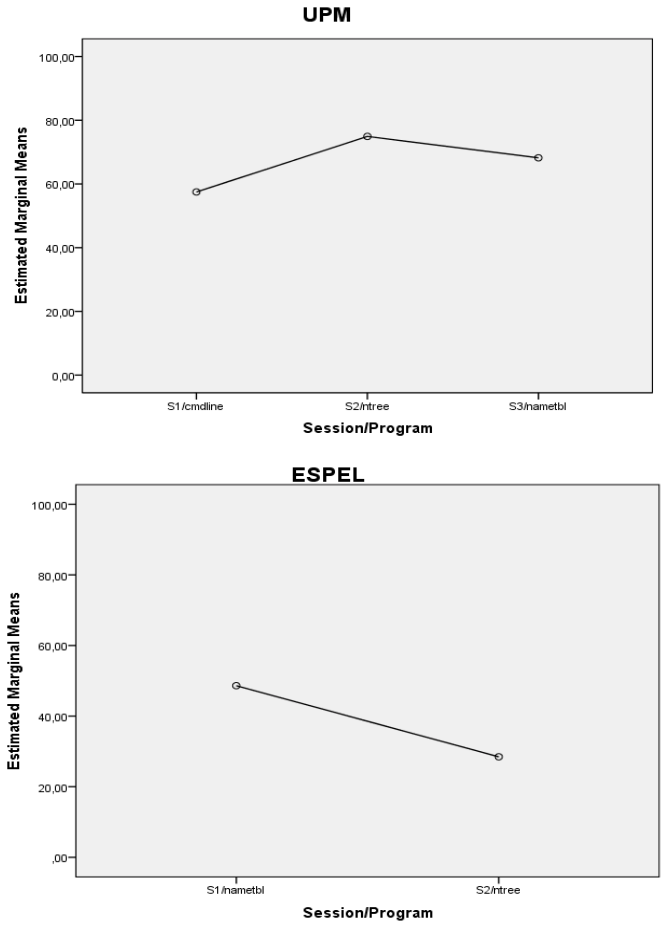

Figura 14: Medias Marginales Estimadas para Sesión/Programa en UPM y ESPEL - InScope

Una de las diferencias que se ha introducido en el diseño de la replicación es el cambio de orden en la aplicación de los programas ntree y nametbl, para poder dilucidar si es la sesión o programa (como afirman los experimentadores originales) los que motivan los efectos observados.

Dado que los resultados obtenidos en ESPEL son diametralmente contrarios a los obtenidos en UPM, la hipótesis más razonable es que el factor causante debe ser la sesión y no el programa.

[3]Los Diagramas de Perfil muestran las medias resultantes de la efectividad de cada técnica de testing, a modo de tendencia o comparación.

El mecanismo por el cual la sesión actúa sobre la efectividad podría residir en un efecto de fatiga. Nótese que el experimento es realizado en un programa académico intensivo y las sesiones experimentales se llevaron a cabo solamente con un día de diferencia entre ellas. Por lo tanto, es perfectamente posible que los sujetos ya llegasen fatigados a las sesiones experimentales (a este respecto, es apreciable que las medias marginales de efectividad siempre son menores en ESPEL que en UPM), y que, en particular, la efectividad de S2 fuera menor que la de S1 (lo cual es fácilmente apreciable en la Figura 14).

La hipótesis de un posible efecto de fatiga es atractiva por dos motivos. En primer lugar, la brusca caída de efectividad entre las sesiones S1 y S2 sería el motivo de que las diferencias S1/nametbl y S2/ntree hayan resultado significativas. Si dicha fatiga no existiera, las diferencias hubieran sido menores y es posible que no significativas, que es precisamente lo observado en UPM (donde las sesiones estuvieron separadas por una semana). En segundo lugar, el tamaño de efecto para el factor sesión/programa (mostrado en la tabla 11), es anormalmente alto en comparación con el del factor técnica (mostrado en la tabla 10). El efecto de fatiga es también comparable con esta observación.

En cualquier caso, los datos obtenidos en ambos experimentos son todavía insuficientes para determinar que factor (sesión o programa) está teniendo realmente influencia, por lo que nuevas replicaciones serán todavía necesarias para aclarar este particular.

Variable respuesta OutScope

En lo que respecta a la variable OutScope, en ningún experimento se observan diferencias significativas. La posibilidad de un efecto de fatiga, indicado anteriormente, encaja perfectamente con el hecho de que S2/ntree sea menos efectiva que S1/nametbl, tal y como se observa en la Figura 15.

Los resultados obtenidos en ESPEL y UPM son contrarios en tendencia, lo que produce la impresión visual de que ambos experimentos son discrepantes. No obstante, nótese el carácter no significativo de

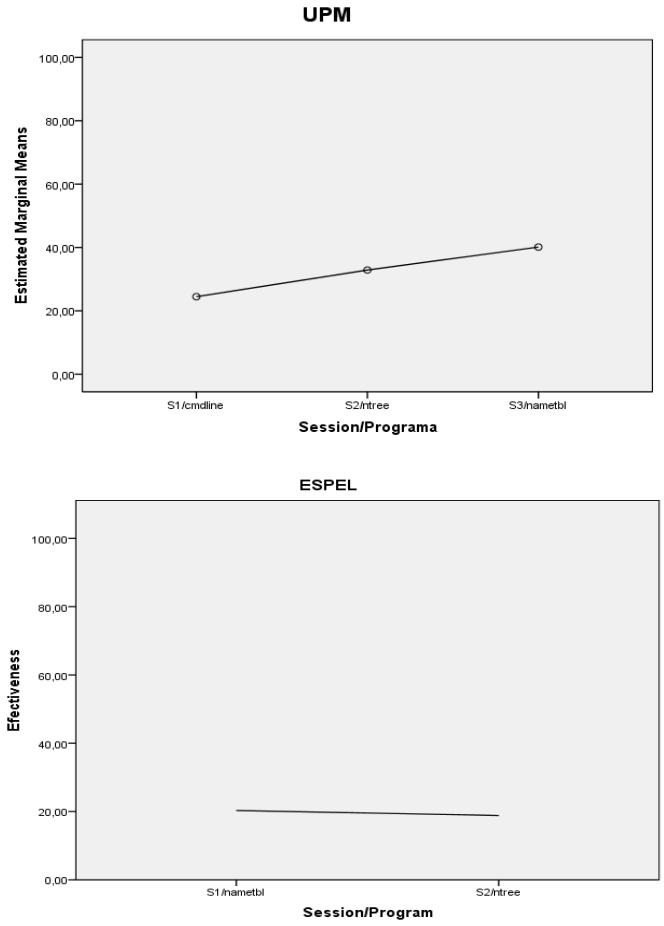

Figura 15: Medias Marginales Estimadas para Sesión/Programa en UPM y ESPEL - OutScope

las diferencias S1-S2 y nametbl-ntree en ambos experimentos. Los resultados no significativos ponen de manifiesto (como por otra parte parece lógico), que ni el programa ni la sesión (con la posible excepción de un efecto de fatiga) tienen ningún efecto en la efectividad de la detección de faltas OutScope. Este hecho refuerza la plausibilidad de que las faltas OutScope sean localizadas exclusivamente en función de las habilidades de los sujetos experimentales, tal y como se indicó en la sección 5.2.2.

5.3.2. Diferencias relativas al factor grupo

Las diferencias relativas entre grupos en ESPEL y UPM son demasiado profundas como para poder aportar tablas y gráficos similares a las usadas anteriormente. No obstante, teniendo en consideración la significación estadística de los resultados, se puede comparar ambos experimentos tal y como se muestra en la tabla 23.

Tabla 23
UPM-ESPEL Comparison - Group Factor

Variable Respuesta	**H0**	
	Upm	**Espel**
InScope	Rechaza	Rechaza
OutScope	Rechaza	Acepta

Variable Respuesta InScope

Tanto el experimento en UPM como en ESPEL se obtienen resultados estadísticamente significativos para el factor grupo. El análisis realizado apunta a que, en ESPEL, los resultados pueden deberse tanto a un efecto de carryover como a un desbalanceo en la experiencia en programación C y uso de técnicas de testing entre grupos experimentales. En UPM, se desconoce si los grupos experimentales estaban balanceados o no (los experimentadores originales no proporcionan dicha información). Adicionalmente, el reporte original [Juristo et al., 2013] apunta a la no existencia de carryover.

Parecería ser que el desbalanceo entre grupos no tuvo un impacto decisivo. El desbalanceo entre grupos experimentales en ESPEL debería manifestarse consistentemente en las faltas InScope y OutScope. Sin embargo, no se observa tal resultado, ya que la diferencia entre grupos para faltas OutScope es no significativa (y, además de ello, distinta en tendencia a InScope).

En lo que respecta al carryover, parecería ser que su existencia no se puede descartar. Esta opinión está basada en dos observaciones.

- El análisis de los factores técnica y sesión/programa tanto en ESPEL como en UPM parece indicar que la identificación de faltas OutScope depende únicamente del sujeto experimental, esto es, de sus conocimientos, experiencia, etc. Si ello fuera cierto, sería lógico que en la replicación no se observára ningún efecto de carryover para faltas OutScope, ya que las técnicas *equivalence partitioning* y *branch testing* no influyen en su identificación. Este hecho no sería cierto para faltas InScope, y por ello en este caso si tiene sentido observar un efecto de carryover.

- De modo completamente informal, parecería que la utilización de EP en primer lugar mejora la efectividad de las técnicas aplicadas a continuación en ambos experimentos(los grupos EP-CR-BT y EP-BT-CR en el experimento original, y el grupo EP-BT en la replicación, aparecen entre los más efectivos)

Una razón que podría explicar la aparición de diferencias significativas en el factor grupo y que no constituirá carryover son los efectos de pequeñas muestras. El experimento original contó con 46 sujetos experimentales, entonces el número de sujetos por grupo es de $46/6 \simeq 8$. En ESPEL, el número de sujetos por grupo es de $23/2 \simeq 12$. Con tan pocos sujetos por grupo, es posible obtener efectos significativos por azar, incluso aplicando correcciones de Bonferroni.

Parece que los efectos de pequeñas muestras pueden añadir ruido en el análisis de los datos (esto es, hacer que ciertos grupos presenten diferencias significativas de otros puramente por azar), pero esto no explica totalmente los resultados. Nótese que el efecto de pequeñas muestras debería actuar consistentemente en las faltas InScope y OutScope, cosa que no ocurre, en UPM, tres grupos presentan diferencias significativas respecto a las faltas InScope, mientras que solo uno respecto a las faltas OutScope (lo que podría tratarse perfectamente de un efecto de pequeñas muestras). En ESPEL, las diferencias son significativas para IsScope pero no para OutScope. Por lo tanto, la existencia de un efecto de carryover para faltas InScope es plausible.

Lamentablemente, la hipótesis de la existencia de carryover es bastante especulativa. Los grupos en ESPEL y UPM no son directamente

compatibles, por lo que todas las inferencias realizadas son indirectas. Por consiguiente, es necesario realizar más replicaciones para confirmar o rechazar la existencia de un efecto carryover.

Variable Respuesta OutScope

El análisis de la variable respuesta InScope ha requerido exponer todos los hallazgos respecto a la variable OutScope, de modo que en este apartado, solo se enunciarán conclusiones:

- Se adscribe la existencia de diferencias significativas en el factor grupo para OutScope a un efecto de pequeñas muestras.
- En consonancia con el análisis realizado para los factores técnica y sesión/programa, se cree que las faltas OutScope son detectadas por los sujetos gracias a sus propias habilidades, por lo que no debería existir un efecto de carryover (que, al fin y al cabo, nace de una relación entre las técnicas de testing y, por lo tanto, requiere de su influencia en la variable respuesta OutScope para poder existir).

Capítulo VI

CONCLUSIONES, LECCIONES APRENDIDAS Y TRABAJOS FUTUROS

Es este capítulo se sintetiza los hallazgos más importantes de este trabajo, haciendo énfasis en las conclusiones, lecciones aprendidas y trabajos futuros.

6.1. Conclusiones

Comparar replicaciones que no son idénticas es complejo. Al eliminar uno de los niveles del factor técnica por razones logísticas (falta de tiempo para realizar las sesiones experimentales), los cambios han afectado en cascada a los factores programa y sesión. En consecuencia, tanto las sesiones como los grupos no son comparables en todos los aspectos. Aun así, la replicación ha ayudado a comprender mejor la influencia de los distintos factores estudiados:

- En primer lugar, se ha confirmado que la técnica *equivalence partitioning* es más efectiva para detectar faltas que están dentro de su alcance y que *branch testing* es más efectiva para las faltas fuera de su alcance. Para el caso de la variable respuesta InScope, esta diferencia en la efectividad podría deberse a que la técnica estructural es más compleja ó, al menos, peor aplicada por los

sujetos, que la funcional. Respecto a la diferencia en efectividad para el caso de la variable respuesta OutScope, se hipotetiza que la mejora en la efectividad de la técnica estructural es debido a que los estudiantes la complementan con una revisión de código, ya que al tener acceso al código fuente (a diferencia de la técnica funcional), es posible que realizaran revisiones para entender mejor el funcionamiento del programa. Los resultados estadísticos muestran que las medidas de efectividad en general resultaron más bajas en ESPEL que en UPM, lo que podría atribuirse a la influencia del contexto. Más precisamente, se cree que pudo influir la modalidad tan intensiva de estudios aplicada en el entrenamiento. Así mismo, pudo sumarse a la modalidad de estudios una posible inexperiencia del capacitador en dictar el curso de V&V, por ser la primera vez y más aún en tales circunstancias.

- En segundo lugar, los resultados obtenidos en los factores sesión/programa y grupo, respecto a la variable respuesta OutScope, apoyan la hipótesis de que ni la técnica EP ni la BT influencian realmente la detección de faltas OutScope. En ambos casos (sesión/programa y grupo), tanto en el experimento original como en la replicación, los resultados no han sido significativos.
- En tercer lugar, parece confirmarse la existencia de algún tipo de efecto de carryover para la variable respuesta InScope. El problema reside, en este caso, en que la conformación de los grupos es diferente en los dos experimentos, y por lo tanto, no son comparables. En consecuencia, es necesario realizar más replicaciones antes de afirmar la existencia de dicho efecto.

Las mayores dificultades han surgido a la hora de dilucidar la influencia del factor sesión/programa respecto a la variable InScope. Los experimentadores originales concluyeron que fue el programa el responsable del efecto sesión/programa en la efectividad en UPM (no olvidar que los dos factores están confundidos). Los experimentadores originales basaban esta conclusión en una comparación entre la efectividad InScope y OutScope para sesión/programa, la cual sugería que las diferencias entre cmdline, nametbl y ntree conjuntamente parecían explicar mejor los datos obtenidos que las diferencias existentes entre

las sesiones S1, S2 y S3.

Sin embargo, los resultados en ESPEL para InScope tienen un patrón completamente opuesto a los resultados de UPM. No solo las tendencias respecto a los programas son distintas en ESPEL y UPM (ntree>nametbl en UPM, mientras que nametbl>ntree en ESPEL), sino que las diferencias en ESPEL son estadísticamente significativas.

Uno de los cambios realizados en la replicación respecto al experimento original fue reducir el número de sesiones de tres a dos, lo que provocó a su vez que uno de los programas utilizados en el experimento original (cmdline) no fuera utilizado. En estas circunstancias, es difícil llegar a una conclusión mínimamente fiable. El análisis tiende a adscribir los efectos observados a la sesión, en lugar del programa. No obstante, existen explicaciones alternativas adicionales. Una particularmente atractiva es la existencia de algún efecto de interacción técnica/programa.

La Figura 16 muestra boxplots para el factor técnica pero, a diferencia de lo realizado en la Figura 12, se ha efectuado una descomposición adicional por sesión/programa. Existen varios valores atípicos que complican la interpretación de la Figura 16, pero puede observarse claramente que la mediana[1] para EP x nametbl es mucho mayor que las restantes combinaciones (BT x nametbl, etc.). Dichos valores son más fáciles de apreciar en el diagrama de perfil (nótese que muestra medias, no medianas) presentado en la Figura 17. Esta interacción no está contemplada en el modelo de análisis ni del experimento original ni en la replicación (que, a este respecto, reutiliza el modelo original a efectos de facilitar la comparación), ya que no es posible su cálculo utilizando una ANOVA de medidas repetidas.

Independientemente del modelo de análisis utilizado, la existencia de tal interacción sería compatible con los resultados obtenidos para técnica y programa en ESPEL, y explicaría por qué se ha observado un efecto de carryover (nótese que la interacción técnica/programa, esto es, que una técnica puede ser más efectiva cuando es aplicada a

[1]La mediana, dentro de la distribución de valores de la efectividad de las técnicas, corresponde al valor medio (no confundir con el valor promedio que corresponde a la media).

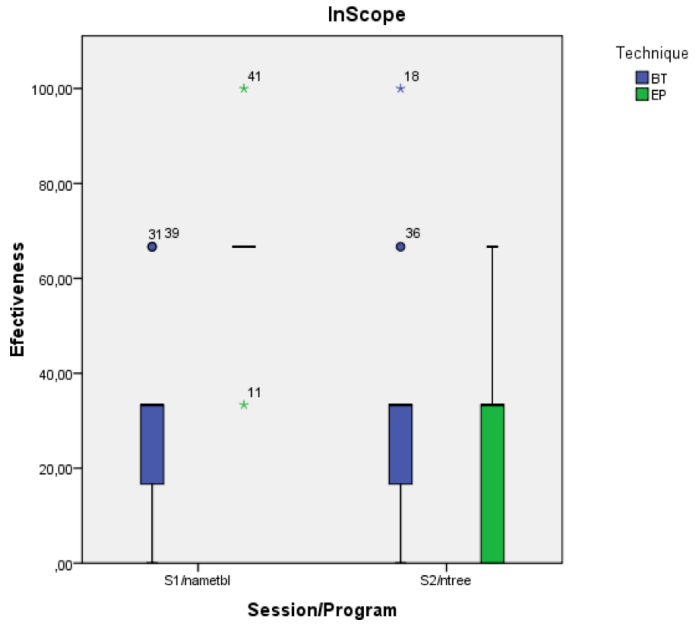

Figura 16: Boxplot de la Interacción Técnica x Sesión/Programa en ESPEL - InScope

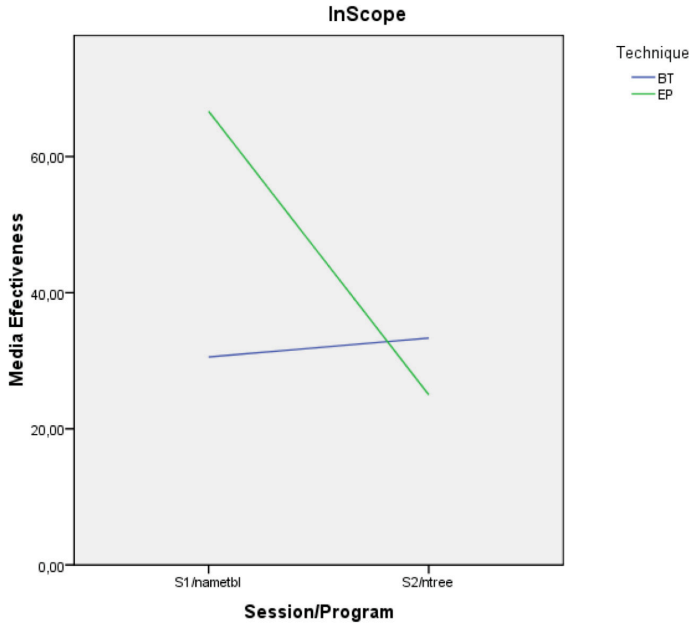

Figura 17: Diagrama de Perfil de la Interacción Technique x Sesión/Programa en ESPEL - InScope

ciertos programas, está confundida con el factor grupo, por lo que el análisis estadístico no puede diferenciar entre ambos tipos de efectos). No obstante, un posible efecto de técnica/programa no explica todos los hallazgos. En particular, se plantea la duda de por qué los grupos EP-BT-CR y EP-CR-BT son tan efectivos en el experimento original,

cuando en la primera sesión el programa ensayado no fue nametbl sino cmdline, y cmdline es considerado como un programa difícil de entender y probar por todos los sujetos participantes en los experimentos. Lo realmente cierto es que todas las explicaciones anteriores son tentativas. Como ya se ha reiterado anteriormente, es ineludible llevar a cabo más replicaciones para conseguir comprender con claridad los efectos de los factores técnica y sesión/programa.

6.2. Lecciones Aprendidas

La replicación realizada pertenece a una familia de experimentos con cierta tradición. Esto implica que la información y las experiencias obtenidas en múltiples replicaciones llevadas a cabo en esta familia, son razonablemente completas y, contando adicionalmente con el reporte del experimento original, la replicación podría haberse llevado a cabo sin necesidad de información adicional.

Sin embargo, visto en retrospectiva, se cree que las posibilidades de éxito de haber procedido de esta forma hubieran sido bastante escasas. Ya se ha indicado con anterioridad que la comunicación con los experimentadores originales fue intensa, por lo que se cree que las coincidencias entre ambos experimentos son notables, pero las diferencias no son menos acusadas. Estas diferencias han podido ser trazadas a características del experimental setting (por ejemplo, carácter intensivo del curso de V&V), pero ello ha sido posible únicamente porque los cambios introducidos en la replicación han sido relativamente pocos. Por ejemplo, se podría haber planteado que la baja efectividad de los sujetos ESPEL en la técnica *branch testing* podría deberse a su poca experiencia con el lenguaje C. Ahora bien, en general los sujetos UPM tampoco tienen experiencia en C y su efectividad es mayor. Por lo tanto, es más razonable adscribir la baja efectividad al training recibido, que si varía, y mucho, entre ESPEL y UPM.

Tal nivel de coincidencia parece difícil de lograr contando únicamente con el reporte del experimento original y los materiales proporcionados. Aspectos como el training ya citado, pero también la ejecución

o la medición de los resultados, por citar ejemplos, no están descritos en el material proporcionado. De no haber contado con la colaboración de los experimentadores originales, especialmente en las fases iniciales, las diferencias entre el experimento inicial y la replicación serían probablemente mucho mayores. Es también probable que dichas diferencias causasen discrepancias en los resultados (como, por ejemplo, las ya indicadas relativas al entrenamiento). Una discusión post experimental con los investigadores originales podría poner de manifiesto las diferencias en el diseño, pero no evitar las discrepancias originales por dichas diferencias. Sin embargo, estos puntos fueron discutidos en las reuniones iniciales que se tuvieron con los experimentadores originales.

Como corolario de lo anterior, la experiencia de haber replicado un experimento previamente realizado por otros experimentadores y, todavía más importante, el intento de comparar los resultados de ambos experimentos, han mostrado que las replicaciones deben ser lo más similares posible al experimento original, para que sea posible adscribir (al menos hipotéticamente) las coincidencias y discrepancias en los resultados a factores y parámetros determinados. Las razones son las mismas indicadas anteriormente, si el experimental setting varia mucho, es posible adscribir las diferencias entre resultados a virtualmente cualquier aspecto, haciendo que la replicación resulte mucho menos esclarecedora de lo que sería en otro caso.

La evidencia más tangible de la efectividad de la comunicación con los experimentadores originales y, su consiguiente transmisión del conocimiento al investigador replicador, se ve reflejada en el nivel de detalle alcanzado en la replicación. Tal es así, que como producto de dicha replicación se obtuvo una publicación en la revista más emblemática de la comunidad de ISE, denominada EMSE (de las siglas en inglés Empirical Software Engineering). La publicación [Fonseca C. et al., 2013] representa un esfuerzo de más de un año que les tomó a sus autores, para su cristalización. Naturalmente, la experiencia alcanzada por la familia de experimentos replicada, lo cual se refleja en sus múltiples publicaciones de impacto (lo cual se detalla en el Capítulo III, Sección 3.1), coadyuvó a alcanzar tal nivel en la replicación.

Obviamente, la mera realización de replicaciones similares no asegura que los resultados obtenidos puedan adscribirse a un factor de forma unívoca. Por una parte, todo experimento está afectado por una cierta probabilidad de error (α y β). Por otra parte, se desconoce qué aspectos de un setting (esto es, variables no controladas) pueden alterar los efectos de los factores. En consecuencia, incluso replicaciones muy similares en su diseño pueden arrojar resultados contradictorios.

Los errores α y β pueden combatirse de una forma relativamente sencilla: aumentando el número de sujetos experimentales. A medida que el número de sujetos experimentales aumenta, los experimentadores pueden mantener los valores de α y β progresivamente más pequeños. Ahora bien; en la práctica la disponibilidad de sujetos experimentales es limitada (por ejemplo, Sjøberg et al. [Sjøberg et al., 2005] reportan que la media de sujetos experimentales por experimento en ISE es de 48.6). En otras palabras, en el mejor de los casos se puede contar con los suficientes sujetos como para asegurar normalidad y librarse de los efectos de pequeñas muestras (entre 30-50 sujetos) [Richy et al., 2004][Graham and Schafer, 1999].

El segundo problema es, en general, irresoluble. Las variables desconocidas son, como su nombre indica, desconocidas, y por lo tanto su efecto no puede estimarse. De todas formas, habitualmente se considera que la aleatorización es efectiva (esto es, elimina los efectos de variables no controladas) a partir de 30 sujetos, que es cuando una muestra de una población genérica se asume que posee características de normalidad.

La replicación realizada cumple con las dos características anteriores. Posee 23 sujetos experimentales que, por su diseño de tipo crossover, se convierten en 23 x 2 = 46 unidades experimentales. Por lo tanto, se trata de un experimento "promedio"en IS. Aún así, y habiendo mantenido razonablemente todos los parámetros y factores bajo control, se ha observado discrepancias respecto al experimento original. Se ha hipotetizado que dichas discrepancias pueden deberse a ciertas causas (como es el caso de las diferencias en los resultados respecto a sesión/programa), pero ello ha sido posible únicamente por que la diferencia entre experimentos es escasa. Si las diferencias entre los ex-

perimentos fuesen más amplias, dicha adscripción, incluso hipotética, no sería posible.

Naturalmente, las causas reales de las discrepancias pueden ser otras. Las replicaciones literales (esto es, lo más parecidas posible al experimento original) representan solo un punto de partida que no significa dejar de realizar replicaciones diferenciadas para explorar, de una forma más general, el conjunto de factores que pueden ejercer un efecto.

Finalmente, se ha constatado que el esfuerzo en la preparación de la replicación es mucho mayor que el destinado a las sesiones experimentales. Por un lado, comprender en detalle el experimento original, así como el training inicial y el procesamiento de los formularios entregados por los sujetos, han demostrado ser mucho más consumidores de tiempo que las sesiones experimentales. De hecho, las sesiones implican un trabajo relativamente reducido.

En resumen, las lecciones aprendidas han sido las siguientes:

- Es importante contar con el apoyo de los experimentadores originales durante, al menos, las fases iniciales de preparación de la replicación, a modo de complementar la información obtenida a través de los reportes y materiales. Contar con toda la información posible es fundamental para que el experimento original y la replicación sean comparables.

- La replicación debe ser lo más similar posible, o incluso idéntica, al experimento original, con la finalidad de poder adscribir las diferencias (o coincidencias) encontradas a variables concretas, pese a que hay autores que afirman lo contrario.

- El mayor esfuerzo de realización del experimento no se encuentra en las sesiones experimentales, sino en la preparación previa y análisis posterior de la información obtenida de los sujetos.

6.3. Trabajos Futuros

La replicación realizada ha confirmado, en lo esencial, los efectos observados en el experimento original. Sin embargo, existen ciertos efectos que todavía no pueden adscribirse con certeza a variables concretas, tales como la diferente efectividad asociada a las sesiones/programas, o los efectos de carryover. La intención a corto plazo es seguir replicando el experimento UPM, alterando el setting tan poco como sea posible, con la intención de determinar sin lugar a dudas qué variables producen qué efectos. Una vez que se entienda bien como se comportan las técnicas de testing bajo estudio (*equivalence partitioning* y *branch testing*) se podrá realizar replicaciones diferenciadas que exploren diferentes settings o poblaciones (e.g.: profesionales con experiencia o entornos industriales).

Referencias

[Basili, 1992] Basili, V. R. (1992). Software modeling and measurement: the goal/question/metric paradigm. Technical Report UMIACS TR-92-96, Departament of Computer Science, University of Maryland, College Park, MD, USA.

[Basili and Selby, 1985] Basili, V. R. and Selby, R. W. (1985). Comparing the effectiveness of software testing strategies. Technical Report TR-1501, Departament of Computer Science, University of Maryland, College Park, MD, USA.

[Basili and Selby, 1987] Basili, V. R. and Selby, R. W. (1987). Comparing the effectiveness of software testing strategies. *IEEE Transactions on Software Engineering*, SE-13(12):78–96.

[Basili et al., 1985] Basili, V. R., Selby, R. W. J., and Hutchens, D. H. (1985). Experimentation in software engineering. Technical report, Air Force Office of Scientific Research and University of Maryland.

[Box et al., 2008] Box, G. E., Hunter, J. S., and Hunter, W. G. (2008). *Estadística para Investigadores Diseño, innovación y descubrimiento.* Editorial Reverté, S. A., 2nd edition.

[Brown, 1980] Brown, B. W. (1980). The crossover experiment for clinical trials. *Biometrics*, 36(1):69–70.

[Camejo R, 2008] Camejo R, A. J. (2008). Globalización, tecnología de la información y flexibilización laboral. *Nómadas Revista Crítica de Ciencias Sociales y Jurídicas*, 19(3).

[Davis and Holt, 1992] Davis, D. D. and Holt, C. A. (1992). *Experimental economics.* Princeton University Press.

[Fonseca C., 2009] Fonseca C., E. R. (2009). Incidencia del uso del sistema de automatización de oficinas en el manejo de la información en el departamento de orientación y bienestar estudiantil del colegio nacional técnico yaruquí. Master's thesis, Universidad Técnica de Ambato.

[Fonseca C., 2012] Fonseca C., E. R. (2012). Definition of a support infrastructure for replicating and aggregating families of software engineering experiments. Available at `http://esem.cs.lth.se/esem2012/idoese/pdf/163_IDOESE2012-EfrainFonseca.pdf`.

[Fonseca C. et al., 2013] Fonseca C., E. R., Espinosa G., E. G., Dieste, O., and Apa, C. (2013). Effectiveness for detecting faults within and outside the scope of testing techniques: An independent replication. *Empirical Software Engineering*, 18(5):1–40.

[Genero et al., 2012] Genero, M., Cruz-Lemus, J. A., Abrahão, S., Insfrán, E., Carsí, J. A., and Caivano, D. (2012). A controlled experiment for assessing the influence of stereotypes on the comprehension of uml sequence diagrams. Available at `http://alarcos.esi.uclm.es/ExpStereotypes/`.

[Gómez, 2012] Gómez, O. (2012). *Tipología de Replicaciones para la Síntesis de Experimentos en Ingeniería del Software*. PhD thesis, Universidad Politécnica de Madrid.

[Gómez et al., 2010] Gómez, O., Juristo, N., and Vegas, S. (2010). Replications types in experimental disciplines. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, number 3, pages 1–10, Bolzano-Bozen, Italy.

[Graham and Schafer, 1999] Graham, J. W. and Schafer, J. L. (1999). On the performance of multiple imputation for multivariate data with small sample size. In Hoyle, R. H., editor, *Statistical Strategies for Small Sample Research*, pages 1–29. Sage Publications.

[Greenwood, 1976] Greenwood, E. (1976). *Experimental sociology: A study in method*. Octagon Books (New York).

[Juristo et al., 2006] Juristo, N., Moreno, A., and Vegas, S. (2006). *Software Testing Tecniques*. Universidad Politécnica de Madrid - Facultad de informática, Boadilla del Monte 28660, 12 edition.

[Juristo and Moreno, 2001] Juristo, N. and Moreno, A. M. (2001). *Basics of Software Engineering Experimentation*. Kluwer Academic Publishers.

[Juristo and Moreno, 2002] Juristo, N. and Moreno, A. M. (2002). Reliable knowledge for software development. *IEEE SOFTWARE*, 19(5):98 – 99.

[Juristo et al., 2002] Juristo, N., Moreno, A. M., and Vegas, S. (2002). A survey on testing technique empirical studies: How limited is our knowledge. *Symposium on Empirical Software Engineering (ISESE'02)*.

[Juristo and Vegas, 2003] Juristo, N. and Vegas, S. (2003). Functional testing, structural testing and code reading: What fault do they each detect? *Empirical Methods and Studies in Software Engineering Experiences from ESERNET*, 2785(12):208–232.

[Juristo et al., 2013] Juristo, N., Vegas, S., and Apa, C. (2013). Effectiveness for detecting faults within and outside the scope of testing techniques: A controlled experiment. Available at `http://www.grise.upm.es/reports.php`, Last visited: May 15th.

[Kamsties and Lott, 1995] Kamsties, E. and Lott, C. (1995). An empirical evaluation of three defect-detection techniques. In *Fifth European Software Engineering Conference (ESEC '95)*, volume 989 of *Lecture Notes in Computer Science*, pages 362–383.

[Kernan et al., 1999] Kernan, W. N., Viscoli, C. M., Makuch, R. W., Brass, L. M., and Horwitz, R. I. (1999). Stratified randomization for clinical trials. *Journal of Clinical Epidemiology*, 52(1):19–26.

[Kitchenham et al., 2003] Kitchenham, B., Fry, J., and Linkman, S. (2003). The case against cross-over designs in software engineering. In *Software Technology and Engineering Practice, 2003. Eleventh Annual International Workshop on*, pages 65–67.

[Latour et al., 1986] Latour, B., Woolgar, S., and Salk, J. (1986). *Laboratory Life: The Construction of Scientific Facts*. Princeton, USA: Princeton University Princeton, USA: Princeton University Press.

[Luo, 2001] Luo, L. (2001). Software testing techniques technology maturation and research strategy. Technical report, Institute for Software Research International Carnegie Mellon University.

[Miller, 1980] Miller, E. F. (1980). Introduction to software testing technology. In *Tutorial: Software Testing & Validation Techniques*, number IEEE Catalog No. EHO 180-0, pages 4–16.

[Myers, 1978] Myers, G. J. (1978). A controlled experiment in program testing and code walkthroughs/inspections. *Communications of the ACM*, 21(9):760–768.

[Richy et al., 2004] Richy, F., Ethgen, O., Bruyère, O., Deceulaer, F., and Reginster, J. (2004). From sample size to effect-size: Small study effect investigation (ssei). *The Internet Journal of Epidemiology*, 1.

[Roper et al., 1997] Roper, M., Wood, M., and Miller, J. (1997). An empirical evaluation of defect detection techniques. *Information and Software Technology*, 39(11):763 – 775.

[Senn, 2002] Senn, S. (2002). *Cross-over Trials in Clinical Research*. John Wiley and Sons Ltd., second edition.

[Shaw, 1990] Shaw, M. (1990). Prospects for an engineering discipline of software. *Software, IEEE*, 7(6):15 –24.

[Sjøberg et al., 2005] Sjøberg, D. I., Han Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N.-K., and Rekdal, A. C. (2005). A survey of controlled experiments in software engineering. *IEEE Transactions on Software Engineering*, 31(9):733 – 753.

[Wohlin et al., 2000] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2000). *Experimentation in software engineering An introduction*. Kluwer Academic Publishers.

[Wohlin et al., 2012] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. (2012). *Experimentation in Software Engineering*. Springer Berlin Heidelberg.

[Wood et al., 1997] Wood, M., Roper, M., Brooks, A., and Miller, J. (1997). Comparing and combining software defect detection techniques: a replicated empirical study. In *Proceedings of the 6th European Software Engineering Conference held jointly with the 5th ACM SIGSOFT international symposium on Foundations of software engineering*, pages 262–277, Zurich, Switzerland.

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE

MAESTRÍA EN INGENIERÍA DE SOFTWARE

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por el Sr. Ing. Efraín Rodrigo Fonseca Carrera MSc., bajo nuestra supervisión.

Ing. Geovanny Raura MIS
DIRECTOR

Ing. Lucas Garcés Ms.C.
DIRECTOR DEL PROGRAMA

Dr. Rodrigo Vaca
SECRETARIO ACADÉMICO