Impacto de las características personales de los programadores en la efectividad de Test-Driven-Development (TDD)

UNIVERSIDAD
NACIONAL
DE LA PLATA

Tesis Doctoral

Autor: Jorge Geovanny Raura Ruiz

Facultad de Informática
Universidad Nacional de La Plata

Mayo 2022

Documento maquetado con TEXIS v.1.0+.

Este documento está preparado para ser imprimido a doble cara.

Impacto de las características personales de los programadores en la efectividad de Test-Driven-Development (TDD)

Tesis Doctoral previa a la obtención del Título de Doctor en Ciencias Informáticas

24/05/2022

Versión 1.0+

Facultad de Informática
Universidad Nacional de La Plata

Mayo 2022

Este trabajo está dedicado a mis padres Guillermo y Esthela, soporte y motor de mi vida

A mi hijo Juan Sebastián, en quien veo reflejado mis propios sueños

A mi compañera de vida Irma, quien supo entenderme y apoyarme en todo momento

y

A mis amigos y colegas que me apoyaron para que este objetivo se haga realidad

Agradecimientos

A todos quienes fueron parte de este cometido.

Gracias por contribuir en este estudio.

Realizar una tesis doctoral implica muchas horas de esfuerzo y dedicación. En general tiene un inicio quizás predecible, pero pocas veces sabemos con lo que nos vamos a encontrar en la búsqueda del conocimiento. La investigación me ha dado la posibilidad de encontrar personas en este camino que sinceramente, sin su ayuda, este trabajo no hubiese podido materializarse.

En primer lugar, debo agradecer a mi Director Oscar Dieste, quien supo guiarme de inicio a fin en el planteamiento, análisis y escritura de esta tesis, pero más que nada, supo entender las diferentes circunstancias por las que tuve que atravesar para conseguir este objetivo.

También debo agradecer a mi co-Directora Claudia Pons, quién siempre me facilitó el camino para hacer que este esfuerzo llegue a feliz término.

No puedo dejar de agradecer también a mi colega y amigo Rodrigo Fonseca, quien me colaboró incondicionalmente en la realización del trabajo experimental y la escritura de publicaciones.

Finalmente, debo agradecer a la Universidad de las Fuerzas Armadas ESPE, al Departamento de Ciencias de la Computación, sus autoridades y compañeros, quienes me apoyaron para cristalizar un reto más en mi desarrollo profesional.

RESUMEN

...

...

Contexto: El desarrollo dirigido por pruebas (Test Driven Development - TDD), es una estrategia de programación propuesta por Kent Beck [3], como alternativa al desarrollo de software tradicional, ha sido una técnica ampliamente estudiada en la ingeniería de software experimental, especialmente con la realización de estudios que intentan demostrar su efectividad en términos de calidad del código y productividad de los programadores. **Objetivos:** En este trabajo de tesis doctoral, se propone la realización de una familia de experimentos [31], para determinar la influencia de factores personales en la Calidad externa y en la Productividad al aplicar TDD en comparación con el desarrollo iterativo con pruebas al final (ITLD). **Metodología:** Se realizó una serie de 7 estudios experimentales en el ámbito académico e industrial, partiendo de un experimento tomado como base. Posteriormente los resultados fueron sintetizados mediante un meta- análisis tipo Individual Patient Data (IPD) con descomposición en subgrupos. **Resultados:** Se obtuvieron diferentes resultados en cuanto a la influencia de los factores humanos sobre la Calidad externa y Productividad, dependiendo del tipo de reclutamiento de los participantes que fueron agrupados como voluntarios (volunteer), no voluntarios (conscripted) y aquellos que participaron de los experimentos como un curso de entrenamiento (Training course), o también agrupados como profesionales y estudiantes. la Calidad externa no produjo diferencias significativas al aplicar TDD, aunque en ciertos casos hubo mejoras al aplicar TDD con estudiantes que participaron como *conscripted*), pero en otros casos la calidad externa decreció cuando fueron estudiantes que participaron como voluntarios. Por otra parte, los desarrolladores que usaron TDD fueron más productivos que aquellos que usaron ITLD. La experiencia en el uso de herramientas de prueba produjo resultados significativos para la Calidad externa y Productividad, aunque esto depende del tipo de reclutamiento y del carácter profesional o estudiante. Así mismo, la experiencia en Java incidió significativamente en la Calidad externa y el conocimiento del entorno Eclipse en la Productividad. La edad y el grado de completitud

o cantidad de código entregado por los participantes al realizar las tareas experimentales fue un factor que influyó significativamente en la Productividad, independientemente de la técnica utilizada. Otro resultado obtenido es que conforme los participantes profesionales tienen mayor edad, su grado de completitud fue disminuyendo, aunque existió cierto interés por realizar un mejor trabajo al aplicar TDD.

Conclusiones: Creemos que uno de los principales aportes de nuestro estudio, que lo consideramos de carácter exploratorio, es haber comprobado cómo la motivación, en este caso determinada por el tipo de reclutamiento, incide en el interés de los sujetos sean profesionales o estudiantes al realizar las tareas experimentales y por tanto influye en su productividad. También observamos que la edad es otro factor humano que debe ser objeto de una mayor investigación en trabajos futuros.

ABSTRACT

...
...

Context: Test Driven Development (TDD), is a programming strategy proposed by Kent Beck[3], as an alternative to traditional software development, has been a widely studied technique in experimental software engineering, especially with the realization of studies that try to demonstrate its effectiveness in terms of code quality and programmers'productivity.

Objectives: In this doctoral thesis, it is proposed to carry out a family of experiments[31], to determine the influence of personal factors on external Quality and Productivity when applying TDD compared to iterative test last development (ITLD).

Methodology: A series of 7 experimental studies were carried out in the academic and industrial field, starting from an experiment taken as a base. Subsequently, the results were synthesized through an Individual Patient Data (IPD) type meta-analysis with subgroup decomposition.

Outcomes: Different results were obtained regarding the influence of human factors on External Quality and Productivity, depending on the type of recruitment of the participants who were grouped as volunteers (volunteer), non-volunteers (conscripted) and those who participated in the experiments as a training course, or also grouped as professionals and students. External quality did not produce significant differences when applying TDD, although in certain cases there were improvements when applying TDD with students who participated as *conscripted*), but in other cases the external quality decreased when they were students who participated as volunteers. On the other hand, developers using TDD were more productive than those using ITLD. The experience in the use of test tools produced significant results for External Quality and Productivity, although this depends on the type of recruitment and the professional or student character. Likewise, experience in Java had a significant impact on External Quality and knowledge of the Eclipse environment on Productivity. The age and degree of completeness or amount of code delivered by the participants when performing the experimental tasks was a factor that significantly influenced producti-

vity, regardless of the technique used. Another result obtained is that as the professional participants get older, their degree of completeness decreased, although there was some interest in doing a better job when applying TDD.

Conclusions: We believe that one of the main contributions of our study, which we consider to be of an exploratory nature, is to have verified how motivation, in this case determined by the type of recruitment, affects the interest of the subjects who are professionals. or students when performing the experimental tasks and therefore influences their productivity. We also note that age is another human factor that should be the subject of further investigation in future work.

Índice

Índice de figuras

Índice de Tablas

Capítulo 1

INTRODUCCIÓN

Resumen:

Este capítulo presenta, a alto nivel, la investigación realizada. Se describe el área de investigación, el problema abordado, los objetivos de investigación, la metodología aplicada y los principales resultados obtenidos. El capítulo finaliza con una descripción de la estructura de la tesis a modo de mapa de carreteras, que facilite al lector, la búsqueda e identificación del material que considere relevante para sus propósitos.

1.1. Área de investigación

Las metodologías ágiles han venido ganando adeptos desde su introducción en la década de los 90. Tal es así que, actualmente, son una de las aproximaciones más utilizadas en el desarrollo de software [2]. El desarrollo ágil se basa en una serie de prácticas, tales como la programación por pares o el desarrollo dirigido por pruebas (Test Driven Development - TDD, por sus siglas en inglés) [3]. Estas prácticas pretenden aumentar la calidad del producto software y la productividad de los programadores.

En el *State of Agile survey* [4] del año 2020, el 30 % de los practicantes ágiles informaron que usan TDD con regularidad. Aunque este porcentaje ha venido disminuyendo (en encuestas anteriores, la adopción de TDD fue de alrededor del 50 %), esta práctica sigue siendo una de las técnicas ágiles más importantes en el desarrollo de software. TDD es una evolución de la técnica *test-first* descrita en *Extreme Programming* [5] y según su creador, Kent Beck, TDD no es una técnica de prueba, sino una técnica de diseño lógica [6]. TDD propone que, en lugar de realizar un diseño o modelo de software, se debe

enfrentar el desarrollo en base al planteamiento de pruebas unitarias antes de la generación efectiva del código. La figura 1.1 muestra el proceso de TDD frente al proceso de desarrollo tradicional (Test-Last Development - TLD). Este proceso consiste de los siguientes pasos:

- En primer lugar, una característica o requerimiento de usuario es seleccionado. A partir de ello, se escribe código de prueba que verifique la correcta realización de una tarea [1] pequeña o parte de un requerimiento del usuario (por ejemplo, un método o parte de la funcionalidad incluida en un método). Dado que el código correspondiente no ha sido desarrollado, esto produce una prueba que falla.
- A continuación, se escribe el código de producción que implementa la funcionalidad a ser probada.
- Una vez implementada la funcionalidad se ejecutan nuevamente las pruebas, así como todas las pruebas anteriores o preexistentes. Si alguna prueba falla, se corrige el código de producción y el conjunto de pruebas se vuelve a ejecutar. Si las pruebas pasan, se re-factoriza tanto el código de producción como las pruebas para mejorar la calidad del código.

Este proceso contrasta con la técnica de TLD que es habitualmente la utilizada en los proyectos de desarrollo tradicionales. En TLD:

- Se escribe en primer lugar el código,
- Luego se escriben las pruebas y,
- Se ejecutan las pruebas. Si fallan, se depuran los errores y se vuelven a ejecutar las pruebas hasta que éstas tienen éxito.

Existe una variante de TLD conocida como el desarrollo iterativo con pruebas posteriores a la codificación (Iterative Test-Last Development - ITLD por sus siglas en inglés). La diferencia entre ambas estrategias es la granularidad de las iteraciones o ciclos de desarrollo-prueba. TLD no especifica si la tarea/requerimiento a desarrollar debe ser grande o pequeño; es el programador el que decide. ITLD preconiza que, aunque las pruebas se escriban y ejecuten después de desarrollar el código de producción, éste código debería desarrollarse de forma incremental mediante la implementación de tareas pequeñas o partes de los requerimientos, al igual que en TDD.

De acuerdo a Beck[3], entre las ventajas de TDD se pueden mencionar la mejora en la comprensión del código, una mayor eficiencia a la hora de

[1] Beck[3] recomienda al desarrollador TDD ser concreto y actuar rápido. En consecuencia, cada ciclo TDD acostumbra a afrontar tareas pequeñas, pero no tiene porqué ser siempre el caso.

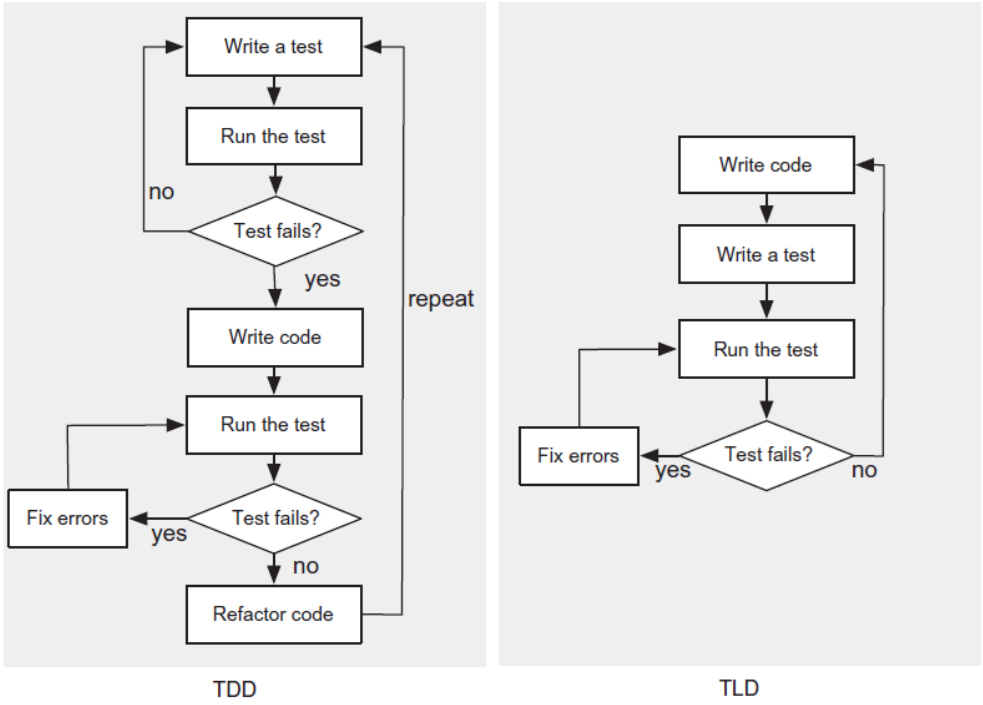


Figura 1.1: Proceso de TDD frente a TLD (figura tomada de [9])

identificar las las causas de los errores en el código, la generación de bancos de pruebas (*safety nets*) que pueden ampliarse a medida que avanza el desarrollo, y la posibilidad de reducir defectos introducidos en el código durante el mantenimiento y depuración.

TDD ha recibido especial atención por parte de la comunidad investigadora, que se ha materializado en la realización de múltiples estudios empíricos[2] al respecto, muchos de ellos de carácter experimental, ej: [7], [8], [9], [10], [11], [12], [13], [14] (por citar algunos).

1.2. Definición del problema

Los estudios empíricos acerca de TDD han sido sintetizados en diferentes revisiones sistemáticas de literatura [15], [16], [17], [18], [19], [20], [21]. Dichas revisiones proporcionan una visión general del tipo de investigación realizada hasta el momento y que podemos resumir en lo siguiente:

- En lo que respecta a la **metodología**, los estudios primarios se han efectuado tanto en la industria (la menor parte) como en el ámbito

[2]El lector podrá comprobar en el Estado de la Cuestión que el número de estudios empíricos acerca de TDD supera el centenar, algo asombroso hoy en día en Ingeniería del Software.

académico, utilizando para ello diferentes métodos de investigación empírica (experimentos, casos de estudio, estudios piloto, o encuestas).

- Los **aspectos generalmente analizados** son el impacto de TDD frente a otras estrategias de programación (tradicional, por pares, etc.), factores que pueden limitar la adopción de TDD [18], el rigor y relevancia de los estudios [22], el contexto de los estudios [15] (como por ejemplo: el entrenamiento previo de los sujetos participantes en los estudios, las tareas experimentales aplicadas, la duración del proyecto y otros aspectos demográficos de los participantes).

- Las **métricas** consideradas son diversas. Por ejemplo, Makinen & Munch [19] enumeran las siguientes: número de defectos, calidad interna y externa, complejidad, tamaño, esfuerzo, mantenibilidad, cobertura, productividad, acoplamiento y cohesión. En lo que respecta a los estudios experimentales, las variables más frecuentemente analizadas son la calidad externa y la productividad incluyendo (en ciertos casos) una serie de covariables:

  - La **Calidad Externa**[3] de un sistema es usualmente operacionalizada como una relación entre el número de casos de prueba de aceptación pasados y el número de defectos por unidad de tamaño del código (por ejemplo, líneas de código u otra medida adecuada) encontrados [16]. Se espera que TDD permita la fácil detección de errores en virtud de que el sistema es sometido a pruebas continuas a un nivel de grano fino.
  - La **Productividad** es expresada generalmente como una relación entre las entradas y salidas [16]. En varios estudios sobre la efectividad de TDD, la productividad mide la velocidad de desarrollo del equipo, y es calculada como la cantidad de funcionalidaes correctas desarrolladas en un determinado período de tiempo (días o meses).

- Las **Covariables** analizadas han sido principalmente la experiencia general de los desarrolladores, la experiencia en un lenguaje de programación, la experiencia en el uso de herramientas de desarrollo, la habilidad en la realización de tareas específicas, el nivel de entrenamiento y educación, la comunicación, el trabajo en equipo, la personalidad y el manejo de conflictos, entre otros.

La mayor parte de las revisiones de literatura coinciden en que existe cierta tendencia a que TDD mejora la Calidad. Sin embargo, si se considera el tipo de investigación, los resultados son inciertos. En la tabla 1.1 se

[3] Todas las referencias posteriores a la "calidad" hacen referencia a la "calidad externa". Por simplicidad, obviaremos el adjetivo "externa".

presenta un comparativo de los efectos de TDD en la Calidad y Productividad. Como se puede advertir, la mayoría de experimentos controlados no presentan diferencias en la Calidad, pero al parecer existe mejora cuando se analizan los datos obtenidos de estudios de caso. En cuanto a la Productividad, los resultados de los estudios primarios analizados en las revisiones de literatura difieren notablemente. Se observa que la Productividad mejora cuando se analizan datos de experimentos controlados. Sin embargo, los estudios de caso proporcionan evidencia mixta, algunos a favor y otros en contra de TDD.

La comparación de los estudios académicos y estudios industriales también indican diferencias en el impacto de TDD sobre la Calidad Externa y la Productividad. Algunos estudios encuentran que TDD mejora estos factores en ambientes académicos, en tanto que en el escenario industrial existe una caída de la Productividad. Considerando esta clasificación, en general los resultados no son concluyentes.

El análisis de la Calidad Interna produce resultados que también son inciertos. En la revisión de literatura de Makinen & Munch [19], por ejemplo, se encuentra que la mayor cantidad de estudios presentan efectos positivos de TDD en cuanto a la densidad de defectos y mantenibilidad del código. En cuanto a la complejidad, el tamaño, la cobertura, el acoplamiento y la cohesión, los resultados son contradictorios o no se observaron efectos.

En cuanto a las covariables, el factor que más frecuentemente ha sido analizado es la experiencia de los desarrolladores. No obstante, **la falta de evidencia empírica es prácticamente inexistente al respecto**.

En resumen, se sabe muy poco acerca de **cómo funciona TDD**. Los estudios de síntesis citados anteriormente reportan diferentes resultados sobre los efectos de TDD en la Productividad y la Calidad dependiendo de los grupos y subgrupos investigados, rigor y relevancia de los métodos de investigación utilizados, así como otros factores. En consecuencia, **todavía ignoramos si TDD mejora la Calidad del software y la Productividad de los programadores** [23].

1.3. Objetivo de la Tesis

Descubrir los factores que influyen en el proceso de desarrollo de software es uno de los objetivos de investigación más importantes en Ingeniería de Software (IS) ([24],[1]). No obstante, además de factores técnicos tales como las técnicas de desarrollo o lenguajes de programación, la investigación en Ingeniería de Software ha establecido que muchos otros aspectos puede influir fuertemente en el desarrollo de software, particularmente las características de las personas que llevan a cabo las actividades de desarrollo ([25],[26]).

El desarrollo de software es un proceso centrado en la persona. Los aspectos humanos desempeñan, en consecuencia, un papel importante ([27];

Tabla 1.1: Tabla resumen comparativo de los efectos de TDD en la calidad y productividad, de acuerdo a los estudios secundarios

Revisión	**Calidad externa**	**Productividad**
Kollanus[17]	Experimento controlado: Sin diferencia Estudios de caso: Mejora Otros: Mejora	Experimento controlado: No concluyente Estudios de caso: Disminuye Otros: Mejora
Turhan et al.[16] Shull et al [8]	Experimento controlado: No concluyente Estudios piloto: Mejora Industria: Mejora	Experimento controlado: mejora Estudios piloto: no concluyente Industria: Disminuye
Rafique & Misic[15]	Experimento académico: Sin diferencia Industria: Mejora	Experimento académico: mejora Industria: Disminuye
Munir et al.[20]	Estudios de Alto Rigor y Alta Relevancia (A): Mejora Estudios de Bajo Rigor y alta Relevancia (B1): Mejora Estudios de Alto Rigor y baja Relevancia (B2): Sin diferencia Estudios de Bajo Rigor y Baja Relevancia (C) : No concluyente	Estudios de Alto Rigor y Alta Relevancia (A): Disminuye Estudios de Bajo Rigor y alta Relevancia (B1): Disminuye Estudios de Alto Rigor y baja Relevancia (B2): Sin diferencia Estudios de Bajo Rigor y Baja Relevancia (C): No concluyente
Makinen[19]	TDD: Algunos efectos positivos, la mayor parte efectos neutrales o no concluyentes	TDD: Algunos efectos negativos, la mayor parte nuetrales o no concluyentes
Bissi et al[21]	Estudios en Industria: Mejora Estudios Académicos: Mejora	Estudios en Industria: Disminuye Estudios Académicos: Mejora

,[28]). Los aspectos humanos han sido investigados en prácticamente todas las actividades de la IS, teniendo un impacto igual o incluso superior a los factores técnicos ([29]).

Un factor que puede explicar los resultados aparentemente contradictorios sobre la efectividad de TDD es **la falta de consideración de las características personales de los programadores**, como pueden ser: la experiencia y el conocimiento previo en TDD, la habilidad para realizar casos de pruebas, el conocimiento del dominio, la motivación, etc. En consecuencia, el objetivo general de esta tesis es:

> *Estudiar la efectividad de TDD, tanto aisladamente como en relación a las características personales de los programadores.*

La falta de conocimiento sólido y empíricamente fundado acerca de la efectividad de TDD, justifica la realización de nuevos estudios que repliquen experimentos anteriores y ensayen TDD bajo nuevas condiciones. Por lo tanto, esta investigación se realizará siguiendo los lineamientos de la ingeniería de software experimental donde, además de examinar la influencia de TDD en la calidad y productividad, **examinaremos la influencia mediadora de los factores personales de los programadores**, tales como la experiencia, la formación académica o el conocimiento de tecnologías informáticas.

1.4. Metodología

La metodología utilizada en la presente investigación es puramente experimental, la cual permite contrastar las creencias y las opiniones con la realidad. De este modo surge el conocimiento científico, donde un conjunto de variables bajo estudio denominadas variables independientes o factores, pueden tomar distintos valores o niveles, a partir de lo cual se investigan sus efectos (variables dependientes o respuesta) [30].

Con el objetivo de obtener mayor evidencia empírica sobre la efectividad de TDD, hemos realizado una familia de experimentos [31] en TDD, tomando como base un experimento que ya ha sido replicado en varias ocasiones [32]. Esto ha permitido reducir los riesgos en la implementación de los experimentos realizados y, además, ha facilitado la comparación con estudios similares.

Debido a que la familia de experimentos fue realizada en diferentes contextos (académicos e industriales), el diseño original fue modificado para adecuarlo a los objetivos de la investigación. Se han realizado cambios al diseño experimental, los factores y niveles considerando el tipo de población. Más concretamente, en el experimento original se aplicó un diseño de medidas repetidas tipo *ABBB*, en tanto que dependiendo del contexto del

experimento, en nuestro estudio se aplicaron diseños tipo *cross over (2x2)* y *cuadrado latino*.

Los experimentos individuales han sido posteriormente sintetizados mediante meta-análisis [33], con la finalidad de lograr un mayor poder estadístico y, por ende, conclusiones más fiables.

1.5. Contribuciones

Durante el desarrollo de la investigación, se han presentado y publicado siete artículos científicos como contribuciones al cuerpo de conocimiento relacionados con la línea de investigación de la presente tesis.

1.5.0.1. Artículos publicados

Los artículos publicados desde el más reciente hasta el más antiguo son:

- **¿Qué factores personales afectan a la calidad y productividad de TDD? Un experimento con profesionales** Raura, G., Pons, C., Fonseca C, Dieste, O. Argentine Symposium on Software Engineering, ASSE 50, 2021, ISSN 2451-7593, pp 96-109. `https://50jaiio.sadio.org.ar/pdfs/asse/ASSE-11.pdf`

  Esta publicación muestra los resultados de uno de los experimentos en el ámbito académico de esta serie de instancias experimentales. Concretamente se muestra la influencia de las características personales de los desarrolladores sobre la Calidad y Productividad en un experimento llevado a cabo con estudiantes de maestría, todos ellos profesionales en activo desempeñando funciones relacionadas con la ingeniería del software. Pudimos determinar que algunos factores humanos como la edad, la función que desempeñan en la organización y el conocimiento previo de la técnica son, probablemente, posibles variables moderadoras y como tales su introducción en el proceso de análisis hace que los efectos de las estrategias de programación ITLD o TDD se vean alterados.

- **Gender gap in computing: A preliminary empirical study** Raura, G., Fonseca C, E.R., Castro, J.W.,Pons, C., Dieste, O. Avances en Ingenieria de Software a Nivel Iberoamericano, CIbSE 2018, 2018, pp. 57 70. Proceedings of the XXI Iberoamerican Conference on Software Engineering, Bogota, Colombia, April 23-27, 2018. Curran Associates 2018, ISBN 978-1-5108-6937-0. `https://dblp.org/db/conf/cibse/cibse2018.html`

  Durante varias décadas se ha observado un bajo interés de las mujeres en seguir carreras relacionadas con las Ciencias de la Computación.

Aprovechando los resultados obtenidos a lo largo de la serie de experimentos realizados en esta investigación, se hizo un meta-análisis para conocer si existen diferencias significativas en cuanto a la calidad y productividad considerando el género de los participantes. Los resultados mostraron que el género no tiene influencia en la efectividad de los desarrolladores, aunque si se observó la escasa participación de las mujeres en nuestros experimentos.

- **Professionals Are Not Superman: Failures beyond Motivation in Software Experiments**, Dieste, O, Fonseca, E.R.C., Raura, G., Rodríguez, P. Proceedings - 2017 IEEE/ACM 5th International Workshop on Conducting Empirical Studies in Industry, CESI 2017, 2017, pp. 27 32, 7968165. 2017. Proceedings of the IEEE/ACM 5th International Workshop on Conducting Empirical Studies in Industry (CESI),ISBN:978-1-5386-1546-1. `https://ieeexplore.ieee.org/document/7968165`

  En este estudio mostramos que la edad es un factor personal que influye sustancialmente en la cantidad de código que realizan los participantes en las tareas experimentales. Aunque este es un aspecto mas bien instrumental que no es atribuíble a la aplicación de la técnica TDD, sin embargo, contradice a la creencia habitual de que los estudios realizados en la industria son potencialmente más fiables que aquellos realizados en el contexto académico.

- **Software engineering reproducible research: A proposal for analyzing the effectiveness of test driven development** Raura, G., Fonseca, E.R., Gualotuña, T., Mejía, C.R. XII Jornadas Iberoamericanas de Ingenieria de Software e Ingenieria del Conocimiento 2017, JIISIC 2017, pp. 279 290. Proceedings of the XII Jornadas Iberoamericanas de Ingenieria de Software e Ingenieria del Conocimiento 2017 (JIISIC'17), ISBN 9781510843967. `https://www.proceedings.com/35254.html`

  En vista de que nuestra investigación consistía de una serie de experimentos que tenían que ser realizados con la mayor similitud posible entre ellos, tuvimos que desarrollar una serie de instrumentos que resultaron efectivos principalmente para el proceso de medición, obtención de los datos crudos y análisis de resultados. El objetivo principal de este artículo, fue presentar una propuesta para configurar experimentos (en nuestro caso sobre el efecto de TDD en la calidad y productividad), que puedan ser reproducibles en condiciones lo más parecidas al experimento original.

- **Experience does not predict performance: The case of the students-academic levels.** Raura, G., Efraín, F., Ponce, A., Dies-

te, O. Ibero-American Conference on Software Engineering CIBSE, 2017, pp. 57 70. Proceedings of the XX Iberoamerican Conference on Software Engineering, Buenos Aires, Argentina, May 22-23, 2017. Curran Associates 2017, ISBN 978-99967-839-2-0. `https://dblp.org/db/conf/cibse/cibse2017.html`.

El objetivo de este estudio fue evaluar si el nivel académico de los estudiantes incide en la calidad y la productividad al aplicar la técnica de TDD, frente a ITLD. Pudimos comprobar en este caso que la experiencia previa de los estudiantes no tuvo efectos significativos en los resultados experimentales.

- **Impact of the programmers'personal characteristics in the effectiveness of Test Driven Development (TDD): Proposal for a series of replications**. Raura, G., Dieste, O., Fonseca, C.E.R. XIX Ibero-American Conference on Software Engineering CIBSE, 2016, pp. 490 498. Proceedings of XIX Ibero-American Conference on Software Engineering, CIbSE 2016, Quito, Ecuador, April 27-29, 2016. Universidad de las Fuerzas Armadas ESPE 2016, ISBN 978-9978-301-81-4. `https://dblp.org/db/conf/cibse/cibse2016.html`.

  Este trabajo fue presentado en un simposio doctoral con el objetivo de mostrar los avances de nuestra investigación y recibir recomendaciones de otros investigadores relacionados con el área.

- **Effectiveness of test-driven development: A replicated experiment**, Dieste, O., Fonseca, E.R., Geovanny Raura, C., Rodríguez, P. XI Jornadas Iberoamericanas de Ingenieria de Software e Ingenieria del Conocimiento, JIISIC 2015, 2015, pp. 53 64. Proceedings of Ingeniería de Software e Ingeniería del Conocimiento. Jornadas Iiberoamericanas. 11TH 2015. (JIISIC'15), ISBN 9781510802087. `https://www.proceedings.com/26032.html`

  Este fue el primer estudio presentado con el objetivo de hacer un primer acercamiento a nuestra línea de investigación. Nos enfocamos principalmente en conocer en la práctica la viabilidad de la realización de nuestro trabajo. Ensayamos un experimento previamente aplicado sobre TDD y que luego fue tomado como base para la presentación de la propuesta de tesis.

1.5.0.2. Artículos en confección

Además de los artículos que ya han sido publicados, se plantea el envío de tres nuevas contribuciones a destacar:

- **Factores personales estudiados en la práctica de TDD. Una revisión sistemática de literatura** Raura, G., Pons, C., Dieste, O,

Fonseca, E.R., Castro, J. Este articulo reporta la revisión sistemática de literatura (SLR) realizada como base del estado del arte de esta tesis. Dada la secases de estudios empíricos que analizan los factores personales y su influencia al aplicar TDD, nos proponemos reportar los hallazgos encontrados, y las carencias existentes en la literatura.

- **Análisis de los factores personales que influyen en TDD. Un experimento industrial** Raura, G., Pons, C., Dieste, O, Fonseca, E.R., Castro, J. La revisión de literatura muestra que los experimento en ingeniería de software realizados en la industria son escasos y sus resultados producen mayor expectativa que aquellos experimentos realizados en el ámbito académico. En este estudio presentamos uno de nuestros experimentos realizados en la industria, destacando los factores personales que influyen en TDD, pero sobre todo las posibles amenazas a la validez de los resultados obtenidos.

- **La práctica de TDD y los factores personales que pueden influir en la Calidad externa y la Productividad. Síntesis de una familia de experimentos.** Raura, G., Pons, C., Dieste, O, Fonseca, E.R., Castro, J. Este estudio aborda la síntesis de los resultados obtenidos de la serie de replicaciones experimentales realizadas en esta tesis. Creemos que el mayor aporte al cuerpo de conocimiento sobre TDD, es la suma de resultados parciales dentro de una familia de experimentos que pueden ser comparables entre sí.

1.6. Estructura de la Tesis

La tesis doctoral se encuentra estructurada en 10 capítulos, incluyendo el capítulo de introducción:

- En el *Capítulo 2* se presenta la revisión del estado de la cuestión. En este apartado se analizan los principales hallazgos relacionados con nuestra investigación.

- En el *Capítulo 3* se plantean las preguntas de investigación. Estas preguntas están orientadas a descubrir el efecto de los factores personales al aplicar la estrategia de desarrollo basado en TDD, en comparación con el enfoque tradicional.

- En el *Capítulo 4* se presenta la metodología de investigación. En este capítulo se detalla el experimento tomado como base de estudio, así como las replicaciones realizadas, considerando sus principales diferencias contextuales o de diseño.

- En el *Capítulo 5* se muestran la ejecución y resultados obtenidos para cada experimento.

- En el *Capítulo 6* se realiza una síntesis de resultados en función de las preguntas de investigación planteadas. Se utiliza para ello un meta-análisis del conjunto de resultados de todas las instancias experimentales.
- En el *Capítulo 7* se abordan las limitaciones y amenazas a la validez respecto a los resultados obtenidos en la investigación.
- En el *Capítulo 8* Se aborda una discusión sobre los hallazgos encontrados
- En el *Capítulo 9* Se mencionan las conclusiones y futuras líneas de investigación.

Capítulo 2

ESTADO DE LA CUESTIÓN

...

...

Resumen:

En este capítulo presentamos una visión general sobre los factores humanos que han sido estudiados al aplicar la estrategia Test Driven Development (TDD). En primer lugar, nos planteamos realizar una revisión sistemática de literatura con el objetivo de obtener evidencia empírica sobre la influencia de los aspectos personales que podrían influir en la calidad y productividad al aplicar TDD.

Sin embargo, al intentar definir la estrategia de búsqueda, no pudimos identificar estudios primarios que abordasen la temática de nuestro interés. Por esta razón, optamos por buscar estudios secundarios que sinteticen los hallazgos encontrados sobre TDD y su relación con la calidad y productividad. En base a los estudios secundarios, identificamos una serie de estudios primarios que cumplen con nuestros criterios de inclusión/exclusión.

Posteriormente, hemos identificado los aspectos personales estudiados en los estudios primarios para, finalmente, hacer una clasificación de factores personales, extraer las variables analizadas, y los principales resultados obtenidos.

2.1. Revisión Sistemática de Literatura

Una revisión sistemática de literatura, conocida por sus siglas en inglés como SLR (Systematic Literature Review), tiene como objetivo principal el identificar, evaluar e interpretar la investigación disponible para responder

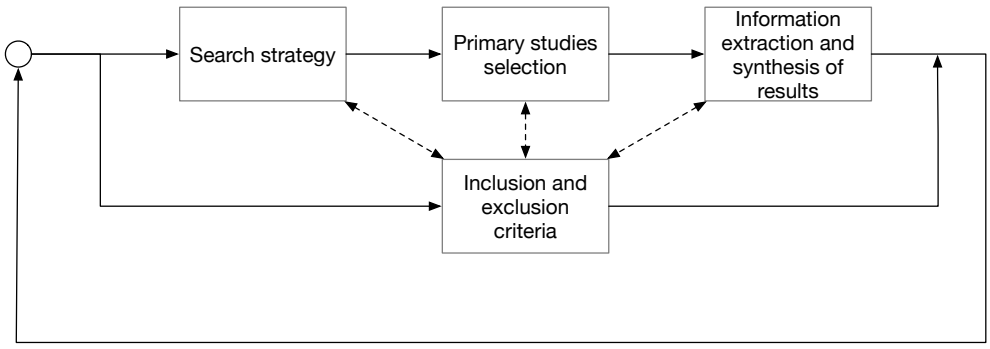


Figura 2.1: Proceso de revisión sistemática de literatura adoptado

a ciertas preguntas de investigación[34]. En este contexto, en primer lugar hemos definido las siguientes preguntas de investigación que deberán ser absueltas con nuestra SLR:

- RQ1: ¿Qué factores humanos se han estudiado en TDD?
- RQ2: ¿Cómo influyen los factores humanos en TDD?

Para realizar la revisión de literatura, nos basamos en las guías propuestas por Kitchenham et al.[34] y Petersen et al.[35], así como de los estudios de[36,37]. El protocolo de revisión aplicado consta de las siguientes etapas (ver Fig. 2.1): (1) definición de criterios de inclusión y de exclusión, (2) formulación de la estrategia de búsqueda, (3) selección de estudios primarios, y (4) extracción de información y síntesis de resultados.

2.1.1. Criterios de inclusión y exclusión

Los criterios de inclusión determinan qué estudios primarios son relevantes para la revisión de literatura. En esencia, buscamos estudios empíricos que estudien, no importa la metodología, factores humanos en relación a la estrategia de programación TDD. En consecuencia, unos criterios de inclusión razonables son los indicados a continuación:

IC1 Estudios empíricos que analizan *cuantitativamente* la relación entre los factores humanos y el desarrollo de software basado en TDD.

IC2 Estudios empíricos mixtos, que analizan *cualitativamente* y *cuantitativamente* la relación entre los factores humanos y el desarrollo de software basado en TDD

Los criterios de exclusión se definieron con el objeto de descartar estudios que, a pesar de su relación con las preguntas de investigación, no proporcionan información significativa en el contexto de TDD. Su inclusión solo causaría ruido en el proceso de investigación. Los criterios de exclusión son:

EC1 Estudios empíricos que *mencionan* (probablemente sin excesivo soporte empírico) la relación entre los factores humanos y el desarrollo de software basado en TDD.

EC2 Estudios empíricos que mencionan la relación entre los factores humanos y el desarrollo de software desde un punto de vista general, incluso aunque sea posible relacionarlos lógicamente con TDD.

EC3 En la medida en que puedan existir dudas al respecto, excluimos de esta revisión estudios que se refieren a la relación entre los factores humanos y el desarrollo de software desde una perspectiva teórica u opinión experta, pero sin fundamento empírico.

Además, se establecieron exclusiones adicionales para mejorar la confiabilidad de las fuentes de información seleccionadas:

EC4 Literatura gris y páginas web, con el objetivo de minimizar el riesgo de sesgo de investigación . No obstante, es conveniente indicar que debido al criterio de exclusión EC3, es poco probable que exista una cantidad sustancial de literatura gris y páginas web cuyos resultados estén empíricamente basados.

EC5 Estudios publicados en otros idiomas además del inglés, dado que las revistas y conferencias de mayor calidad en Ingeniería de Software Experimental usan el idioma inglés.

2.1.2. Estrategia de Búsqueda

La estrategia de búsqueda establecida para esta investigación incluye las siguientes actividades: (1) Definición de la cadena de búsqueda y (2) Ajuste del grupo de control.

El origen de los términos de la cadena de búsqueda generalmente se los justifica en el curso ordinario de un proceso de SLR, por ejemplo, a través de un grupo de control de literatura (CG)[?]. El CG es un conjunto de estudios que cumplen fielmente con la IEC propuesta, a partir de la cual se establece la cadena de búsqueda, extrayendo los términos más usuales y relevantes relacionados con el tema de investigación. Sin embargo, los estudios, donde los factores humanos están explícitamente relacionados con el desarrollo de software basado en TDD, resultaron difíciles de identificar, lo que nos obligó a cambiar la estrategia de búsqueda. Como alternativa, planeamos identificar estudios sobre TDD y, a partir de dichos estudios, localizar estudios primarios candidatos sobre los que aplicar criterios de inclusión y exclusión.

La estrategia de identificar estudios primarios a partir de estudios secundarios, a menudo es aplicada cuando la investigación en un determinado campo ya tiene varios años de evolución, partiendo del supuesto de que un estudio secundario debería contener todos los estudios primarios que se ajustan a los criterios de inclusión y exclusión ahí definidos. Esta estrategia ha

sido adoptada por ejemplo, en el estudio de [38], donde los investigadores parten del análisis de estudios secundarios realizados sobre TDD para obtener los estudios primarios candidatos que han sido publicados en un determinado periodo de tiempo (que sería el de la fecha de publicación del último estudio secundario). Con el objeto de extraer aquellos artículos publicados dentro del periodo de tiempo no considerado en los estudios secundarios previos, se realiza una nueva búsqueda para de esta forma obtener una lista de estudios candidatos actualizada. Esta estrategia evita un esfuerzo considerable de investigación bibliográfica.

Partiendo de esta premisa, configuramos nuestra cadena para realizar la búsqueda de estudios primarios. Según [39], la cadena de búsqueda para encontrar estudios secundarios tiene dos componentes principales: el primero debe contener los sinónimos más comúnmente utilizados con respecto a los estudios secundarios. El otro debe incluir los sinónimos más relevantes sobre el tema de investigación.

Utilizamos la cadena propuesta en [39] para estructurar el primer componente de nuestra cadena, pero excluimos términos irrelevantes como "meta-analysis ", y agregamos otros sinónimos relevantes sobre estudios secundarios como "in-depth review ". Nuestro segundo componente incluyó los sinónimos más relevantes de nuestro tema de investigación.

Finalmente, realizamos un proceso iterativo de prueba y error alternando los términos seleccionados alrededor de la cadena de búsqueda en la Base de datos digital Scopus (DDB). Seleccionamos la siguiente cadena de búsqueda: *(("test-driven development" OR "test driven development" OR "tdd" OR "test first" OR "test-first") AND ("literature review" OR "systematic literature review" OR "mapping study" OR "systematic map" OR "literature analysis" OR "in-depth review"))*, dado que devolvió un número manejable de estudios, con títulos estrechamente relacionados con nuestro tema de investigación.

2.1.3. Selección de estudios secundarios

Para obtener los estudios secundarios, la cadena de búsqueda definida en la sección anterior fue adaptada a diferentes bases de datos digitales. Obtuvimos 974 estudios candidatos distribuidos de la siguiente manera: Scopus 517, SpringerLink 277, IEEEXplore 9 y ACM 171. Se revisaron exhaustivamente los estudios candidatos para identificar estudios secundarios sobre TDD. Como resultado, se obtuvieron *7 estudios secundarios*(seis revisiones sistemáticas de literatura y un meta-análisis) que sintetizan los resultados obtenidos en un amplio número de estudios primarios (más de 100) que comparan TDD y otras estrategias de desarrollo, en especial el desarrollo tradicional. Estos estudios se muestran en la tabla 2.1.

Tabla 2.1: Estudios secundarios

Código	Estudio
CS1	The effects of test-driven development on external quality and productivity: A meta-analysis [15]
CS2	How Effective is Test-Driven Development? [16]
CS3	Test-Driven Development - Still a Promising Approach? [17]
CS4	Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review [18]
CS5	Effects of Test-Driven Development: A Comparative Analysis of Empirical Studies [19]
CS6	Considering rigor and relevance when evaluating test driven development: A systematic review [20]
CS7	The effects of test driven development on internal quality, external quality and productivity: A systematic review [21]

2.1.4. Selección de estudios primarios

Para obtener la selección de estudios primarios candidatos, revisamos los estudios primarios referenciados en los estudios secundarios. Esto arrojó un total de *114 estudios primarios candidatos*, mostrados en la Tabla 2.2.

Tabla 2.2: Estudios primarios candidatos obtenidos de los estudios secundarios

Cód.	Estudios secundarios	Primary studies
CS1	The effects of test-driven development on external quality and productivity: A meta-analysis [15]	[40], [41], [42], [43], [44], [45], [7], [8], [12], [46], [47], [48], [9], [49], [50], [51], [52], [53]
CS2	How Effective is Test-Driven Development? [16]	[10], [45], [54], [55], [56], [57], [8], [12], [58], [59], [13], [60], [9], [49], [50], [61], [62], [63], [52], [64], [65]
CS3	Test-Driven Development - Still a Promising Approach? [17]	[40], [10], [66], [42], [43], [44], [67], [45], [54], [68], [7], [56], [8], [12], [69], [70], [46], [59], [13], [60], [71], [47], [9], [49], [50], [72], [51], [73], [61], [62], [63], [52], [74], [53], [65]
CS4	Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review [18]	[75], [40], [10], [66], [41], [42], [76], [45], [54], [7], [56], [8], [11], [77], [12], [69], [70], [46], [78], [79], [13], [80], [60], [81], [71], [82], [47], [83], [9], [84], [49], [51], [85], [86], [73], [61], [62], [63], [87], [52]
CS5	Effects of Test-Driven Development: A Comparative Analysis of Empirical Studies [19]	[40], [10], [43], [88], [68], [56], [8], [12], [69], [46], [60], [71], [47], [84] [49], [89], [63], [90], [52]
CS6	Considering rigor and relevance when evaluating test driven development: A systematic review [20]	[91], [40], [10], [92], [41], [93], [43], [88], [45], [94], [54], [7], [56], [8], [12], [69], [95], [96], [97], [98], [59], [60], [71], [47], [9], [49], [50], [89], [51], [73], [61], [62], [63], [90], [52]
CS7	The effects of test driven development on internal quality, external quality and productivity: A systematic review [21]	[91], [99], [40], [10], [41], [44], [45], [68], [7], [56], [8], [69], [70], [96], [46], [60], [71], [47], [49], [50], [89], [51], [73], [62], [100], [63], [64]

La mayoría de los 114 estudios candidatos sobre TDD, no tenían como objeto de investigación los factores humanos. La revisión del abstract, introducción y conclusiones (apartados utilizados habitualmente en los protocolos de revisiones de literatura para la selección de estudios ej: [101]) no

permitió aplicar los criterios de inclusión/exclusión con el suficiente rigor y, en consecuencia, no fue posible identificar los estudios primarios en una fase temprana.

Como alternativa, nos planteamos la lectura en profundidad de los estudios candidatos que fueron registrados en una tabla denominada Matriz de Estudios Candidatos (MEC) en donde se identificó con una (X) aquellos estudios que presentaron resultados de acuerdo a los criterios de inclusión de la sección 2.1.1. Además, se marcó con un signo de admiración (!) aquellos estudios que presentaban resultados cualitativos, que proporcionaban opiniones respecto a, o discutían, los factores humanos que podían haber influenciado sus hallazgos.

Una revisión sistemática de Literatura conlleva un notable esfuerzo y tiempo para su realización (como se evidencia en algunos estudios reportados en [102]), por lo que se optó por realizar un estudio independiente a esta tesis con la participación de otros investigadores. Por este motivo, los estudios de la MEC se distribuyeron de manera proporcional y aleatoria a cuatro investigadores (Rodrigo Fonseca, Jhon Castro, Fernando Uyaguari y Geovanny Raura) para su revisión de acuerdo a los criterios de inclusión/exclusión indicados en la sección 2.1.1.

2.1.5. Extracción de datos

En la Matriz de Estudios Candidatos (MEC) se definieron seis campos: el nombre del estudio, los autores, los factores personales de interés, el tipo de estudio realizado (experimento, estudio de caso, etc.), un campo para registrar observaciones importantes y finalmente el investigador designado para revisar el estudio. Por razones de espacio esta tabla no ha sido incluida pero se encuentra accesible en `https://github.com/georaura/tddexperiments/tree/master/appendices`.

Durante el proceso de revisión, cada investigador realizó un análisis exhaustivo de los estudios asignados. Para aprovechar el esfuerzo, se marcaron las secciones de los artículos donde los autores mostraban resultados, comentarios o conclusiones respecto a los factores humanos objeto de nuestro estudio. Para mantener un adecuado control de la revisión, se realizaron reuniones con todos los investigadores de manera semanal. En promedio, se analizaron 8 estudios por semana, dando un tiempo total de revisión de 14 semanas aproximadamente.

Con el propósito de no sesgar la selección de estudios y, o, extracción de datos, se utilizó el mecanismo de triangulación propuesto por [103]. Este mecanismo consiste en analizar los fenómenos desde diferentes perspectivas para evitar amenazas a la validez en estudios cualitativos y dar una mayor fortaleza a los hallazgos. En concreto, un investigador (el proponente de esta tesis) definió una segunda tabla denominada Matriz de Características del Estu-

dio y Efectos Observados -MCE^{2}O- (Por razones de espacio, esta tabla se encuentra disponible en `https://github.com/georaura/tddexperiments/tree/master/appendices`) con el propósito de realizar una segunda revisión de aquellos artículos que fueron marcados sea con (X) o con (!) en la tabla MEC.

En la tabla -MCE^{2}O- se identificaron el tipo de estudio (experimento, caso de estudio, encuesta, etc.), el lugar de realización del estudio (academia o industria), el número de sujetos participantes, el tipo de sujetos (estudiantes de grado o postgrado, profesionales). Se complementó esta tabla con los efectos observados, detallando una hipótesis principal, una hipótesis secundaria, las métricas utilizadas, los resultados obtenidos y observaciones importantes. **A todos los efectos, la MCE^{2}O constituye la culminación de la extracción de datos**.

Finalmente, en una reunión de todos los investigadores, se resolvieron las divergencias encontradas y se refinó la MCE^{2}O. Como resultado, identificamos *26 estudios primarios* como se indica en la Tabla 2.3 denominada Matriz de Estudios Primarios y Factores Humanos (MEPFH)

2.1.6. Resultados reportados en los estudios secundarios

Los estudios secundarios identificados realizan una síntesis, desde su propia perspectiva, de los estudios empíricos en TDD. En esta sección presentamos los principales hallazgos reportados en los estudios secundarios. Nuestro interés se centra principalmente en los efectos de TDD sobre la calidad y la productividad y la influencia de los factores humanos, aunque eventualmente analizamos otros aspectos relacionados con las variables de nuestro interés. Al final de la sección presentamos un resumen con los principales hallazgos encontrados.

2.1.6.1. Turhan et al. [16]

Turhan et al, realiza una revisión sistemática de literatura con el objetivo de obtener evidencia empírica sobre la calidad interna y externa del código, la productividad y la calidad de los test desarrollados aplicando TDD. En total, los investigadores analizaron 32 estudios empíricos (8 experimentos controlados, 14 estudios pilotos, y 10 casos de uso en la industria).

Los autores analizaron estudios que informaron resultados cuantitativos de los efectos de TDD sobre las variables respuesta. Sin embargo, indican que **la comparación directa de la resultados cuantitativos en todos los ensayos fue imposible, ya que los ensayos midieron la efectividad de TDD de maneras diferentes.** En su lugar, asignaron a cada ensayo un valor cualitativo de ”mejor”, ”peor”, ”mixto” o ”inconcluso / sin diferencias”. Los resultados son los siguientes:

Tabla 2.3: Matriz de Estudios Primarios y Factores Humanos - MEPFH (Estudios cuantitativos (X), Estudios Mixtos (X)(!)

	PERSONAL FACTORS														
Author	Overall Personnel Experience	Programming language experience	Tool experience	Task-specific expertise	Training level or workshops	Schedule pressure	Analyst capabilities	Psycology Factors	Communication	Team Size	Methodological discipline employed	Age	Level of education	Personality
[9]							X							
[76]								!						X
[57]					X									
[104]				X										
[54]								X						
[105]									X					
[85]									x					
[106]	X													
[100]										x				
[107]	x							!						
[8]					x									
[70]	x	!											!	
[84]	x							!						
[77]	!						X							
[108]	!							x						
[78]				X				!						
[11]	x							!						
[12]				X	!									
[109]				X	!						!			
[63]			!					X						
[71]	!	x												
[91]	X	!												
[99]				X										
[110]	X													
[14]	X	!	!											
[111]		X	!											

- Los autores no encontraron diferencias significativas en cuanto a la calidad interna al aplicar TDD en comparación con Test-Last Development (TLD), aunque TDD parece producir mejores resultados sobre el grupo de control para ciertos tipos de métricas (complejidad y reutilización); otras métricas (acoplamiento y cohesión) obtienen peores resultados al utilizar TDD. Otra observación es que TDD produce código que es menos complejo a nivel de método / clase, pero más complejo a nivel de paquete / proyecto. Por otra parte, los autores indican que **las diferencias en la calidad interna pueden deberse a otros factores como la motivación, la habilidad, la experiencia, y efectos de aprendizaje de los participantes.**

- Los estudios sugieren que TDD mejora la calidad externa especialmente en estudios industriales. En experimentos controlados y estudios piloto, la evidencia empírica presenta resultados contradictorios. En promedio, TDD mejora la calidad externa.

- No existe consistencia sobre el efecto de TDD en la productividad. De acuerdo a los autores, este es uno de los aspectos que ha generado la mayoría de controversias y discusión. Por un lado, hay quienes sostienen que la curva de aprendizaje que se requiere para aplicar TDD podría decrementar la productividad al inicio, aunque no hay un consenso sobre los efectos a largo plazo. Por otro lado, hay quienes argumentan que TDD podría incrementar la productividad por diferentes razones como: se pueden detectar errores rápidamente, al realizar test automáticos se espera que la cantidad de errores vaya disminuyendo, la corrección de errores sea más sencilla debido a la simplicidad del diseño, etc. Por otro lado, se observan también diferentes medidas de productividad en los estudios primarios como el esfuerzo de desarrollo y mantenimiento, la cantidad de código o características producidas durante tiempo y la cantidad de código o características producidas por unidad de esfuerzo de desarrollo, lo que sugiere de acuerdo a la evidencia disponible, que TDD no tenga un efecto consistente sobre la productividad.

- Se encuentra evidencia de que TDD mejora la calidad de las pruebas.

En cuanto a las características personales, los autores realizan una clasificación de estudios por **el grado de experiencia de los participantes**, dividiéndolos en estudiantes de pregrado (menor nivel de rigurosidad), graduados y profesionales (mayor nivel de rigurosidad) [1]. No se presentan resultados directamente relacionados con el grado de experiencia, mas bien

[1] Nótese que la dicotomía estudiante vs. profesional puede también interpretarse en términos de factores humanos

se presentan los efectos sobre la calidad y productividad clasificados por tipos de estudio: experimentos controlados, estudios piloto y estudios de uso en la industria. **En los estudios en la industria (que generalmente son realizados con la participación de profesionales con mayor experiencia que los estudiantes) los resultados favorecen a TDD en la calidad interna y externa** (de un total de cuatro estudios, tres fueron clasificados como Mejora en favor de TDD para la calidad interna, y seis de un total de siete estudios también fueron clasificados como Mejora en favor de TDD para la calidad externa). **En lo referente a la productividad, los estudios en la industria muestran que la misma disminuye al utilizar TDD en comparación con el desarrollo tradicional** (de un total de siete estudios en la industria, 5 fueron clasificados indicando que TDD empeora la productividad).

2.1.6.2. Kollanus[17]

Kollanus analiza 40 estudios empíricos (25 experimentos, 14 casos de estudio y una encuesta). De acuerdo al autor, los estudios muestran resultados contradictorios respecto a los efectos de TDD en la productividad y en la calidad interna y externa.

No se analizan factores personales. Como única salvedad, se menciona que **los participantes de los estudios suelen ser estudiantes o profesionales que no tienen experiencia con TDD**. El autor hipotetiza que, de utilizar programadores experimentados que estén acostumbrados a utilizar TDD, los resultados de los estudios podrían ser diferentes.

2.1.6.3. Causevic et. al[18]

Causevic et al, realizan una revisión de literatura y analizan los factores que limitan la adopción de TDD en la industria. Se identificaron siete factores limitantes en 48 estudios empíricos (25 experimentos, 20 casos de estudio y 2 encuestas), de los cuales dos tienen relación con factores humanos:

- **La falta de experiencia o conocimiento previo de TDD por parte de los programadores**. Los autores mencionan que los participantes en los experimentos (ya sea estudiantes o profesionales) recibieron principalmente capacitación o tutoriales sobre cómo aplicar TDD. En varios casos, el conocimiento mejoró a medida que los participantes progresaron con el experimento. Se identificaron cuatro estudios referentes a la experiencia o conocimiento previo en TDD. En dos estudios de caso realizados en la industria con la participación de profesionales, los desarrolladores atribuyeron los problemas de implementación con TDD a la falta de educación o experiencia en esta técnica. Además, otros dos estudios reportan **diferencias significativas en la forma**

de aplicar TDD entre desarrolladores experimentados y novatos. Los autores concluyen que la falta de conocimiento o experiencia en TDD podría crear problemas en su adopción.

- **La falta de habilidad del desarrollador en la escritura de casos de prueba**. Se identificaron tres estudios primarios referentes a este aspecto. Dos de los estudios reportaron experiencias negativas en desarrollo de pruebas aplicando TDD. Uno de estos estudios fue realizado en el ámbito académico; los estudiantes expresaron dificultades para llegar a desarrollar buenos casos de prueba al utilizar TDD. Otro de los estudios fue realizado en un contexto mixto entre estudiantes y profesionales, se observó que **la falta de habilidades para escribir casos de prueba, es un factor limitante para adoptar TDD.** Los autores además indican que los estudios primarios identificados, no investigan directamente cómo se diseñan los casos de prueba y si el diseño de casos de prueba para TDD es diferente de cómo los ingenieros de pruebas experimentados lo están realizando.

Otros factores limitantes que se identificaron son: el mayor tiempo de desarrollo, falta de anticipación al diseño, problemas específicos con el dominio y herramientas de pruebas, poco cumplimiento del protocolo TDD, y problemas al aplicar TDD partiendo de código heredado.

2.1.6.4. Rafique and Misic [15]

Los autores presentan un meta-análisis de 27 experimentos sobre los efectos en la calidad externa y la productividad al aplicar TDD.

Los resultados indican que TDD tiene un pequeño efecto positivo sobre la calidad externa pero poco o ningún efecto claramente observable sobre la productividad. Tanto el modelo de efectos fijos y aleatorios mostraron un tamaño de efecto muy pequeño, y estadísticamente no significativo para las dos variables respuesta. Sin embargo, **al analizar los estudios industriales en comparación con los estudios académicos, se encuentran ciertas diferencias: Se mejora la calidad y disminuye la productividad en los estudios industriales, aunque los resultados tampoco fueron estadísticamente significativos.**

En cuanto a la calidad externa, los autores indican que **la diferencia podría explicarse por la experiencia de los participantes y por el tamaño de las tareas experimentales. Con respecto a la experiencia, mencionan que es razonable pensar que los desarrolladores industriales (de quienes se puede esperar que tengan un nivel de experiencia mucho mayor que los académicos), es probable que logren una mayor conformidad con TDD y de esta manera conseguir una mejora en la calidad.** En relación al tamaño de la tarea, la evidencia

empírica muestra que los beneficios de TDD pueden observarse con el tiempo, por lo tanto, los autores concluyen que en los experimentos académicos, que generalmente son de corta duración, la tendencia es que los tamaños de efecto sean menores que en los estudios en la industria.

En lo que respecta a la productividad, los autores observan que muchos estudios académicos han reportado problemas con la correcta aplicación de la técnica de TDD. Argumentan que los desarrolladores con mayor experiencia pueden lograr niveles más altos de conformidad con el proceso y, por lo tanto, prestar más atención y dedicar más tiempo a los procesos específicos de TDD. Se menciona que actividades como pruebas unitarias, refactorización y similares, pueden incidir en el aumento de la duración del proyecto y por lo tanto disminuir la productividad de los desarrolladores más experimentados.

Dadas las diferencias en los experimentos realizados con estudiantes y profesionales en el ámbito académico e industrial, **los autores también analizaron la experiencia de los desarrolladores como una variable moderadora para intentar explicar sus hallazgos. Se identificaron únicamente tres estudios de este tipo, los cuales mostraron resultados contradictorios:**

- En el primer estudio[84] se encontró que los profesionales terminaron la tarea en menos tiempo, y este resultado fue estadísticamente significativo. La diferencia se atribuyó a una mayor rapidez en la codificación y mayor nivel de experiencia en programación. Sin embargo, una mayor proporción de programas preparados por los estudiantes pasaron las pruebas de aceptación, pero este resultado no fue estadísticamente significativo.

- En el segundo estudio[11] los resultados difirieron notablemente de los del anterior: Se encontró que los profesionales generalmente tardaban más en completar el programa que los grupos de novatos. Los resultados fueron estadísticamente significativos para uno de los tres grupos de sujetos con los que se realizó este experimento. Este resultado fue coincidente con los obtenidos por los autores al comprar los estudios académicos e industriales, asumiendo que los estudiantes tienen menos experiencia que los profesionales. En este caso, no se reportaron resultados respecto a la calidad.

- En el tercer estudio[112] los resultados indicaron una pequeña correlación entre la experiencia del programador y la calidad externa en el grupo que utilizó TDD, pero esta observación no fue estadísticamente significativa.

Otro resultado reportado por los autores, aunque no tiene una relación directa con factores humanos, es la influencia de la complejidad de la tarea. Altos niveles de complejidad podrían hacer más difícil la aplicación de TDD,

ya que implica mayor esfuerzo y un impacto negativo en la productividad. Los autores recomiendan que este aspecto debería ser estudiado con más detalle para obtener conclusiones definitivas.

2.1.6.5. Makinen and Munch [19]

Makinen y Munch realizan una revisión de literatura integradora (integrative literature review) para analizar las experiencias de los estudios empíricos existentes de la industria y academia sobre los efectos de TDD.

Esta revisión de literatura engloba 19 publicaciones (ocho en el contexto académico, ocho en el contexto industrial y tres mixtos). Los casos de estudio y experimentos con profesionales fueron catalogados en el contexto industrial, en tanto que los experimentos controlados y cuasi-experimentos con estudiantes fueron catalogados en el contexto académico. Con esta clasificación extraen conclusiones en función de las variables respuesta que han sido reportadas en los estudios primarios: densidad y número de defectos, cobertura de código, complejidad, acoplamiento y cohesión, tamaño del código, esfuerzo, mantenibilidad, calidad externa y productividad.

En una agregación de resultados, los autores mencionan que la mayoría de efectos positivos de TDD fueron aquellos que analizaron la densidad y número de defectos, aunque algunos estudios también reportaron efectos positivos en relación a la calidad externa, complejidad, mantenibilidad y tamaño del código. La mayoría de efectos negativos reportados hacen referencia al esfuerzo y productividad, lo que podría indicar, señalan los autores, **la presencia de factores de contexto ocultos que tienen influencia en el efecto de TDD.**

En cuanto a efectos relacionados con factores humanos, los autores encuentran que algunos estudios primarios reportaron diferencias entre desarrolladores expertos y estudiantes. En uno de los estudios analizados [84], se encuentra que los expertos generalmente fueron más rápidos al escribir código que los estudiantes, aunque ambos grupos escribieron código de pruebas más rápido que el código de producción. En este caso, la productividad fue asociada de forma indirecta con la velocidad del desarrollo.

Sobre la calidad externa y el efecto de la experiencia con TDD, los autores hacen un análisis cualitativo en estudios que presentan resultados de cuestionarios y entrevistas. En este caso, no reportan estudios que comparen estudiantes y profesionales, sin embargo, al analizar la calidad interna, los autores mencionan que **la experiencia de los desarrolladores en general parece ser un factor para explicar cómo trabajan de manera individual y qué tan bien pueden adherirse al proceso de desarrollo basado en pruebas**. De la evidencia encontrada, los autores concluyen por ejemplo, que es posible que los estudiantes que no estén familiarizados con el concepto de TDD no puedan lograr una cobertura de código tan alta como sus pares de la industria, además que la complejidad del código escrito

por los desarrolladores en la industria es mayor que aquella desarrollada por los estudiantes.

2.1.6.6. Munir et. al.[20]

Munir et. al, analizan los efectos de TDD en la calidad y productividad, de acuerdo a: 1) rigor del método experimental aplicado (por ejemplo, los estudios donde el contexto, el diseño y las amenazas a la valides fueron adecuadamente descritos, se los clasificó como de alto rigor) y 2:) la relevancia del estudio (entendida como el grado de impacto en la industria en función de cuatro indicadores: el tipo de sujetos, la metodología de investigación, el análisis de un sistema real en la industria o de pequeños programas como tareas experimentales, y si el contexto de estudio fue real o en laboratorio). Para ello, realizan una revisión sistemática de literatura que incluye 41 estudios primarios (25 experimentos, 13 estudios de caso y 3 encuestas). Los estudios primarios son clasificados en cuatro grupos de acuerdo a su rigor y relevancia.

Los autores indican que las conclusiones que pueden obtenerse varían en función del grupo al que pertenecen los estudios. **Los estudios clasificados como de alta relevancia y rigor indican que TDD tiene un efecto positivo sobre la calidad externa del código**, aunque los autores advierten que esta conclusión debería justificarse con un mayor número de estudios que los disponibles en su revisión. Para los otros grupos, no se advierten mayores diferencias entre TDD comparado con TLD.

En esta revisión de literatura, **no se analizan factores humanos directamente, sin embargo los estudios de mayor rigor y relevancia (aquellos con mayor impacto en la industria), son estudios hechos con profesionales con más experiencia que los estudios de menor impacto, hechos en ambientes académicos con estudiantes que se asume son menos experimentados**. Por lo tanto, las conclusiones entes mencionadas podrían extrapolarse desde esta perspectiva.

2.1.6.7. Bissi et al. (2016)[21]

Esta revisión sistemática de literatura analiza los efectos de TDD sobre la calidad interna y externa, y la productividad. Los autores analizaron 27 estudios primarios (15 experimentos, 9 casos de estudio, 2 cuestionarios y una simulación). Las conclusiones del estudio indican que:

- El 76 % de estudios confirman un incremento en la calidad interna, y un 88 % de estudios reportan un aumento significativo en la calidad externa.
- Alrededor del 44 % de estudios indican que la productividad decrece al aplicar TDD en comparación con TLD. En el resto de casos, no se

reportaron diferencias o existió un incremento (en un 28 % para ambos casos).

En esta revisión sistemática de literatura no se analizan factores humanos propiamente dichos, aunque los estudios son clasificados de acuerdo a su carácter académico (donde los sujetos son estudiantes) o industrial (con sujetos profesionales). El análisis independiente de dichos grupos muestra que **en los estudios académicos se produce un incremento de productividad, en comparación con los estudios en la industria, donde la productividad decrece**. En relación a la **calidad externa**, todos los estudios que analizan este factor **en la industria** (9 estudios) reportaron **efectos positivos de TDD**. Mientras que de un total de 8 estudios, uno sólo reportó efectos negativos y otro **sin diferencias en el ámbito académico**.

2.1.7. Conclusiones de los estudios secundarios

La tabla 2.4 muestra un resumen de los estudios secundarios analizados:

Tabla 2.4: Resumen del efecto de factores humanos en estudios secundarios

Estudio	**Overall Personnel Experience**	**Task-specific expertise**	**Psycology Factors and Training level or workshops**
Turhan et al (2010)	(*) Tdd mejora la calidad interna y externa. La productividad disminuye en estudios con profesionales	Se menciona la habilidad del desarrollador como posible factor que afectan los resultados de los estudios en cuanto a la calidad interna.	Se mencionan como posibles factores que afectan los resultados de los estudios en cuanto a la calidad interna.
Kollanus (2010)	(*) Se menciona como posible factor que afecta los resultados de los estudios		
Causevic et. al (2011)		Algunos estudios reportan diferencias significativas en la aplicación de TDD entre desarrolladores experimentados y novatos. Concluyen que la falta de conocimiento o experiencia en TDD y la habilidad del desarrollador para escribir casos de prueba podría crear problemas en su adopción.	
Rafique and Misic (2013)	En estudios que comparan estudiantes con profesionales, TDD mejora la calidad externa y disminuye la productividad de los profesionales en comparación con los estudiantes.		
Makinen and Munch (2014)	(*) Algunos estudios reportaron diferencias entre desarrolladores expertos y estudiantes. Un estudio mostró que los expertos fueron más rápidos al excribir código que los estudiantes. Al analizar la calidad interna se menciona a la experiencia como un posible factor que efacta el proceso de desarrollo con TDD.		
Munir et. al (2014)	(*) TDD tiene un efecto positivo sobre la calidad externa del código, en estudios clasificados como de alta relevancia y realizados en un contexto industrial con profesionales.		
Bissi et al. (2016)	(*) TDD tiene un efecto positivo sobre la productividad en estudios académicos. En estudios en la industria, la productividad decrece. En los dos casos se reportaron efetos positivos sobre la calidad externa		

(*) La experiencia del desarrollador no fue analizada de forma directa sino como parte de estudios realizados con profesionales y estudiantes

Los factores humanos que son analizados en los estudios secundarios son: la experiencia, la habilidad de los sujetos para aplicar TDD en comparación con otras estrategias, la motivación de los participantes de los experimentos y otros efectos relacionados con el aprendizaje de las técnicas. En las siguientes secciones sintetizamos estos hallazgos por cada uno de estos factores.

2.1.7.1. Overall Personnel Experience

La mayoría de estudios secundarios (excepto el de Causevic et al.[18]) mencionan el efecto de la experiencia de los desarrolladores como un factor

humano que pude incidir en la efectividad de TDD. Este efecto es observado principalmente al comparar estudios en el ámbito académico e industrial. Un único estudio (Rafique and Misic[15]) presenta resultados cuantitativos respecto a la experiencia como variable moderadora. Para el resto de estudios hemos considerado las diferencias entre estudiantes y profesionales asumiendo razonablemente que la experiencia de los desarrolladores en el ámbito académico e industrial es diferente.

Respecto a la productividad y la experiencia, el estudio de Rafique and Misic[15] señala que al aplicar TDD, algunos estudios primarios mostraron que se obtiene un incremento en la productividad en los programadores menos experimentados (estudiantes) en comparación con los programadores más experimentados (profesionales). Además, al comparar estudios académicos e industriales, este efecto es coincidente con los resultados de Turhan et al[16] y Bissi et al[21]): La productividad disminuye en estudios con profesionales comparados con estudios con estudiantes. Makinen and Munch[19], por otro lado, encontraron un estudio donde se observa que los profesionales escribieron código más rápido que los estudiantes. Rapidez y productividad no significan lo mismo, pero los autores señalan que la escritura de pruebas acelera de alguna manera la velocidad de implementación de las historias de usuario o afectan la velocidad con la que los desarrolladores escriben líneas de código fuente.

En relación a la calidad y la experiencia, los resultados de Turhan et al.[16], Munir et. al.[20] y Bissi et al.[21] señalan que TDD tiene un efecto positivo en la calidad externa en estudios con profesionales y estudiantes. Al analizar la calidad interna, Turhan et al.[16] de igual forma reporta un efecto positivo de TDD en estudios en la industria. Makinen and Munch[19], también menciona a la experiencia como un factor que podría afectar a los resultados de los estudios en razón de algunas diferencias de calidad interna reportadas entre desarrolladores expertos y estudiantes.

Kollanus[17] sólo advierte que puede existir un efecto de la experiencia al aplicar TDD, aunque no se hace un análisis específico al respecto.

2.1.7.2. Task-specific expertise

El conocimiento previo que tenían los participantes sobre TDD, bien puede catalogarse dentro de los aspectos relacionados con la experiencia. Sin embargo, lo hemos considerado de manera independiente, por ser un factor que está específicamente relacionado con el conocimiento de la técnica del desarrollo basado en pruebas (*Task-specific expertise*) que podría incidir notablemente en los resultados. Por ejemplo, el estudio de Causevic et. al[18] menciona que algunos estudios reportaron diferencias significativas entre desarrolladores que ya conocían la técnica y desarrolladores novatos. Concluyen que la falta de conocimiento o experiencia en TDD así como la habilidad del desarrollador para escribir casos de prueba, puede ser un factor

limitante para adoptar esta técnica. Este factor también es mencionado por Turhan et al.[16] como un aspecto que puede afectar a los resultados de los estudios.

2.1.7.3. Otros factores humanos (Psycology Factors and Training level or workshops)

El estudio secundario de Turhan et al.[16] menciona posibles efectos de la motivación y el aprendizaje del programador. Al analizar la calidad interna, los autores señalan que TDD parece producir resultados diversos dependiendo de los tipos de métricas analizadas (por ejemplo, TDD parece que produce mejores resultados si se analiza la complejidad y reutilización, pero empeora al analizar el acoplamiento y cohesión). Dada esta diversidad de resultados, los autores indican que las diferencias en la calidad interna pueden deberse a otros factores como la motivación, la habilidad, la experiencia, y efectos de aprendizaje de TDD de los participantes, aunque no se presentan resultados al respecto.

2.1.7.4. Notas finales

A manera de conclusión podemos señalar que la experiencia de los desarrolladores es el factor humano mayormente estudiado en los estudios secundarios, aunque no de manera directa, sino mediante la comparación de estudios académicos e industriales con la participación de estudiantes y profesionales. El único estudio (Rafique and Misic[15]) que ha intentado analizar este aspecto de forma cuantitativa, sólo pudo identificar tres estudios primarios relacionados, por lo que no consiguieron establecer una correlación confiable debido a la falta de datos sobre el nivel de experiencia en las diversas áreas tales como pruebas, refactorización, entre otros aspectos. La tabla 2.5 muestra los resultados sobre la Calidad y Productividad que han sido analizados en relación con la experiencia. Podemos observar que la Calidad Externa mejora en función de la experiencia, aunque la Productividad disminuye para el caso de los profesionales.

2.1.8. Resultados reportados en los estudios primarios

Los **factores principales** son los siguientes: Overall personnal experience[106] [107] [70] [84] [11] [91] [110] [14] [77] [108] [71], task-specific expertise[104] [78] [12] [109] [99], psicology factors[54] [108] [63] [76] [107] [84] [78] [11], programming language experience[71] [111] [70] [91] [14], training level or worshops[57] [8] [12] [109], analyst capabilities[113] [77], communication[105] [85], team size[100], y personality[76].

Tabla 2.5: Efecto de la experiencia sobre la Calidad y Productividad en estudios secundarios

	Experiencia		
Estudio	**Calidad interna**	**Calidad externa**	**Productividad**
[16] (*)	Mejora	Mejora	Disminuye con profesionales
[15]		Mejora con profesionales	Disminuye con profesionales
[20] (*)		Mejora con profesionales	
[21] (*)		Mejora	Disminuye con profesionales

(*) La experiencia del desarrollador no fue analizada de forma directa sino como parte de estudios realizados con profesionales y estudiantes

2.1.8.1. Experiencia personal en desarrollo

Con respecto a la experiencia en desarrollo **overall personnel experience**, por un lado encontramos que algunos estudios primarios afirman que no existen diferencias significativas entre: (i) los años de experiencia de los grupos de desarrolladores [70], en aspectos del desarrollo con TDD como el refactoring [91], (ii) los grupos de sujetos en función de su experiencia y las habilidades en las prácticas de TDD [14]. El trabajo de Höfer y Philipp [11] afirma que no existen diferencias significativas en la conformidad con el proceso de desarrollo con TDD entre novatos y expertos, ni en la calidad de las pruebas.

Por otro lado, existen algunos estudios primarios que afirman que existen diferencias significativas [110] [84] [107] [106]. Según Latorre [110], la capacidad de aplicar TDD depende inicialmente de la experiencia. Los desarrolladores más experimentados alcanzan un alto nivel de conformidad con el proceso, y obtienen un mejor rendimiento al realizar las tareas, sin embargo, el uso de TDD tuvo un impacto mínimo en la productividad. El trabajo de Müller, y Höfer [84] afirma que los expertos logran un mejor desempeño en la conformidad con las reglas de TDD, cobertura de bloque de código, tiempo de ejecución de pruebas unitarias, cambios en el código de pruebas y de la aplicación. Domino et al. [107] afirman que existen diferencias significativas en el rendimiento obtenido en algunas tareas para grupos con mayor experiencia cuando utilizaron TDD. Según Janzen y Saiedian [106] los estudiantes menos

experimentados se encuentran más abiertos a adoptar TDD. Mientras que el trabajo de Janzen y Saiedian [108] afirma que los programadores maduros son más propensos a elegir TDD en lugar de TLD. Los programadores principiantes se mostraron más renuentes a adoptar TDD.

2.1.8.2. Experiencia en tareas específicas

En cuanto al factor **task-specific expertise**, el desarrollo con TDD mejora las habilidades de refactorización a medida que los programadores adquieren experiencia, y mejora la confianza en el equipo con respecto a la calidad del código y simplifica el mantenimiento [109]. Además, según Aniche y Gerosa [99] la experiencia y el conocimiento previo del desarrollador son críticos al crear software orientado a objetos. Los sujetos sin experiencia previa en TDD tienen más dificultades al aplicar esta técnica aunque la diferencia no es significativa [78], y gastan más tiempo al realizar las pruebas que en la codificación [12]. Sin embargo, según Lui y Chan [104] los equipos sin experiencia previa en TDD, una vez adoptada esta técnica presentan mejoras en: (i) la calidad del software, (ii) estimación de tareas, y (iii) seguimiento de avances en el desarrollo y disciplina.

2.1.8.3. Factores psicológicos

Con relación a **psicology factors**, los sujetos experimentales (estudiantes de postgrado) muestran niveles de aceptación o rechazo a TDD durante el desarrollo del experimento [54]. Según el trabajo de Vu et al. [63], la ambición y motivación del equipo pueden tener más efecto que el enfoque de desarrollo aplicado. Los sujetos prefirieron TDD sobre TLD aunque las diferencias no fuero significativas. En el trabajo de Höfer y Philipp [11], los autores no observan diferencias significativas en la motivación al trabajar en parejas y con la participación en el experimento.

2.1.8.4. Experiencia en el lenguaje de programación

Con respecto al factor **programming language experience**, las habilidades de los desarrolladores como el lenguaje de programación y las pruebas unitarias tienen un impacto significativo en su productividad, pero no en la calidad externa [111]. Los resultados del trabajo de Madeyski [71] están en la misma línea de los obtenidos por Fucci et al. [111]. Madeyski [71] afirma que los desarrolladores con más experiencia en la tecnología usada pueden ser más efectivos para escribir pruebas y realizar código. Mientras que Janzen et al. [70] afirma que no existe una diferencia significativa entre TDD y TLD en términos de la experiencia en programación.

2.1.8.5. Nivel de entrenamiento o capacitación

En cuanto al factor **training level or worshops**, el trabajo de Geras [57] identifica que muy pocas organizaciones brindan capacitación en pruebas de software, y afirma que mejorar las habilidades en pruebas podría mejorar la efectividad de TDD. En el experimento realizado por Gupta y Jalote [8], los sujetos experimentales (estudiantes de postgrado y pregrado) afirman que necesitan más entrenamiento en TDD en comparación con CDD (Conventional Code Development).

2.1.8.6. Capacidad de análisis

Con relación al factor **analyst capabilities**, TDD parece que mejora la comprensión del programa según el experimento realizado por Müller y Hagner [113]. El trabajo de Huang et al. [77] concluye que la capacidad de análisis de los sujetos varía según su nivel y experiencia profesional. Los sujetos con mayor nivel y experiencia aumentan la probabilidad de escribir pruebas de mayor calidad, pero fallan más en el análisis de cobertura.

2.1.8.7. Comunicación

En cuanto a la **communication**, el estudio realizado en la industria por parte de Kobayashi et al. [105] encontró una estrecha relación entre la comunicación del equipo y la aplicación de TDD, lo cual redujo el costo de documentación. El trabajo de Sfetsos et al. [85] también encuentra que TDD tiene estrecha relación con la comunicación. La comunicación continua y la colaboración entre desarrolladores y clientes son aspectos importantes para empresas grandes y pequeñas [85].

2.1.8.8. Tamaño del equipo de desarrollo

Con respecto al factor **team size**, Turnu et al. [100] concluye que el número de desarrolladores que contribuyen a un proyecto FLOSS (Free Libre Open Source Software) no está relacionado con el uso de la práctica de TDD. La productividad disminuye cuando se adopta completamente esta técnica.

2.1.8.9. Personalidad

En cuanto al factor **personality**, el experimento en la industria realizado por Domino et al. [76] concluyó que los sujetos con estilos de resolución de conflictos más integradores, tuvieron mejores resultados de rendimiento y se mostraron más colaboradores.

2.1.8.10. Otros factores humanos mencionados

Dentro de nuestra selección de estudios primarios algunos autores han mencionado otros aspectos humanos: **Tool experience** ([63] [14] [111]), **Methodological discipline employed** ([109]), **Level of education** ([70]). Los resultados más relevantes de estos estudios (aquellos marcados con X) ya fueron analizados en las secciones anteriores, por lo cual no se incluye un análisis independiente de estos factores.

2.1.9. Preparación de la síntesis

Nuestro propósito principal ha sido identificar y clasificar los factores humanos que podrían influir en la calidad y productividad de los desarrolladores, para de esta manera categorizar nuestros hallazgos dentro de un marco más o menos estandarizado. No era de nuestro interés realizar una revisión sistemática de literatura sobre marcos de trabajo que aborden este objetivo. Sin embargo, realizamos diversas búsquedas no sistemáticas principalmente en Google Scholar y en IEEEXPLORE, utilizando una diversidad de cadenas, como por ejemplo: ("human Factors.ºR "Human Issues") AND ("Software quality.ºR "Software productivity"). En general las cadenas de búsqueda nos devolvieron una gran cantidad de artículos, pero en su gran mayoría no se relacionaron con los factores humanos que afecten a la calidad y productividad de los desarrolladores de software.

Como conclusión de esta búsqueda informal, encontramos que no existe un único framework para analizar los diferentes factores humanos en Ingeniería de Software. Existen trabajos que proponen diferentes frameworks dependiendo de los intereses particulares de los investigadores [114] [115] [116] [117] [118] [119] [120] [121] [122] [1]. El trabajo de França et al. [120] propone una teoría de la motivación laboral y la satisfacción laboral de los ingenieros de software (TMS-SE), basada inicialmente en la teoría de la satisfacción laboral y las características laborales mejorada y adaptada al contexto de desarrollo de software.

Para medir los factores psicológicos, usualmente el framework utilizado es el Five-Factor Model (FFM). El FFM (también conocido como *Big Five* está compuesto por cinco tipos de factores de personalidad: apertura a la experiencia, escrupulosidad, extraversión, amabilidad y neuroticismo. Este framework ha sido utilizado por ejemplo por Mellblom et al. [121] para investigar la conexión entre los tipos de personalidad de acuerdo al modelo FFM y el agotamiento en los ingenieros de software.

El trabajo de Trendowicz y Münch [122] reporta una descripción general de los factores que inciden en la productividad de acuerdo a los profesionales del software. Encuentran que la productividad de los procesos de desarrollo de software dependen significativamente de las capacidades de los desarrolladores, así como de las herramientas y métodos que utilizan. Los autores

categorizan los factores en cuatro grupos: producto, personal, proyecto y proceso. Por otra parte, Pirzadeh[1] identifica y caracteriza los factores humanos que influyen en el proceso de desarrollo desde las perspectiva del ciclo de vida del desarrollo y la gestión del software.

En vista de que no existe un único framework, hemos seleccionado la propuesta de Pirzadeh[1], debido a la sencilla y práctica clasificación que realiza de los factores humanos en tres niveles: individual, interpersonal y organizacional.

2.1.10. Síntesis de los estudios primarios

De acuerdo al framework de[1], podemos indicar que los factores humanos asignados en la categoría de *Individual* han sido los más analizados (22 artículos marcados como (X) y 14 artículos marcados como (!)). Le siguen los categorizados como *Interpersonal* (con 2 artículos marcados como (X) 3 artículos marcados como (!) y finalmente *Organizational* (con 3 artículos marcados como (X) y 1 artículo marcado como (!)). Como se desprende de la tabla 2.3), los principales factores humanos que influyen en TDD, en función al total de estudios marcados con (X) son: *Overall personnal experience (8 estudios), task-specific expertise (5 estudios) y Psycology Factors (4 estudios)*. El resto de factores tienen dos o menos estudios marcados con (X). A continuación presentamos una síntesis de resultados de acuerdo al framework de[1]:

2.1.10.1. Factores individuales

En esta categoría se discuten los aspectos humanos individuales de los desarrolladores y su efecto en la aplicación de TDD.

Con respecto al factor **overall personnel experience**, varios estudios ([110],[84],[107]) coinciden en líneas generales en que los desarrolladores con más experiencia tienen un mejor rendimiento y desempeño con las reglas de TDD. Si se considera la preferencia de TDD sobre TLD,[108] encuentran que los programadores más experimentados tienden a elegir TDD en lugar de TLD. Sin embargo, otro trabajo de los mismos autores[106] afirma que los estudiantes menos experimentados son quienes están más abiertos a adoptar TDD. En contraste con los trabajos anteriores, hay otros autores que afirman que no existen diferencias significativas entre los grupos de sujetos a la vista de su experiencia[70] [14].

Las diferencias anteriores entre ambos grupos, podrían deberse a factores como el diseño experimental, por ejemplo, si trabajaron solos o en parejas, o el tiempo destinado para realizar la tarea experimental. En el trabajo de LaTorre[110], los desarrolladores debían trabajar de manera individual. Mientras que en el estudio de Janzen et al.[70] los sujetos debían trabajar en parejas. En el estudio de Fucci et al.[14] los sujetos tuvieron 4 horas para

realizar la tarea experimental. Sin embargo, en el estudio de LaTorre[110], los participantes tenían cerca de un mes para terminar el desarrollo.

Con relación al factor **task-specific expertise**, el trabajo de Marchenko et al.[109] afirma que a medida que los desarrolladores adquieren experiencia en TDD, sus habilidades de refactorización y la confianza del equipo de trabajo mejoran la calidad del código. Resultados similares encuentran Lui y Chan[104] con respecto a los equipos que sin tener experiencia previa TDD, una vez empiezan a adoptar la técnica presentan mejoras en la calidad del código. Huang y Holcombe[12] afirman que quienes no tienen experiencia en TDD gastan más tiempo en las pruebas que en la codificación. Sin embargo, el trabajo de Kollanus y Isomöttönen[78] indica que los sujetos que no tienen experiencia previa en TDD tienen más impedimentos cuando aplican TDD, no obstante esa diferencia no es significativa. Estas diferencias podrían deberse por ejemplo al nivel de su formación. En el trabajo de Kollanus y Isomöttönen[78] los participantes eran estudiantes de magíster e incluso algunos tenían experiencia laboral. Mientras que en el estudio de Huang y Holcombe[12] los participantes eran estudiantes de pregrado.

En cuanto al factor **programming language experience**, la habilidad en el lenguaje de programación tiene un impacto significativo en la productividad, pero no en la calidad externa[111]. Los desarrolladores con mayor experiencia en la tecnología usada pueden ser mas efectivos escribiendo código y pruebas[71]. Sin embargo, los resultados del estudio de Janzen et al.[70] muestran que no existe diferencia significativa entre TDD y TLD con respecto a la experiencia en programación. Estas diferencias en los resultados podrían deberse, por ejemplo al tiempo destinado para el entrenamiento y al ambiente en el cual se realizaron los estudios. El estudio de Fucci et al.[111] fue ejecutado en un ambiente académico, y los sujetos recibieron 6 sesiones prácticas de laboratorio (de 3 horas cada una) sobre pruebas unitarias y desarrollo basado en pruebas en java. Mientras que el estudio de Janzen et al.[70] se realiza en un ambiente profesional y los desarrolladores recibieron 3 cursos. El primer curso introdujo C++ a programadores de C experimentados y tuvo una duración de cuatro días, con un segmento de desarrollo basado en pruebas en el último día del curso. El 2do y 3er curso introdujeron el desarrollo basado en pruebas para programadores Java experimentados. Ambos fueron cursos de dos días.

2.1.10.2. Factores interpersonales

En esta sección se presentan los resultados de artículos que abordan cuestiones relacionadas con la comunicación, la colaboración y la cooperación de las personas cuando actúan en grupo, cuando trabajan en una organización y cómo estos aspectos influyen en el proceso de desarrollo.

El factor **Communication** fue abordado por[105] y Sfetsos et al.[85] en estudios que analizan la programación por pares y TDD como prácticas del

desarrollo ágil. Los dos autores coinciden en que la efectividad de TDD tiene una estrecha relación con la comunicación del equipo. Por un lado, se mejoran las pruebas y se reduce el costo de documentación de los programas [105] y, por otro lado, la comunicación continua entre desarrolladores y clientes son aspectos considerados importantes para adoptar prácticas de desarrollo ágiles como TDD [85]. También se observan factores que limitan adoptar las prácticas de desarrollo ágiles cuando los equipos de desarrollo son de diferentes culturas.

El efecto del **tamaño del equipo de desarrollo** en la productividad al aplicar TDD, fue analizado por [123]. El autor concluye que la productividad disminuye cuando se adopta TDD en proyectos de software libre que son desarrollados por varios programadores. Además la cantidad de desarrolladores que contribuyen con el proyecto, no tiene relación con la práctica de TDD.

2.1.10.3. Factores organizacionales

Los artículos en este apartado, discuten los factores humanos involucrados en el nivel organizacional del proceso de desarrollo. Por ejemplo, el entorno de la organización (entorno de trabajo), talleres de capacitación, metodologías de desarrollo adoptadas por las organizaciones y los factores humanos que afectan su adopción. Con relación a estos factores, únicamente identificamos dos estudios relacionados con **training level or workshops**. Geras [57] y Gupta [8] hacen un análisis de la percepción de los desarrolladores al aplicar TDD. Las encuestas reflejan que muy pocas organizaciones brindan capacitación en pruebas de software, lo cual podría mejorar la efectividad de TDD en comparación con el desarrollo convencional de código.

2.1.10.4. Síntesis de los estudios primarios

La Tabla 2.6 presenta, para los factores humanos principales, un resumen de la síntesis realizada.

2.2. Síntesis general de la revisión de literatura

La revisión de literatura planteada en la sección 2.1 tuvo como objetivo identificar los factores humanos que se han estudiado en TDD y su posible influencia en los resultados obtenidos cuando los desarrolladores aplican esta técnica en comparación con otras técnicas de desarrollo tradicional. Como se muestra en la tabla 2.7, hemos identificado 9 factores humanos que se han mencionado en estudios primarios o secundarios como posibles aspectos que pueden influir al aplicar TDD. Se ha considerando en el análisis la clasificación propuesta por Pirzadeh [1].

Tabla 2.6: Resumen de la Síntesis

Factor humano	Existen diferencias significativas	No existen diferencias significativas	Posibles razones divergencia
Overall personnel experience	Los desarrolladores con mayor experiencia tienen un mejor rendimiento, y existe mayor conformidad con el proceso TDD [110] [84] [107]. Los desarrolladores más experimentados tienden a elegir TDD en lugar de TLD [108].	Los trabajos de Janzen et al. [70] y Fucci et al. [14] afirman que no existen diferencias significativas entre los grupos de sujetos en perspectiva con su experiencia.	**Forma de trabajo:** En LaTorre [110] los desarrolladores trabajaron individualmente. En tanto que en el estudio de Janzen et al. [70] los desarrolladores trabajaron en parejas. **Tiempo destinado para la tarea experimental:** En Fucci et al. [14] los desarrolladores tuvieron 4 horas para realizar la tarea experimental, pero en LaTorre [110] tuvieron cerca de un mes.
Task-specific expertise	A medida que la experiencia de los desarrolladores en TDD aumenta: (i) sus habilidades de refactorización y confianza del equipo mejoran [109], (ii) la calidad del código mejora [104].	Los sujetos sin experiencia previa en TDD tienen más dificultades cuando aplican TDD, pero esa diferencia no es significativa [78].	**Nivel de formación:** En el trabajo de Kollanus y Isomöttönen [78] los sujetos eran estudiantes de magíster y algunos tenían experiencia laboral. No obstante, en el trabajo de Huang y Holcombe [12] los sujetos eran estudiantes de pregrado.
Programming language experience	La habilidad en el lenguaje de programación tiene un impacto significativo en la productividad [111]. Los desarrolladores con mayor experiencia en la tecnología usada pueden ser más efectivos escribiendo código y pruebas [71].	El trabajo de Janzen et al. [70] afirma la no existencia de diferencia significativa entre TDD y TLD.	**Tiempo destinado para el entrenamiento y el ambiente en el cual se realizaron los estudios:** En Fucci et al. [111] los sujetos estaban en un ambiente académico y recibieron 6 sesiones prácticas de laboratorio. Sin embargo, en Janzen et al. [70] los sujetos pertenecían a un ambiente profesional, y recibieron 3 cursos (el 1ro para desarrolladores C experimentados con una duración de 4 días, el 2do y 3ero con una duración de 2 días cada uno).
Training level or workshops	La efectividad de TDD puede mejorar si lo hacen sus habilidades en pruebas [57]. Los sujetos necesitan más entrenamiento en TDD en comparación con el desarrollo tradicional.	No hay	N/A
Communication	Tiene relación con la aplicación de TDD reduciendo los costos de documentación [105] [85].	No hay	N/A

Tabla 2.7: Factores humanos analizados en estudios primarios y secundarios clasificados de acuerdo a [1]

Tipo Factor	**Factores**	**Estudios primarios**	**Estudios secundarios**
Individuales	Overall Personnel Experience	[106] [107] [70] [84] [11] [91] [110] [14] [77] [108] [71]	[16], [17], [15], [19], [20], [21]
	Task-specific expertise	[104] [78] [12] [109] [99]	[16], [18]
	Psicology factors	[54] [108] [63] [76] [107] [84] [78] [11]	[16]
	Programming language experience	[71] [111] [70] [91] [14]	
	Analyst capabilities	[113] [77]	
Interpersonales	Communication	[105], [85]	
	Team size	[100]	
	Personality	[76]	
organizacionales	Training level or worshops	[57] [8] [12] [109]	[16]

En síntesis podemos indicar que la experiencia de los participantes es el factor más analizado tanto en estudios primarios como secundarios. Los estudios que analizan la experiencia, en su mayoría comparan a estudiantes con profesionales quienes generalmente son los más experimentados. Por otra parte, las métricas utilizadas han sido diversas (ver la tabla `https://github.com/georaura/tddexperiments/tree/master/appendices`), razón por la cual no hemos conseguido realizar un meta-análisis que nos permita obtener conclusiones más sólidas sobre los efectos de los factores humanos sobre TDD. Sin embargo, los estudios (en su mayoría experimentos) demuestran, sin lugar a duda, que existe una influencia positiva o negativa de los diferentes factores humanos analizados en los resultados obtenidos al aplicar TDD.

Capítulo 3

PREGUNTAS DE INVESTIGACIÓN

Lo importante es no dejar de hacerse preguntas

Albert Einstein

Resumen:

En este capítulo planteamos distintas preguntas de investigación y el objetivo general del estudio.

3.1. Preguntas de investigación

De nuestro análisis del estado de la cuestión, se desprende que existen muy pocos estudios que hayan analizado los factores factores humanos que podrían influir en la calidad externa del software y la productividad de los programadores al aplicar TDD. Apenas logramos identificar 26 estudios primarios, de los cuales 17 son estudios experimentales y el resto corresponden a estudios de caso (3), encuestas (3), simulaciones (1), un estudio exploratorio y un estudio mixto (encuesta y experimento).

Los estudios primarios analizan principalmente la experiencia general en desarrollo (8), la experiencia en un lenguaje de programación (2) y la experiencia en la realización de tareas específicas (5). Otros aspectos menos analizados han sido los factores psicológicos (3), la capacidad de análisis (2), el nivel de entrenamiento (2), la comunicación (2), el tamaño del equipo (1) y la personalidad (1).

Los estudios que analizan estos aspectos personales llegan, en general, a resultados que han sido contradictorios o no concluyentes. Por ejemplo, en el trabajo de Hofer y Philipp [11] se afirma que no existen diferencias

significativas en la conformidad del proceso de desarrollo con TDD entre novatos y expertos. Por el contrario, Latorre[110] señala que la capacidad de aplicar TDD depende inicialmente de la experiencia; los desarrolladores más experimentados alcanzan un alto nivel de conformidad con el proceso, y son más productivos.

Otro aspecto observado es que muchas investigaciones presentan problemas metodológicos. Por ejemplo, cuando se analiza la calidad interna,[15],[19] y[17] advierten de problemas en la comparación de estudios primarios por la utilización de distintas métricas y criterios de medición. En este caso, las métricas más usuales han sido la cobertura de pruebas y el número de casos de prueba, pero también se midieron el tamaño del método, la complejidad ciclomática, el acoplamiento y la cohesión. La heterogeneidad de estudios no permite que los resultados sean fácilmente comparables, lo que produce que la agregación de estudios sea engorrosa y de lugar a amenazas a la validez[23]. Por esta razón Rafique and Misic[15] en su meta-análisis analizan únicamente la calidad externa y la productividad, dejando de lado otros factores como la calidad interna.

Debido a los problemas encontrados en los distintos estudios primarios, creemos importante realizar nuevos estudios que consideren las características de los sujetos y que, además, produzcan resultados que faciliten su agregación como una familia de experimentos y su comparación con estudios previos. La calidad externa y la productividad han sido las variables respuesta más frecuentemente usadas, y su agregación es factible. En consecuencia, nos planteamos las siguientes preguntas de investigación:

- **RQ1: ¿Cuál es la influencia de los factores humanos sobre la calidad externa cuando se utiliza TDD?**
- **RQ2: ¿Cuál es la influencia de los factores humanos sobre la productividad cuando se utiliza TDD?**

Por otro lado, cuando comenzamos a realizar nuestros estudios especialmente con profesionales, fuimos evidenciando que muchos de ellos entregaban las tareas sin apenas haber realizado código alguno (a lo que denominamos grado de completitud de la tarea). Creemos que este comportamiento también podría estar asociado a factores humanos como la motivación de los participantes. Por esta razón, se optó por la inclusión de una tercera pregunta de investigación:

RQ3: Cuál es la influencia de los factores humanos sobre el grado de completitud de las tareas experimentales cuando se utiliza TDD?

Al realizar la revisión de la literatura, se identificaron diferentes categorías de factores personales que pueden influir en el desarrollo de software: factores individuales, factores interpersonales y factores organizacionales.

Basados en esta clasificación, hemos seleccionado los factores más relevantes de acuerdo a nuestros intereses y también en función de las limitaciones del contexto de experimentación y los recursos disponibles para la ejecución de la investigación.

En lo referente a nuestros intereses, hemos escogido aquellos factores donde más evidencia empírica se ha encontrado en la literatura, como es el caso de la experiencia personal ya sea en desarrollo, en el uso de herramientas, la experiencia profesional en general, entre otros. Si logramos comparar nuestros resultados con los obtenidos en otros estudios, en nuestra opinión, podremos contribuir de forma más efectiva con el cuerpo de conocimiento existente hasta la fecha en cuanto a la influencia de estas variables.

Al referirnos a las limitaciones del contexto de experimentación, en nuestro caso los experimentos serán ejecutados dentro de un salón de clases con sesiones experimentales de corta duración. Aún cuando los participantes serán profesionales y estudiantes, las sesiones experimentales no se realizarán en entornos normales de trabajo o con proyectos reales de desarrollo ya que no tenemos la posibilidad de configurar diseños experimentales donde sea necesario largos periodos de tiempo para su realización. Factores como la comunicación, el tamaño del equipo de desarrollo, el nivel de entrenamiento o capacitación previa, no podrían ser objeto de análisis objetivo en estas condiciones.

Por nuestra falta de conocimiento específico en determinados campos del conocimiento (como por ejemplo la psicología) y, en virtud de los limitados recursos disponibles para la ejecución de los experimentos, hemos descartado el análisis de factores donde no seamos capaces de aplicar adecuadamente las técnicas y herramientas idóneas para la obtención de resultados confiables.

Por lo expuesto, en este estudio hemos considerado el análisis de los siguientes factores personales:

- Edad,
- Nivel de educación,
- Experiencia profesional,
- Experiencia en programación,
- Experiencia en lenguaje java,
- Experiencia en el uso del framework JUnit,
- Uso previo de herramientas de pruebas,
- Uso previo de la técnica de TDD,
- Experiencia en la técnica TDD,

- Entrenamiento previo en el desarrollo de pruebas unitarias,
- Conocimiento del entorno de desarrollo Eclipse y función actual en la organización.

Capítulo 4

METODOLOGÍA DE INVESTIGACIÓN

Si no conozco una cosa, la investigaré

Louis Pasteur

En el presente capítulo se presenta el marco metodológico que ha guiado el desarrollo de esta tesis, el cual consiste en la ejecución de una serie de replicaciones experimentales correspondientes a una familia de experimentos. Dichas replicaciones tienen como objetivo responder las preguntas de investigación planteadas en el Capítulo 3.

Hemos realizado siete de experimentos tomando como guía a la familia de experimentos propuesta por N. Juristo y su equipo de la Universidad Politécnica de Madrid (UPM), durante el proyecto FiDiPro EISEL [32]. Cinco de los siete experimentos fueron realizados en la academia, y los dos restantes en la industria. Cada experimento debió introducir distintos cambios de acuerdo al contexto donde se llevó a cabo.

Para una mejor comprensión de la metodología de investigación aplicada, en primer lugar se describe el experimento base abreviado como ESPE2015, de acuerdo a las guías propuestas por Jedlitschka & Pfahl (2005) [124]. A paso seguido se describen cada una de las replicaciones realizadas desde la perspectiva de sus diferencias a nivel de diseño y ejecución. Estas diferencias han sido descritas siguiendo las guías propuestas por Carver et al. [125] para reportar replicaciones.

Las replicaciones realizadas son del tipo literales nativas tanto internas como externas (de acuerdo a [126]) con ciertas diferencias en su contexto o metodología. Las replicaciones literales nativas pretenden mantener las características lo más cercanas al experimento original, esto es, la replicación ha sido realizada por los mismos investigadores, utilizando los mismos protocolos, la misma operacionalización y diferentes muestras de la misma po-

blación. Son internas cuando se realiza en el mismo lugar y externas cuando se la realiza en sitios distintos).

En este capítulo se mencionan únicamente las diferencias referentes a la manera en que se planteó la realización del experimento desde la perspectiva metodológica, dejando para el capítulo 5 otras consideraciones como el contexto y demás razones que motivaron las diferencias de una replicación a otra.

4.1. Descripción del experimento base ESPE2015

La falta de conocimiento acerca de la efectividad de TDD ha motivado la realización de estudios que repliquen experimentos anteriores o ensayen TDD bajo nuevas condiciones. Con este enfoque, un equipo de investigadores liderado por Natalia Juristo de la Universidad Politécnica de Madrid en el marco del proyecto de investigación [32], iniciaron una línea de investigación sobre el impacto de TDD en el desarrollo de software.

Para nuestro estudio, nos hemos inspirado en la familia de experimentos definido por este equipo de investigación. El objetivo principal de este proyecto fue estudiar la efectividad del desarrollo basado en pruebas en comparación con la técnica de desarrollo incremental en función de dos variables respuesta: la calidad externa del código y la productividad de los desarrolladores. Los resultados iniciales de esta investigación [127], no arrojaron efectos significativos al comparar los dos técnicas, tanto para la variable respuesta Calidad como para la Productividad, aunque en esta última se aprecia un p-valor bajo (p-valor = 0.116). Por otra parte, la tarea experimental sí produjo resultados significativos para las dos variables (p-valor <0.000 en los dos casos).

A partir del análisis de los resultados que fueron obtenidos por el equipo de investigación de N. Juristo, nos planteamos la realización de una serie de experimentos, considerando principalmente las características personales de los desarrolladores como pueden ser su edad, su experiencia, el conocimiento previo de las técnicas, el nivel de educación, entre otros factores que no fueron específicamente analizados en estos experimentos, pero que consideramos pueden influir significativamente en las variables Calidad y Productividad que fueron estudiadas.

En esta sección se reporta el experimento base ESPE2015 que es el punto de partida a la serie de experimentos realizados para mejorar el entendimiento acerca de la efectividad de TDD.

4.1.1. Factores y sus niveles

4.1.1.1. Slicing

La especificación de requisitos basadas en historias de usuario fueron propuestas como una de las primeras técnicas de programación extrema[5]. A menudo estas historias de usuario pueden ser muy grandes y deben ser descompuestas para que sean más fáciles de estimar y para que puedan realizarse en el menor tiempo posible (típicamente entre dos semanas y dos meses, como se expone en los principios del desarrollo ágil[128]).

Para conseguir esta descomposición, las historias de usuario pueden ser divididas como si fuesen rodajas de un pastel, donde cada parte debe contener una funcionalidad coherente y demostrable (un sub-producto) que incluye el desarrollo tanto de la interfaz de usuario, el almacenamiento o persistencia de datos y otras capas definidas en la arquitectura. Esta estrategia se la define como slicing vertical[129].

En la práctica, la técnica de Slicing sigue una estrategia de descomposición del problema en una serie de sub-problemas (llamadas slicies), de tal suerte que la yuxtaposición de los mismos componen la funcionalidad total requerida y por tanto el código de producción que debe desarrollarse.

4.1.1.2. Estrategia de programación

La **aproximación al desarrollo**, tiene dos niveles o tratamientos: TDD y ITLD Para TDD, se utilizó la siguiente estrategia de codificación:

1. Descomponer el proyecto en pequeñas y manejables sub-tareas para ser implementadas
2. Escribir la mínima prueba necesaria asociada a cada tarea antes de escribir el código de producción.
3. Obtener una rápida retroalimentación una vez ejecutadas las pruebas, antes de decidir si la tarea fue completamente implementada como se esperaba.
4. Refactorizar o cambiar de forma incremental el código de producción, con el objetivo de mejorar el diseño del sistema y la estructura del código, sin introducir cambios en la funcionalidad.

La estrategia ITLD sigue en general los mismos pasos que TDD, y la diferencia radica en el orden de aplicación de las pruebas y de la codificación. En ITLD, el código de producción se escribe antes que los test pero por pasos o incrementos. A diferencia de la técnica conocida como Test Last Development (TLD) en donde las pruebas son realizadas una vez que se completa todo el código de producción necesario para cumplir una determinada funcionalidad.

Se ha utilizado ITLD en lugar de TLD como nivel de control, ya que por lo general los programadores luego de haber realizado el código de producción, hacen sólo unos pocos casos de prueba. Esto conllevaría a comparar TDD con una estrategia TLD donde casi no existan pruebas. Con ITLD se intenta forzar de cierta manera a la realización de pruebas de manera incremental para cada *slice*, haciéndola comparable con TDD de una forma más natural.

4.1.1.3. Nivel de especificación

El **nivel de especificación de la tarea** tiene dos niveles o tratamientos: Sliced y NoSliced.

Para operacionalizar el factor Sliced, la especificación de la tarea a realizar por el sujeto ha sido descompuesta en una serie de *slices*. Cada paso corresponde a un ciclo del proceso de TDD. Para cada paso, se proporciona una especificación textual del comportamiento del código, y un ejemplo que puede usarse como caso de prueba.

La versión *NoSliced*, al contrario, es una especificación que contiene simplemente una descripción textual de las funciones a codificar.

Como ejemplo, vamos a suponer que la tarea a realizar es una calculadora. En la versión *Sliced*, la especificación se entregaría como una serie de pequeñas tareas a realizar, como sumar y un ejemplo de caso de prueba, luego restar y otro ejemplo de caso de prueba, y así sucesivamente para cada operación que se requiere sea implementada. En la versión *NoSliced*, se entregaría una descripción completa de todas las operaciones que deben ser codificadas para la calculadora. Las tareas experimentales utilizadas en este estudio se detallan en la sección 4.1.5.

4.1.2. Variables respuesta y métricas

Los estudios que analizan la efectividad de TDD han reportado sus resultados utilizando diferentes métricas tanto para evaluar la calidad del código desarrollado como la productividad de los programadores. Para evaluar la productividad, algunos estudios como los de Huang [77], Janzen y Saiedian [95] utilizan como métrica el número de líneas de código fuente desarrolladas por mes (LOC/person*month). Sin embargo, el uso de Líneas de Código (LOC) puede ocultar el esfuerzo realizado por el programador en la refactorización de código durante el proceso de desarrollo basado en TDD [59].

Para evitar este efecto, otros estudios se han basado en la medición de historias de usuario o test de aceptación pasados, ya que toman en cuenta la funcionalidad y la calidad de los productos de software desarrollados. Entre los estudios que utilizan los test de aceptación o historias de usuario, podemos mencionar el de Muller [9] que puede considerarse como uno de los primeros trabajos reportados en TDD que analiza la el entendimiento del

programa en base al número de historias de usuario pasadas y la reutilización adecuada de métodos existentes.

Hemos definido a la Calidad externa del código desarrollado en función de las historias de usuario pasadas de la siguiente manera:

1. El Número de historias de usuario que fueron abordadas por el programador. Se representa como TUS (Tacklet User Stories) y contabiliza si por lo menos una aserción dentro de un conjunto de aserciones, pertenecientes a una historia de usuario, ha sido realizada satisfactoriamente. Si al menos una aserción tiene éxito, indica que el programador ha intentado abordar (*"tackle"*) la historia. Esta variable se define como:

$$\#TUS = \sum_{i=1}^{\#us} \#Assert_i(Pass) \geq 0 \mapsto True \tag{4.1}$$

   Donde #Tus es el total de historias de usuario que componen la tarea experimental. Esta variable no es usada para evaluar los experimentos, sino para calcular las variables respuesta Calidad y Productividad, como se indica a continuación:

2. Calidad externa: representa la conformidad de los requerimientos frente al código desarrollado por los sujetos. Dicha conformidad se establece mediante test de aceptación. Se define como:

$$QLTY = \frac{\sum_{i=1}^{\#tus} QLTY_i}{\#TUS} \tag{4.2}$$

   Donde $QLTY_i$ es la calidad de la i-ésima historia de usuario abordada (TUS), y se encuentra definida como:

$$QLTY_i = \frac{\#Assert_i(Pass)}{\#Assert_i(All)} \tag{4.3}$$

   En posteriores estudios basados en test de aceptación como los de Pancur y Ciglaric [89]; George [7]; Madeyski [59] se define a la Calidad externa como el número o el porcentaje de test pasados satisfactoriamente.

   A diferencia de algunos estudios antes citados en donde se considera a la Productividad como el número de historias de usuario implementadas por hora, en nuestro caso no consideramos el tiempo de desarrollo de forma directa. Como habíamos señalado en el capítulo anterior, debido al contexto de realización de los experimentos (esto es, en un ámbito académico), las tareas experimentales se realizan en un tiempo acotado y más corto del necesario para que un desarrollador promedio complete todos los requisitos. En virtud de que los programadores generalmente usan todo el tiempo disponible para la realización de la

tarea experimental, no tiene mayor sentido utilizar como variable al tiempo de desarrollo como medida para calcular la Productividad.

Por otro lado, nuestro objetivo es reproducir de la manera más cercana posible el experimento realizado por los investigadores de la UPM. Por tal motivo hemos utilizado las mismas variables respuesta y métricas definidas en estos estudios y que a su vez coinciden con los de Erdogmus et al.[45] y Fucci et al.[14]. Por tanto, la productividad de los desarrolladores se ha definido como:

3. Productividad: representa la cantidad de trabajo realizado por los sujetos de manera satisfactoria, y se define como el número de aserciones pasadas frente al número total de aserciones:

$$PROD = \frac{\#Assert(Pass)}{\#Assert(All)} \tag{4.4}$$

4.1.3. Hipótesis Experimentales

El experimento base posee tres hipótesis experimentales para cada variable respuesta. La primera hace referencia a la influencia de ITLD y TDD en la calidad (QLTY) y productividad (PROD) del trabajo realizado por los programadores:

$$H_{10}\text{: } \mu(QLTY)_{ITLD} = \mu(QLTY)_{TDD}$$
$$H_{11}\text{: } \mu(QLTY)_{ITLD} <> \mu(QLTY)_{TDD}$$

$$H_{20}\text{: } \mu(PROD)_{ITLD} = \mu(PROD)_{TDD}$$
$$H_{21}\text{: } \mu(PROD)_{ITLD} <> \mu(PROD)_{TDD}$$

La segunda hipótesis hace referencia a Slicing y la estrategia por slicing.

4.1.4. Covariables

Esta tesis pretende determinar la influencia de las características personales de los programadores. Por este motivo, estudiamos los efectos de diferentes covariables, tales como la edad de los sujetos, la experiencia en desarrollo y el conocimiento previo de TDD, entre otros aspectos. Dichas covariables en conjunto con los factores definidas en la sección 4.1.2, permitirán descubrir si alguna característica personal influye en la efectividad de los programadores cuando usan TDD o ITLD. Las covariables utilizadas se muestran en la siguiente tabla:

Tabla 4.1: Covariables

Covariable	Descripción	Métrica
Edad	Indica la edad del sujeto participante	Medida mediante un valor numérico: Tipo entero positivo
Nivel de Educación	Determina el nivel de educación del sujeto, el mismo que puede ser estudiante de pre-grado, estudiante de post-grado u otro.	Medida mediante escala nominal: 1 Bachiller en Ciencias de la Computación 2 Master 3 Otro
Experiencia en programación	Indica la experiencia del sujeto en programación	Medida mediante escala ordinal de likert: 1 Sin experiencia (<2 años) 2 Novato (2 - 5 años) 3 Intermedio (6 - 10 años) 4 Experto (>10 años)
Uso de herramientas de pruebas	Indica si el sujeto ha utilizado o no herramientas automáticas de pruebas	Medida mediante escala nominal: 1 Si 2 No
Experiencia en lenguaje Java	Determina si el sujeto tiene experiencia con el uso del lenguaje Java	Medida mediante escala escala ordinal de likert: 1 Sin experiencia (<2 años) 2 Novato (2-5 años) 3 Intermedio (6-10 años) 4 Experto (>10 años)
Experiencia en JUnit	Indica si el sujeto tiene experiencia previa utilizando el framework JUnit	Medida mediante escala escala ordinal de likert: 1 Sin experiencia (10 años) 2 Novato (2-5 años) 3 Intermedio (6-10 años) 4 Experto (>10 años)
Uso de la técnica TDD	Permite conocer si el sujeto ha tenido conocimiento previo en la técnica TDD	Medida mediante escala nominal: 1 Si 2 No
Experiencia en TDD	Si el sujeto tiene experiencia en la técnica de TDD, este indicador permite determinar los años de experiencia en su uso	Medida mediante escala escala ordinal de likert: 1 Sin experiencia (10 años) 2 Novato (2-5 años) 3 Intermedio (6-10 años) 4 Experto (>10 años)
Entrenamiento previo en desarrollo de pruebas unitarias	Determina si el sujeto ha recibido entrenamiento previo en el desarrollo de pruebas unitarias	Medida mediante escala nominal: 1 Si 2 No
Conocimiento del entorno Eclipse	Determina si el sujeto tiene conocimiento previo del IDE de desarrollo Eclipse	Medida mediante escala nominal: 1 Si 2 No
Función actual en la organización	Indica cuál es la función que se encontraba desempeñando el sujeto dentro de la Organización	Medida mediante escala nominal: 1 Manager 2 Developer 3 Analyst 4 Other

Los valores de estas covariables fueron obtenidos mediante la aplicación de un cuestionario al inicio de cada experimento. Sin embargo, dependiendo de las condiciones de cada experimento, algunas de ellas fueron descartadas. Por ejemplo, para la mayor parte de experimentos realizados en la academia, incluyendo el experimento base, la edad de los participantes no era un factor diferencial. En cambio, los participantes si se diferenciaban en su nivel de educación.

Para cada una de las covariables especificadas en la Tabla 4.1 existe una hipótesis post-hoc asociada, sin embargo no se describen todas ellas de manera explícita. Por ejemplo, si consideramos la edad, las hipótesis post-hoc son:

1. Ha.0 No hay ninguna relación entre edad del sujeto participante y la efectividad en la aplicación de ITLD, TDD
2. Ha.1 Existe una relación entre edad del sujeto participante y la efectividad en la aplicación de ITLD, TDD

Las hipótesis alternativas son de dos colas (2-tailed) en todos los casos. Además, las hipóstesis serán analizadas separadamente para cada uno de los

experimentos reportados en la sección 5. Como podrá comprobar el lector, haremos incapié en aquellas donde se observen resultados significativos o que merezcan ser analizadas con detenimiento. Es importante indicar que las hipótesis post-hoc, al ser formuladas sobre covariables, no proporcionan información sobre relaciones causales; esto sólo es posible cuando las hipótesis se formulan sobre los factores experimentales. Las hipótesis post-hoc sugieren la existencia de una relación, pero su carácter correlacional o causal debe establecerse en futuras investigaciones.

Durante la realización de los experimentos, adicionalmente a los cuestionarios demográficos, se aplicó otro cuestionario luego de cada sesión experimental orientado a recabar el nivel de dificultad percibida por los participantes respecto a la estrategia de programación y la tarea experimental. Con ello intentamos entender si existió alguna influencia de los factores instrumentales.

4.1.5. Tareas Experimentales.

En el diseño experimental se incluyó a la tarea como una variable de bloqueo. La tarea fue presentada con un único nivel de definición en el experimento propuesto por N. Juristo y su equipo, pero en posteriores experimentos se ensayaron los niveles definidos como *Sliced* y *NoSliced*. En nuestro estudio, decidimos incluir este factor tanto en el experimento base como en las replicaciones. Las descripciones de las tareas fueron presentadas a los participantes en idioma inglés, aunque al momento de la ejecución tanto del experimento base como de las posteriores replicaciones, fuimos notando que en algunos participantes esto presentaba cierta dificultad para el entendimiento del problema. Para evitar este efecto pernicioso, se optó por realizar una explicación general de las tareas o aclarar dudas para cada caso. A continuación se describen las tareas con sus correspondientes niveles:

MarsRover API (MR). Este es un ejercicio de programación que tiene por objetivo el desarrollo de una serie de métodos públicos o API (Aplication Program Interface), que simula el movimiento de un vehículo hacia distintos puntos con diferentes orientaciones (Norte, Sur, Este, Oeste), dentro de un planeta representado por un plano de coordenadas. No se requiere la implementación de una interfaz de usuario, y se espera que el programa retorne la ubicación del vehículo como una cadena de caracteres donde se indique las coordenadas (x, y) en el plano y su orientación, además de la ubicación de posibles obstáculos encontrados.

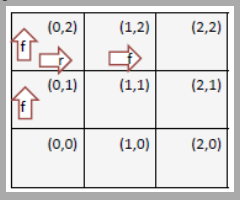
Se utilizaron dos descripciones para la tarea MarsRover API. La primera consistió en una especificación conteniendo cuatro requerimientos y dos ejemplos de test de aceptación sencillos (NoSliced). En la segunda descripción, los cuatro requerimientos fueron divididos en diez historias de usuario mucho más detalladas y adicionalmente contenían un test de aceptación (Sliced).

A continuación se presenta la especificación de la tarea MarsRover API, versión *NoSliced* entregado a los sujetos en idioma inglés:

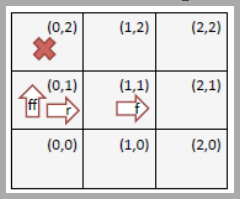
MarsRover API (MR) NoSliced

The API manages a rover that moves on a planet (squared grid) of arbitrary size (x,y). The rover starts the movement at position (0,0). The direction of the movement can be N (north), S (south), E (east) and W (west). The rover starts facing North.
The rover receives one string of commands: l (left), r (right), f (forward) and b (backward). l and r change the rover's direction counter- and clockwise, respectively, but not the position. f and b move the rover 1 position on the grid towards the direction it is facing or away from it, respectively. The facing does not change. When the rover moves over the edges of the planet, it spawns on the opposite side.
The planet (grid) may contain obstacles. Obstacles are defined as a list of coordinates "(obs1X,obs1Y)(obs2X,obs2Y)..." When the rover finds an obstacle during a tour, it skips the current command (i.e.: does not move to the cell where the obstacle is located) and continue executing the remaining commands.

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



Example of a rover's tour on a 3x3 planet in response to the command "ffrf". The starting position is (0,0) facing N. After the first (forward) command, the rover moves to position (0,1) facing north. Subsequent commands keep the rover moving. The expected output is (1,2,E). With two more f's, the rover would spawn over the right edge to the final position (0,2,E).



Example of a rover's tour on a 3x3 planet in response to the command

"ffrf", with one obstacle in position (0,2). After the first (forward) command, the rover moves to position (0,1) facing north. The 2nd f command does not change the rover's position, because there is an obstacle in (0,2). This second f command is thus skipped. The expected output is (1,1,E)(0,2).

La especificación de la tarea MarsRover API, versión *Sliced* es la siguiente:

MarsRover API (MR) Sliced

Develop an API that moves a rover around a planet. The planet is represented as grid with x and y coordinates.
The rover has also a direction that it is facing. The direction can be north (N), south (S), west (W) or east (E).
The input received by the rover is a string representing the commands it needs to execute.

1. The planet
The planet where the rover moves is represented as a squared grid, with size (x, y).
Requirement: Define a planet of size (x, y). Example: (100,100) creates a planet of size 100x100.

2. Landing
When the rover lands on the planet, it starts its journey at the start of the grid and faces north.
Requirement: When the rover lands on the planet its position shall be (0,0) facing north.
Example: An empty command (i.e.: "") to the rover results in returning its landing status (0,0,N).

3. Turning
The rover turns right or left. It remains on the same grid's cell. Its orientation change accordingly.
Requirement: Compute the position of the rover after turning left (command "l") or right (command "r").
Example: A rover in position (0,0,N) is in position (0,0,E) after executing command "r".
A rover in position (0,0,N) is in position (0,0,W) after executing command "l".

4. Moving
The rover moves forward or backward one grid's cell with reference to its facing. The rover direction does not change.
Requirement: Compute the position of the rover after moving forward (command "f")
or backward (command "b") one cell on the grid.
Example: A rover in position (7,6,N) arrives at (7,7,N) after executing a "f" command.
A rover in position (5,8,E) arrives at (4,8,E) after executing a "bçommand.

5. Moving and turning combined
The rover shall be able to execute arbitrary sequences of "f", "b", "l" and "r" commands.
Requirement: Compute the position of the rover after executing commands in series.
Example: A rover in position (0,0,N) arrives at position (2,2,E) after executing "ffrff".

6. Wrapping
Since the planet is a sphere the rover wraps at the opposite edge once it moves over it.
Requirement: Compute the position of the rover moving over the edges. The rover shall spawn on the opposite side.
Example: A rover on a planet of size 100x100, which moves backward (command "b")
after landing (remember that landing takes place always in position (0,0,N)), arrives at position (0,99,N).

7. Placing obstacles
Obstacles can be placed on specific cells of the grid.
Requirement: Define the obstacles as a string (x1,y1)(x2,y2)... Place the obstacles on the grid.
Example: "(1,1)(4,5)" defines two obstacles, one in position (1,1) and another in position (4,5).
Notice that the planet grid should be greater or equal than 6x6.

8. Locating a single obstacle
The rover might encounter (i.e.: tries to move onto) an obstacle. When it does it should report the obstacle and
continue executing the remaining commands.
Requirement: Compute the position of a rover encountering an obstacle, and report the obstacle.

The same obstacle should be reported only once.
Example: A rover just landed (position (0,0,N)). There is one obstacle on coordinates (2,2) of the planet.
The rover executes "ffrfff" and reports (1,2,E)(2,2).
Notice that the same obstacle is encountered twice but reported only once.

9. Locating multiple obstacles
The rover might encounter multiple obstacles. When it does, it should report all of them once,
and in the order they have been encountered.
Requirement: Compute the position of the rover encountering obstacles,and report the obstacles encountered
in the order they are encountered. The same obstacle shall be reported only once.
Example: A rover just landed (position (0,0,N)). There are two obstacles at coordinates (2,2) and (2,1) on the planet.
The rover executes "ffrfffrflf" and reports (1,1,E)(2,2)(2,1).
Notice that the first obstacle is encountered twice but reported only once.

10. A tour around the planet
The rover goes on a tour around the planet encountering several obstacles, and wrapping in both axes.
Requirement: Compute the position of a rover that executes a series of commands that result in moving
along both axes in both directions, encountering several obstacles and wrapping from both edges of the planet.
Example: The rover lands on a 6x6 planet with obstacles at (2,2), (0,5) and (5,0).
It executes the command "ffrfffrbbblllfrfrbbl" and returns (0,0,N)(2,2)(0,5)(5,0)
Congratulations, you are done!

Además de la especificación de la tarea, se entregó un template escrito en Java con el objeto de que los sujetos no pierdan tiempo en configurar la estructura básica del programa y facilitarles la realización del proceso de ejecución de pruebas de aceptación. Para MR, el template contenía 13 líneas de código (sin contar espacios en blanco ni comentarios) con 5 funciones públicas. Se incluyó además un test que genera el framework Junit por defecto con 9 líneas de código. Para cada función se incluyeron comentarios describiendo los parámetros de entrada y de salida esperada.

Robert Martin's Bowling Score Keeper (BSK). Esta tarea tiene por objetivo calcular el marcador de un único juego de bolos. Igual que MR,

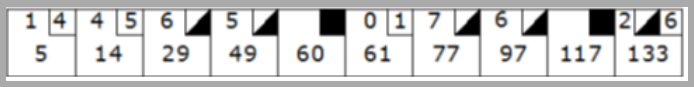
esta tarea consiste en la implementación de un algoritmo de programación, sin que sea necesario el desarrollo de una interface de usuario. BSK también fue descrito con dos niveles de especificación (o de abstracción) de manera similar a MR. El primer nivel de especificación contiene cuatro requerimientos (similar a MR - NoSliced) y en el segundo nivel se especifican se contemplan trece historias de usuario (3 más que MR - Sliced), con ejemplos de test de aceptación. Las tareas en versión *Sliced* y *NoSliced* de BSK se presentan a continuación en idioma inglés tal como se les entregó a los sujetos:

Bowling Score Keeper (BSK) NoSliced

The game consists of 10 frames as shown above. In each frame the player has two opportunities to knock down 10 pins. The score for the frame is the total number of pins knocked down, plus bonuses for strikes and spares.
A spare is when the player knocks down all 10 pins in two tries. The bonus for that frame is the number of pins knocked down by the next roll. So in frame 3
above, the score is 10 (the total number knocked down) plus a bonus of 5 (the
number of pins knocked down on the next roll.)

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

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



Bowling Score Keeper (BSK) Sliced

The objective is to develop an application that can calculate the score of a single bowling game using TDD.
There is no graphical user interface.
You work only with objects and JUnit test cases in this assignment. You won't need a main method.
The application's requirements are divided into a set of user stories, which

serve as your to-do list.
You should be able to incrementally develop a complete solution without an upfront
comprehension of all the game's rules.
For this exercise, don't read ahead, and handle the requirements one at a time in the order provided.
Solve the problem using TDD, starting with the first story's requirement.
Remember to always lead with a test case, taking hints from the examples provided.
Only when a story is done, move on to the next one.
A story is done when you are confident your program correctly implements the functionality stipulated by the story's requirement.
This implies all of your test cases for that story and all of the test cases for the previous stories pass.
You may need to tweak your solution as you progress towards more advanced stories.

1. Frame
Each turn of a bowling game is called a frame. 10 pins are arranged in each frame.
The goal of the player is to knock down as many pins as possible in each frame.
The player has two chances, or throws, to do so. The value of a throw is given by the number of pins knocked down in that throw.
Story: As the scorekeeper, I want to be able to record a frame as composed of two throws.
The first and second throws should be distinguishable.
Example: [2, 4] is a frame with two throws, in which two pins were knocked down in the first throw and four pins were knocked down in the second.

2. Frame Score
An ordinary frame's score is the sum of its throws.
Story: As the scorekeeper, I want to be able to compute the score of an ordinary frame after a player has rolled both throws.
Examples: The score of the frame [2, 6] is 8. The score of the frame [0, 9] is 9.

3. Game
A single game consists of 10 frames.
Story: As the scorekeeper, I want to define a game as a sequence of 10 frames.
Example: The sequence of frames [1, 5] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] represents a game.

You may reuse this game from now on to represent and test different scenarios, modifying only a few frames each time.

4. Partial Game
When the player rolls a throw, the throw is automatically recorded in the correct frame.
Story: As the scorekeeper, when the a player rolls throws, I want the game to keep track of the frames and figure out in
which frame to place the next throw depending on the past throws. You think this is easy.
Maybe for now. We'll see.
Example: If the game currently consists of the frames [1, 5] [3, 6] [7, 2] [3, ?] and the
player rolls a throw with a value of 4, the game becomes [1, 5] [3, 6] [7, 2] [3, 4]. Another roll with a value of 5 transforms the game to [1, 5] [3, 6] [7, 2] [3, 4][5, ?].

5. Game Score
The score of a bowling game is the sum of the individual scores of its frames.
Story: As the scorekeeper, I want to know a player's game?s current score at all times.
Example: The score of the game [1, 5] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] is 81.
Partial scores are possible for an incomplete game if the frame scores are known up to the last complete frame.
The score of the game [1, 5] [3, 6] [7, ?] is 15. The frame [7, ?] is not yet complete.

Strike
A frame is called a strike if all 10 pins are knocked down in the first throw. In this case, there is no second throw.
A strike frame can be written as [10, 0]. The score of a strike equals 10
plus the sum of the next two throws of the subsequent frame.
Story: As the scorekeeper, I want to be able to recognize a strike frame, compute its score after the next frame has been completed, and compute the game's score.
Examples: Suppose [10, 0] and [3, 6] are consecutive frames.
Then the first frame is a strike and its score equals $10 + 3 + 6 = 19$.
The game [10, 0] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] has a score of 94. The partial game [10, 0] [3, 6] has a score of 28.

7. Spare
A frame is called a spare when all 10 pins are knocked down in two throws. The score of a spare frame is 10 plus the value of the first throw from the subsequent frame.
Story: As the scorekeeper, I want to be able to recognize a spare frame, compute the score of a game containing a spare frame after the first throw of the next frame has been rolled, and compute the game's score.
Examples: [1, 9], [4, 6], [7, 3] are all spares. If you have two frames [1, 9] and [3, 6] in a row, the spare frame?s score is 10 + 3 = 13. The game [1, 9] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6]
has a score of 88. The partial game [1, 9] [3, 6] has a score of 22.

8. Strike and Spare
A strike can be followed by a spare. The strike?s score is not affected when this happens.
Story: As the scorekeeper, I want to make sure that the score of a strike is computed right when it's followed by a spare.
Examples: In the sequence [10, 0] [4, 6] [7, 2], a strike is followed by a spare. In this case, the score of the strike is 10 + 4 + 6 = 20, and the score of the spare is 4 + 6 + 7 = 17.
The game [10, 0] [4, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] has a score of 103.

9. Multiple Strikes
Two strikes in a row are possible. You must take care when this happens for the computation of the first strike's score requires the values of throws from two subsequent frames.
Story: As the scorekeeper, I want to make sure that I can record two consecutive strikes correctly in the game,
and correctly compute the score of the first strike after the next two throws have been rolled.
Examples: In the sequence [10, 0] [10, 0] [7, 2], the score of the first strike is 10 + 10 + 7 = 27.
The score of the second strike is 10 + 7 + 2 = 19.
The game [10, 0] [10, 0] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] has a score of 112.
The score of the partial game [10, 0] [10, 0] [7, ?] is 27 (we can?t yet compute the scores of the last two frames).

10. Multiple Spares
Two spares in a row are possible. The first spare?s score is not affected when this happens.

Story: As the scorekeeper, I want to be able to compute the score of a game with two spares in a row,
and the scores of the first spare after the next spare has been completed.
Example: The game [8, 2] [5, 5] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 6] has a score of 98.

Spare as the Last Frame
When a game's last frame is a spare, the player will be given a bonus throw. However, this bonus
throw does not belong to a regular frame. It is only used to calculate the score of the last spare.
Story: As the scorekeeper, I hate it when the last frame is a spare: let the game please figure
out that the next roll is a bonus throw and compute the score of the last frame and the
whole game based on the value of that bonus throw.
Example: The last frame in the game [1, 5] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 8] is a spare.
If the bonus throw is [7], the last frame has a score of 2 + 8 + 7 = 17. The game has a score of 90.

12. Strike as the Last Frame
When a game?s last frame is a strike, the player will be given two bonus throws. However,
these two bonus throws do not belong to a regular frame.
They are only used to calculate score of the last strike frame.
Story: As the scorekeeper, I hate it even more when the last frame of a game is a strike:
let the game please figure out that the next rolls are bonus throws and compute the score of the last frame
and the whole game based on the value of those bonus throws.
Example: The last frame in the game [1, 5] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [10, 0] is a strike.
If the bonus throws are [7, 2], the last frame?s score is 10 + 7 + 2 = 19. The game's score is 92.

13. Bonus is a Strike
Further bonus throws are not granted when a game?s last frame is a spare and the bonus throw is a strike.
Story: As the scorekeeper, I hate it most when the last frame is spare and the bonus throw is a strike:
please God, let the game figure this scenario out correctly.

Example: In the game [1, 5] [3, 6] [7, 2] [3, 6] [4, 4] [5, 3] [3, 3] [4, 5] [8, 1] [2, 8], the last frame is a spare.
If the bonus throw is [10], the game's score is 93.

14. Best Score
A perfect game consists of all strikes (a total of 12 of them including the bonus throws), and has a score of 300.
Story: As the scorekeeper, I love it when the game is just a sequence of strikes, including the bonus throws,
because I know that the player then deserves a perfect score of 300.
Example: A perfect game looks like [10, 0] [10, 0] [10, 0] [10, 0] [10, 0] [10, 0] [10, 0] [10, 0] [10, 0] [10, 0]
with bonus throws [10, 10]. Its score is 300.

15. Random Game
Story: As the scorekeeper, I want to make sure that the game
[6, 3] [7, 1] [8, 2] [7, 2] [10, 0] [6, 2] [7, 3] [10, 0] [8, 0] [7, 3] [10] has a score of 135.

Congratulations, you are done!

Se entregó además de la especificación de requerimientos, un template en lenguaje Java que consistió en este caso de dos clases: la primera con 17 líneas de código (sin comentarios, debido a su simplicidad) y con 4 funciones públicas a ser implementadas por los sujetos. La segunda clase consta de 34 líneas de código con 7 funciones públicas y dos constructores. En esta clase también se incluyeron comentarios que describían los parámetros y resultados esperados de cada función. Además se adicionó una clase de pruebas generada por defecto por la herramienta Junit con 9 líneas de código.

Spreadsheet (SS) consiste en el desarrollo de un programa similar a una hoja de cálculo tipo MS Excel. Una hoja contiene filas y columnas que almacenan números, cadenas de caracteres o fórmulas. Los números están restringidos a números enteros positivos o negativos. El objetivo es realizar operaciones de suma, resta multiplicación, división de enteros, resto de la división y concatenación, siguiendo las convenciones de MS Excel. (por ejemplo: -1+2", -B3*(4+1)"). Para disminuir la complejidad en cuanto a las reglas de precedencia de operadores, se requiere el uso de paréntesis para las operaciones de multiplicación, división y módulo. En otros casos se debe aplicar la evaluación de izquierda a derecha (por ejemplo: -1+2*3"debe devolver el valor de 9). El uso de espacios en la cadena de evaluación también está permitido. Adicionalmente, se debe realizar el control de errores como es el caso de la división por cero, referencias circulares, referencias a celdas

incorrectas, celdas vacías, etc.

La descripción de SS en idioma inglés en sus versiones *Sliced* y *NoSliced* se presentan a continuación:

Spreadsheet (SS) NoSliced

This exercise aims creating a basic spreadsheet. The goal is not to develop the GUI, but the code that implements
the data structures, performs the operations and returns the results back to the GUI. The spreadsheet contains cells, organized in rows and columns (similar to MS Excel).
The cells store numbers, strings (enclosed by simple quotes ", e.g.: 'This is a string'), or formulas (starting with the = sign).
In this version, numbers shall be restricted to (positive or negative) integers. The operations are: addition (+), subtraction (-), multiplication (*), integer division (/), module (%) and concatenation (&).
Formulas shall follow Excel conventions (e.g.: "=1+2", "=B3*(4+1)"). To avoid tedious parsing, precedence of */ % shall be explicitly specified using parentheses. Otherwise, evaluation proceeds strictly from left to right. For instance, "=1+2*3" is evaluated to 9 no matter the correct value is 7. Arbitrary spaces can be inserted between symbols (e.g.: "= 1 + 2*3" is valid)
Errors shall be caught during evaluation and an Error or Circular error message shall be returned.
Error is the general error message and shall be returned when operations yield wrong results
(e.g.: division by zero, references to incorrect cells, empty cells, etc.).
Circular shall be returned when formulas make circular references (causing the recursion never end).

Spreadsheet (SS) Sliced

This exercise aims creating a basic spreadsheet. The goal is not to develop the user interface, but the code that implements the data structures and performs the operations.
The spreadsheet follows Excels conventions.
The spreadsheet implements three operations: set(), get() and evaluate(). Set and get assign and return cells values.
Evaluate takes the cells content and returns the result (e.g.: a cell containing "=1+1" evaluates to 2).

1. Setup
There are two basic operations: set(), that assigns a value to a cell
get(), that retrieves the value of a cell (e.g.: for editing)
Requirement: Define the operations set() and get().
Example: if set(”A1”, ”1”) is run, a subsequent get(”A1”) gives: ”1”.

2. Number handling
The spreadsheet should handle correctly formatted integer numbers
Requirement: The spreadsheet shall be able to store and return integer numbers, both signed and unsigned.
When a cell containing an integer number is evaluated, the result of the evaluation shall be the number itself.
Example: if set(”A1”, ”-1”) is run, a subsequent call to evaluate(”A1”) returns the same number: ”-1”.

3. Wrongly formatted integers
The spreadsheet detects incorrectly formatted integers
Requirement: When a number does not follow the integer format (e.g.: contains a decimal point, special symbols,
characters, etc.), the evaluation shall return the error message ”Error”.
Example: if set(”A1”, ”5A”) is run, a subsequent call to evaluate(”A1”) returns ”Error”.

4. String handling
The spreadsheet should handle arbitrary strings
Requirement: The spreadsheet shall be able to store and return strings. All strings are entered between single quotes.
When a cell containing a string is evaluated, the result of the evaluation shall be the same string without quotes.
Removing the quotes eases displaying strings in the screen correctly, as well as concatenating with other strings (see below).
Example: if set(”A1”, ”a string”) is run, a subsequent call to evaluate(”A1”) returns the same string without quotes: ”a string”.

5. Unquoted strings
The spreadsheet detects incorrectly formatted strings
Requirement: When a string does not have heading or trailing quotes, the evaluation shall return the error message ”Error”.
Example: if set(”A1”, ”a string”) is run, a subsequent call to evaluate(”A1”) returns ”Error”.

6. Simple formulas

The spreadsheet evaluates simple formulas (without operators or cell references)
Requirement: When a cell contains a ”=” sign, followed by a string or integer number, the evaluation of that cell shall return the corresponding string or integer (see Req. 2, 4).
Example: if set(”A1”, ”=a string”) is run, a subsequent call to evaluate(”A1”) returns ”a string”.

7. Simple formulas with errors
The spreadsheet detects errors in simple formulas
Requirement: When a cell contains a -”sign, followed by a wrong string or integer number (see Req. 3, 5), the evaluation of that cell shall return the error message #Error.
Example: if set(”A1”, ”=a string”) is run, a subsequent call to evaluate(”A1”) returns ”Error”.

8. Cell references
The spreadsheet handles cell references in formulas
Requirement: When a formula contains a reference to a cell (following Excel convention, e.g.: A5), the evaluation shall be recursive, i.e.: the referenced cell is evaluated, and the result is returned by the formula.
Example: if A5 contains the integer ”5” and set(”A1”, ”=A5”) is run, a subsequent call to evaluate(”A1”) returns ”5”.

9.
Errors in cell references
The spreadsheet detects errors when cell references in formulas are used
Requirement: When the value contained in a cell referenced by a formula is incorrect, the evaluation shall return the error message ”Error”.
Example: if A5 contains the wrong integer ”5A” and set(”A1”, ”=A5”) is run, a subsequent call to evaluate(”A12”) returns ”Error”.

10. Circular references
The spreadsheet detects circular references in formulas
Requirement: When a formula contains circular references, the evaluation shall return the error message ”Circular.”
Example: if A5 contains the formula ”=A1” and set(”A1”, ”=A5”) is run, a subsequent call to evaluate(”A1”) returns ”Circular”.

11. Integer operations
The spreadsheet performs the integer addition, subtraction, multiplication, division, and module.

Requirement: When the operators + - * / % are used in formulas, the evaluation shall return the results of the addition, subtraction, multiplication, division, and module operations, respectively.
Usual precedence does not apply, i.e.: formulas are always evaluated from left to right.
Example: if set(”A1”, ”=1+1*2”) is run, a subsequent call to evaluate(”A1”) returns ”4”.

12. Errors in integer operations
The spreadsheet identifies errors in integer operations
Requirement: When an operation cannot be performed because (1) the operators are incorrect integer numbers or (2) there is a division by zero, the evaluation shall return the error message #Error.
Example: if set(”A1”, ”=1+1A”) is run, a subsequent call to evaluate(”A1”) returns ”Error”.

String operations
The spreadsheet performs the concatenation of strings
Requirement: When the operator & is used in formulas, the evaluation shall return the concatenation of the respective string operators.
Example: if set(”A1”, ”=a string”) is run, a subsequent call to evaluate(”A1”) returns ”astring”.

14. Errors in string operations
The spreadsheet identifies errors in string concatenation
Requirement: When a concatenation cannot be performed because the strings are wrongly formatted (see Req. 5) the evaluation shall return the error message .Error.”
Example: if set(”A1”, ”=astring”) is run, a subsequent call to evaluate(”A1”) returns ”Error”.

15. Parentheses
The spreadsheet handles parentheses in formulas
Requirement: When a formula contains parentheses, the internal part shall be evaluated first.
Example: if set(”A1”, ”=1+(1*2)”) is run, a subsequent call to evaluate(”A1”) returns ”3”.

16. Errors using parentheses
The spreadsheet detects unbalanced parentheses
Requirement: When a formula contains unbalanced parentheses, the

evaluation shall return the error message ”Error”.
Example: if set(”A1”, ”=1+(1*2”) is run, a subsequent call to evaluate(”A1”) returns ”Error”.

17. Spaces
Formulas can contain arbitrary spaces
Requirement: Unnecessary spaces in formulas are ignored.
Example: if set(”A1”, ”=1+ (1 * 2)”) is run, a subsequent call to evaluate(”A1”) returns ”3”.

Congratulations, you are done!

4.1.6. Complejidad relativa de las tareas experimentales.

Las funcionalidades de MR y BSK son diferentes; sin embargo su estructura y nivel de complejidad es homogéneo.

MR tiene una estructura de datos que consiste de una matriz de NxN celdas y donde cada celda puede almacenar un obstáculo. Los obstáculos no tienen ningún comportamiento asociado y se podrían representar como un tipo de dato simple, por ejemplo, un dato de tipo boleano que indique la presencia o ausencia del obstáculo en una coordenada determinada. Existen seis operaciones que fueron definidas en la clase MarsRover y entregadas como parte del template. Estas operaciones son:

- Matrix (Inicializa la matriz y permite la asignación de obstáculos en las celdas)
- Command parsing
- Forward moves
- Backward moves
- Left truns
- Right turns

De estas operaciones, las que presentan un mayor nivel de complejidad son fordward y backward, e implica la realización de tres condicionales en secuencia, siendo estas funciones donde se producen la mayor cantidad de faltas. Las otras operaciones conllevan un menor esfuerzo, aunque para la asignación de obstáculos a una celda se tiene que realizar una transformación de tipos de datos que puede requerir de algunos ciclos para su adecuada depuración.

De forma análoga, BSK también necesita de una estructura de datos, en este caso de un arreglo limitado a 10 elementos en lugar de una matriz de NxN en el caso de MR. Este menor nivel de complejidad es balanceado con la incorporación de variables adicionales para contabilizar los bonos que podrían obtenerse en el juego. Esta característica puede modelarse mediante la declaración de un objeto conteniendo tipos de datos enteros. BSK tiene seis operaciones principales:

- Add a frame
- Add a bonus throws
- Is a spare
- Is a strike
- Frame score
- Frame game score.

Una de las operaciones más complejas en la tarea BSK, es el cálculo del puntaje de un *frame*. Este cálculo depende si el lanzamiento contempla un bono (*spare o strike*), la posición del frame dentro del juego y en qué caso el siguiente frame es un strike. El cálculo del puntaje total se lo puede realizar de forma análoga a como se transforman los comandos en MR (realizando un recorrido por todo el arreglo). El resto de operaciones, en general tienen la misma complejidad que MR.

En cuanto a SS, este programa tiene una estructura que también contiene seis operaciones principales por implementar:

- Evaluate
- Unquote
- isCorrecltyQuoted
- isQuoted
- isNumber
- isFormula.

Siguiendo el mismo protocolo para la instrumentación de los experimentos, se entregó junto con la especificación del ejercicio, un template escrito en Java[130] conteniendo 41 líneas de código (sin espacios ni comentarios) con 8 funciones públicas. Se incluyó además un test que genera automáticamente el framework Junit conteniendo 9 líneas de código. Se incluyeron comentarios para describir los parámetros de entrada y de salida esperada

de aquellas funciones donde no era tan evidente entender el comportamiento esperado.

La operación más compleja para el caso de SS es la evaluación de expresiones (*Evalaute*) que debe ser ingresada como una cadena de caracteres. Esta evaluación puede resolverse mediante la implementación de una función recursiva en la que debe realizarse las conversiones necesarias para realizar las operaciones aritméticas y devolver su resultado en caso de no existir un error. Las operaciones restantes no implican mayor grado de complejidad. Por ejemplo, *isNumber* retorna verdadero si la cadena es un número. Esta característica puede implementarse utilizando una expresión regular en una sola línea de código.

SS puede considerarse distinto en complejidad a MR y BSK. Para el caso de SS, la estrategia de solución implica el diseño de un algoritmo recursivo que exige un alto grado de abstracción. Para MR y BSK, como ya se ha comentado, se debe diseñar un algoritmo para operar sobre una estructura de datos tipo matricial. Por esta razón, su nivel de complejidad puede considerarse relativamente menor que el de SS.

4.1.7. Diseño Experimental

En el experimento base, se analizaron los factores *Estrategia de programación* con los niveles ITLD, y TDD; y el factor *Nivel de definición de la tarea* con los niveles Sliced y NoSliced. Las tareas experimentales fueron los problemas MarsRover API (MR) y Bowling Scorekeeper (BSK).

Spredsheet (SS) fue utilizado en un único experimento ya que se intentó conocer la Calidad y Productividad obtenida por los participantes antes de recibir ningún entrenamiento en las técnicas ITLD y TDD. La introducción de este factor, demandaba un mayor tiempo para la ejecución de los experimentos, y dadas las restricciones temporales, se optó por no ensayar esta condición y por tanto no se descartó este programa como tarea experimental.

El diseño experimental utilizado en el experimento base fue de tipo *cross over (2x2)* between-subjects de medidas repetidas con dos factores, como se muestra en la tabla 4.2:

Tabla 4.2: Diseño Experimental

Grupos	**Sesión Experimental 1 ITL**	**Sesión Experimental 2 - TDD**
G1	BSK /Sliced	MR/No Sliced
G2	BSK/ No Sliced	MR/Sliced
G3	MR/Sliced	BSK/No Sliced
G4	MR/No Sliced	BSK/Slicing

4.1.8. Poder Estadístico

El poder estadístico es la probabilidad de que la hipótesis nula sea rechazada cuando la hipótesis alternativa es verdadera. Para realizar una prueba de hipótesis con garantías, es importante realizar previamente un análisis del poder estadístico ([131]). El análisis del poder estadístico permite calcular el tamaño mínimo de la muestra necesaria para poder llegar a conclusiones fiables acerca de los efectos observados.

Para las investigaciones en Ingeniería de Software, Ramos Román & Dolado Cosín[132] recomiendan utilizar un poder estadístico del 80 % (equivalente a un error tipo II del 20 %), con un nivel de significación (error tipo I) del 5 %. Nuestro análisis sigue estas recomendaciones. Ramos Román & Dolado Cosín[132] también recomiendan utilizar tamaños medios de efecto poblacional determinados de acuerdo a las definiciones de Cohen[131] (por ejemplo $d = 0{,}5$, $r = 0{,}3$, $f = 0{,}25$, etc.). En nuestro caso, hemos decidido que la diferencia mínima que queremos identificar es de 15 puntos para la Calidad y 25 puntos para la Productividad, lo que supone tamaños de efecto d medios.

Desde un punto de vista técnico, hemos implementado el análisis del poder estadístico mediante una simulación de Monte-Carlo del diseño experimental *cross-over* utilizado en la tesis, con 500 interacciones para cada tamaño muestral (de 6 a 120 sujetos por grupo)[1]. Las varianzas poblacionales las hemos tomado, a falta de mejor referencia, del experimento base ESPE2015.

En estas condiciones, el análisis del poder estadístico para detectar efectos de covariables (que es lo que más nos interesa) se muestra en la Fig. 4.1 para la Calidad, y en la Fig. 4.2 para la Productividad. Necesitaremos 39 sujetos por grupo (en total $39x2$ =78 sujetos) para la variable Calidad y 90

[1]Recomendamos al lector interesado la revisión del código R, ya que no comentamos aquí todos los detalles de la simulación.

por grupo (180 totales) para la Productividad. Este último valor es bastante elevado ya que, aunque el diseño es intra-sujetos, las covariables son por su propia naturaleza inter-sujetos, por lo que no se benefician de la reducción de varianza propiciada por el diseño.

Esto contrasta con el análisis del poder estadístico para detectar efectos de factores, que se muestran en las Figs. 4.3 y 4.4. Puede observarse que en este último caso los valores son mucho más razonables, ya que los factores son intra-sujetos. Más concretamente, se precisan 18 y 12 sujetos por grupo (36 y 12 sujetos totales) para Calidad y Productividad, respectivamente.

En retrospectiva, hemos realizado siete experimentos. Todos los experimentos tienen tamaños muestrales pequeños, con la excepción de ESPE2015, que tenía 43 sujetos totales, Por este motivo, además de los experimentos individuales, hemos decidido realizar un meta-análisis *Individual Patient Data* siguiendo a[133] para identificar los efectos de las covariables.

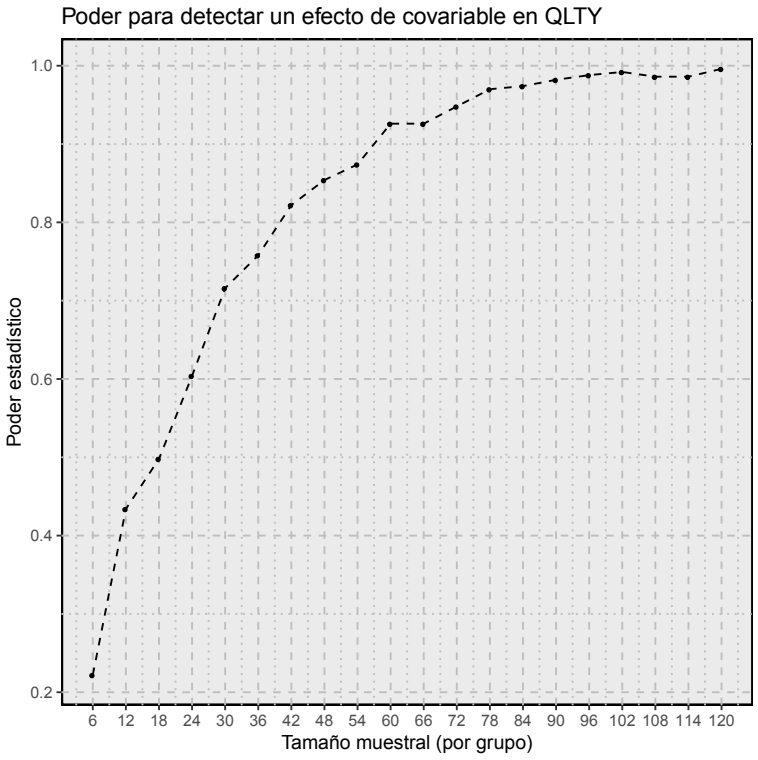


Figura 4.1: Poder estadístico de la variable respuesta Calidad

4.1.9. Selección de sujetos

Para la selección de sujetos experimentales se ha utilizado muestreo por conveniencia; esto es: los participantes fueron seleccionados entre miembros

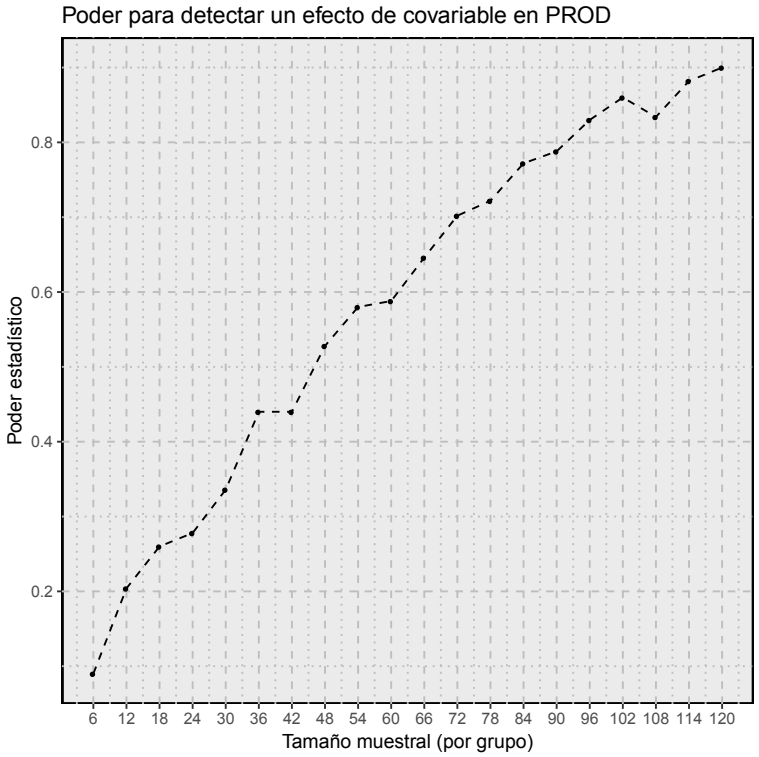


Figura 4.2: Poder estadístico de la variable respuesta Productividad

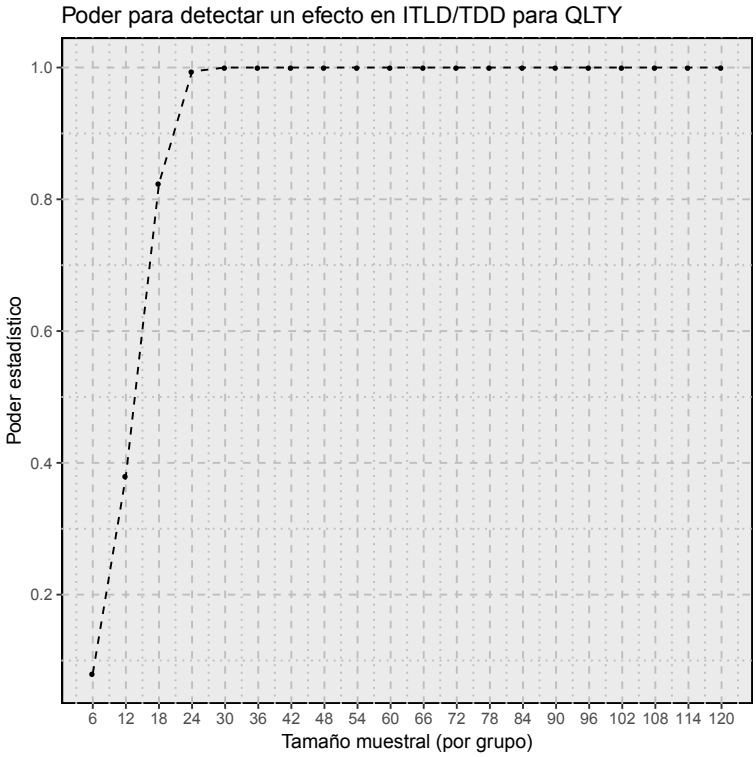


Figura 4.3: Poder estadístico de la variable respuesta Calidad

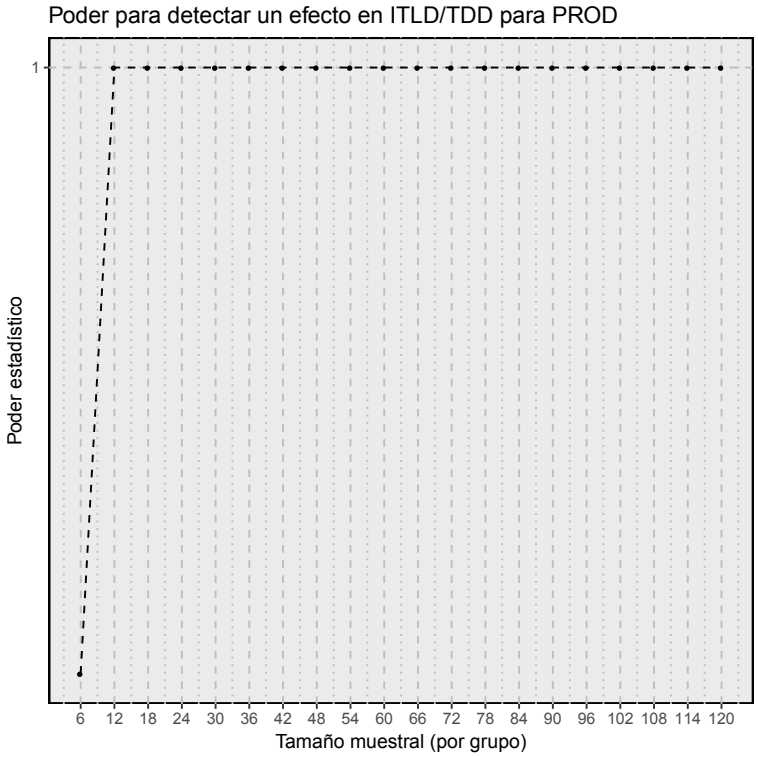


Figura 4.4: Poder estadístico de la variable respuesta Productividad

de distintas universidades, empresas o instituciones con las que el investigador tenía algún tipo de relación profesional. Dependiendo del caso, los sujetos experimentales fueron estudiantes de carrera en cursos avanzados, estudiantes de doctorado, profesionales con diversos niveles de experiencia. Para el caso del experimento base, los sujetos seleccionados fueron estudiantes universitarios que al momento se encontraban cursando el quinto y sexto grado de la Carrera de Ingeniería de Sistemas e Informática, en los cursos de Desarrollo de Software y Programación Avanzada.

4.1.10. Asignación de los Sujetos a Tratamientos

El estudio fue un experimento controlado. Por tanto, la asignación de sujetos a los tratamientos se realizó de manera aleatoria en distintos grupos de acuerdo al diseño experimental aplicado (que se muestra en la sección 4.1.7).

4.1.11. Protocolo Experimental

El protocolo experimental utilizado está compuesto por seis actividades realizadas en tres semanas con sesiones de dos días por semana como se indica en la tabla 4.3. A continuación se describen éstas actividades en orden cronólogico:

- Durante el primer día se realizó una sesión de entrenamiento en pruebas unitarias.
- El segundo día se realizó una sesión de entrenamiento de las técnicas de Slicing e ITLD.
- El tercer día se realizó la primera sesión experimental con la aplicación de la técnica ITLD.
- El cuarto y quinto día se realizaron sesiones de entrenamiento en la técnica de TDD.
- Finalmente, el sexto día se ejecutó la segunda sesión experimental con la aplicación de la técnica TDD.

Las sesiones de entrenamiento tuvieron una duración de dos horas, en tanto que, para la ejecución de las sesiones experimentales los estudiantes tardaron en promedio dos horas con treinta minutos.

Previo al inicio de la primera sesión de entrenamiento, se les pidió al los participantes que rellenaran el cuestionario demográfico en línea para obtener las covariables descritas en 4.1.4. Durante esta primera sesión de entrenamiento, los sujetos recibieron capacitación sobre la realización de pruebas

unitarias utilizando como lenguaje de programación Java y el framework de pruebas JUnit V4.

La capacitación realizada durante el primero y segundo día consistió en la aplicación de las técnicas de ITLD basadas en la descomposición de tareas (Slicing). Se utilizó una dinámica en la cual cada sujeto debía resolver un slice de cada ejercicio de entrenamiento delante de sus compañeros en turnos. Esta estrategia permitió captar la atención del grupo y evitar que se desconcentrasen durante la realización de las tareas solicitadas. El tercer día se ejecutó la primera sesión experimental y al final del mismo los participantes llenaron el cuestionario post-experimental.

Los días cuatro y cinco, se dedicaron a la capacitación en la aplicación de la técnica TDD y, de manera similar a la sesión de entrenamiento con ITLD, los participantes realizaron frente a sus compañeros cada una de los requerimientos identificadas en los slices. Finalmente se ejecutó la segunda sesión experimental con el uso de TDD, y al igual que en la primera sesión experimental, se solicitó llenar una encuesta al concluir la ejecución del experimento.

Es importante señalar que dadas las condiciones de organización y tiempo disponible de los alumnos, se estableció un tiempo de tres horas para la ejecución de las sesiones experimentales. Sin embargo, el tiempo promedio utilizado fue menor (dos horas con treinta minutos).

El entrenamiento y ejecución del experimento estuvo a cargo del investigador proponente de este estudio y se utilizaron los materiales y ejercicios que fueron facilitados por el Dr. Oscar Dieste, quien forma parte del equipo de investigación.

Tabla 4.3: Cronograma ESPE2015

Semana 1		**Semana 2**		**Semana 3**	
Día1 (2 horas)	**Día2 (2 horas)**	**Día3 (2,5 horas)**	**Día4 (2 horas)**	**Día5 (2 horas)**	**Día6 (2,5 horas)**
Entrenamiento en pruebas unitarias	Entrenamiento en las técnicas ITL/slicing	Sesión experimental,ITL	Entrenamiento en la técnica TDD Parte I	Entrenamiento en la técnica TDD Parte II	Sesión experimental,TDD

4.1.12. Instrumentación

Las tareas experimentales fueron realizadas utilizando el lenguaje de programación Java y JUnit como framework de pruebas. En todos los casos se entregó a los sujetos la estructura básica de código indicada en la sección 4.1.5, con el propósito de que los sujeto se enfoque principalmente en la resolución del problema y en la escritura de las pruebas necesarias. El código estaba alojado en un repositorio de Github habilitado para el efecto[130]. Como IDE de desarrollo se utilizó Eclipse con un plugin específicamente diseñado para facilitar la medición de las variables respuesta. Para ello, se introdujeron ciertos cambios dentro del framework JUnit con el objeto de obtener de forma automática las variables respuesta y facilitar el proceso de

medición. Este plugin genera un reporte en un archivo de texto conteniendo las métricas para todos los casos de prueba con los respectivos porcentajes de casos pasados, no pasados, errores, números de casos de prueba y finalmente la medida de las variables respuesta (QLTY y PROD).

4.1.13. Procedimiento de medición

Para determinar los valores cuantitativos de las variables respuesta QLTY y TUS, se desarrollaron diferentes casos de prueba (13 para BSK, y 11 para MR) estructurados como historias de usuario. Cada método de prueba contiene por lo menos una aserción aunque generalmente más. Para BSK se confeccionaron un total de 48 casos de prueba y 56 aserciones, en tanto que para MR se prepararon un total de 52 casos de prueba con 79 aserciones. MR contiene un 7,6 % de casos de prueba adicionales a BSK, y al contrario un 29,1 % de menos aserciones que BSK. Consideramos que estas diferencias no son relevantes a objetos de la medición.

La medición de las variables respuesta se lo realizó de forma semi-automática. En un primer paso, se depuraron los errores a los cambios introducidos por los sujetos dentro del template proporcionado ya que impedían la ejecución de los casos de prueba. Esta depuración en ningún caso afectó a la lógica de programación. Una vez corregidos los errores, se realizó el proceso de medición de forma automática usando el plugin JUnit modificado al que nos hemos referido en la sección 4.1.12.

Al realizar el proceso de medición semi-automática, pudimos notar que independientemente de los valores obtenidos para las variables QLTY y PROD, algunos sujetos entregaron código casi sin haber tocado el esqueleto entregado. Este comportamiento motivó a que se incluyera dentro del reporte de mediciones, una nueva variable respuesta que se denominó: *grado de completitud de la tarea realizada*, con tres valores posibles (nótese que la variable es ordinal): *Nothing* cuando el código era prácticamente el mismo que el entregado por los sujetos, *Insignificant* cuando los cambios eran menores (generalmente cuando la variable instrumental TUS era inferior al 30 por ciento del total esperado aproximadamente), y *Acceptable*, para los casos restantes.

4.1.14. Análisis de Datos

El diseño del presente experimento es una variante de los diseños *cross-over*. Este tipo de diseños experimentales han sido analizados por Vegas et al. **??**. El método de análisis más adecuado es un modelo mixto donde el sujeto experimental se especifica como un factor aleatorio. En R, dicho modelo sería el siguiente:

$$y = Strategy + Slicing + Task + (1|Subject),$$

donde y representa la calidad del código o productividad de los desarrolladores.

4.2. Replicación México-UADY2015

El experimento UADY2015 es una replicación de tipo externa del experimento base ESPE2015, esto es, que se realizó en un lugar distinto al de ESPE2015. En este caso, el experimento fue ejecutado con estudiantes de la Facultad de Licenciatura en Ciencias de la Computación de la Universidad Autónoma de Yucatán en México (UADY). Los estudiantes fueron invitados a participar en un curso taller como parte del evento académico denominado Jornadas de Ingeniería de Software FMAT2015. La diferencia principal con respecto al experimento base fue principalmente en el protocolo seguido. En este caso, se realizó un curso taller de 5 días de duración, dos de los cuales se utilizaron para las sesiones experimentales y tres para el entrenamiento, como se muestra en la siguiente tabla:

Tabla 4.4: Cronograma UADY y UNLP

Tiempo (min)	**Día 1**	**Día 2**	**Día 3**	**Día 4**	**Día 5**
15	Introducción	Slicing	Primera Sesión de Evaluación (ITL)	Desarrollo basado en pruebas (TDD)	Segunda Evaluación (TDD)
15	Cuestionario demográfico				
60	Introducción al desarrollo ágil				
60	Pruebas Unitarias con Junit	Incremental Test Last (ITL)			
15	Receso	Receso	Receso	Receso	Receso
75	Tarea de control 1	Tarea de control 2	Feedback	Tarea de control 3	Feedback

Otro aspecto a considerar es que, a diferencia del experimento base, en esta replicación los sujetos tuvieron sesiones de entrenamiento más intensivas y en días consecutivos. En cuanto a las variables demográficas, nos encontramos con un grupo de sujetos más diverso en cuanto a su nivel de formación (existieron estudiantes de pre-grado que se encontraban cursando entre el cuarto y séptimo semestre, con excepción de un único participante a nivel de maestría), lo cual era de esperarse dado el contexto del experimento. También podemos comentar que en este caso la inscripción fue voluntaria y la única motivación que tuvieron los sujetos fue la entrega de un certificado de participación.

4.3. Replicación Argentina-UNLP2015

Esta replicación del experimento base fue realizada en la Universidad Nacional de la Plata (UNLP) en Argentina, con un grupo de estudiantes principalmente de doctorado que en su momento se encontraban colaborando en los distintos laboratorios de investigación de la Universidad. Con excepción del grado académico de los participantes, esta replicación no difiere en gran medida de la replicación UADY2015. Podemos precisar que la motivación en este caso fue la colaboración que los participantes quisieron

dar a la doctora Claudia Pons, perteneciente al Laboratorio de Investigación y Formación en Informática Avanzada (LIFIA).

4.4. Replicación ESPE2016

Esta replicación fue realizada nuevamente en la Universidad de las Fuerzas Armadas ESPE de Ecuador. Al igual que el experimento base 4.1, los sujetos experimentales fueron estudiantes de la Carrera en Ingeniería de Sistemas e Informática. ESPE2016 no difiere sustancialmente de ESPE2015 más que en el orden cronólogico de realización de los experimentos. El objetivo de esta repliación fue incrementar la muestra poblacional dentro del mismo contexto del experimento ESPE2015.

4.5. Replicación Quito2016

La replicación Quito2016 fue realizada en el ámbito industrial. Los participantes fueron programadores pertenecientes a grupos de desarrollo de aplicaciones del Comando Conjunto de las Fuerzas Armadas del Ecuador (FF.AA), y desarrolladores de empresas afiliadas a la Asociación Ecuatoriana de Software (AESOFT). Este experimento se enmarcó en dos cursos de entrenamiento en técnicas de desarrollo ágil. En esta replicación se introdujeron cambios sustanciales en relación al experimento base ESPE2015, y por esta razón hemos creído conveniente reportar los cambios de acuerdo a las guías de Carver[125], como se verá en las secciones subsiguientes.

4.5.1. Motivación para la realización de la replicación Quito2016

Uno de los aspectos principales que motivaron la realización de la replicación Quito2016 fue contrastar los resultados de los experimentos en el contexto académico con el ámbito empresarial. Asimismo, se esperaba identificar la influencia que pueden tener las características personales del programador que no fueron consideradas hasta el momento en el ámbito académico como fueron la edad, la función actual en la organización y la experiencia previa en desarrollo.

Por otra parte, se decidió incorporar la estrategia de desarrollo que el sujeto conozca (típicamante Test Last Development TLD), a la que denominamos *YourWay*, con el objetivo de evaluar en qué medida la formación impartida en ITLD y TDD era efectivamente asimilada por los programadores profesionales.

4.5.2. Nivel de interacción con los experimentadores originales

La replicación Quito2016 fue guiada principalmente por el Dr. Oscar Dieste (uno de los experimentadores originales de la UPM) quien capacitó a los participantes en las técnicas ITLD y TDD. Además condujo la realización de las sesiones experimentales. El entrenamiento en pruebas unitarias y la medición de datos fueron hechas por el proponente de este estudio.

En el caso del experimento Quito2016, las replicación también puede clasificarse como literal (es decir, la replicación se asemeja al experimento original tanto como sea posible), conjunta (algunos de los experimentadores originales participaron en la replicación) y externa, ya que el experimento se realizó en el mismo sitio que el experimento ESPE2015, pero la población es distinta.

4.5.3. Cambios respecto al Experimento base

Existen diferencias entre el experimento base ESPE2015 y esta replicación. Las diferencias principales radican en el tipo de población, los factores y en el diseño experimental, las cuales se describen a continuación.

4.5.3.1. Factores y niveles de la repliación Quito2016

Uno de los cambios introducidos en la replicación Quito2016 con respecto al experimento base fue el factor **Aproximación al desarrollo**. En esta replicación se introduce el nivel denominado **YourWay** (YW), para analizar si la formación impartida afecta a los desarrolladores participantes ya sea volviéndolos menos productivos o provocando que escriban código de menos calidad. Por lo tanto, la **Aproximación al desarrollo** en la replicación tiene tres niveles: YW, ITL y TDD.

En cuanto al factor **Nivel de especificación de la tarea**, que para el caso del experimento base incluye los niveles **Slicing y No Slicing**, se omite este factor. En esta replicación, en principio, se esperaba un número de sujetos adecuado para dividirlos en grupos con muestras representativas. Sin embargo, como se verá en la sección de resultados, existieron varios sujetos que no asistieron a ninguna de las sesiones en las que se inscribieron, y en otros casos, como era previsible, no estuvieron presentes en todo el proceso experimental. Por este motivo, la inclusión de un factor adicional complicaba la obtención de datos, independientemente del poder estadístico que se lograba alcanzar; el riesgo de que alguna combinación de niveles (esto es, alguna celda de la tabla de diseño) resultase vacía o con pocos sujetos era considerable.

4.5.3.2. Variables respuesta y métricas de la replicación

Tanto las variables respuesta así como las métricas se mantuvieron idénticas al experimento base.

4.5.3.3. Hipótesis Experimentales de la replicación

Como se ha indicado, para la replicación se introduce el nivel YourWay (YW) en el factor **Aproximación al desarrollo** por lo que se plantean nuevas hipótesis respecto al experimento base:

La calidad y su efecto en la aplicación de YW, TDD e ITL:

H_{10}: $\mu(QLTY)_{YW} = \mu(QLTY)_{ITLD} = \mu(QLTY)_{TDD}$ H_{11}: $\mu(QLTY)_{YW} <> \mu(QLTY)_{ITLD} <> \mu(QLTY)_{TDD}$

La productividad y su efecto en la aplicación de YW, TDD e ITL:

H_{20}: $\mu(PROD)_{YW} <> \mu(PROD)_{ITLD} = \mu(PROD)_{TDD}$ H_{21}: $\mu(PROD)_{YW} <> \mu(PROD)_{ITLD} <> \mu(PROD)_{TDD}$

En cuanto a las covariables, se mantienen las mismas que en el experimento base.

4.5.3.4. Diseño experimental de la replicación

Para incrementar el poder estadístico, en el diseño del experimento base se utilizó un diseño between-subjects (entre-sujetos) del tipo *cross over* ($2x2$) con los factores: *Aproximación al desarrollo* con los niveles TDD e ITLD y el *Nivel de especificación de la tarea* con los niveles Sliced y no-Sliced. Como variable de bloqueo se utilizó la *tarea experimental* con los niveles BSK y MR.

En la replicación se introduce el nivel YourWay para el factor *Aproximación al desarrollo*, y el nivel Spredsheet (SS) en el factor secundario *tarea experimental*. Se descarta el Nivel de especificación de la tarea (Sliced vs no-Sliced). Dadas las limitaciones en cuanto al número de sujetos. EL diseño resultante que se muestra en la tabla4.5, puede clasificarse como crossover 3x3, aunque la mayoría de los lectores probablemente reconocerán un cuadrado latino de dimensión 3. Las dos denominaciones son, en este caso, equivalentes.

Tabla 4.5: Diseño Experimental de la Replicación

Grupos	**Sesión Experimental 1 YW**	**Sesión Experimental 2 ITL**	**Sesión Experimental 3 TDD**
G1	BSK	SS	MR
G2	SS	MR	BSK
G3	MR	BSK	SS

4.5.3.5. Selección de sujetos para la replicación

En el experimento base así como en la serie de experimentos derivados, la población de sujetos experimentales fueron estudiantes de distintos niveles de las Universidades UADY, UNLP y ESPE. Para la replicación Quito2016, la población experimental fueron profesionales en el campo de la ingeniería de sistemas informáticos o afines que se encontraban al momento prestando sus servicios como desarrolladores de aplicaciones para el Comando conjunto de las Fuerzas Armadas del Ecuador (FF.AA) y como desarrolladores de distintas empresas afiliadas a la Asociación Ecuatoriana de Software AESOFT.

Los sujetos fueron invitados a participar en dos horarios, uno para los profesionales pertenecientes a las FFAA dentro de sus horas habituales de trabajo y como parte de sus cursos habituales de actualización profesional. Para el grupo de desarrolladores pertenecientes a las empresas de la AESOFT, se organizó las sesiones durante la tarde una vez concluida su jornada laboral habitual. Una de las dificultades percibidas para este grupo de personas fue la distancia que existía entre su lugar de trabajo (generalmente en la ciudad de Quito-Ecuador) y el sitio de realización de los cursos (situado en la Universidad de las Fuerzas Armadas ESPE, al Nor-Occidente de Quito) al que en promedio se tarda entre 45 a 60 minutos en llegar. Se observó la escasa participación de este segundo grupo, especialmente a partir del segundo día, posiblemente debido a este factor. Además la participación era voluntaria, al contrario del carácter de asistencia obligatoria para los programadores de las FF.AA.

4.5.3.6. Protocolo Experimental de la replicación

El protocolo experimental seguido en esta replicación, en general, es muy parecido al del experimento base. El cambio introducido es la inclusión de una sesión experimental en el primer día de entrenamiento, en donde se pide a los sujetos que resuelvan el ejercicio planteado de la manera en que habitualmente lo hacen (YourWay). El cronograma seguido para la replicación con los programadores de las FF.AA y de la AESOF es el siguiente:

Tabla 4.6: My caption

Sesión 1		Sesión2		Sesión3	
T1	T2	T3	T4	T5	T6
Entrenamiento en pruebas unitarias	Sesión experimental de base (Your Way)	Entrenamientoen las técnicas ITL/slicing Entrenamiento en la técnica TDD	Sesión experimental,ITL	Entrenamiento en la técnica TDD	Sesión experimental,TDD

La duración del entrenamiento fue de 4 horas diarias durante 4 días (16 horas en total), de las cuales se destinaron seis horas para las sesiones experimentales (dos para cada técnica ensayada). Como se puede observar, las sesiones experimentales tuvieron una fase de entrenamiento previo a la aplicación de las técnicas ITLD y TDD, como sucedió en el experimento

base. Pero a diferencia de este, en este caso se realizó una primera sesión experimental en el primer día del entrenamiento aplicando la técnica YW. Asimismo, se solicitó a los participantes rellenar los cuestionarios demográficos el primer día del curso y después de cada sesión experimental. Dado el contexto de este experimento, se incluyeron las covariables *edad*, *experiencia previa* y *función que desempeñaban los sujetos en la organización*.

Con excepción de los cambios aquí indicados, el resto de aspectos como la instrumentación, el protocolo de recolección de datos, medición y procedimiento de análisis, siguen las mismas consideraciones indicadas en el experimento base ESPE2015.

4.6. Replicación Costa Rica-BABEL2016

El experimento BABEL2016 es una replicación de tipo externa del experimento base ESPE2015. La replicación fue realizada en Costa Rica con un grupo de desarrolladores de la empresa Grupo Babel Software. Esta fue la segunda replicación realizada en la industria y, aunque en principio se esperaba usar el diseño de la replicación Quito2016 dado el contexto del experimento (sobre todo por el número de participantes) los cambios introducidos fueron mínimos con relación al experimento base.

En lo referente a la metodología, se analizaron las mismas variables y se utilizó el mismo diseño experimental que en el experimento base. Además se aplicó el mismo protocolo para la ejecución del experimento utilizado en UADY2015 y UNLP2015, esto es, con sesiones de entrenamiento y experimentación realizadas de forma continua por 5 días.

4.7. Replicación UTN2017

Esta replicación es la última de la serie de experimentos realizados en la presente investigación. La replicación UTN2017 fue hecha en un contexto académico con un grupo de estudiantes de la Maestría en Ingeniería de Software de la Universidad Técnica de Ibarra de Ecuador, dentro del módulo Gestión de Proyectos de Software impartido por el proponente de este estudio. Sin embargo, a efectos de investigación, la replicación UTN2017 debe contarse como industrial, ya que los participantes eran profesionales en activo que se encontraban realizando estudios de cuarto nivel de especialización.

En cuanto al diseño experimental seguido para UTN2017, éste no se aparta sustancialmente de lo realizado en el experimento base. La diferencia más importante radica en el tiempo utilizado para la ejecución de este experimento que fue de 4 días en dos fines de semana consecutivos (viernes y sábados).

Capítulo 5

RESULTADOS

We have the duty of formulating, of summarizing, and of communicating our conclusions, in intelligible form, in recognition of the right of other free minds to utilize them in making their own decisions.

Ronald A. Fisher

Resumen:

En el presente capítulo se muestran los principales resultados de la investigación, tanto desde el punto de vista experimental como observacional. El estudio experimental se realiza mediante el análisis de las variables respuesta de los experimentos individuales. El estudio observacional se realiza en base al análisis de la influencia de otros factores que podrían incidir en los resultados experimentales, como son las características personales de los programadores, los cuales poseen un interés especial para la presente tesis.

El capítulo se encuentra dividido en secciones donde se presentará, para cada experimento, la siguiente información:

a) ejecución, incluyendo la descripción de la muestra y desviaciones respecto a la planificación inicial,
b) estadísticos descriptivos,
c) pruebas de hipótesis, incluyendo análisis post-hoc, y
d) estudio observacional, incluyendo análisis de subgrupos si se estimase necesario.

.

A lo largo de la investigación, hemos podido comprobar que los valores de las variables Calidad y Productividad sólo representan parcialmente la actividad de los sujetos experimentales. El *grado de completitud* de la tarea experimental, esto es, el esfuerzo que los sujetos han puesto en la confección de código, independientemente de la Calidad o

Productividad alcanzadas, parece también una variable respuesta importante. Dado que esta variable no estaba inicialmente prevista en el diseño experimental, será analizada en un apartado específico al final de cada sección.

Para la realización de este capítulo se ha utilizado R[134], y los paquetes *broom*[135], *car*[136], *ggplot2*[137], *multcomp*[138], *psych*[139], *xlsx*[140] y *xtable*[141], *stringr*[142], *lmerTest*[143], *rcompanion*[144].

5.1. Experimento Quito2016

5.1.1. Ejecución

Este experimento fue realizado en la Universidad de las Fuerzas Armadas ESPE en Sangolquí (Quito - Ecuador) en Mayo de 2016. Los sujetos experimentales fueron profesionales en activo, reclutados mediante invitaciones realizadas en coordinación con representantes de las instituciones participantes.

5.1.1.1. Muestra

En el experimento Quito2016 participaron 23 profesionales, pertenecientes a:

1. Grupos de desarrollo de aplicaciones de software del Comando Conjunto de las Fuerzas Armadas del Ecuador (FF.AA).
2. Desarrolladores de empresas afiliadas a la Asociación Ecuatoriana de software (AESOFT).

Tal y como se observa en la tabla 5.113, la mayoría de los sujetos son desarrolladores menores de 40 años. Casi todos poseen estudios superiores, normalmente en informática. La mayor parte de los sujetos (con sólo 3 excepciones) poseen más de dos años de experiencia profesional y más de 2 años de experiencia en programación. El 65 % ha programado en Java, que es el lenguaje de programación utilizando en esta investigación.

Por otro lado, la mayor parte de sujetos (excepto en tres casos) no han utilizado pruebas unitarias, y tampoco conocen el uso de herramientas de pruebas y del framework Junit. Como era esperable dado lo anterior, los sujetos tampoco tienen experiencia en TDD. Precisamente por este motivo, de forma previa a las sesiones experimentales, hemos formado a los sujetos específicamente en pruebas unitarias y TDD.

Tabla 5.1: Características demográficas de los sujetos del experimento Quito2016

EXPERIMENTO: Quito2016		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	5
	30 >= Edad < 40 años	11
	40 >= Edad < 50 años	6
	Edad >= 50 años	1
Nivel de Educación	Other	1
	Undergraduate	0
	Bachelor	20
Experiencia profesional	Sin experiencia (<2 años)	3
	Novato (2 - 5 años)	8
	Intermedio (6 - 10 años)	6
	Experto (>10 años)	6
Experiencia en programación	Sin experiencia (<2 años)	3
	Novato (2 - 5 años)	11
	Intermedio (6 - 10 años)	7
	Experto (>10 años)	2
Experiencia en lenguaje Java	Sin experiencia (<2 años)	8
	Novato (2 - 5 años)	15
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	22
	Novato (2 - 5 años)	1
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de herramientas de pruebas	Yes	3
	No	20
Uso de la técnica TDD	Yes	1
	No	22
Experiencia en TDD	Sin experiencia (<2 años)	23
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	3
	No	20
Conocimiento del entorno Eclipse	Yes	13
	No	10
Función actual en la organización	Manager	2
	Developer	15
	Analyst	6

5.1.1.2. Preparación

El día previo a la realización del experimento se prepararon los computadores con las herramientas de software necesarias, esto es: la máquina virtual de Java, el framework Junit, el código para empaquetar y realizar mediciones de los experimentos y el IDE de desarrollo Eclipse. A cada participante se le dotó con un computador personal con similares características.

5.1.1.3. Realización

El experimento se realizó durante 4 días, en sesiones de mañana (desarrolladores FF.AA) y tarde (desarrolladores AESOF). Cada sesión tuvo una duración de cuatro horas. Durante las sesiones se impartió formación en TDD y se realizaron las tareas experimentales, conforme al siguiente protocolo:

- **Sesión 1:** En primera instancia, se solicitó que los sujetos llenaran el cuestionario demográfico. A continuación, se llevó a cabo la primera tarea experimental (solucionar BSK, MR o SS aplicando la estrategia YW). Finalmente, se realizó una primera sesión de formación en pruebas unitarias utilizando el framework Junit.

- **Sesión 2:** Se impartió formación en la técnica de ITLD.

- **Sesión 3:** Se realizó la segunda tarea experimental experimental (solucionar BSK, MR o SS aplicando la estrategia ITLD). Posteriormente, se realizó una primera etapa de capacitación en TDD.

- **Sesión 4:** Se realizó una segunda etapa de capacitación en TDD, seguida por la realización de la tercera tarea experimental (solucionar BSK, MR o SS aplicando la estrategia TDD).

Cada tarea experimental duró unas dos horas y media, con leves desviaciones. Para evitar cansancio, cada sesión tuvo un pausa establecida de 20 minutos aproximadamente. Se pidió a los sujetos que no compartiesen información sobre las tareas experimentales durante los recesos. La capacitación estuvo a cargo de Oscar Dieste y Geovanny Raura (proponente de esta tesis).

5.1.1.4. Desviaciones

El experimento se llevó a cabo de acuerdo al protocolo establecido salvo el primer día, donde la sesión de la mañana tuvo que retrasarse unos 30 minutos aproximadamente. La razón fue que que algunos sujetos tuvieron dificultad para encontrar el lugar indicado para la realización del experimento.

5.1.1.5. Reducción del conjunto de datos

Se produjo una reducción progresiva en el número de sujetos que asistieron al experimento a lo largo de los cuatro días de duración del mismo. El primer día se presentaron 23 sujetos, pero sólo 22 sujetos entregaron la primera tarea experimental (que implicaba la aplicación de la estrategia YW). La segunda y tercera tarea experimentales fueron entregadas por 19 y 15 sujetos, respectivamente. Sólo 14 sujetos realizaron en su integridad las tres tareas experimentales de que consta el experimento.

5.1.2. Estadísticos descriptivos

Describiremos por separado cada una de las variables respuestas obtenidas: Calidad y Productividad. Para una mejor compresión representaremos en el mismo gráfico un diagrama box-plot y un diagrama de perfil, donde los puntos rojos representan las medias.

5.1.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
YW	45.62	40.06	-0.20	-1.92
ITLD	61.42	35.88	-0.79	-0.93
TDD	47.74	37.72	-0.28	-1.71

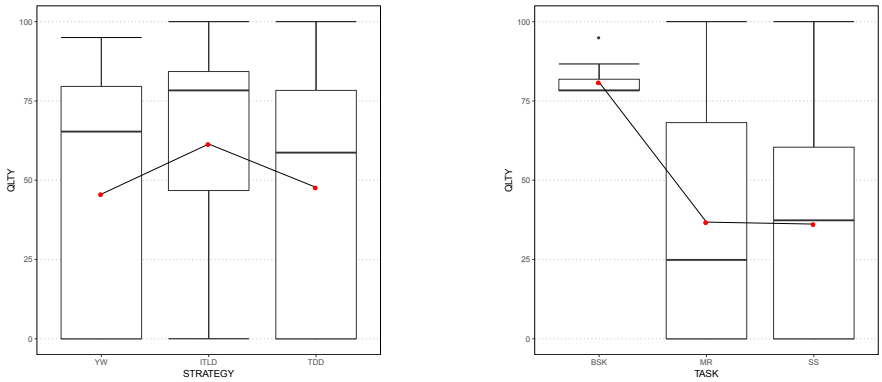
Tabla 5.2: Estadísticos descriptivos para QLTY

Como se observa en la tabla 5.114, las calidades son intermedias con desviaciones estándar altas. Ello significa que existen marcadas diferencias entre sujetos (algunos sujetos realizaron la tarea de manera satisfactoria, pero otros sólo consiguieron resultados deficientes). En promedio, ITLD obtiene un 61 % de Calidad y supera a YW y TDD, aunque no en gran medida (15 y 13 puntos, respectivamente). La figura 5.107a muestra el box-plot para el factor estrategia. La impresión que proporciona este gráfico corrobora lo indicado anteriormente: ITLD parece comportarse mejor que YW y TDD, pero en escasa medida.

La curtosis (o apuntamiento de la curva de distribución) es de tipo platicúrtica o negativa (< 0), e indica que los datos se agrupan poco respecto de la media, lo que refrenda la interpretación inicial de las diferencias entre sujetos. La asimetría es negativa, lo que indica una mayor densidad de puntos a la derecha de la media. Los valores de curtosis son bastante altos, lo que sugiere que pueden existir problemas respecto a la normalidad de los datos.

Las tareas experimentales no tienen un excesivo interés a efectos de investigación, por lo que no adjuntamos una tabla con los estadísticos descriptivos, pero si los box-plot en la figura 5.107b. MR y SS muestran unas

medias, medianas y dispersión muy parecidas. En el caso de BSK, los sujetos obtienen valores de calidad muy uniformes. La media de BSK es superior a las de MR y SS. En otras palabras: el box-plot anticipa un posible efecto significativo para la tarea.



(a) Estrategia de Programación (b) Tarea

Figura 5.1: Box plots para la variable respuesta QLTY

5.1.2.2. Productividad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
YW	15.45	17.09	0.81	-0.29
ITLD	13.40	11.30	0.14	-1.86
TDD	14.19	12.97	0.25	-1.39

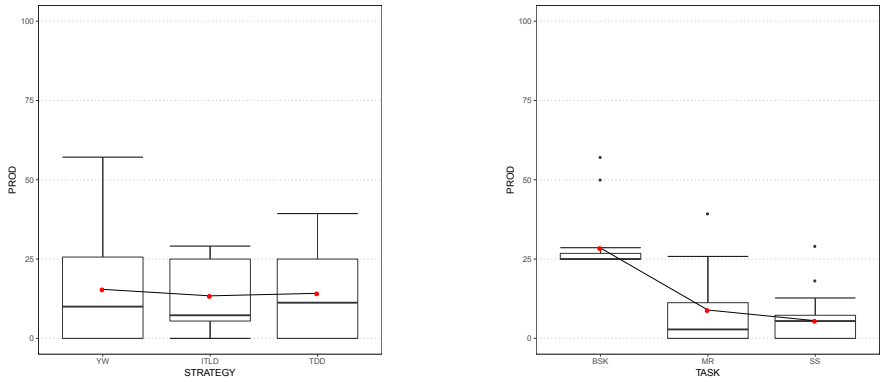
Tabla 5.3: Estadísticos descriptivos para PROD

En cuanto a la productividad, la tabla 5.115 muestra claramente que los promedios son menores que los de Calidad. Esto indica que, si bien los sujetos han conseguido implementar de forma razonablemente correcta (en promedio) la funcionalidad requerida, lo han hecho sobre un subconjunto reducido de la funcionalidad total (esto es, unas pocas historias de usuario). Ninguna estrategia de programación (YW, ITLD, TDD) destaca sobre las restantes, tal y como se aprecia en la figura 5.108a.

En cuanto a la distribución, se producen dispersiones menores que en el caso de la Calidad, aunque ello era esperable debido a los también menores valores promedio de la productividad. La curtosis es de tipo platicúrtica al igual que para la Calidad. Sin embargo, la Productividad presenta asimetría positiva (aunque no muy marcada), lo que indica que un número considerable de sujetos tienen productividades inferiores al promedio. Los valores son, en todos los casos, cercanos o menores a 1, lo cual sugiere que los valores están normalmente distribuidos.

El impacto de la tarea (BSK, MR o SS) es claramente visible en la figura 5.108b, poseyendo las mismas características que en el caso de la Calidad, esto es, los mejores valores están siempre asociados a la tarea BSK.

En general, los estadísticos descriptivos y box-plot para la Productividad sugieren las mismas conclusiones que para el caso de la Calidad: nulo efecto de la estrategia de programación, y un posible efecto de tarea.



(a) Estrategia de Programación (b) Tarea

Figura 5.2: Box plots para la variable respuesta PROD

Vale mencionar que en este experimento también queríamos comprobar si el nivel de especificación de la tarea experimental (slicing) tuvo algún efecto sobre la Calidad y Productividad, sin embargo, esto no fue posible porque había pocos sujetos como para ejecutar el experimento con 3 niveles en la estrategia de programación.

5.1.3. Prueba de hipótesis

5.1.3.1. Análisis estadístico

El análisis de los datos experimentales se ha realizado de la forma siguiente (el análisis de la Productividad es similar:

```
# Variable respuesta Calidad
lmq <- lmer(QLTY ~  1 + STRATEGY +
                        TASK +
                        GROUP + # Esta variable
                            representa la secuencia en que
                             las tareas han sido
                            realizadas
                        (1 | subjectID),
                    data = all_expData)
```

GROUP representa la secuencia en que los sujetos realizaron las tareas experimentales. Dicha secuencia no es relevante a efectos de responder a

las preguntas de investigación de la presente tesis, pero es necesaria para realizar un análisis correcto de los datos experimentales, tal y como indica Vegas et al.[145]

La tabla 5.116 muestra los resultados del análisis respecto a las variables respuesta Calidad y Productividad en formato de tabla ANOVA. Tal y como se anticipó en la sección 5.8.2, la Tarea arroja resultados significativos, tanto para Calidad (tabla 5.116a) como Productividad (tabla 5.116b). Por el contrario, la Estrategia de Programación también resulta significativa para Calidad (no así para la Productividad). Esto se debe a los mejores resultados de la estrategia ITLD, como ya mencionamos en la sección 5.8.2 y lo comprobaremos seguidamente mediante análisis post-hoc.

La variable GROUP, aunque está acotada a pequeños p-valores, no resulta estadísticamente significativa. Esto implica que la secuencia en que las tareas han sido implementadas por los sujetos experimentales no resulta relevante.

Tabla 5.4: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	5072.89	2536.44	2.00	22.87	3.88	0.0355
TASK	16587.08	8293.54	2.00	23.41	12.67	0.0002
GROUP	4667.11	2333.56	2.00	11.82	3.57	0.0614

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	45.42	22.71	2.00	19.41	0.70	0.5078
TASK	3258.72	1629.36	2.00	19.60	50.36	0.0000
GROUP	152.65	76.33	2.00	20.12	2.36	0.1201

5.1.3.2. Análisis post-hoc

La existencia de resultados significativos en el análisis estadístico exige realizar un análisis post-hoc. Este análisis se muestra, para la calidad, en la tabla 5.117, mientras que para la Productividad aparece en la tabla 5.118.

El análisis post-hoc confirma que la estrategia ITLD produce código con mayor calidad que YW (unos 27 puntos porcentuales), pero ninguna otra diferencia (por ejemplo: TDD vs. ITLD resulta estadísticamente no significativa). En lo referente a las tareas, las tablas 5.117 y 5.118 muestran claramente que la tarea BSK obtiene mayores valores de Calidad y Productividad que las tareas MR y SS. Esto se apreciaba en los box-plot de la sección 5.8.2. Las diferencias son notables, llegando a producirse hasta 48 puntos de diferencia entre BSK y SS para la Calidad. Las tareas MR y SS se

comportan de forma parecida, aunque para la Productividad su diferencia resulta estadísticamente significativa.

Tabla 5.5: Resultados del análisis post-hoc para QLTY

(a) STRATEGY

	Estimate	Std. Error	z value	Pr(>|z|)
ITLD - YW	27.07	9.72	2.78	0.01
TDD - YW	11.03	9.99	1.10	0.51
TDD - ITLD	-16.05	10.80	-1.49	0.30

(b) TASK

	Estimate	Std. Error	z value	Pr(>|z|)
MR - BSK	-34.53	9.85	-3.51	0.00
SS - BSK	-48.11	10.03	-4.79	0.00
SS - MR	-13.58	10.48	-1.30	0.40

Tabla 5.6: Resultados del análisis post-hoc para PROD

	Estimate	Std. Error	z value	Pr(>|z|)
MR - BSK	-15.73	2.48	-6.33	0.00
SS - BSK	-24.15	2.48	-9.73	0.00
SS - MR	-8.41	2.67	-3.15	0.00

5.1.3.3. Chequeo del análisis estadístico

Existen tres condiciones [1] que deben cumplirse para tener la seguridad de que los resultados del análisis estadístico mediante el modelo mixto sean fiables[146]:

1) Existe una relación lineal entre el modelo completo (y cada uno de los factores) y la variable respuesta (para esto seguiremos las recomendaciones de[147]),
2) Homogeneidad de varianzas entre los niveles de los factores, y
3) Los residuos del modelo están normalmente distribuidos.

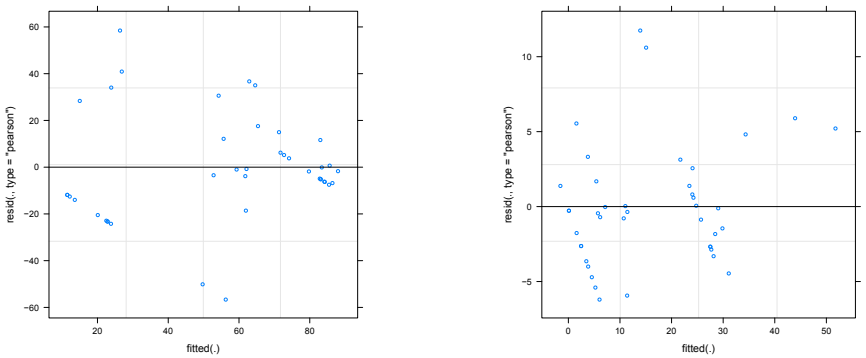
Tal y como describiremos en los apartados siguientes, las condiciones 1) y 2) se cumplen razonablemente, en tanto que la 3) no. Esto pone en cuestión la fiabilidad del análisis mostrado en la tabla 5.116, aunque la significación estadística de la Tarea puede observarse claramente en los box-plot de las

[1] A estas tres condiciones podría añadírsele la condición de independencia. No estudiamos esta condición ya que los datos provienen de un experimento aleatorizado, por lo que independencia debería garantizarse por diseño.

figuras 5.107 y 5.108. En cualquier caso, los resultados del análisis deben tomarse con la debida cautela.

Linealidad

La relación lineal entre el modelo y las variables respuesta puede comprobarse mediante un gráfico de dispersión. En el eje X se representa los valores ajustados del modelo, mientras que en el eje Y se muestran los residuos de Pearson de cada variable respuesta (Calidad o Productividad). R proporciona directamente este gráfico mediante el comando *plot*. Los gráficos resultantes se muestran en la figura 5.109.



(a) Calidad (b) Productividad

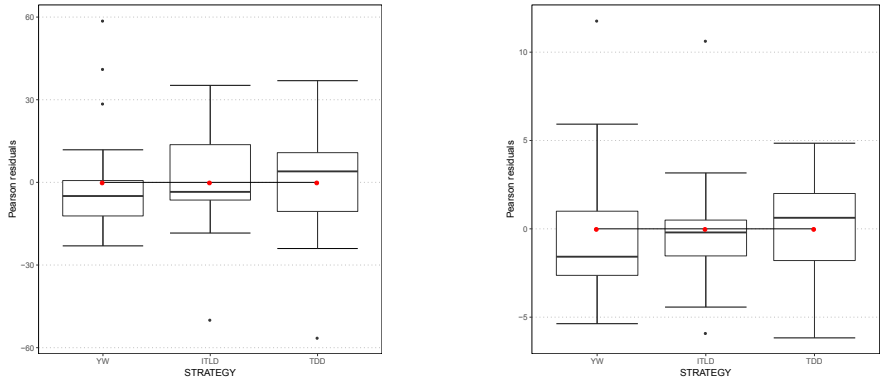
Figura 5.3: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

Cuando existe una relación lineal entre el modelo y la variable respuesta, el gráfico muestra los puntos aleatoriamente distribuidos alrededor del valor $y = 0$, y a lo largo de todos los valores de x. De existir otra relación, por ejemplo cuadrática, los puntos tenderían a formar una gráfica $y = x^2$. En el caso de la figura 5.109, si bien no se observa una nube de puntos, tampoco se observa otro patrón discernible. Por lo tanto, podemos aceptar (o mejor dicho, no rechazar) que las variables respuesta pueden representarse como una función lineal de los factores.

Una comprobación similar debe realizarse para cada factor del modelo. En este caso, debemos extraer a mano los residuos de Pearson de cada factor antes de generar el gráfico. El código correspondiente es (para el factor STRATEGY):

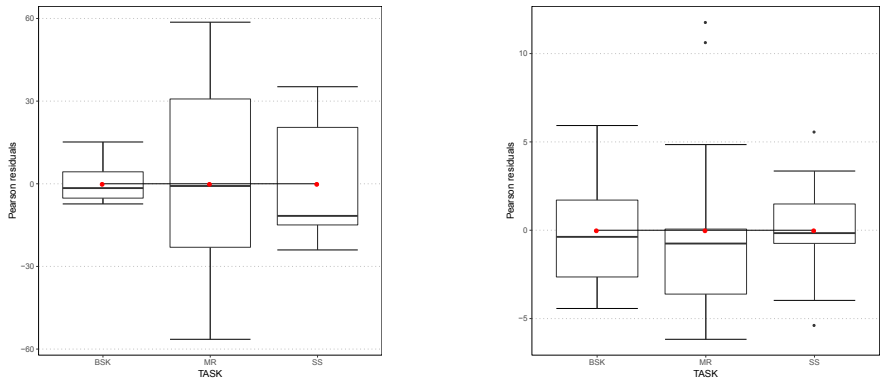
```
tmp <- data.frame(lmq@frame$STRATEGY,
                  resid(lmq, type="pearson"))
```

Si las variables independientes hubieran sido predictores continuos, la linealidad podría ponerse de manifiesto mediante una figura similar a la figu-



(a) Calidad (b) Productividad

Figura 5.4: Chequeo de la relación lineal entre el factor Estrategia y los residuos de Pearson



(a) Calidad (b) Productividad

Figura 5.5: Chequeo de la relación lineal entre el factor Tarea y los residuos de Pearson

ra 5.109. Al tratarse de factores, no es posible crear un gráfico de dispersión y tendremos que acudir a gráficos de box-plot. Para determinar la linealidad, deberemos observar la relación entre las medias/medianas de cada uno de los niveles de los factores y el valor $y = 0$, así como las dispersiones de los mismos (que hacen el papel de "nube de puntos"). Los gráficos resultantes se muestran así:

- Para el factor Estrategia, en la figura 5.110a para Calidad y en la figura 5.110b para la Productividad.

  Podemos observar que, en todos los casos, las medias (puntos rojos) y las medianas (línea central de cada caja) se localizan razonablemente en una línea recta horizontal que pasa por cero. Se observa que a medida que pasamos del factor YW a ITLD y a TDD, la media de los residuos se incrementa. No obstante, observando las dispersiones de los niveles de los factores, puede comprobarse que están razonablemente enfrentadas, sin que aparezcan patrones ascendentes o descendentes que sugieran no linealidad.

- Para el factor Tarea, en las figuras 5.111a y 5.111b para Calidad y Productividad, respectivamente.

  Las medias y medianas están alineadas y cercanas a cero en todos los casos. Las dispersiones son parecidas, con la excepción de ITLD para la Calidad, que es mucho más amplia que las restantes. No obstante, no se observa ningún patrón de no linealidad.

Homogeneidad de varianzas

La homogeneidad de varianzas se estudia utilizando el mismo gráfico mostrado en la figura 5.109, pero esta vez buscando patrones de embudo. El reducido tamaño muestral perjudica la identificación de patrones claros, sin embargo, no se aprecia la existencia de ninguna figura en embudo que sugiera heterogeneidad de varianzas.

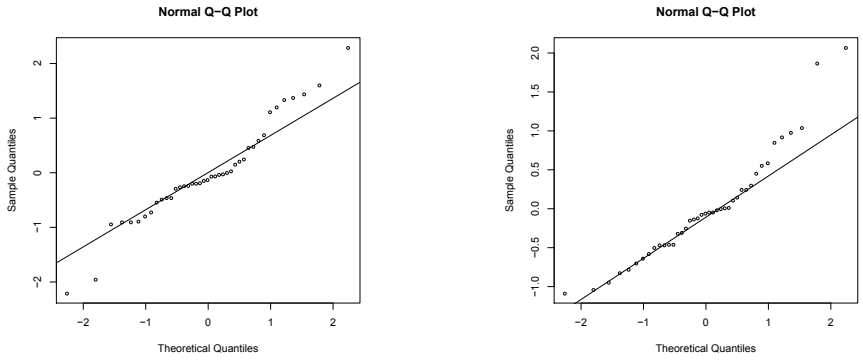
Normalidad de residuos

En los modelos mixtos, es necesario estudiar la normalidad tanto de los factores fijos como de los factores aleatorios (nótese que hay dos fuentes de varianza).

En lo atinente a los factores fijos, tal y como puede apreciarse en los gráficos QQ mostrados en la figura 5.112, la distribución de los residuos de las variables respuesta QLTY y PROD se solapan razonablemente con la distribución normal, representada en la línea recta diagonal, produciéndose desviaciones sólo en los extremos, lo cual es un hecho bastante corriente.

El test de Shapiro-Wilks confirma la normalidad de los residuos para la calidad ($W = 0{,}96$, $p - value = 0{,}12$). Sin embargo, en el caso de la produc-

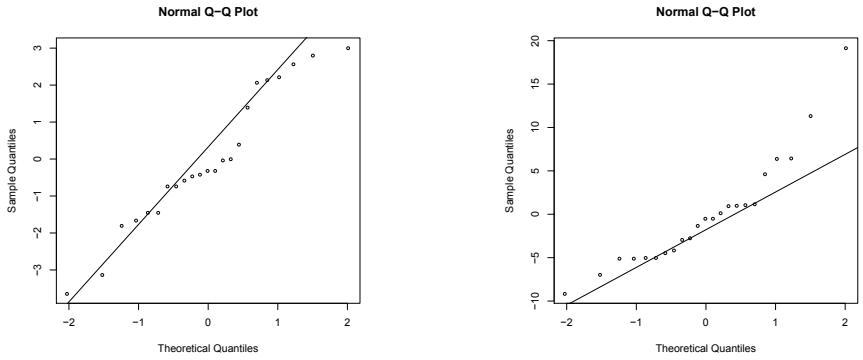
tividad, el test indica que los residuos no están distribuidos normalmente ($W = 0{,}93$, $p - value = 0{,}01$).



(a) Calidad (b) Productividad

Figura 5.6: Gráficos QQ para los factores fijos

En lo que respecta a los factores aleatorios, la situación es muy parecida a la anterior. Tal y como puede observarse en la figura 5.113, el gráfico QQ muestra que los residuos se separan notablemente de la linea diagonal, pero el test de Shapiro-Wilks confirma la normalidad de la Calidad ($W = 0{,}95$, $p - value = 0{,}26$). Ello no ocurre, al igual que en los factores fijos, para la Productividad ($W = 0{,}89$, $p - value = 0{,}02$).



(a) Calidad (b) Productividad

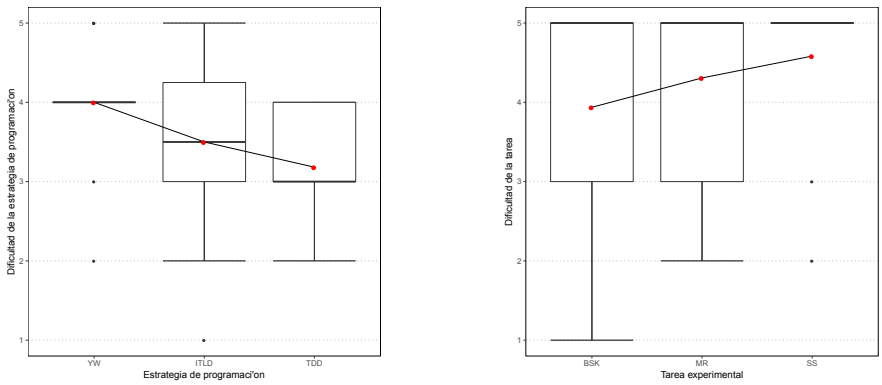
Figura 5.7: Gráficos QQ para los factores aleatorios

La falta de normalidad de residuos para la variable respuesta PROD produce que los análisis estadísticos no sean fiables. Cuando esto sucede, lo habitual es repetir el análisis utilizando métodos no paramétricos o aplicar transformaciones a los datos. Hemos aplicado esta última estrategia. Dado

que existen varios ceros en la variable respuesta PROD, hemos optado por usar la transformación "raíz cuadrada" en lugar del logaritmo, para no perder datos. Procediendo de esta forma, conseguimos que todos los residuos de los factores fijos sean normales ($W = 0{,}96$, $p - value = 0{,}16$), como también los aleatorios ($W = 0{,}95$, $p - value = 0{,}26$). Los resultados del análisis no cambian. La única excepción es la diferencia entre las tareas SS y MR, que pasan a ser no significativas, tal como se muestra en la tabla 5.119. De esta forma, los resultados respecto al factor tarea coinciden completamente para Calidad y Productividad.

Tabla 5.7: Resultados del análisis post-hoc para PROD una vez aplicada la transformación *raíz cuadrada*

	Estimate	Std. Error	z value	Pr(>|z|)
MR - BSK	-2.66	0.51	-5.23	0.00
SS - BSK	-3.83	0.51	-7.44	0.00
SS - MR	-1.17	0.54	-2.15	0.08



(a) Dificultad percibida de la estrategia de programación

(b) Dificultad percibida de la tarea

Figura 5.8: Efecto de la dificultad percibida por los sujetos

5.1.3.4. Influencia de factores instrumentales

Los resultados obtenidos en las secciones anteriores podrían deberse a factores instrumentales (no debidos al fenómeno en sí, sino al modo en que se ha investigado, esto es, al diseño experimental). Por una parte, la estrategia de programación no es algo que los sujetos experimentales conozcan de antemano, sino que fue enseñada mediante cursos de formación específicos. Del mismo modo, las tareas fueron seleccionadas específicamente para el experimento. Lo lógico sería esperar que los resultados obtenidos dependan, al

menos parcialmente, de la complejidad de las estrategias de programación y tareas experimentales.

Dificultad de la estrategia de programación

Los sujetos informaron, mediante una encuesta post-experimental, su punto de vista sobre la complejidad de las diferentes estrategias de programación usando una escala de Likert de 1 a 5 puntos (1 = más compleja, 5 = más sencilla). Las medianas y las medias difieren entre categorías, tal y como puede comprobarse en la figura 5.114a. YW es la estrategia considerada más sencilla, mientras que TDD es vista como la más difícil. El test Kruskal-Wallis entre las estrategias de programación y la dificultad percibida arroja un p-valor = 0,03 estadísticamente significativo, lo cual confirma la impresión visual obtenida en la la figura 5.114a.

Dado que existen tres estrategias de programación (YW, ITLD, TDD), la diferencia significativa en la dificultad percibida pueden darse en todas las combinaciones posibles, o sólo en algunas de ellas. A continuación mostramos los resultados de los test de Wilcoxon para cada par de estrategias:

- $p - valor_{YW,ITLD} = 0{,}23$
- $p - valor_{YW,TDD} = 0{,}005$
- $p - valor_{ITLD,TDD} = 0{,}42$

La única diferencia estadísticamente significativa surge entre YW y TDD. En los casos YW-ITLD y ITLD-TDD, las diferencias son menores y no significativas. Estos resultados sugieren que las diferencias entre YW y TDD podrían estar infladas tanto para la Calidad como la Productividad. No obstante, en el caso particular de YW-TDD, las diferencias en Calidad y Productividad no son significativas. Esto impica que es poco probable que el factor instrumental *Dificultad de la estrategia de programación* esté actuando.

Dificultad de la Tarea experimental

La dificultad de la tarea posee características muy simulares a las indicadas anteriormente para la estrategia de programación, tal y como puede observarse en la figura 5.114b: las medianas son similares y las disperciones relativamente parecidas. Las medias divergen ligeramente, aunque esto debe tomarse con cautela al tratarse de una escala ordinal.

El test Kruskal-Wallis entre las tareas experimentales y la dificultad percibida arroja un p-valor = 0,35, lo cual sugiere que todas las tareas experimentales poseen una complejidad comparable. Esto, al contrario a lo que pueda parecer, no es un resultado sencillo de comprender. Si la dificultad de la tarea no explica las diferencias entre BSK y MR/SS que señalan los análisis estadísticos de la tabla 5.116, entonces otro aspecto relacionado con

la tarea está causando dichas diferencias. No es el momento de indagar las causas subyacentes, pero es un aspecto interesante que merece investigar en el futuro.

5.1.4. Análisis de subgrupos

La demografía de los participantes (véase sección 5.8.1.1) indica que existen diferencias sustanciales entre sujetos en lo que respecta a:

- Edad.
- Experiencia profesional.
- Experiencia en programación.
- Experiencia en Java.
- Conocimiento del entorno Eclipse.
- Función actual en la organización.

En las secciones siguientes estudiaremos si la calidad del código o la productividad de los programadores están relacionadas con algunas de las características anteriores. Sería posible estudiar otras variables (las restantes que aparecen en la tabla 5.113). Sin embargo, los sujetos no presentan diferencias sustanciales en dichas características (nivel de educación, uso de herramientas de pruebas, experiencia en el framework jUnit, uso de TDD, experiencia en TDD y entrenamiento en pruebas unitarias), por lo que su análisis resultaría estéril.

5.1.4.1. Edad

Hemos calculado la influencia de la edad sobre la Calidad y la Productividad de la forma siguiente (para la Productividad el análisis es similar):

```
# Variable respuesta Calidad
lmq <- lmer(QLTY ~ 1 +
                   STRATEGY * age +
                   TASK +
                   GROUP +
                   (1 | subjectID),
            data = all_expData)
```

El resultado del análisis indica que la Edad no posee un efecto significativo ni en solitario ni en interacción con la Estrategia de Programación, tal y como puede observarse en la tabla 5.120. Cabe una pequeña excepción: La interacción $STRATEGY * age$ es casi significativa (p-valor=0.059).

En la tabla 5.120 también puede observarse que el efecto de la Estrategia de Programación pasa a no ser significativo para la Calidad (sin incluir

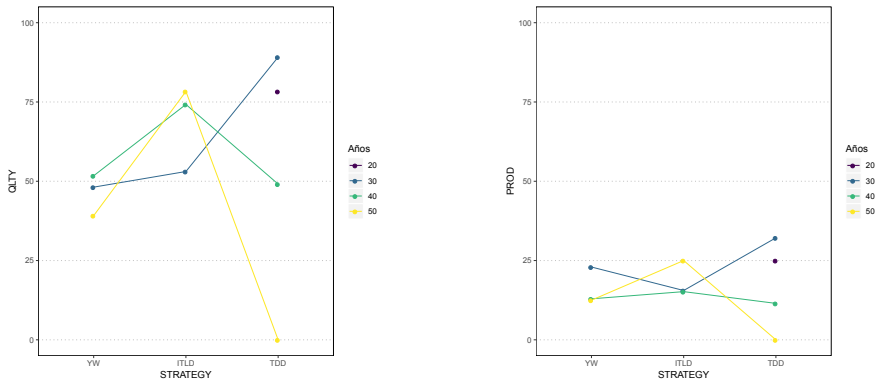
Tabla 5.8: Resultados del análisis estadístico para la Edad

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3616.29	1808.14	2.00	26.02	3.27	0.0539
age	195.66	195.66	1.00	22.03	0.35	0.5578
TASK	7918.95	3959.47	2.00	21.10	7.17	0.0042
GROUP	5417.30	2708.65	2.00	12.13	4.90	0.0275
STRATEGY:age	3498.24	1749.12	2.00	25.78	3.17	0.0589

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	6.09	3.05	2.00	24.43	2.05	0.1501
age	2.09	2.09	1.00	24.39	1.40	0.2473
TASK	59.22	29.61	2.00	20.29	19.95	0.0000
GROUP	10.77	5.38	2.00	16.97	3.63	0.0488
STRATEGY:age	5.65	2.83	2.00	24.00	1.90	0.1708



(a) Calidad (b) Productividad

Figura 5.9: Efecto de la Edad. La edad se reporta redondeado a décadas. Por ejemplo, el valor 20 representa el rango de 15-24 años de experiencia. El valor 30 representa el rango 25-34, etc. El valor redondeado se escoge en el centro del intervalo para facilitar la lectura de los gráficos y, al mismo tiempo, evitar cambios súbitos.

la edad, el resultado era estadísticamente significativo), lo que sugiere la existencia de algún tipo de relación entre la Estrategia de Programación y la Edad (de no existir, lo normal sería que los p-valores disminuyeran al introducir nuevas variables). Creemos que esta relación es precisamente lo que denota la interacción $STARTEGY * age$ indicada más arriba.

Dicha relación también puede observarse gráficamente, aunque con dificultades, en la figura 5.115. Dado que el rango de edades es muy amplio (24 - 52), hemos discretizado la edad en decadas de la forma siguiente:

```
all_expData$ageInDecades <- round(all_expData$age/10)*10
```

lo cual produce un gráfico de perfil, más sencillo de interpretar que un scatter plot:

- Si la Edad (sin interacción con la Estrategia de Programación) ejerciese un efecto en la variable respuesta (Calidad o Productividad), entonces deberíamos observar líneas (de colores, cada una correspondiente a una década) en paralelo, apiladas unas encima de otras.
- Si la interacción *STRATEGY * age* ejerciese un efecto en la variable respuesta, deberíamos ver dientes de sierra definidos por un patrón ascendente o descendente de edad.

Las líneas de la figura 5.115 se superponen, no se apilan. Esto implica, tal y como indica la tabla 5.120, que el efecto de la Edad es no significativo tanto para la Calidad como la Productividad.

Sin embargo, si son claramente visibles los dientes de sierra, especialmente para la Calidad. Los sujetos se comportan de manera similar para YW e ITLD, aunque de forma bastante sorpresiva, los resultados para ITLD parecen un poco mejores que los resultados de YW. La clave estriba en la estrategia TDD. Los sujetos de 30 años parecen incrementar su efectividad al usar TDD, mientras que los sujetos de 40 y 50 años muestran eficacias claramente decrecientes. En otras palabras: TDD funciona peor a medida que aumenta la edad de los sujetos. No obstante, al tratarse de efectos no significativos al nivel $\alpha = 0{,}05$, esta tendencia deberá confirmarse en futuros experimentos.

5.1.4.2. Experiencia profesional

De forma similar a la Edad, hemos calculado la influencia de la Experiencia Profesional de los sujetos sobre la Calidad y la Productividad. La tabla 5.121 muestra los resultados del análisis. La Experiencia Profesional resulta estadísticamente significativa para la Calidad (nótese que la línea verde es "paralela" a las líneas morada y amarilla), aunque no para la Productividad (las tres líneas se superponen, aunque por muy poco), en la figura 5.116a. Al igual que en el caso de la Edad, una vez introducido el

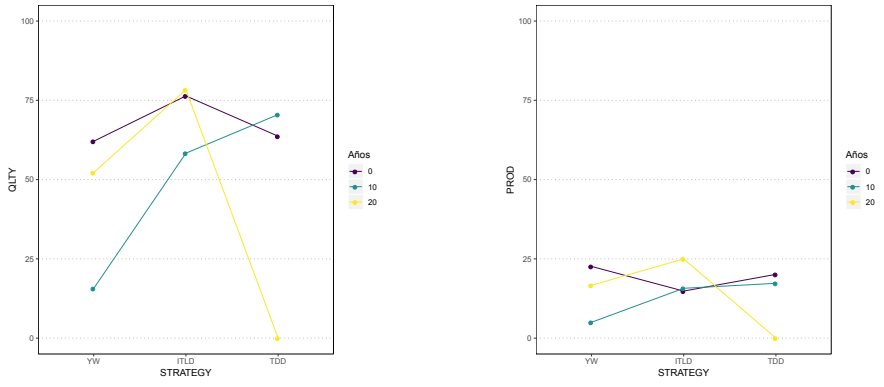
Tabla 5.9: Resultados del análisis estadístico para la Experiencia Profesional

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	2909.15	1454.57	2.00	31.00	2.39	0.1082
experienceYears	2643.65	2643.65	1.00	31.00	4.35	0.0454
TASK	13497.38	6748.69	2.00	31.00	11.09	0.0002
GROUP	5829.75	2914.88	2.00	31.00	4.79	0.0154
STRATEGY:experienceYears	1556.57	778.28	2.00	31.00	1.28	0.2925

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	4.39	2.19	2.00	24.76	1.21	0.3164
experienceYears	7.00	7.00	1.00	17.11	3.85	0.0664
TASK	82.11	41.06	2.00	24.05	22.56	0.0000
GROUP	18.09	9.05	2.00	14.07	4.97	0.0233
STRATEGY:experienceYears	4.20	2.10	2.00	24.55	1.15	0.3317



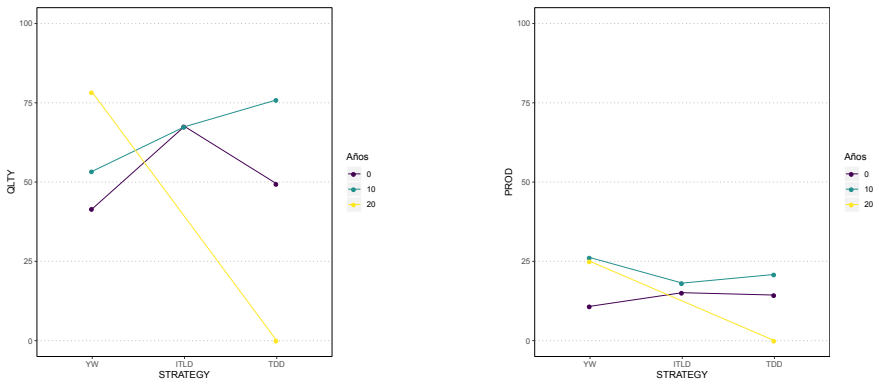
(a) Calidad (b) Productividad

Figura 5.10: Efecto de la experiencia profesional. La experiencia se redondea a décadas, como en el gráfico de perfil anterior.

aspecto personal en el análisis, la Estrategia de Programación deja de ser estadísticamente significativa.

Como la variable *experienceYears* es numérica, el modelo mixto proporciona directamente una estimación de su efecto, equivalente al parámetro B de un modelo de regresión. La Experiencia Profesional posee una influencia negativa ($B = -0{,}79$ puntos/año) para la Calidad

5.1.4.3. Experiencia en programación



(a) Calidad (b) Productividad

Figura 5.11: Efecto de la experiencia en programación. La experiencia se reporta redondeada a décadas.

En el caso de la experiencia en programación, el análisis no proporciona ningún resultado significativo. Por brevedad, no mostraremos la tabla de análisis. Si mostraremos, sin embargo, el gráfico de perfil en la figura 5.117. El gráfico de perfil sugiere que la Experiencia en Programación ejerce un efecto en la Calidad y Productividad similar al de la Experiencia Profesional y la Edad (aunque, es conveniente indicarlo nuevamente, los resultados no son estadísticamente significativos).

De nuevo, una vez introducido el aspecto personal en el análisis, la Estrategia de Programación deja de ser estadísticamente significativa.

5.1.4.4. Experiencia en programación Java

El análisis estadístico mostrado en la tabla 5.122 indica que la Experiencia en el Lenguaje de Programación Java muestra una influencia positiva, con p-valores (a efectos prácticos) estadísticamente significativos ($p - value = 0.01$ y $p - value = 0.06$) tanto para la Calidad como para la Productividad, respectivamente.

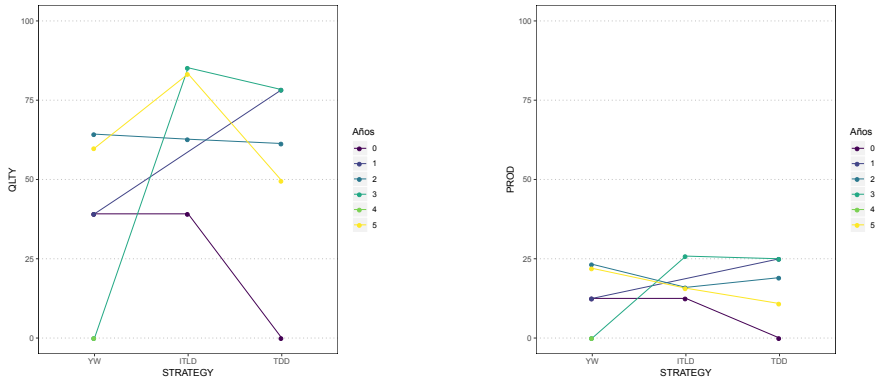
Tabla 5.10: Resultados del análisis estadístico para la Experiencia en Java

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	798.16	399.08	2.00	31.00	0.66	0.5230
javaExperience	4664.94	4664.94	1.00	31.00	7.74	0.0091
TASK	17640.89	8820.45	2.00	31.00	14.63	0.0000
GROUP	6658.24	3329.12	2.00	31.00	5.52	0.0089
STRATEGY:javaExperience	169.91	84.95	2.00	31.00	0.14	0.8691

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	6.78	3.39	2.00	24.21	2.44	0.1081
javaExperience	5.52	5.52	1.00	19.44	3.98	0.0603
TASK	98.16	49.08	2.00	20.93	35.37	0.0000
GROUP	12.21	6.11	2.00	16.71	4.40	0.0291
STRATEGY:javaExperience	2.11	1.05	2.00	23.08	0.76	0.4790



(a) Calidad (b) Productividad

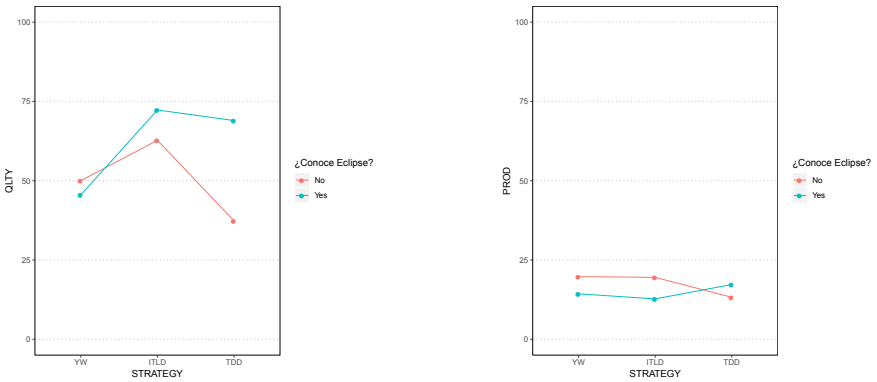
Figura 5.12: Efecto de la experiencia en programación Java. La experiencia se reporta redondeada a años, ya que los sujetos reportan experiencias únicamente entre 0-5 años.

De nuevo, una vez introducido el aspecto personal en el análisis, la Estrategia de Programación deja de ser estadísticamente significativa. Como ya hemos indicado anteriormente, de no existir relación entre la Estrategia de Programación y la Experiencia en el Uso del Lenguaje de Programación Java, dicho p-valor no debería sufrir alteraciones sustanciales y, normalmente, su valor disminuye, no aumenta. Es razonable suponer, por lo tanto, que si existe dicha relación.

En la figura 5.118 puede observarse con una relativa claridad (menor de la que nos gustaría) que a mayor experiencia, mayor Calidad. La productividad está muy igualada. El efecto es considerable para la Calidad ($B = 4.99$, aunque no para la Productividad ($B = 0.18$). Por poner los efectos en contexto, una persona con 10 años de experiencia en Java produciría un código con 49.9 puntos más de calidad que una persona sin experiencia. Se trata sin duda de un efecto muy notable que explica por qué la la Estrategia de Programación deja de ser estadísticamente significativa.

Aunque se observan dientes de sierra sugiriendo una interacción *STRATEGY * javaExperience*, las diferencias no son suficientes para obtener resultados estadísticamente significativos.

5.1.4.5. Conocimiento del entorno Eclipse



(a) Calidad (b) Productividad

Figura 5.13: Efecto del conocimiento del entorno Eclipse.

Otro aspecto analizado es el conocimiento del entorno de desarrollo Eclipse. No adjuntamos la tabla de análisis ya que todos los resultados son no significativos. La figura 5.119 sugiere que el Conocimiento del Entorno Eclipse podría mejorar la Calidad, pero las líneas están bastante superpuestas, lo que resulta plenamente coherente con el análisis estadístico. Adicionalmente, la Estrategia de Programación sigue siendo estadísticamente significativa, lo que pone de manifiesto que el Conocimiento del Entorno Eclipse no tiene

impacto sustancial en la Calidad.

5.1.4.6. Función actual en la organización

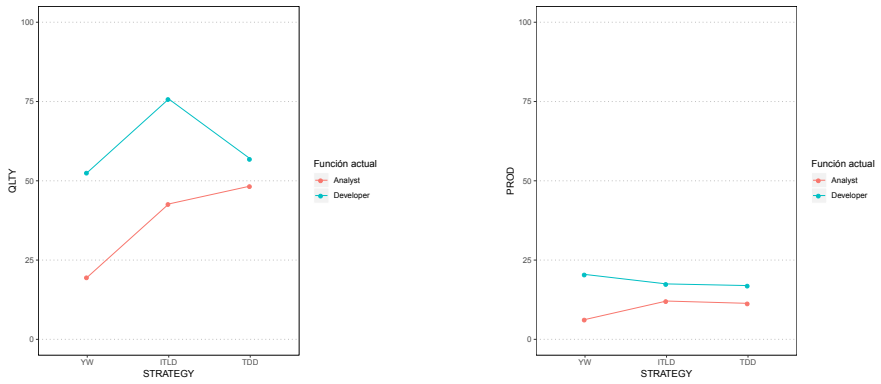
Tabla 5.11: Resultados del análisis estadístico para la Función Actual en la Organización

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3504.07	1752.03	2.00	29.00	2.74	0.0812
currentFunction	2079.52	2079.52	1.00	29.00	3.25	0.0817
TASK	16984.28	8492.14	2.00	29.00	13.29	0.0001
GROUP	4399.93	2199.96	2.00	29.00	3.44	0.0456
STRATEGY:currentFunction	754.39	377.20	2.00	29.00	0.59	0.5607

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	5.36	2.68	2.00	22.04	1.42	0.2623
currentFunction	3.42	3.42	1.00	15.40	1.82	0.1971
TASK	90.97	45.48	2.00	20.26	24.13	0.0000
GROUP	10.19	5.09	2.00	14.01	2.70	0.1017
STRATEGY:currentFunction	0.31	0.16	2.00	21.71	0.08	0.9208



(a) Calidad (b) Productividad

Figura 5.14: Efecto de la función actual en la organización

Existen tres tipos de perfiles: Gestores, analistas y desarrolladores. Como sólo había dos gestores en la muestra de sujetos experimentales, los hemos eliminado del conjunto de datos y hemos analizado únicamente el comportamiento de los analistas y desarrolladores.

Tal y como se observa en la tabla 5.123, la función que se encontraban desempeñando los sujetos en la organización ejerce un efecto (a efectos

prácticos) estadísticamente significativo en la Calidad, aunque no en la Productividad. Al igual que en otros casos, el p-valor de la Estrategia de Programacion supera, al introducir la característica personal, el nivel $\alpha = 0{,}05$.

Este efecto se muestra en la figura 5.120, donde se observa claramente que los desarrolladores (línea verde) consiguen mejores resultados que los analistas (línea roja). Las interacciones no son estadísticamente significativas. En general, los resultados se parecen mucho y, probablemente, están relacionados, con la Experiencia en Programación Java.

5.1.4.7. Resumen general

Tabla 5.12: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Variable		STRATEGY * Variable			
	B	p-value	B ITLD	p-value ITLD	B TDD	p-value TDD
Edad	0.52	0.56	0.72	0.68	**-3.64**	**0.02**
Experiencia Profesional	**-0.79**	**0.05**	0.32	0.86	-2.13	0.16
Experiencia en Programación	-0.58	0.92	3.54	0.21	-2.13	0.31
Experiencia en Programación Java	**4.99**	**0.01**	2.66	0.6	1.4	0.8
Conocimiento del Entorno Eclipse	6.98	0.1	13.68	0.49	13.97	0.5
Función Actual en la Organización	**16.38**	**0.08**	15.44	0.52	-11.93	0.6

Las tablas 5.124 y 5.125 muestran el resumen de los análisis de subgrupos realizados. Estrictamente hablando, la mayor parte de las variables estudiadas no ejercen ningún impacto en la Calidad o la Productividad, con excepción de la Experiencia Profesional y la Experiencia en Programación Java. Relajando un poco la condición $\alpha = 0{,}05$, otras variables (resaltadas en negrita en las tablas) pueden considerarse influyentes. Los resultados para la Calidad y Productividad son coherentes entre sí.

Todos los aspectos personales estudiados tienen una cierta relación entre sí. La Edad, y las distintas experiencias, comparten la dimensión *tiempo*. Esto produce que exista cierta correlación entre las mismas. Por ejemplo, la existencia de efectos significativos relacionados con la Experiencia Profesional (negativos) y con la Experiencia en Java (positivos) se podría explicar porque los sujetos más jóvenes y con menor experiencia profesional, generalmente tienen mayor conocimiento del lenguaje Java.

Tabla 5.13: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Variable		STRATEGY * Variable			
	B	p-value	B ITLD	p-value ITLD	B TDD	p-value TDD
Edad	0	0.25	0	0.78	**0.03**	**0.06**
Experiencia Profesional	**0**	**0.07**	0	0.59	0.01	0.27
Experiencia en Programación	0	0.8	0.03	0.23	0.02	0.21
Experiencia en Programación Java	**0.18**	**0.06**	0.1	0.24	0	0.84
Conocimiento del Entorno Eclipse	0.04	0.33	0.1	0.76	0.88	0.39
Función Actual en la Organización	1.11	0.2	0	1	0.24	0.71

Del mismo modo, se podría explicar el menor desempeño de los analistas por la misma razón anterior, aunque invertida: los analistas son habitualmente de mayor edad que los desarrolladores, y por lo tanto con menor experiencia profesional.

De todas formas, un único experimento proporciona evidencia limitada acerca del fenómeno de estudio. Deberemos aguardar hasta el meta-análisis del Capítulo 6 para determinar si los efectos observados son consistentes a lo largo del conjunto de experimentos realizados.

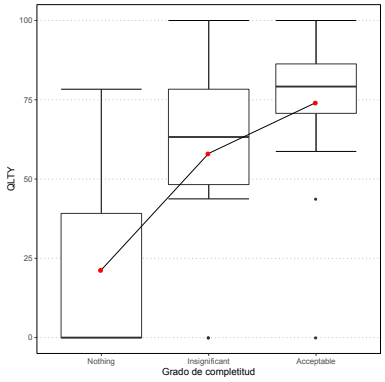
5.1.5. Grado de completitud

5.1.5.1. Aspectos generales

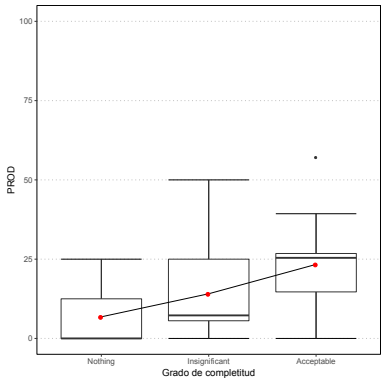
Además de la medición de las variables respuesta, el código que entregaron los sujetos fue revisado manualmente, independientemente de las puntuaciones obtenidas para las variables Calidad y Productividad. Inmediatamente pudimos percibir que algunos sujetos habían trabajado considerablemente, mientras que otros apenas habían tocado el código. En algunos casos, los sujetos devolvieron exactamente el mismo código que habían recibido al inicio de la tarea experimental.

Para estudiar este fenómeno, a cada tarea entregada por los sujetos se le asignó una etiqueta: *Nothing*, cuando el código entregado fue *prácticamente* el mismo código que el descargado de *github*; *Insignificant*, cuando los cambios realizados estaban restringidos a unas pocas líneas de código; y *Aceptable* en los casos restantes. Observe que dichos grados de completitud no implican puntuaciones concretas de QLTY o PROD, sino que en realidad

incluye códigos que exhiben un amplio rango de valores, tal y como puede observarse con claridad en la figura 5.121.



(a) Calidad



(b) Productividad

Figura 5.15: Relación entre el grado de completitud y la calidad y productividad

El número de sujetos que no realizó ningún trabajo sustancial en las tareas experimentales es bastante alto. La Tabla 5.126 muestra los detalles. De un total de 41 tareas recogidas, 11 sujetos no hicieron nada. En 12 casos, los sujetos hicieron una mínima cantidad de trabajo. Únicamente en 18 casos, los sujetos realizaron el trabajo esperado.

5.1.5.2. Impacto de la estrategia de programación y la tarea experimental

Un primer aspecto que deberíamos comprobar es si el grado de completitud de las tareas está relacionado con la estrategia de programación o la tarea experimental. Esto podría sugerir, por ejemplo, si alguna tarea fue especialmente compleja, o si los sujetos dejaron de trabajar a medida que

	Número de tareas entregadas
Nothing	11
Insignificant	12
Acceptable	18

Tabla 5.14: Grado de completitud

progresaban las sesiones experimentales, por citar dos ejemplos.

La *estrategia de programación* no parece estar relacionada con el grado de completitud de las tareas, como se muestra en la tabla 5.127. La estrategia YW muestra un mayor número de tareas vacías, pero esto podría ser explicado por las deserciones que se dieron después de la primera sesión experimental, lo que redujo el número de tareas vacías en la 2^{da} (ITLD) y 3^{ra} sesión (TDD).

	Nothing	Insignificant	Acceptable
YW	6	5	7
ITLD	2	6	4
TDD	3	1	7

Tabla 5.15: Estrategia de programación

Este juicio informal puede corroborarse mediante la realización de un test χ^2, el cual no indica ninguna relación entre la estrategia de programación y el grado de finalización de las tareas ($\chi^2 = 5{,}44$, $df = 4$, $p - value = 0{,}25$).

	Nothing	Insignificant	Acceptable
BSK	3	4	9
MR	3	3	7
SS	5	5	2

Tabla 5.16: Grado de completitud por tarea experimental (BSK, MR, SS)

Por el contrario, grado de completitud si parece estar relacionado con las tareas. SS (que es la tarea considerada más compleja, tal y como puede apreciarse en la figura 5.114b) es la tarea entregada más veces con completitudes de tipo *Nothing* e *Insignificant*. BSK y MR tienen muchas más entregas de tipo *Acceptable*. No obstante, las diferencias entre SS y BSK/MR no son tan elevadas como para que el test χ^2 muestre diferencias significativas ($\chi^2 = 5{,}22$, $df = 4$, $p - value = 0{,}27$).

5.1.5.3. Impacto de las variables demográficas

La sección anterior ha permitido poner de manifiesto que las causas subyacentes al grado de completitud no se deben a los factores experimentales,

sino que deben tener relación con otros aspectos como, por ejemplo, las características personales de los participantes, que es lo que interesa a esta investigación. Esta suposición se basa en el comportamiento mostrado por los sujetos durante la realización de las sesiones experimentales. Varios sujetos abandonaron el experimento de forma temprana, indicando que no estaban interesados en el entrenamiento asociado. Los que permanecieron podrían clasificarse en dos grupos: Aquellos que mostraron interés en los temas enseñados, o los que comenzaron a distraerse en lugar de prestar atención al entrenamiento. Muchos de este último grupo eran profesionales experimentados. Los primeros eran principalmente jóvenes profesionales. Este comportamiento puede relacionarse fácilmente con los resultados de la sección 5.8.4.

En las secciones siguientes estudiaremos si las variables demográficas que hemos recogido, explican el grado de completitud de las tareas entregadas por los sujetos experimentales. Las variables que examinaremos serán las mismas que en la sección 5.8.4, esto es:

- Edad.
- Experiencia profesional.
- Experiencia en programación.
- Experiencia en Java.
- Conocimiento del entorno Eclipse.
- Función actual en la organización.

Edad

La edad parece afectar al grado de terminación de las tareas. Los sujetos de mayor edad mostraron mayor probabilidad de presentar tareas incompletas, lo cual es claramente visible en el gráfico tipo box-plot mostrado en la figura 5.122. No obstante, el test Kruskal-Wallis entre la edad y el grado de completitud arroja un p-valor = 0,16, lo cual es insuficiente para confirmar la impresión visual obtenida en la figura 5.122.

Experiencia profesional

Los años de experiencia (véase la figura 5.123) exhiben un patrón similar al mostrado para la edad, probablemente debido a que la edad y los años de experiencia están moderadamente correlacionados (véase la figura 5.122). No obstante, las diferencias son más acentuadas: el test Kruskal-Wallis entre los años de experiencia y el grado de completitud arroja un p-valor = 0,02, lo cual confirma el impacto (negativo) de la experiencia profesional en el

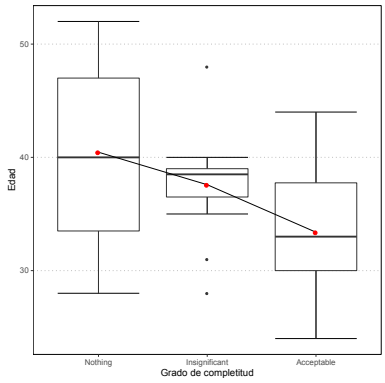


Figura 5.16: Relación entre la edad y el grado de completitud de las tareas

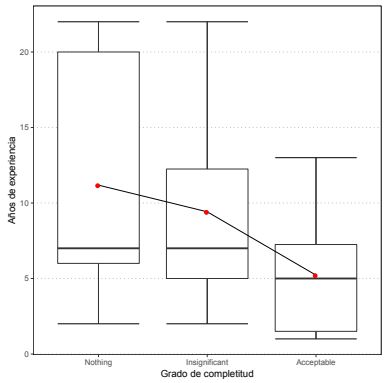


Figura 5.17: Relación entre los años de experiencia profesional y grado de completitud de las tareas

grado de completitud de las tareas: los profesionales con mayor experiencia entregan más frecuentemente tareas con completitud *Nothing* e *Insignificant*.

Experiencia en programación

La experiencia en programación no está fuertemente relacionada con la presentación de tareas vacías, como se muestra en la Fig. 5.124. La mayoría de los sujetos tenían entre 5 y 8 años de experiencia, y la distribución de experiencias fue bastante similar en todas las categorías de terminación de tareas. El test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,38, lo cual confirma la impresión visual obtenida en la figura 5.124.

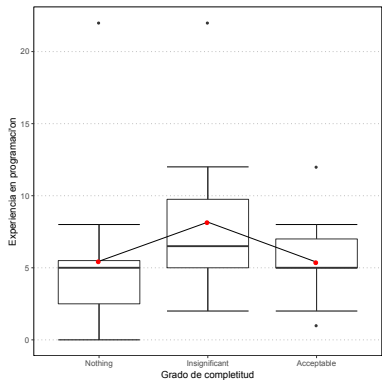


Figura 5.18: Relación entre la experiencia en programación y el grado de completitud de las tareas

Experiencia en programación Java

Por el contrario, la experiencia de programación con el lenguaje Java si influyó en la presentación de tareas incompletas (véase la figura 5.125). El test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud arroja un p-valor = 0,01.

Conocimiento del entorno Eclipse

	Nothing	Insignificant	Acceptable
No	8	5	6
Yes	3	7	12

Tabla 5.17: Conocimiento del entorno eclipse

En cuanto al conocimiento del entorno de desarrollo Eclipse y el grado de completitud de las tareas, los resultados del test χ^2 son no-significativos ($\chi^2 = 4{,}41$, $df = 2$, $p - value = 0{,}11$).

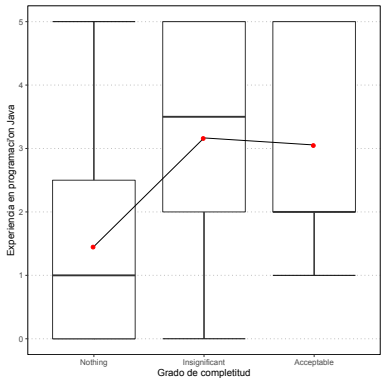


Figura 5.19: Relación entre la experiencia en Java y el grado de completitud de las tareas

Función actual en la organización

	Nothing	Insignificant	Acceptable
Analyst	5	4	1
Developer	5	8	16
Manager	1	0	1

Tabla 5.18: Funcion actual en la organización

De forma similar a lo obtenido para el conocimiento del entorno de desarrollo Eclipse, los resultados del test χ^2 son no significativos ($\chi^2 = 7{,}88$, $df = 4$, $p - value = 0{,}1$) para la función que se encontraban desempeñando los sujetos en la organización.

5.1.5.4. Resumen general impacto variables demográficas

Los resultados del análisis de la influencia de las variables demográficas estudiadas frente al grado de completitud de las tareas realizadas se sintetizan en la tabla 5.131. Como hemos advertido, las variables: *Experiencia profesional* y *Experiencia en Java* tienen influencia significativa en el grado de completitud de las tareas. Estos resultados concuerdan con el análisis realizado en la sección 5.8.4, es decir, se confirma el hecho de que los sujetos menos experimentados son los que más se esfuerzan en completar las tareas experimentales. Además, es razonable suponer que los profesionales más jóvenes tienen un mayor conocimiento del lenguaje Java. En síntesis, los resultados obtenidos de esta replicación nos conducen a pensar que la calidad y productividad de los desarrolladores se han visto mayormente afectadas por las características personales de los programadores (particularmente la

experiencia profesional y la experiencia en Java), antes que por las estrategias de programación ensayadas (YW, ITLD, TDD).

Tabla 5.19: Resumen de la influencia del grado de completitud de las tareas

	p-value
Edad	0.16
Experiencia Profesional	**0.02**
Experiencia en Programación	0.38
Experiencia en Java	**0.01**
Conocimiento del Entorno Eclipse	0.11
Función Actual en la Organización	0.1

5.2. Experimento CostaRicaBabel2016

5.2.1. Ejecución

El experimento Babel2016, al igual que el experimento Quito2016, corresponde a una replicación del experimento base ESPE2015 como se describió en el capítulo de Metodología (véase sección 4.6). El experimento fue realizado entre el 5 de diciembre hasta el 9 de diciembre de 2016 en el Grupo Empresarial Babel Software de San José de Costa Rica. Los sujetos experimentales fueron desarrolladores profesionales empleados de la empresa, quienes participaron como parte de los programas de capacitación continua que se realizan de manera permanente.

5.2.1.1. Muestra

En el experimento Babel2016 participaron 10 profesionales, pertenecientes a la empresa Grupo Babel. En la tabla 5.132, se observa que la mayoría de los sujetos tienen menos de 30 años de edad y poseen estudios superiores (Bachelor) generalmente relacionados al campo de la informática. Sin embargo, la experiencia profesional de la mayoría es inferior a 2 años. Además los sujetos indican tener entre 2 y 5 años de experiencia en programación. En cuanto a la experiencia en el lenguaje Java, el 50 % son novatos (entre 2 y 5 años) y el 50 % restante no tiene experiencia (menos de 2 años) en este lenguaje.

Ningún sujeto indica conocer herramientas de prueba aunque dos sujetos dicen conocer la técnica TDD; sin embargo, ninguno posee experiencia en desarrollo con esta técnica. Tres sujetos indican haber recibido algún tipo de entrenamiento previo en desarrollo con pruebas unitarias y la mayoría conocen el entorno de desarrollo Eclipse que fue utilizado durante la ejecución de los experimentos. La mayoría de sujetos se encontraba desempeñando las

funciones de desarrollador dentro de la empresa. Uno de ellos era Tester y tres ocupaban otras funciones.

Tabla 5.20: Características demográficas de los sujetos del experimento Babel2016

EXPERIMENTO: Babel2016		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	7
	30 >= Edad < 40 años	3
	40 >= Edad < 50 años	0
	Edad >= 50 años	0
Nivel de Educación	Other	1
	Undergraduate	0
	Bachelor	8
Experiencia profesional	Sin experiencia (<2 años)	8
	Novato (2 - 5 años)	2
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Experiencia en programación	Sin experiencia (<2 años)	2
	Novato (2 - 5 años)	7
	Intermedio (6 - 10 años)	1
	Experto (>10 años)	0
Uso de herramientas de pruebas	Yes	0
	No	10
Experiencia en lenguaje Java	Sin experiencia (<2 años)	5
	Novato (2 - 5 años)	5
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	10
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	2
	No	8
Experiencia en TDD	Sin experiencia (<2 años)	10
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	3
	No	7
Conocimiento del entorno Eclipse	Yes	7
	No	3
Función actual en la organización	Tester	1
	Developer	6
	Another	3

5.2.1.2. Preparación

Antes de la realización del experimento, los participantes instalaron en su computador personal las herramientas de software con las mismas características y versiones utilizados en todos los experimentos; esto es: la máquina virtual de Java, el framework Junit, el plugin para realizar mediciones de los experimentos y el IDE de desarrollo Eclipse.

5.2.1.3. Realización

La ejecución del experimento se realizó durante 4 días en sesiones de cuatro horas diarias a partir de las 5 de la tarde en las instalaciones de la empresa. Durante las sesiones se impartió formación en TDD y se realizaron las tareas experimentales, conforme al siguiente protocolo:

- **Sesión 1:** Previo al inicio del entrenamiento los sujetos llenaran el cuestionario demográfico. A continuación, se realizó una primera sesión de introducción al desarrollo ágil y formación en pruebas unitarias utilizando el framework Junit. A diferencia del experimento Quito2016, no se realizó una sesión experimental aplicando la técnica YW.
- **Sesión 2:** Se impartió formación en la técnica de ITLD.
- **Sesión 3:** Se realizó la primera tarea experimental que consistió en solucionar BSK o MR con slicing o sin slicing aplicando la estrategia ITLD. Posteriormente, se realizó una primera etapa de capacitación en TDD.
- **Sesión 4:** Durante esta sesión se realizó una segunda etapa de capacitación en TDD, seguida por la realización de la segunda tarea experimental (la tarea no realizada en la sesión 3).

De forma similar al experimento Quito2016, cada sesión experimental duró unas dos horas y media, con leves desviaciones. Cada sesión tuvo un pausa establecida de 20 minutos aproximadamente para evitar cansancio de los participantes. La capacitación estuvo a cargo de Geovanny Raura con la colaboración de otro investigador (Rodrigo Fonseca). Se utilizaron los materiales de entrenamiento preparados por Oscar Dieste quien también participó como capacitador en el experimento original.

5.2.1.4. Desviaciones

El protocolo se llevó a cabo de acuerdo a lo establecido. Sin embargo, durante las sesiones de entrenamiento, no todos los participantes llegaron puntuales (con un promedio de 15 minutos de retraso) debido principalmente a la densidad de tráfico y la dificultad que existía para movilizarse hacia el lugar de la capacitación en las horas programadas.

5.2.1.5. Reducción del conjunto de datos

Se produjeron variaciones en el número de sujetos que asistieron al experimento a lo largo de los cuatro días de duración del mismo. Se presentaron inicialmente 10 sujetos, pero sólo 9 realizaron la primera tarea experimental (que implicaba la aplicación de la estrategia ITLD). La segunda tarea experimental (aplicación de la estrategia TDD) fue entregada por 7 sujetos. Sólo 6 sujetos realizaron en su integridad las dos tareas experimentales planificadas para este experimento. Este comportamiento se explica de cierta forma ya que algunos profesionales invitados tenían tareas pendientes por resolver propias de su trabajo y por tanto le dieron prioridad a estas actividades antes que a la participación en las sesiones experimentales.

5.2.2. Estadísticos descriptivos

Se describen por separado los estadísticos descriptivos para cada una de las variables respuesta: Calidad y Productividad. En los gráficos mostrados, utilizamos puntos rojos para representar las medias.

5.2.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	54.69	43.79	-0.34	-1.92
TDD	39.73	46.71	0.25	-2.13

Tabla 5.21: Estadísticos descriptivos para QLTY

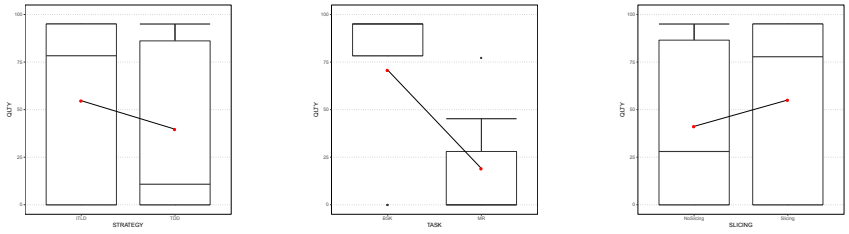
En la tabla 5.133 se observa que la Calidad es intermedia para el caso de la estrategia ITLD y un poco menor para la estrategia TDD, con desviaciones estándar altas en ambos casos. Esto implica que no todos los sujetos hicieron las tareas de forma razonablemente satisfactoria, existiendo marcadas diferencias entre ellos. El solapamiento de las cajas de la figura 5.126a) permite visualizar que es poco probable que la prueba de hipótesis nos de un resultado significativo para la estrategia de programación.

En cuanto a la distribución, la curtosis es de tipo Platicúrtica o negativa (< 0), e indica que los datos se agrupan poco respecto de la media, lo que explica las diferencias entre sujetos. La asimetría en este caso es negativa para ITLD y positiva para TDD, lo que confirma que existe una mayor densidad en la distribución a la izquierda de la media para el primer caso, y hacia la derecha para el segundo. Esto implica que más sujetos estuvieron por debajo del promedio en el caso de ITLD, en tanto que para TDD más sujetos estuvieron por encima del promedio. Como se puede observar en los datos demográficos de la tabla 5.132, un par de participantes afirmaron tener conocimiento previo en TDD, sin embargo dada la limitada cantidad

de sujetos, no podemos afirmar si las diferencias en la QLTY se relacionan con este hecho.

La figura 5.126b muestra el impacto que tiene la tarea (BSK o MR) en la dispersión de resultados. De acuerdo a la gráfica, se advierte que existen diferencias significativas con un alto rendimiento para los sujetos que realizaron BSK, y con un pobre rendimiento de los sujetos que hicieron MR (posteriormente complementaremos el análisis de este hecho, al indagar sobre la dificultad de la tarea percibida por los sujetos).

Por otro lado, como se aprecia en la figura 5.126c, no se observan diferencias significativas para el nivel de definición de la tarea, lo que nos hace suponer que las tareas presentadas con un grado de definición detallado (Slicing), no aportó mayormente en la calidad del código desarrollado por lo sujetos.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.20: Box plots para la variable respuesta QLTY

5.2.2.2. Productividad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	18.86	17.52	0.25	-1.40
TDD	9.61	13.25	0.65	-1.67

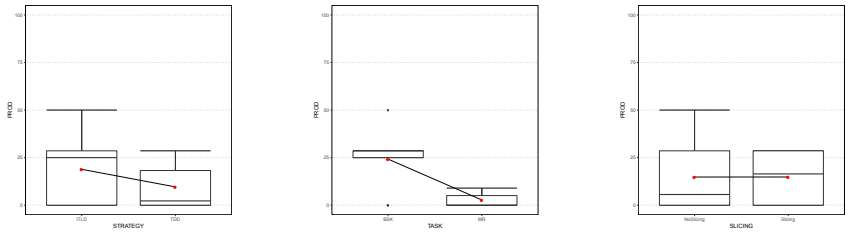
Tabla 5.22: Estadísticos descriptivos para PROD

En cuanto a la Productividad, la tabla 5.134 muestra que los promedios son menores que los de la Calidad. Esto se explica de manera análoga a lo indicado en el experimento Quito2016, donde sólo unas pocas historias de usuario fueron correctamente implementadas. Sin embargo, ITLD destaca en promedio sobre TDD con aproximadamente el doble de diferencia.

Las dispersiones obtenidas para la Productividad son ligeramente menores comparadas con la Calidad. La curtosis es negativa en ambos casos. En cuanto a la asimetría, esta vuelve a tener una valor muy bajo, cercano a cero. La interpretación de éstos estadísticos es similar a lo realizado para la Calidad.

En la figura 5.127, se aprecia el solapamiento entre las distribuciones de los datos de ITLD y TDD. Ello podría producir resultados no significativos durante el análisis, no así para la tarea (BSK o MR), donde se aprecian diferencias en las medias y no existe solapamiento entre las distribuciones, lo que sugiere la existencia de resultados significativos para la tarea (al igual que para la Calidad, BSK produce mejores resultados de Productividad que MR). Por otra parte, tampoco se aprecia una influencia del nivel de definición de la tarea sobre la Productividad aunque el grado de dispersión es mucho menor que para la Calidad.

En síntesis podemos anotar que existe un nulo efecto de la Estrategia de Programación tanto para la QLTY como para la PROD, y un posible efecto de la Tarea (aunque no se ve un efecto significativo de su nivel de definición *slicing*).



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.21: Box plots para la variable respuesta PROD

5.2.3. Prueba de hipótesis

5.2.3.1. Análisis estadístico

En este y los siguientes experimentos se consideró para el análisis, el factor *Slicing* o nivel de definición de la tarea.

```
> lmq <- lmer(QLTY ~  1 +
+                   STRATEGY +
+                   TASK +
+                            SLICING + # Esta variable representa el nivel de descripció
+                   GROUP +
+                   (1 | subjectID),
+                   data = all_expData
+ )
> lmp <- lmer(PROD ~  1 +
+                   STRATEGY +
+                   TASK +
```

```
+                    SLICING +
+                            GROUP +
+                    (1 | subjectID),
+                    data = all_expData
+ )
```

Los resultados del análisis estadístico respecto a las variables respuesta Calidad y Productividad se muestran en la tabla 5.135. Como ya se anticipó en la sección 5.9.2, sólo la Tarea arroja resultados significativos, tanto para la Calidad (tabla 5.135a) como para la Productividad (tabla 5.135b). Ninguna de las Estrategias de Programación estudiadas mejora la Calidad del código o la Productividad de los programadores. Es notable haber obtenido algún resultado significativo (en este caso la Tarea), dado el escaso número de sujetos experimentales involucrados.

En virtud al diseño experimental considerado para esta replicación, esto es, con dos tareas experimentales (MR y BSK), no es pertinente la realización de un análisis post-hoc; los estadísticos descriptivos muestran claramente que la Calidad y Productividad obtenidos para BSK supera a los de MR.

Tabla 5.23: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	677.88	677.88	1.00	4.60	1.10	0.3466
TASK	8811.96	8811.96	1.00	5.06	14.27	0.0126
SLICING	1951.42	1951.42	1.00	5.10	3.16	0.1344
GROUP	1120.03	1120.03	1.00	7.69	1.81	0.2163

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	298.90	298.90	1.00	11.00	2.60	0.1350
TASK	1661.71	1661.71	1.00	11.00	14.47	0.0029
SLICING	54.26	54.26	1.00	11.00	0.47	0.5062
GROUP	339.23	339.23	1.00	11.00	2.95	0.1137

5.2.3.2. Chequeo del análisis estadístico

Las asunciones del análisis estadístico son prácticamente las mismas que para el experimento Quito2016. En la sección 5.8.3.3, indicábamos que las asunciones de los modelos mixtos eran:

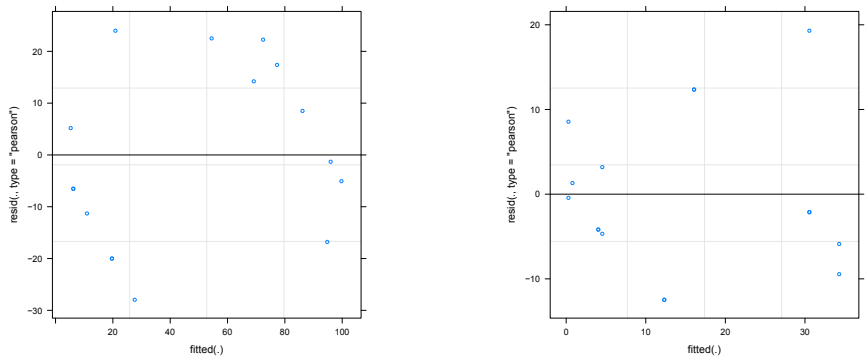
1) Existe una relación lineal entre los factores y la variable respuesta,
2) Homogeneidad de varianzas entre los niveles de los factores, y
3) Los residuos del modelo están normalmente distribuidos.

En el caso de este experimento (y en los restantes experimentos de la tesis) no es necesario comprobar la condición 1) de linealidad. Esto se debe a que los factores tienen únicamente dos niveles, lo que produce que la relación entre ambos sea siempre lineal.

Analizaremos, por tanto, únicamente las condiciones 2) y 3). En este caso, ambas condiciones se cumplen como se demuestra en las siguientes secciones. De ello se deduce que el análisis presentado en la tabla 5.135 es fiable. Al igual que en el experimento Quito2016, se mantiene la significación estadística de la Tarea como sugerían los box-plot de las figuras 5.126 y 5.127.

Homogeneidad de varianzas

El estudio de la homogeneidad de varianzas se realiza de la misma forma que en el experimento Quito2016. En los gráficos presentados en 5.128 no se aprecian patrones de embudo que nos sugiera heterogeneidad entre varianzas tanto para QLTY como para PROD, aunque el reducido tamaño muestral no permite la identificación de patrones claros.



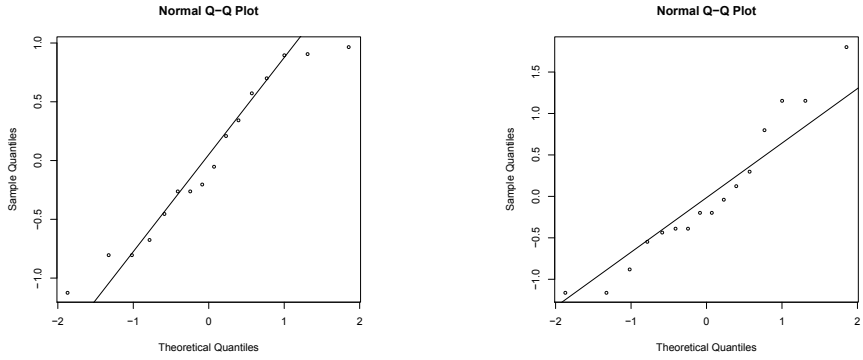
(a) Calidad (b) Productividad

Figura 5.22: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

Normalidad de residuos

Al igual que en el experimento Quito2016, estudiamos la normalidad tanto de los factores fijos como de los factores aleatorios. En lo que respecta a los factores fijos, podemos notar en la figura 5.129 que la distribución de los residuos de las variables respuesta QLTY y PROD se solapan razonablemente con la distribución normal, representada en la línea recta diagonal, con ciertas desviaciones en los extremos, lo cual es un hecho frecuente como se ha comentado anteriormente. Al realizar el test de Shapiro-Wilks confirmamos la normalidad de los residuos tanto para la Calidad ($W = 0{,}94$,

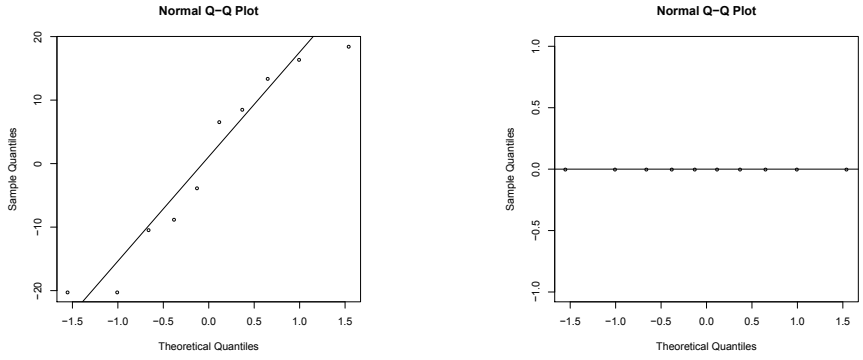
$p-value = 0{,}36$) como para la Productividad ($W = 0{,}94$, $p-value = 0{,}33$).



(a) Calidad (b) Productividad

Figura 5.23: Gráficos QQ para los factores fijos

En lo que respecta a los factores aleatorios, la situación es muy parecida a la anterior. Tal y como puede observarse en la figura 5.130, el gráfico QQ muestra que los residuos se ajustan razonablemente a la distribución normal para QLTY. El test de Shapiro-Wilks nos permite confirmar esta impresión visual ($W = 0{,}91$, $p - value = 0{,}3$). Para la productividad, no es posible realizar tal análisis, ya que los efectos aleatorios son muy pequeños e iguales a cero, tal y como se puede apreciar en la figura 5.129b. Estos ceros se deben a una simplificación que realiza el análisis *lmer* mas no a un problema asociado con los datos experimentales.



(a) Calidad (b) Productividad

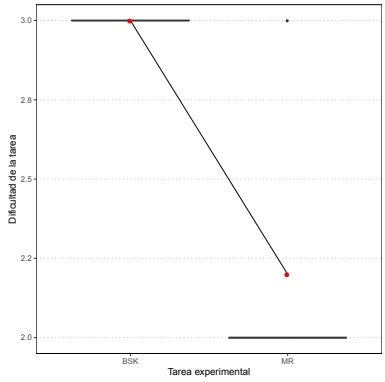
Figura 5.24: Gráficos QQ para los factores aleatorios

5.2.3.3. Influencia de factores instrumentales

Como sucedió en el experimento Quito2016, las tareas fueron seleccionadas ad-hoc siendo razonable pensar que las diferencias entre las tasas de Calidad y Productividad obtenidas al aplicar las estrategias ITLD y TDD pudieran depender en parte de la complejidad de la tarea.

Dificultad de la Tarea experimental

La figura 5.131a muestra la existencia de valores nulos para la estrategia TDD, por tal motivo, no es factible el análisis de la dificultad percibida de la tarea. El test Kruskal-Wallis entre la tarea experimental y la dificultad percibida arroja un p-valor = 0,16.



(a) Dificultad percibida de la tarea

Figura 5.25: Efecto de la dificultad de la tarea percibida por los sujetos

5.2.4. Análisis de subgrupos

De acuerdo a los datos demográficos de los participantes (véase sección 5.9.1.1) podemos establecer que existen diferencias que merecen considerarse en los siguientes aspectos:

- Edad.
- Experiencia profesional.
- Experiencia en programación.
- Experiencia en Java.
- Conocimiento del entorno Eclipse.
- Función actual en la organización.
- Entrenamiento previo en desarrollo de pruebas unitarias.

En lo referente al Uso de la Técnica de TDD, si bien dos personas indicaron que han utilizado esta técnica antes de participar en el experimento, sin embargo en este análisis no se lo ha considerado dada la poca cantidad de sujetos en los grupos. A diferencia del experimento Quito2016, en esta replicación si se incluye el análisis del entrenamiento previo en desarrollo de pruebas unitarias. En las secciones siguientes vamos a tratar de establecer si alguna de estas características guarda relación con la calidad del código o con la productividad de los desarrolladores.

5.2.4.1. Edad

De forma análoga a lo realizado en el experimento Quito2016, hemos analizado la interacción $STRATEGY * age$ y confeccionado un gráfico de perfil para determinar el impacto de la edad de los sujetos en la Calidad y Productividad. El código de R utilizado es el siguiente:

```
> lmq <- lmer(QLTY ~ 1 +
+                     STRATEGY * age +
+                     TASK +
+                     SLICING +
+                     GROUP +
+                     (1 | subjectID),
+              data = all_expData)
> lmp <- lmer(PROD ~  1 +
+                      STRATEGY * age +
+                      TASK  +
+                      SLICING +
+                      GROUP +
+                      (1 | subjectID),
+              data = all_expData)
```

El resultado muestra que la edad posee una influencia negativa para la calidad (B =-1.1) y una influencia positiva para la productividad (B =1.41). Los efectos no son significativos en los dos casos ($P - value$ =0.33 y $P - value$ =0.77).

Sin embargo, es más probable para el caso de la Calidad, que estemos en presencia de un error tipo II provocado por el reducido número de sujetos experimentales. En la figura 5.132 se puede apreciar que la Calidad obtenida por los sujetos dentro del rango de 20 a 29 años de edad supera ampliamente a la del grupo de 30 a 39 años. Este efecto no se observa para la Productividad.

En síntesis, existe cierta tendencia a mejorar la Calidad cuando las personas de mayor edad (de 30 a 39 años) aplican TDD. Por otro lado, la Productividad de todos los participantes tiende a decrecer al aplicar la es-

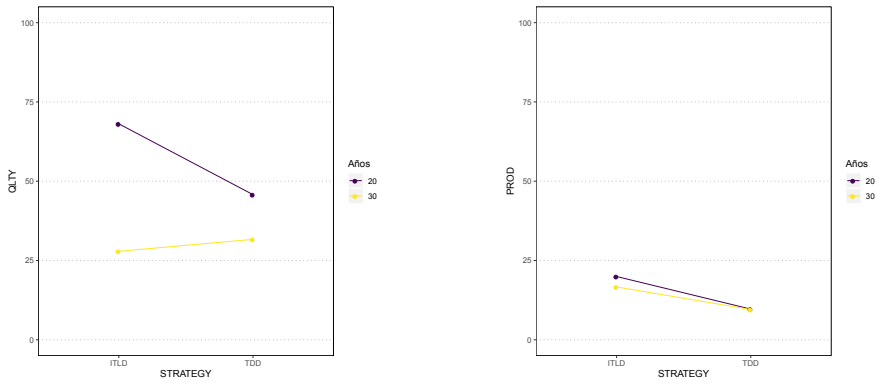
Tabla 5.24: Resultados del análisis estadístico para la Edad

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	338.69	338.69	1.00	5.21	0.60	0.4713
age	627.10	627.10	1.00	6.41	1.12	0.3290
TASK	8266.83	8266.83	1.00	4.02	14.71	0.0184
SLICING	2022.89	2022.89	1.00	4.28	3.60	0.1261
GROUP	982.10	982.10	1.00	6.43	1.75	0.2313
STRATEGY:age	464.14	464.14	1.00	4.86	0.83	0.4064

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	190.03	190.03	1.00	5.73	1.77	0.2340
age	10.09	10.09	1.00	5.95	0.09	0.7696
TASK	1799.22	1799.22	1.00	4.86	16.75	0.0100
SLICING	107.87	107.87	1.00	4.88	1.00	0.3633
GROUP	409.38	409.38	1.00	5.49	3.81	0.1032
STRATEGY:age	263.07	263.07	1.00	5.30	2.45	0.1751



(a) Calidad (b) Productividad

Figura 5.26: Efecto de la Edad. La edad se reporta redondeada a décadas. Por ejemplo, el valor 20 representa el rango de 20 a 29 años de experiencia.

trategia TDD. No obstante, estos resultados no son fiables debido a que no alcanzan la significación estadística.

5.2.4.2. Experiencia profesional

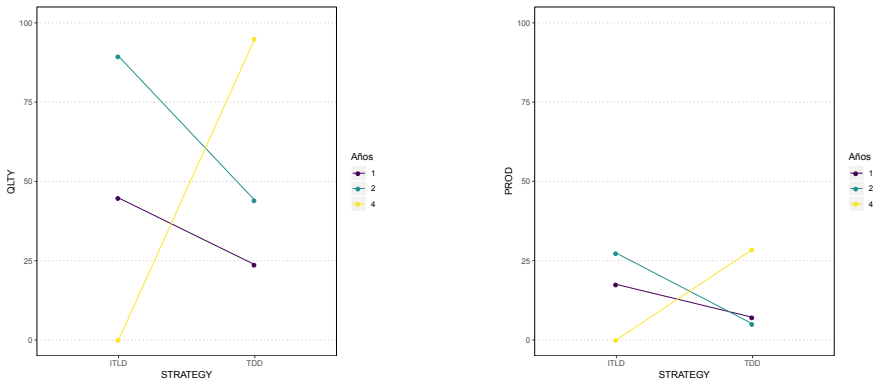
Tabla 5.25: Resultados del análisis estadístico para la Experiencia Profesional

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	2381.65	2381.65	1.00	2.35	31.40	0.0209
experienceYears	65.80	65.80	1.00	6.58	0.87	0.3845
TASK	4487.98	4487.98	1.00	2.30	59.17	0.0110
SLICING	31.71	31.71	1.00	2.44	0.42	0.5736
GROUP	89.72	89.72	1.00	6.81	1.18	0.3138
STRATEGY:experienceYears	1892.06	1892.06	1.00	2.28	24.94	0.0287

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	719.55	719.55	1.00	1.54	61.99	0.0317
experienceYears	1.19	1.19	1.00	5.48	0.10	0.7609
TASK	692.16	692.16	1.00	1.50	59.63	0.0348
SLICING	74.99	74.99	1.00	1.61	6.46	0.1559
GROUP	5.86	5.86	1.00	5.77	0.50	0.5051
STRATEGY:experienceYears	433.01	433.01	1.00	1.49	37.30	0.0499



(a) Calidad (b) Productividad

Figura 5.27: Efecto de la experiencia profesional. La experiencia se reporta redondeada a años. Por ejemplo, 1.5 años de experiencia se redondea a 2. No realizamos una discretación más amplia (lustros, décadas) debido a que todos los valores de ExperienceYears oscilan entre 0 y 4.

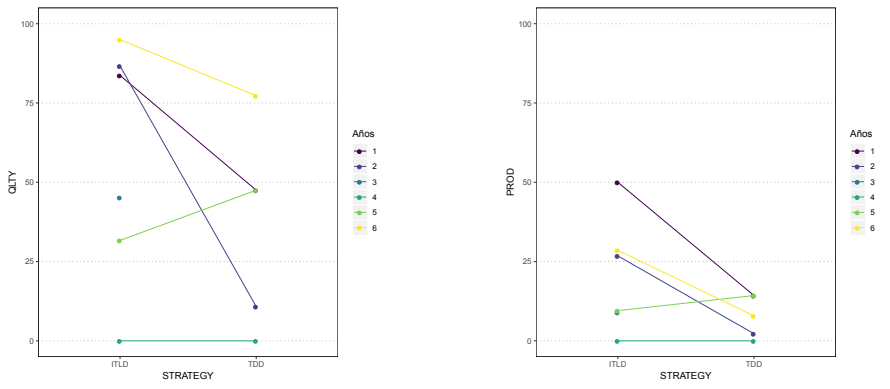
La experiencia profesional posee una influencia negativa tanto para la

Calidad como para la Productividad (B =-5.09, B =-5.73). También se observa que los efectos no son significativos ($p-value$ =0.38 y $p-value$ =0.76) en los dos casos, aunque al observar la figura 5.133 podríamos sugerir respecto a la Calidad, que se está produciendo un error tipo II, esto es, un falso negativo.

Además de los efectos principales, la tabla 5.137 muestra interacciones significativas $STRATEGY * experienceYears$ tanto para la Calidad como para la Productividad. Siendo $STRATEGY$ un factor con dos niveles (ITLD y TDD), y $experienceYears$ una variable tipo ratio, la interacción debe entenderse como el efecto que 1 año de experiencia tiene en TDD, comparado con ITLD. Por ejemplo, el efecto de la interacción es B =29.2 lo que significa que los programadores que usan TDD obtienen 29.2 puntos más que los que usan ITLD. Esta cifra depende de los años de experiencia profesional, esto es, para 2 años de experiencia profesional, el efecto sería $2 \times 29{,}2 = 58{,}4$. Para la productividad, el efecto es $B = 13{,}95$. En ambos casos, los resultados son estadísticamente significativos ($p - value$ =0.03 y $p - value$ =0.05 para la Calidad y Productividad, respectivamente).

Debemos indicar que la interacción $STRATEGY * experienceYears$ antes mencionada, es también visible en el gráfico 5.133. La interacción se manifiesta como el cruce de la línea correspondiente a 4 años de experiencia con las demás. En general, una interacción se manifiesta siempre como un cruce de líneas, pero el análisis puede resultar estadísticamente no significativo (como es el caso, por ejemplo, de la figura 5.135 relacionada con la experiencia en programación Java).

5.2.4.3. Experiencia en programación



(a) Calidad (b) Productividad

Figura 5.28: Efecto de la experiencia en programación. La experiencia se reporta redondeada a años, como en el caso anterior.

Tabla 5.26: Resultados del análisis estadístico para la Experiencia en programación

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	703.20	703.20	1.00	4.08	1.16	0.3413
programmingExperience	18.68	18.68	1.00	6.45	0.03	0.8661
TASK	6191.81	6191.81	1.00	4.05	10.20	0.0325
SLICING	1660.71	1660.71	1.00	3.44	2.74	0.1849
GROUP	822.68	822.68	1.00	6.56	1.36	0.2850
STRATEGY:programmingExperience	320.60	320.60	1.00	3.78	0.53	0.5098

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	153.12	153.12	1.00	9.00	1.21	0.3001
programmingExperience	107.22	107.22	1.00	9.00	0.85	0.3816
TASK	1382.47	1382.47	1.00	9.00	10.91	0.0092
SLICING	23.35	23.35	1.00	9.00	0.18	0.6778
GROUP	225.34	225.34	1.00	9.00	1.78	0.2151
STRATEGY:programmingExperience	24.57	24.57	1.00	9.00	0.19	0.6700

El análisis muestra una influencia negativa para las variables Calidad y Productividad (B = -1.87, B = -2.29). Los efectos no son significativos en ambos casos (p-valor = 0.87 y p-valor = 0.38). En la figura 5.134, se aprecia que en general la Calidad y Productividad decrecen al aplicar TDD, excepto para el grupo de sujetos en el rango de 5 años de experiencia en programación, donde las variables respuesta obtienen mejores resultados al aplicar TDD. No obstante, la falta de un patrón claro en los gráficos de perfil hace que los efectos no se puedan interpretar con claridad.

5.2.4.4. Experiencia en programación Java

La influencia del uso del lenguaje de programación Java es positiva para la Calidad (B =4.61) y negativa para la Productividad (B =-1.23). Vale subrayar que para el caso de la variable Calidad, el efecto se acerca al nivel de significación estadística ($P-value$ =0.06), no así para el caso de la Productividad donde el efecto no es estadísticamente significativo ($P-value$ =0.2). Los gráficos de la figura: 5.135, muestran que los desarrolladores con más experiencia en Java (con 5 o más años), no presentan diferencias sustanciales de Calidad o Productividad al aplicar TDD o ITLD, aunque la Calidad para este grupo de sujetos bordea el 100 %, en contraste con la Productividad que apenas supera el 25 %. Los desarrolladores en el rango de 2 años de experiencia en Java, tienden a mejorar la Calidad y Productividad al aplicar TDD, en cambio los desarrolladores con menos experiencia en Java obtienen peores resultados con TDD. Si bien la Calidad se aproxima al nivel de sig-

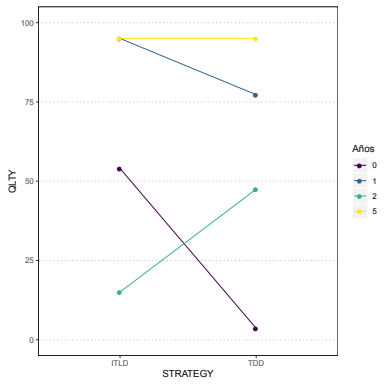
Tabla 5.27: Resultados del análisis estadístico para la Experiencia en Java

(a) Calidad

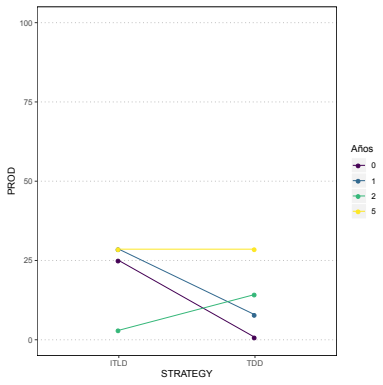
	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1639.08	1639.08	1.00	4.01	2.24	0.2090
javaExperience	3504.99	3504.99	1.00	8.53	4.78	0.0581
TASK	2903.43	2903.43	1.00	5.02	3.96	0.1030
SLICING	460.52	460.52	1.00	4.20	0.63	0.4703
GROUP	5011.09	5011.09	1.00	5.50	6.84	0.0432
STRATEGY:javaExperience	999.95	999.95	1.00	7.92	1.36	0.2767

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	592.19	592.19	1.00	9.00	6.19	0.0345
javaExperience	179.57	179.57	1.00	9.00	1.88	0.2038
TASK	564.52	564.52	1.00	9.00	5.90	0.0380
SLICING	6.70	6.70	1.00	9.00	0.07	0.7973
GROUP	670.62	670.62	1.00	9.00	7.01	0.0266
STRATEGY:javaExperience	311.95	311.95	1.00	9.00	3.26	0.1044



(a) Calidad



(b) Productividad

Figura 5.29: Efecto de la experiencia en programación Java. La experiencia se reporta redondeada a años.

nificación estadística, la poca cantidad de sujetos hace que éstos resultados deban tomarse con cautela.

5.2.4.5. Conocimiento del entorno Eclipse

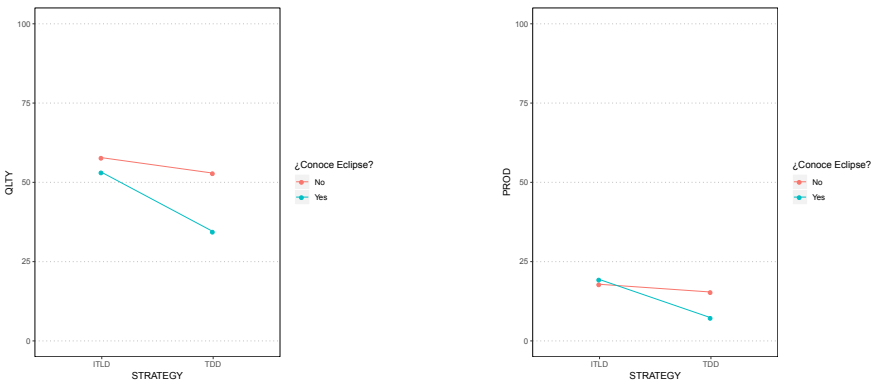
Tabla 5.28: Resultados del análisis estadístico para el conocimiento de entorno Eclipse

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	125.67	125.67	1.00	3.18	0.28	0.6286
knowEclipse	0.88	0.88	1.00	6.79	0.00	0.9657
TASK	8862.56	8862.56	1.00	3.47	20.09	0.0151
SLICING	2312.31	2312.31	1.00	3.42	5.24	0.0952
GROUP	692.17	692.17	1.00	6.84	1.57	0.2515
STRATEGY:knowEclipse	1147.22	1147.22	1.00	3.09	2.60	0.2026

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	109.41	109.41	1.00	3.89	1.30	0.3194
knowEclipse	0.32	0.32	1.00	5.79	0.00	0.9528
TASK	1731.78	1731.78	1.00	4.20	20.59	0.0094
SLICING	106.46	106.46	1.00	3.95	1.27	0.3243
GROUP	198.29	198.29	1.00	5.97	2.36	0.1758
STRATEGY:knowEclipse	268.99	268.99	1.00	3.56	3.20	0.1571



(a) Calidad (b) Productividad

Figura 5.30: Efecto del conocimiento del entorno Eclipse.

El conocimiento del entorno de desarrollo Eclipse presenta una influencia negativa tanto para la Calidad como para la Productividad (B =20.98 y B =9.03 respectivamente). Tal y como se aprecia en la figura 5.136, los

sujetos que tenían conocimiento del entorno Eclipse obtienen peores resultados de Calidad tanto al aplicar ITLD como TDD. De hecho, los sujetos que manejan Eclipse empeoran al usar TDD.

A efectos prácticos, ocurre lo mismo con la Productividad. No obstante, los efectos no son significativos ($p - value$ =0.97 y $p - value$ =0.95).

5.2.4.6. Función actual en la organización

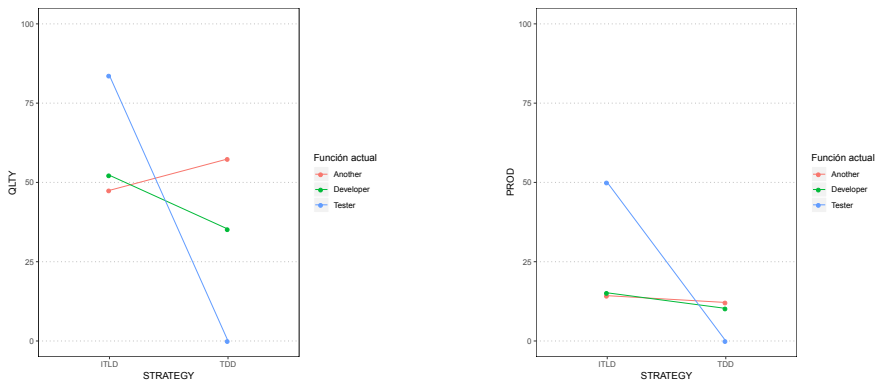
Tabla 5.29: Resultados del análisis estadístico para la Función Actual en la Organización

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1636.87	1636.87	1.00	1.02	12.87	0.1684
currentFunction	68.03	34.01	2.00	5.76	0.27	0.7743
TASK	4199.57	4199.57	1.00	1.09	33.02	0.0954
SLICING	3322.14	3322.14	1.00	1.11	26.12	0.1056
GROUP	224.46	224.46	1.00	5.86	1.76	0.2334
STRATEGY:currentFunction	1852.40	926.20	2.00	2.00	7.28	0.1207

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	697.80	697.80	1.00	1.35	18.00	0.0970
currentFunction	23.68	11.84	2.00	4.97	0.31	0.7498
TASK	752.73	752.73	1.00	1.68	19.41	0.0651
SLICING	189.20	189.20	1.00	1.74	4.88	0.1762
GROUP	47.28	47.28	1.00	5.48	1.22	0.3156
STRATEGY:currentFunction	411.80	205.90	2.00	2.00	5.31	0.1585



(a) Calidad (b) Productividad

Figura 5.31: Efecto de la función actual en la organización

El efecto de la función de los participantes en la organización sobre la Calidad y la Productividad es negativo en los dos casos (B =-5.17 y B =-2.2). Los efectos tampoco son significativos ($p-value$ =0.77 y $p-value$ =0.75).

De acuerdo a los gráficos presentados en 5.137, llama la atención que el único sujeto que indicó cumplir funciones como Tester dentro de la empresa (como se desprende de la tabla **??**, obtuvo los peores resultados tanto en la Calidad como en la Productividad al aplicar TDD. Sin embargo, como se visualiza en los gráficos, el obtener un cero por ciento de efectividad al aplicar TDD en ambos casos nos induce a pensar que el participante posiblemente entregó la tarea en blanco al realizar la segunda sesión experimental (este comportamiento no es nuevo y ya fue identificado en el experimento Quito2016. En la sección 5.9.5 se analizará con más detalle este aspecto).

5.2.4.7. Entrenamiento previo en desarrollo de pruebas unitarias

Tabla 5.30: Resultados del análisis estadístico para el conocimiento de Pruebas Unitarias

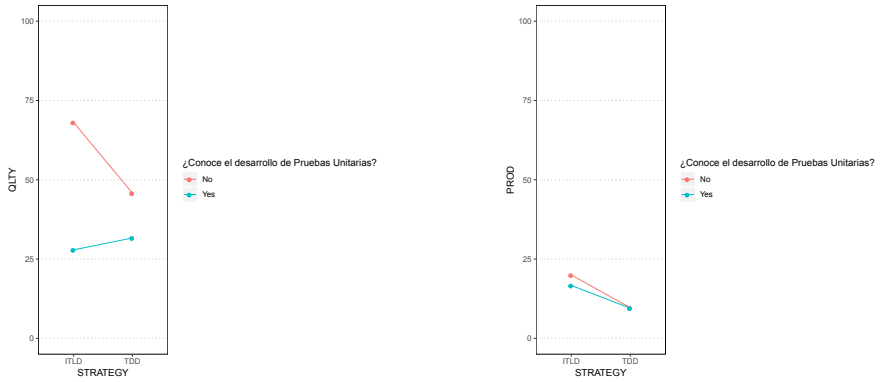
(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	699.49	699.49	1.00	3.46	1.10	0.3614
UTTraining	550.90	550.90	1.00	6.09	0.87	0.3868
TASK	8348.44	8348.44	1.00	4.63	13.16	0.0173
SLICING	2121.03	2121.03	1.00	4.79	3.34	0.1296
GROUP	709.16	709.16	1.00	6.70	1.12	0.3270
STRATEGY:UTTraining	198.02	198.02	1.00	4.38	0.31	0.6037

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	391.34	391.34	1.00	9.00	3.14	0.1104
UTTraining	26.26	26.26	1.00	9.00	0.21	0.6574
TASK	1795.05	1795.05	1.00	9.00	14.38	0.0043
SLICING	88.56	88.56	1.00	9.00	0.71	0.4214
GROUP	389.80	389.80	1.00	9.00	3.12	0.1110
STRATEGY:UTTraining	110.08	110.08	1.00	9.00	0.88	0.3722

El análisis muestra que el entrenamiento previo en desarrollo de pruebas unitarias presenta una influencia negativa para la Calidad(B =-11.88) y positiva para la Productividad (B =8.51). Sin embargo los efectos no son significativos en ambos casos (B =0.39 y B =0.66). En los gráficos presentados en 5.138, se aprecia que aquellos sujetos que indicaron conocer el desarrollo de pruebas unitarias, mejoraron la Calidad al aplicar TDD, al contrario de aquellos que indicaron no conocer el desarrollo de pruebas unitarias. Sin embargo, se aprecia una disminución de la Productividad al aplicar TDD en ambos casos.



(a) Calidad (b) Productividad

Figura 5.32: Efecto del conocimiento de Pruebas Unitarias.

5.2.4.8. Resumen general

Las tablas 5.143 y 5.144 sintetizan los resultados de los análisis de subgrupos realizados. En este caso, únicamente la *Experiencia en Java* se aproxima al nivel de significación para la variable Calidad, no así para la Productividad. Se pueden observar valores notablemente superiores de B para la Calidad con respecto a la Productividad en cuanto al *Conocimiento del Entorno Eclipse* y el *Entrenamiento previo en desarrollo de pruebas unitarias*. Al parecer existe una fuerte correlación entre estas dos variables y la Calidad del código desarrollado. Sin embargo, la limitada cantidad de sujetos no permite alcanzar el poder estadístico suficiente como para que podamos hacer afirmaciones razonables.

Tabla 5.31: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Edad	-1.1	0.33	-3.53	0.41
Experiencia profesional	-5.09	0.38	**29.2**	**0.03**
Experiencia en Programación	-1.87	0.87	5.92	0.51
Experiencia en Programación Java	**4.61**	**0.06**	13.59	0.28
Conocimiento del Entorno Eclipse	20.98	0.97	-39.96	0.2
Conocimiento de Pruebas Unitarias	-11.88	0.39	-16.65	0.6

Tabla 5.32: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Edad	1.41	0.77	-2.36	0.18
Experiencia profesional	-5.73	0.76	**13.95**	**0.05**
Experiencia en Programación	-2.29	0.38	1.47	0.67
Experiencia en Programación Java	-1.23	0.2	7.21	0.1
Conocimiento del Entorno Eclipse	9.03	0.95	-18.82	0.16
Conocimiento de Pruebas Unitarias	8.51	0.66	-11.51	0.37

5.2.5. Grado de completitud

5.2.5.1. Aspectos generales

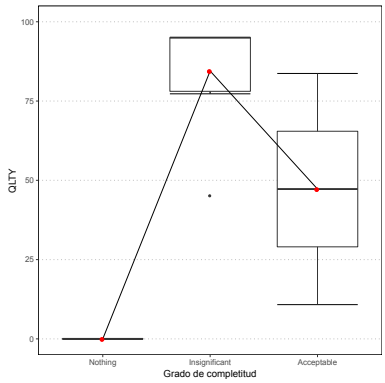
En esta sección analizamos manualmente el código entregado por los sujetos. Como en el caso de Quito2016, se utilizó una escala de Likert designando la calificación de *Nothing* a los sujetos que no hicieron nada de código, textitInsignificant para los que realizaron unas pocas líneas, y *Aceptable* para los que hicieron el trabajo como se esperaba.

En Tabla 5.145 se cuantifica el grado de completitud del trabajo realizado. De un total de 16 tareas recogidas, 6 sujetos no hicieron nada, 8 sujetos hicieron una mínima cantidad de trabajo y 2 sujeto realizó el trabajo como era esperado.

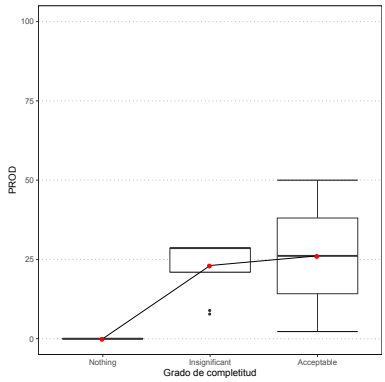
	Número de tareas entregadas
Nothing	6
Insignificant	8
Acceptable	2

Tabla 5.33: Grado de completitud

Podemos advertir que el comportamiento de los sujetos influye en la QLTY y PROD. A modo de ejemplo, el promedio de la QLTY de la primera sesión, incluidos los sujetos que no realizaron un trabajo sustancial, es del 54.69 %; cuando se excluyen estos sujetos, la QLTY aumenta a 83.7 %. El incluir las tareas vacías en el análisis de datos puede comprometer significativamente la interpretación de los resultados.



(a) Calidad



(b) Productividad

Figura 5.33: Relación entre el grado de completitud y la calidad y productividad

5.2.5.2. Impacto de la estrategia de programación y la tarea experimental

Podemos ver que la *estrategia de programación* no parece estar relacionada con el grado de finalización de las tareas, como se muestra en la tabla 5.146. Tanto la estrategia ITLD como TDD muestran el mismo número de tareas vacías y poco trabajadas. Solo un sujeto hizo la tarea de manera aceptable en la primera sesión experimental (ITLD).

	Nothing	Insignificant	Acceptable
ITLD	3	5	1
TDD	3	3	1

Tabla 5.34: Estrategia de programación

Mediante la realización de un test χ^2, podemos confirmar que no existe una relación entre la estrategia de programación y el grado de finalización de las tareas ($\chi^2 = 0{,}25$, $df = 2$, $p - value = 0{,}88$).

	Nothing	Insignificant	Acceptable
BSK	2	6	1
MR	4	2	1

Tabla 5.35: Grado de completitud por tarea experimental (BSK, MR)

El grado de completitud, por otra parte, parece tener cierta relación con las tareas. BSK es la tarea donde menos sujetos tienen un grado de completitud tipo *Nothing*. Sin embargo las diferencias entre BSK y MR no significativas como muestra el test χ^2 ($\chi^2 = 2{,}46$, $df = 2$, $p - value = 0{,}29$).

5.2.5.3. Impacto de las variables demográficas

En base al análisis realizado en la sección anterior, una vez más hemos podido notar que el grado de completitud de las tareas experimentales aparentemente no está directamente relacionado con los factores experimentales sino más bien con otros aspectos como las características personales de los sujetos. En las siguientes secciones analizaremos si el grado de completitud de las tareas se encuentra relacionado con las variables demográficas. Vamos a analizar las mismas variables demográficas de la sección 5.9.4, esto es:

- Edad.
- Experiencia profesional.
- Experiencia en programación.
- Experiencia en Java.

- Conocimiento del entorno Eclipse.
- Función actual en la organización.
- Entrenamiento previo en desarrollo de pruebas unitarias.

Edad

El gráfico tipo box-plot mostrado en la figura 5.140. muestra que la Edad tiene cierta relación con el grado de completitud de las tareas. Los sujetos de mayor edad fueron los que más presentaron tareas incompletas, con excepción del único sujeto que entregó la tarea de tipo *Acceptable*. El test Kruskal-Wallis entre la edad y el grado de completitud no arroja resultados significativos (p-valor = 0,45).

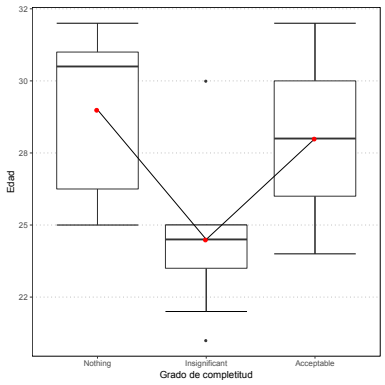


Figura 5.34: age

Experiencia profesional

La experiencia profesional no indica tener influencia en la presentación de las tareas. La figura 5.141 muestra una mayor dispersión para el grado de completitud *Insignificant*, con una mediana inferior a los dos años de experiencia. El test Kruskal-Wallis entre los años de experiencia y el grado de completitud arroja un p-valor = 0,41 no significativo y confirma lo observado en la gráfica 5.141.

Experiencia en programación

En la Fig. 5.142, se aprecia que los sujetos con mayor experiencia en programación son los que menos grado de completitud obtienen. Sin embargo, el test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,22, por lo que no podemos confirmar desde un punto de vista estadístico esta tendencia.

Experiencia en programación Java

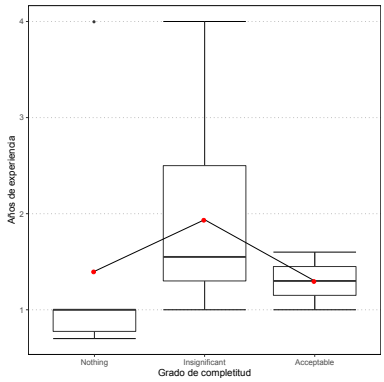


Figura 5.35: ExperienceYears

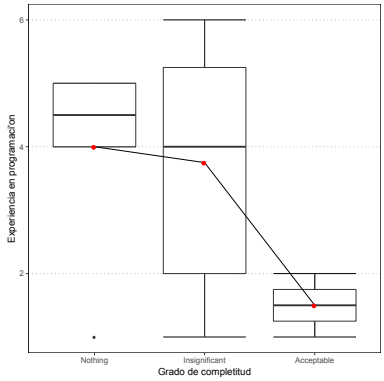


Figura 5.36: programmingExperience

La experiencia de programación con el lenguaje Java no influyó en la presentación de tareas incompletas (véase la figura 5.143). El test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud arroja un p-valor = 0,5, lo cual confirma la impresión visual obtenida.

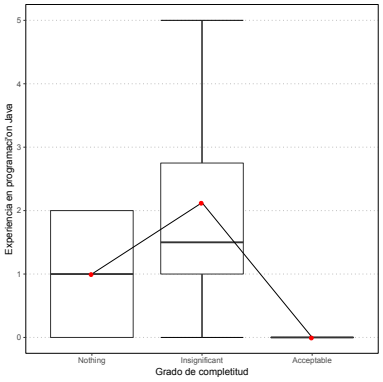


Figura 5.37: javaExperience

Conocimiento del entorno Eclipse

	Nothing	Insignificant	Acceptable
No	1	3	1
Yes	5	5	1

Tabla 5.36: Conocimiento del entorno eclipse

El conocimiento del entorno de desarrollo Eclipse no tiene influencia en el grado de completitud de las tareas experimentales. Los resultados del test χ^2 confirman este hecho ($\chi^2 = 1{,}07$, $df = 2$, $p - value = 0{,}59$).

Función actual en la organización

	Nothing	Insignificant	Acceptable
Another	2	3	0
Developer	3	5	1
Tester	1	0	1

Tabla 5.37: Funcion actual en la organización

En esta instancia experimental, la función que indicaron desempeñar los sujetos dentro de la organización tampoco presenta influencia en el grado de completitud de las tareas. Los resultados del test χ^2 no son significativos ($\chi^2 = 4{,}18$, $df = 4$, $p - value = 0{,}38$). Podemos advertir que el único

sujeto que obtuvo un grado de completitud *Acceptable*, se encontraba desempeñando la función de *Tester* dentro de la empresa, aunque este grado de completitud sólo lo alcanzó en una de las tareas experimentales. El limitado núemero de sujetos participantes no nos permite examinar de manera más objetiva estos resultados.

Entrenamiento previo en desarrollo de pruebas unitarias

	Nothing	Insignificant	Acceptable
No	2	7	1
Yes	4	1	1

Tabla 5.38: Entrenamiento previo en pruebas unitarias

El entrenamiento previo que algunos sujetos indicaron tener en el desarrollo de pruebas unitarias tampoco presenta influencia estadísticamente significativa en el grado de completitud de las tareas. El test χ^2 confirma este hecho ($\chi^2 = 4{,}44$, $df = 2$, $p - value = 0{,}11$).

5.2.5.4. Resumen general impacto variables demográficas

La tabla 5.151 resume los resultados de la influencia de las variables demográficas estudiadas frente al grado de completitud de las tareas realizadas. Como se puede observar, en todos los casos, los p-valores están lejos del nivel de significación. No obstante, el reducido tamaño muestral de esta instancia experimental no permite obtener mayores conclusiones al respecto.

Tabla 5.39: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Edad	0.45
Experiencia profesional	0.41
Experiencia en programación	0.22
Experiencia en Java	0.5
Conocimiento del entorno Eclipse	0.59
Función actual en la organización	0.38
Entrenamiento previo en desarrollo de pruebas unitarias	0.11

5.3. Experimento EcuadorESPE2015

5.3.1. Ejecución

ESPE2015 es el experimento base de la familia de experimentos realizado durante la presente investigación. El experimento fue realizado en la

Universidad de las Fuerzas Armadas ESPE de Ecuador con estudiantes de pregrado durante los periodos académicos regulares de clases comprendidos entre abril-agosto del 2015. Este experimento es el primero realizado en el ámbito académico.

5.3.1.1. Muestra

En el experimento ESPE2015 participaron 43 estudiantes del quinto y sexto grado de la Carrera de Ingeniería de Sistemas e Informática en la Universidad de las Fuerzas Armadas ESPE. La tabla 5.152 muestra que la totalidad de los sujetos son jóvenes (<30 años) y con poca experiencia en los diferentes aspectos considerados para este estudio. Como únicas excepciones, podemos indicar que:

- Aproximadamente el 70 % de los sujetos ha utilizado herramientas de prueba.
- Aproximadamente el 50 % de los mismos tiene experiencia en el entorno de desarrollo Eclipse.

5.3.1.2. Preparación

Para la realización de los experimentos se utilizaron los laboratorios de computación de la Universidad, donde se instalaron las herramientas de software necesarias para la realización de los experimentos; esto es: la máquina virtual de Java, el framework Junit, el plugin para empaquetar y realizar mediciones de los experimentos y el IDE de desarrollo Eclipse.

5.3.1.3. Realización

El experimento se realizó como parte de las asignaturas de Desarrollo de software y Programación Avanzada correspondientes al quinto y sexto nivel de la Carrera de Ingeniería en Sistemas e Informática respectivamente. El experimento se realizó en dos días a la semana durante tres semanas dentro del horario de clase de las asignaturas, con un horario de dos horas para el entrenamiento y dos horas con treinta minutos para la realización de las tareas experimentales. Se definió el siguiente protocolo de ejecución:

- **Semana 1:** Previo al inicio del entrenamiento los sujetos llenaron el cuestionario demográfico. En el día uno de la primera semana se realizó una sesión de introducción al desarrollo ágil y formación en pruebas unitarias utilizando el framework Junit. En el segundo día de la primera semana se capacitó en las técnicas ITLD y slicing.

Tabla 5.40: Características demográficas de los sujetos del experimento ESPE2015

EXPERIMENTO: ESPE2015		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	43
	30 >= Edad < 40 años	43
	40 >= Edad < 50 años	43
	Edad >= 50 años	43
Nivel de Educación	Other	0
	Undergraduate	43
	Bachelor	0
Experiencia profesional	Sin experiencia (<2 años)	43
	Novato (2 - 5 años)	43
	Intermedio (6 - 10 años)	43
	Experto (>10 años)	43
Experiencia en programación	Sin experiencia (<2 años)	3
	Novato (2 - 5 años)	36
	Intermedio (6 - 10 años)	4
	Experto (>10 años)	0
Uso de herramientas de pruebas	Yes	30
	No	13
Experiencia en lenguaje Java	Sin experiencia (<2 años)	5
	Novato (2 - 5 años)	36
	Intermedio (6 - 10 años)	2
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	38
	Novato (2 - 5 años)	5
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	5
	No	38
Experiencia en TDD	Sin experiencia (<2 años)	40
	Novato (2 - 5 años)	3
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	1
	No	42
Conocimiento del entorno Eclipse	Yes	20
	No	23
Función actual en la organización	Tester	0
	Developer	0
	Student	43

- **Semana 2:** En el primer día de la segunda semana se realizó la primera sesión experimental, que consistió en la solución de las tareas BSK o MR con o sin slicing y aplicando la técnica ITLD. Posteriormente, en el segundo día de la segunda semana, se realizó una primera sesión de entrenamiento en la técnica TDD.
- **Semana 3:** Durante el primer día de la tercera semana se realizó la segunda fase de capacitación en la técnica TDD. Finalmente, en el segundo día de la tercera semana, se realizó la segunda tarea experimental, en este caso, solucionar BSK o MR con o sin Slicing y aplicando la estrategia TDD.

La capacitación estuvo a cargo de Geovanny Raura con la colaboración de Rodrigo Fonseca, y se utilizaron los materiales de entrenamiento preparados por Oscar Dieste.

5.3.1.4. Desviaciones

En general, el protocolo se cumplió de acuerdo a lo establecido. Al utilizarse los horarios regulares de clases de los estudiantes, las horas de inicio y culminación de las actividades experimentales y de la capacitación se cumplieron dentro de lo establecido con un promedio de cinco minutos de retraso.

5.3.1.5. Reducción del conjunto de datos

Existieron ciertas variaciones en el número de sujetos que asistieron al experimento a lo largo de las tres semanas de duración del mismo. Se presentaron 43, y de ellos 42 sujetos realizaron la primera tarea experimental, es decir la aplicación de la estrategia ITLD. La segunda tarea experimental fue entregada por 33 sujetos quienes aplicaron la estrategia TDD. La reducción de algunos sujetos en la segunda tarea experimental puede explicarse en razón de que esta actividad fue optativa para quienes querían mejorar sus promedios en las asignaturas antes indicadas.

5.3.2. Estadísticos descriptivos

Se describen por separado cada una de las variables respuestas de Calidad y Productividad obtenidas.

5.3.2.1. Calidad

La tabla 5.153 muestra que la calidad es intermedia, con desviaciones estándar relativamente elevadas. La asimetría y la curtosis para ITLD y TDD son opuestas y no excesivamente altas; esto ayuda a que los residuos del modelo de análisis se aproximen más fácilmente a la normalidad.

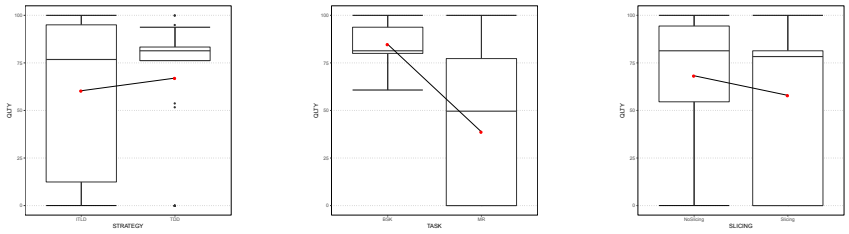
Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	60.36	39.27	-0.64	-1.26
TDD	67.00	33.47	-1.31	0.04

Tabla 5.41: Estadísticos descriptivos para QLTY

El box-plot mostrado en la figura 5.144 es probablemente más sencillo de comprender. Las medianas (y las medias, tal y como muestra la tabla 5.153), son muy parecidas, pero una mayor cantidad de sujetos obtiene calidades más altas usando TDD. Este hecho podría tener algún impacto en el análisis a favor de TDD.

Por su parte, la tarea BSK obtiene mayores valores de calidad que MR, tal y como indica la figura 5.144b. Esto sugiere que los resultados del análisis pueden resultar significativos para la tarea.

En este experimento ensayamos, además de los factores STRATEGY y TASK, el factor SLICING. El efecto de este factor puede observarse en la figura 5.144c. Como se aprecia, los sujetos que aplican slicing parece que obtienen mejores valores de Calidad que los sujetos que no lo aplican, aunque las diferencias no serán probablemente estadísticamente significativas.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.38: Box plots para la variable respuesta QLTY

5.3.2.2. Productividad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	20.39	24.26	1.42	1.53
TDD	45.05	30.02	-0.27	-1.48

Tabla 5.42: Estadísticos descriptivos para PROD

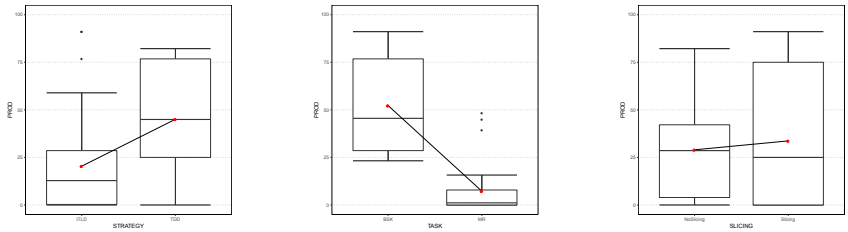
Podemos observar en la tabla 5.154 que los promedios de la variable Productividad son menores que los de Calidad. Esto implica que solo unas pocas historias de usuario fueron correctamente implementadas. La estrategia

TDD es superior a ITLD con más del doble de diferencia. Las desviaciones estándar son elevadas.

La asimetría y curtosis es positiva para el caso de ITLD, y negativa para el caso de TDD. Las dispersiones obtenidas son menores que en el caso de la Calidad, aunque no en gran medida al ser comparadas.

En la figura 5.145 se aprecia que, al igual que las medias, las medianas también difieren considerablemente, obteniendo un valor más alto para la estrategia TDD. Se aprecia que existe un impacto de la tarea para la Productividad: la media se encuentra muy superior para BSK en comparación con MR. Por otra parte, el nivel de definición de la tarea (slicing) parece no tener influencia sobre la Productividad.

En conclusión, se puede apreciar un posible efecto de la estrategia de programación, así como un posible efecto de la tarea para la Productividad.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.39: Box plots para la variable respuesta PROD

5.3.3. Prueba de hipótesis

5.3.3.1. Análisis estadístico

El análisis estadístico se ha realizado de igual modo que lo realizado en el experimento Babel2016. Los resultados de la ANOVA respecto a las variables respuesta Calidad y Productividad se muestran en la tabla 5.155. Los resultados son, en general, no significativos con dos excepciones:

- La estrategia de programación ha resultado significativa para la variable respuesta Productividad. No es necesario realizar un análisis post-hoc; la tabla 5.154 indica claramente que la Productividad obtenida con TDD es mayor que la obtenida con ITLD.
- Por otra parte, tal y como anticipábamos en la sección 5.10.2, la tarea arroja resultados significativos tanto para Calidad como Productividad. De nuevo, la tarea BSK obtiene mejores valores que MR.

Tabla 5.43: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	604.80	604.80	1.00	70.00	0.74	0.3914
TASK	39061.50	39061.50	1.00	70.00	48.03	0.0000
SLICING	3088.74	3088.74	1.00	70.00	3.80	0.0553
GROUP	7.22	7.22	1.00	70.00	0.01	0.9252

(b) Productividad

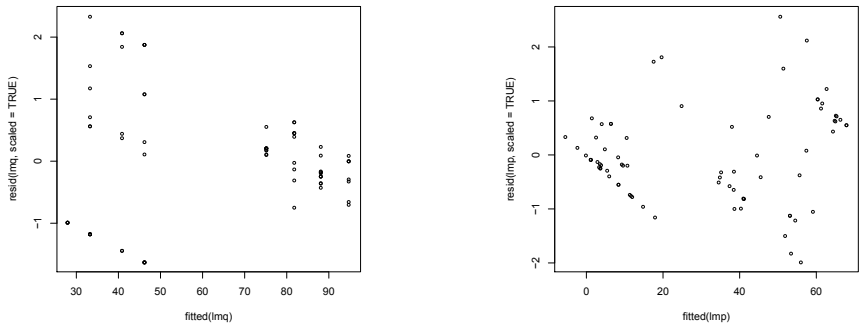
	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3355.23	3355.23	1.00	36.70	13.61	0.0007
TASK	29252.75	29252.75	1.00	36.78	118.65	0.0000
SLICING	395.75	395.75	1.00	37.18	1.61	0.2130
GROUP	191.23	191.23	1.00	40.94	0.78	0.3836

5.3.3.2. Chequeo del análisis estadístico

El chequeo del análisis estadístico sigue el mismo patrón que el experimento Babel 2016. Como se ha mencionado, de las tres asunciones dadas para los modelos mixtos (linealidad entre factores, homogeneidad de varianzas y distribución normal de residuos) comprobaremos la segunda y tercera condición ya que en este experimento al igual que en los restantes, los factores tienen solo dos niveles, por lo que la linealidad siempre se cumple.

Homogeneidad de varianzas

Los gráficos de valores predichos vs. residuos estandarizados se muestran en la figura **??**. No se aprecia la existencia de ninguna figura en embudo que sugiera heterogeneidad entre varianzas.



(a) Calidad (b) Productividad

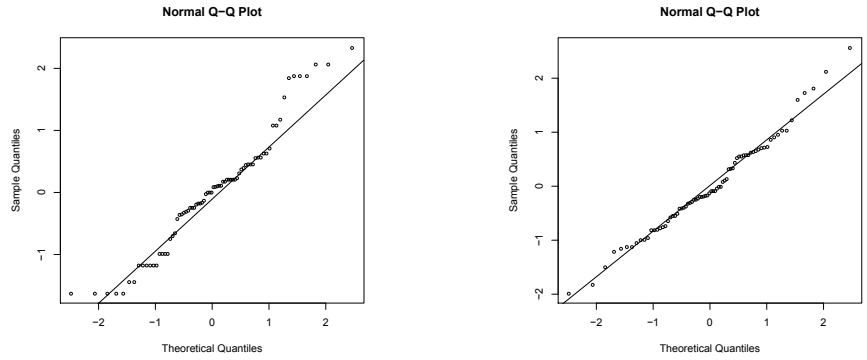
Figura 5.40: Chequeo de la homogeneidad de varianzas

Normalidad de residuos

Estudiamos tanto los factores fijos como los factores aleatorios. En lo que respecta a los factores fijos, podemos apreciar en la figura 5.147, que la distribución de los residuos de la variable respuesta PROD se solapan razonablemente con la distribución normal, representada en la línea recta diagonal. Como ha ocurrido en los experimentos anteriores, notamos que se producen desviaciones sólo en los extremos. En cuanto a la variable QLTY, podemos notar una gran desviación en los extremos.

El test de Shapiro-Wilks confirma la normalidad de los residuos para la Productividad ($W = 0{,}98$, $p - value = 0{,}37$). Sin embargo, en el caso de la Calidad, el test indica que los residuos no están distribuidos normalmente como ya lo habíamos advertido ($W = 0{,}96$, $p - value = 0{,}01$). Ninguna de las transformaciones aplicadas ha logrado alcanzar la normalidad. Adicionalmente, la asimetría y curtosis de los datos son en general mayor que 1. En consecuencia, los análisis referidos a la Calidad deben tomarse con cautela.

En relación a los efectos aleatorios, la situación es la misma que en el caso de los factores fijos: Normalidad para la Productividad ($W = 0{,}95$, $p - value = 0{,}07$) pero no para la Calidad ($W = 0{,}92$, $p - value = 0{,}01$).



(a) Calidad (b) Productividad

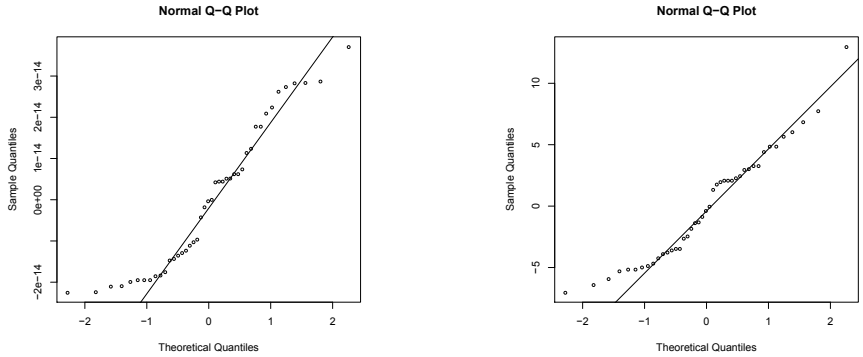
Figura 5.41: Gráficos QQ para los factores fijos

5.3.3.3. Influencia de factores instrumentales

En esta sección analizamos la dificultad de la tarea experimental de acuerdo a la percepción de los participantes. Como hemos advertido, las tareas fueron seleccionadas ad-hoc y por ende los resultados obtenidos podrían deberse al diseño experimental.

Tarea experimental

Utilizamos una escala de Likert de 1 a 5 puntos para calificar la comple-

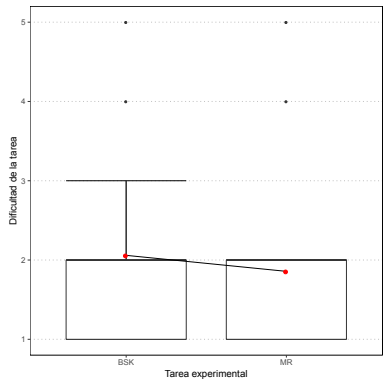


(a) Calidad (b) Productividad

Figura 5.42: Gráficos QQ para los factores aleatorios

jidad de la tarea, podemos observar en la figura 5.149a, que la complejidad percibida para MR es prácticamente similar a la complejidad de BSK, aunque el análisis estadístico indicó la existencia de diferencias significativas entre BSK-MR tanto para la variable QLTY como para la variable PROD. El test Kruskal-Wallis entre las tarea experimental y la dificultad percibida arroja un p-valor = 0,57, lo cual confirma la impresión visual obtenida en la figura 5.149a.

De acuerdo a la percepción de los sujetos, la dificultad de la tarea no juega un papel relevante, aun cuando estadísticamente existen diferencias significativas. En esta replicación los sujetos llenaron únicamente una de las dos encuestas post experimentales lo cual no nos permite explicar de forma categórica estos resultados.



(a) Dificultad percibida de la tarea

Figura 5.43: Efecto de la dificultad de la tarea percibida por los sujetos

5.3.4. Análisis de subgrupos

Conforme a los datos demográficos de los participantes (véase sección 5.10.1.1), podemos indicar que existen diferencias que merecen considerarse en los siguientes aspectos:

- Experiencia en programación.
- Experiencia en Java.
- Conocimiento del entorno Eclipse.
- Experiencia en el Framework JUnit.

5.3.4.1. Experiencia en programación

Tabla 5.44: Resultados del análisis estadístico para la Experiencia en programación

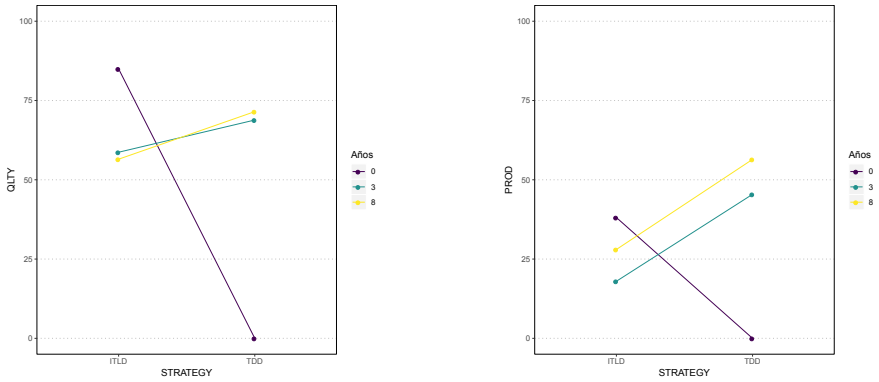
(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	724.71	724.71	1.00	68.00	0.87	0.3541
programmingExperience	36.25	36.25	1.00	68.00	0.04	0.8353
TASK	38024.53	38024.53	1.00	68.00	45.68	0.0000
SLICING	2891.07	2891.07	1.00	68.00	3.47	0.0667
GROUP	9.23	9.23	1.00	68.00	0.01	0.9165
STRATEGY:programmingExperience	309.95	309.95	1.00	68.00	0.37	0.5438

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	167.81	167.81	1.00	36.41	0.70	0.4077
programmingExperience	571.43	571.43	1.00	40.77	2.39	0.1299
TASK	28582.79	28582.79	1.00	35.33	119.48	0.0000
SLICING	456.39	456.39	1.00	35.52	1.91	0.1758
GROUP	130.77	130.77	1.00	39.76	0.55	0.4640
STRATEGY:programmingExperience	217.39	217.39	1.00	35.36	0.91	0.3469

El análisis de la experiencia en programación muestra una influencia negativa para las variable Calidad (B =-0.84) y positiva para la Productividad (B =1.03). Los efectos no son significativos en ambos casos ($P-valor$ =0.84 y $P-valor$ =0.13). No obstante, los gráficos 5.150a y 5.150b) sugieren que los programadores más experimentados mejoran al aplicar TDD tanto en la Calidad como en la Productividad (nótese las líneas amarillas y verdes). Los programadores nóveles fallan estrepitosamente al aplicar TDD. Hay que recalcar que con la introducción de la experiencia en programación, las variables respuesta dejan de ser significativas. Este es un hecho ya ocurrió en Quito2016 y nos permite advertir que ciertos fatores personales, puden



(a) Calidad (b) Productividad

Figura 5.44: Efecto de la experiencia en programación. La experiencia se reporta redondeada en años. Por ejemplo, el valor 0.5 representa 1 años de experiencia, mientras que 2.3 representa dos años de experiencia.

insidir en los resutados obtenidos. Esta tendencia la iremos corroborando durante el análisis de las siguientes instancias experimentales.

5.3.4.2. Experiencia en programación Java

Tabla 5.45: Resultados del análisis estadístico para la Experiencia en Java

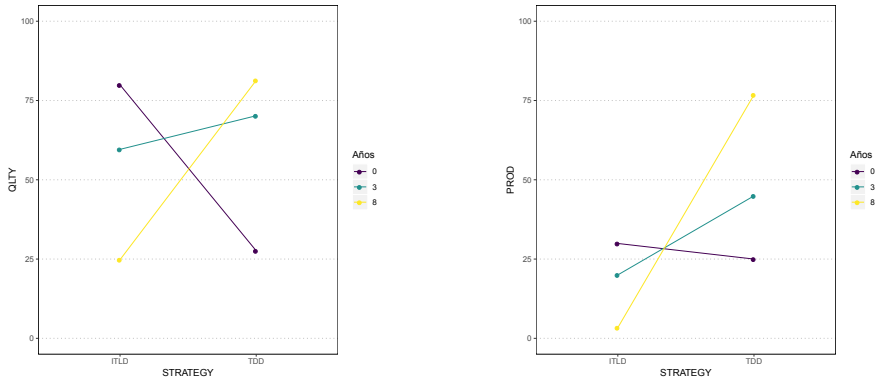
(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1240.14	1240.14	1.00	68.00	1.50	0.2248
javaExperience	22.88	22.88	1.00	68.00	0.03	0.8684
TASK	32801.42	32801.42	1.00	68.00	39.69	0.0000
SLICING	2845.06	2845.06	1.00	68.00	3.44	0.0679
GROUP	12.87	12.87	1.00	68.00	0.02	0.9011
STRATEGY:javaExperience	716.86	716.86	1.00	68.00	0.87	0.3550

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	228.88	228.88	1.00	32.87	0.91	0.3471
javaExperience	338.85	338.85	1.00	35.03	1.35	0.2537
TASK	25641.17	25641.17	1.00	35.26	101.93	0.0000
SLICING	426.57	426.57	1.00	36.11	1.70	0.2011
GROUP	67.71	67.71	1.00	39.26	0.27	0.6068
STRATEGY:javaExperience	191.54	191.54	1.00	31.59	0.76	0.3895

El uso del lenguaje de programación Java presenta una influencia negativa para la Calidad y positiva para la Productividad (B =-1.77 y B =0.66). Los efectos no son significativos ($P-value$ =0.87 y $P-value$ =0.25). En este



(a) Calidad (b) Productividad

Figura 5.45: Efecto de la experiencia en programación Java. La experiencia se reporta redondeada a años como en el caso anterior.

caso, el gráfico 5.151 muestra que los programadores con más experiencia en Java obtienen mejores resultados en cuanto a la Calidad y Productvidad al aplicar TDD. Los programadores nóveles fallan estrepitosamente al aplicar TDD.

5.3.4.3. Conocimiento del entorno Eclipse

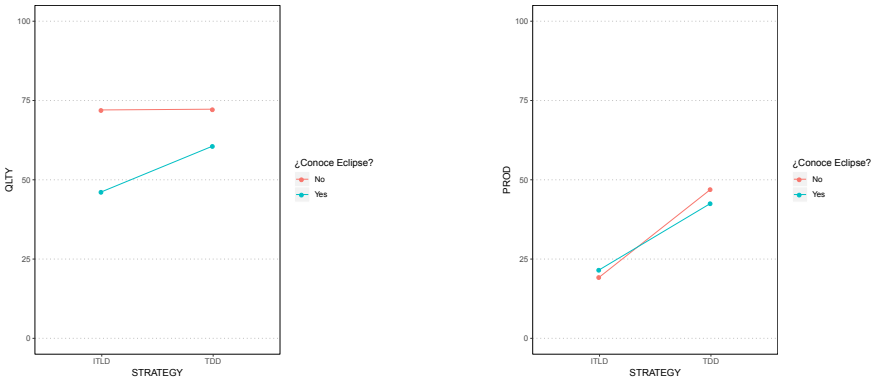
Tabla 5.46: Resultados del análisis estadístico para el conocimiento de entorno Eclipse

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	452.32	452.32	1.00	68.00	0.62	0.4331
knowEclipse	5819.64	5819.64	1.00	68.00	8.00	0.0061
TASK	38721.58	38721.58	1.00	68.00	53.23	0.0000
SLICING	2873.89	2873.89	1.00	68.00	3.95	0.0509
GROUP	4.00	4.00	1.00	68.00	0.01	0.9411
STRATEGY:knowEclipse	958.37	958.37	1.00	68.00	1.32	0.2551

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3134.28	3134.28	1.00	35.50	12.74	0.0011
knowEclipse	3.05	3.05	1.00	37.41	0.01	0.9119
TASK	29271.71	29271.71	1.00	35.35	118.97	0.0000
SLICING	423.01	423.01	1.00	35.63	1.72	0.1982
GROUP	184.61	184.61	1.00	39.64	0.75	0.3916
STRATEGY:knowEclipse	252.33	252.33	1.00	33.49	1.03	0.3185



(a) Calidad (b) Productividad

Figura 5.46: Efecto del conocimiento del entorno Eclipse.

El conocimiento del entorno de desarrollo Eclipse presenta una influencia estadísticamente significativa para la Calidad ($P - value$ =0.01), con un tamaño de efecto fuertemente negativo (B =-25.08). Como se aprecia en la figura 5.152a, los participantes que indicaron conocer el IDE Eclipse alcanzan menores calidades que los sujetos que no conocen Eclipse. En el caso particular de TDD, los sujetos que conocen Eclipse mejoran ostensiblemente la Calidad.

En lo que respecta a la variable Productividad, el conocimiento del IDE Eclipse posee una influencia positiva (B =3.27). Sin embargo, los efectos no son signficativos ($P - value$ =0.01). En este caso, todos los participantes obtuvieron una mejor Productividad al aplicar TDD, como se observa en la figura 5.152b.

5.3.4.4. Experiencia en el Framework JUnit

La experiencia en el Framework Junit presenta una influencia negativa tanto para la Calidad, como para la Productividad (B =-5.24 y B =-1.1). Sin embargo, los efectos no son significativos en ambos casos ($P - value$ =0.34 y $P - value$ =0.21). La interacción *STRATEGY*jUnitExperience* resulta significativa para la Productividad, no así para la Calidad. Esto indica que los programadores que aplican TDD obtienen B =8.13 puntos más que los que usan ITLD.

En las figuras 5.153a y 5.153b podemos notar que los participantes más experimentados con el Framework Junit, mejoran tanto en la Calidad como en la Productividad al utilizar TDD, al contrario de los menos experimentados.

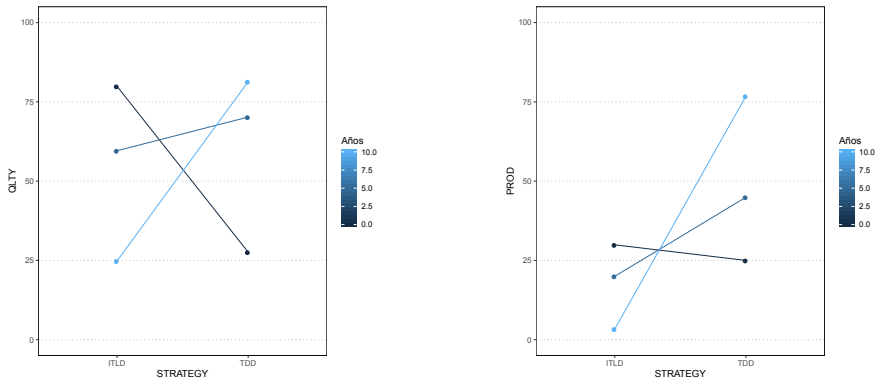
Tabla 5.47: Resultados del análisis estadístico para la Experiencia en el Framework JUnit

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	831.44	831.44	1.00	68.00	1.01	0.3177
jUnitExperience	749.05	749.05	1.00	68.00	0.91	0.3428
TASK	36103.54	36103.54	1.00	68.00	43.99	0.0000
SLICING	2961.00	2961.00	1.00	68.00	3.61	0.0617
GROUP	6.24	6.24	1.00	68.00	0.01	0.9307
STRATEGY:jUnitExperience	234.38	234.38	1.00	68.00	0.29	0.5948

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	2194.25	2194.25	1.00	34.07	9.84	0.0035
jUnitExperience	356.77	356.77	1.00	36.64	1.60	0.2139
TASK	24647.95	24647.95	1.00	35.22	110.51	0.0000
SLICING	599.03	599.03	1.00	35.55	2.69	0.1101
GROUP	54.70	54.70	1.00	40.38	0.25	0.6231
STRATEGY:jUnitExperience	1050.57	1050.57	1.00	32.60	4.71	0.0374



(a) Calidad (b) Productividad

Figura 5.47: Efecto de la experiencia en el Framework JUnit. La experiencia se reporta en lustros (5 años). Por ejemplo, el valor 0 representa 0-4 años de experiencia. El valor 5 representa 5-9 años de experiencia, etc.

5.3.4.5. Resumen general

En las tablas 5.160 y 5.161 mostramos un resumen de los resultados de los análisis de subgrupos realizados. Como se puede observar, el *Conocimiento del entorno de desarrollo Eclipse* es la única variable que tiene un impacto significativo sobre la Calidad, no así para la Productividad.

Tabla 5.48: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	-0.84	0.84	2.55	0.54
Experiencia en Programación Java	-1.77	0.87	4.31	0.35
Conocimiento del Entorno Eclipse	**-25.08**	**0.01**	14.48	0.26
Experiencia en Framework JUnit	-5.24	0.34	3.77	0.59

Tabla 5.49: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	1.03	0.13	2.18	0.35
Experiencia en Programación Java	0.66	0.25	2.25	0.39
Conocimiento del Entorno Eclipse	3.27	0.91	-7.56	0.32
Experiencia en Framework Junit	-1.1	0.21	**8.13**	**0.04**

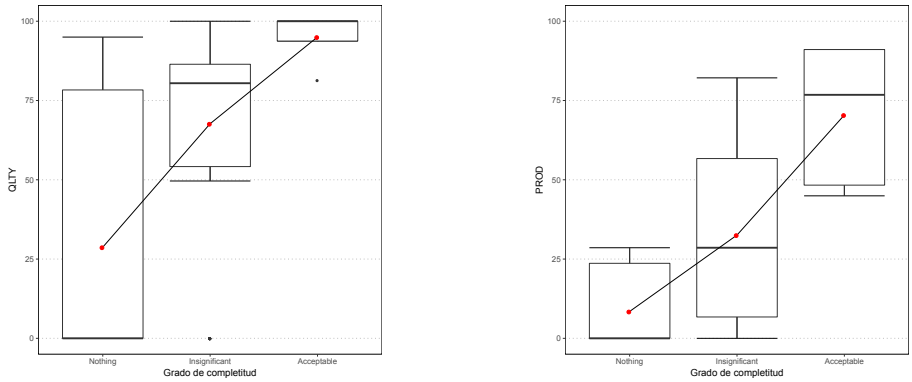
5.3.5. Grado de completitud

5.3.5.1. Aspectos generales

Analizamos manualmente el código entregado por los sujetos con las mismas consideraciones hechas para los anteriores experimentos. Observamos que algunos sujetos no hicieron ningún código (*Nothing*), la mayoría realizaron unas pocas líneas de código (*Insignificant*); y la menor parte obtuvo una calificación de *Aceptable*.

La relación entre el grado de completitud frente a las variables QLTY y PROD, se presentan en la figura: 5.154. Adicionalmente, en la tabla En Tabla 5.162 se cuantifica el grado de completitud del trabajo realizado.

De un total de 75 tareas recogidas, 12 sujetos no hicieron nada. Por otro lado, 58 hicieron una mínima cantidad de trabajo y 5 sujetos realizaron el



(a) Calidad (b) Productividad

Figura 5.48: Relación entre el grado de completitud y la calidad y productividad

trabajo como era esperado.

	Número de tareas entregadas
Nothing	12
Insignificant	58
Acceptable	5

Tabla 5.50: Grado de completitud

Si consideramos las tareas calificadas como *Nothing* en la medición del estudio, la QLTY y PROD disminuyen en promedio, como hemos venido observando en experimentos anteriores. A modo de ejemplo comparamos que el promedio de la QLTY de la segunda sesión, incluidos los sujetos que no realizaron un trabajo sustancial, es del 67 %; cuando se excluyen estos sujetos, la QLTY aumenta a 91.7 %. El hecho de incluir las tareas vacías compromete nuestro análisis significativamente.

5.3.5.2. Impacto de la estrategia de programación y la tarea experimental

La tabla 5.163, nos permite ver que al parecer la *estrategia de programación* no parece estar relacionada con el grado de finalización de las tareas. Existe el mismo número de sujetos que no hicieron nada tanto para TDD como para ITLD. Existe una mínima diferencia entre el núemero de sujetos que hicieron las tareas de manera aceptable y la estrategia de programación.

Podemos indicar que no existe relación entre la estrategia de programación y el grado de finalización de las tareas. Esto se comprueba mediante la realización de un test χ^2 que arroja los siguientes resultados: $\chi^2 = 0{,}86$,

	Nothing	Insignificant	Acceptable
ITLD	6	34	2
TDD	6	24	3

Tabla 5.51: Estrategia de programación

$df = 2$, $p - value = 0{,}65$.

	Nothing	Insignificant	Acceptable
BSK	4	33	3
MR	8	25	2

Tabla 5.52: Grado de completitud por tarea experimental (BSK, MR)

La tabla 5.164 muestra cierta relación entre el grado de completitud y las tareas. En este caso, el doble de sujetos no hicieron nada en la tarea MR, aunque casi el mismo número de sujetos entregaron las dos tareas de forma aceptable. Sin embargo, las diferencias entre BSK y MR no significativas como muestra el test χ^2 ($\chi^2 = 2{,}31$, $df = 2$, $p - value = 0{,}31$).

5.3.5.3. Impacto de las variables demográficas

En las siguintes secciones analizaremos si el grado de completitud de las tareas, se encuentra reacionado con las variables demográficas obtenidas de los sujetos experimentales. Vamos a analizar las mismas variables de la sección 5.10.4, esto es:

- Experiencia en programación.
- Experiencia en Java.
- Conocimiento del entorno Eclipse.
- Entrenamiento previo en desarrollo de pruebas unitarias.

Experiencia en programación

La experiencia en programación parece tener alguna relación con la presentación de tareas vacías, como se muestra en la Fig. 5.211. Podemos notar que los sujetos más experimentados, en promedio hicieron un trabajo más aceptable. El test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,32, que no resulta estadísticamente significativo.

Experiencia en programación Java

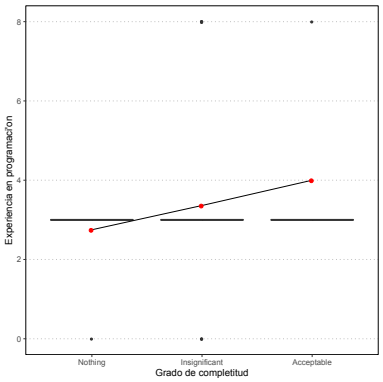


Figura 5.49: Experiencia en programación

La experiencia en desarrollo con el lenguaje Java, al igual que la experiencia en programación, al parecer influye en el grado de presentación de las tareas (véase la figura 5.212). Aunque el test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud arroja un p-valor = 0,23 el cual no resulta estadísticamente significativo.

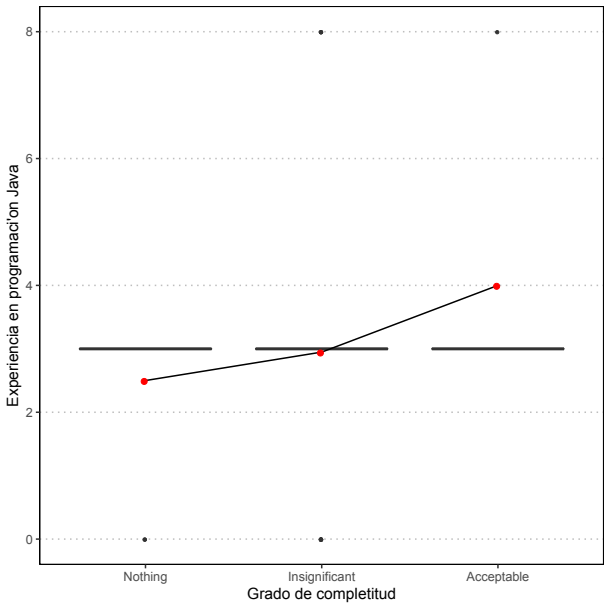


Figura 5.50: javaExperience

Conocimiento del entorno Eclipse

El conocimiento del entorno de desarrollo Eclipse no tiene influencia en el grado de completitud de las tareas experimentales. El test χ^2 nos indica

	Nothing	Insignificant	Acceptable
No	7	32	2
Yes	5	26	3

Tabla 5.53: Conocimiento del entorno eclipse

este hecho ($\chi^2 = 0{,}51$, $df = 2$, $p - value = 0{,}78$).

Función actual en la organización

	Nothing	Insignificant	Acceptable
No	11	57	5
Yes	1	1	0

Tabla 5.54: Entrenamiento previo en pruebas unitarias

El entrenamiento previo en el desarrollo de pruebas unitarias tampoco presentan influencia en el grado de completitud de las tareas. El test χ^2 confirma este hecho ($\chi^2 = 1{,}82$, $df = 2$, $p - value = 0{,}4$). Por lo tanto no es significativo.

5.3.5.4. Resumen general impacto variables demográficas

La tabla 5.167 resume los resultados de los análisis de la influencia de las variables demográficas estudiadas frente al grado de completitud de las tareas realizadas. Ninguna de las variables ejercen un impacto que sea estadísticamente significativo.

Tabla 5.55: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Experiencia en programación	0.32
Experiencia en Java	0.23
Conocimiento del entorno Eclipse	0.78
Entrenamiento previo en desarrollo de pruebas unitarias	0.4

5.4. Experimento MexicoUADY2015

5.4.1. Ejecución

El experimento UADY2015 es una replicación del experimento base. El experimento fue realizado en la Universidad Autónoma de Yucatán de México con estudiantes de pregrado como parte de un evento académico orga-

nizado por la Universidad. Este es el segundo experimento realizado en el ámbito académico con similares características al experimento ESPE2015.

5.4.1.1. Muestra

En el experimento UADY2015 participaron 35 estudiantes que se encontraban cursando (excepto por un estudiante de maestría) el cuarto, quinto, sexto y séptimo grados en la Facultad de Licenciatura en Ciencias de la Computación de la Universidad Autónoma de Yucatán de México (UADY). En la tabla 5.168 se observa que todos los participantes son jóvenes menores de 30 años sin experiencia profesional. Las diferencias mas notables de este grupo de sujetos que podemos destacar es que aproximadamente el 40 % de los mismos tiene experiencia en el entorno de desarrollo Eclipse. Casi todos los participantes indicaron tener experiencia tanto en programación como en lenguaje Java en el rango de 2 a 5 años. En los otros aspectos, el grupo es bastante homogéneo excepto por unos pocos estudiantes que no superan el 10 % en promedio.

5.4.1.2. Preparación

De forma similar a los anteriores experimentos realizados, se instalaron previamente las herramientas de software necesarias; esto es: la máquina virtual de Java, el framework Junit, el plugin para empaquetar y realizar mediciones de los experimentos y el IDE de desarrollo Eclipse. Se utilizó uno de los laboratorios de la Universidad para la realización de las sesiones de entrenamiento y de los experimentos.

5.4.1.3. Realización

Esta instancia experimental se realizó como parte de un evento académico denominado Jornadas de Ingeniería de Software FMAT2015, organizado por la Facultad de Ciencias de la Computación de la UADY. Tanto las sesiones de entrenamiento como las sesiones experimentales se realizaron como un curso taller de 5 días con un horario de cuatro horas por día. El protocolo seguido se detalla a continuación:

- **Día 1:** Previo al inicio del entrenamiento los sujetos llenaran el cuestionario demográfico. Posteriormente se realizó una sesión de introducción al desarrollo ágil y formación en pruebas unitarias utilizando el framework Junit. Los estudiantes tuvieron un receso de 15 minutos, luego de lo cual se les pidió realizar un ejercicio práctico como tarea de control.
- **Día 2:** Durante el segundo día se realizó una fase de capacitación en la técnica de slicing e Incremental Test Last Development (ITLD).

Tabla 5.56: Características demográficas de los sujetos del experimento UADY2015

EXPERIMENTO: UADY2015		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	35
	30 >= Edad < 40 años	35
	40 >= Edad < 50 años	35
	Edad >= 50 años	35
Nivel de Educación	Other	0
	Undergraduate	34
	Bachelor	0
Experiencia profesional	Sin experiencia (<2 años)	35
	Novato (2 - 5 años)	35
	Intermedio (6 - 10 años)	35
	Experto (>10 años)	35
Experiencia en programación	Sin experiencia (<2 años)	0
	Novato (2 - 5 años)	31
	Intermedio (6 - 10 años)	3
	Experto (>10 años)	1
Uso de herramientas de pruebas	Yes	5
	No	30
Experiencia en lenguaje Java	Sin experiencia (<2 años)	2
	Novato (2 - 5 años)	33
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	33
	Novato (2 - 5 años)	2
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	2
	No	33
Experiencia en TDD	Sin experiencia (<2 años)	35
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	3
	No	32
Conocimiento del entorno Eclipse	Yes	13
	No	22
Función actual en la organización	Tester	0
	Developer	0
	Student	35

Posteriormente hubo un receso de 15 minutos seguido por la realización de una segunda tarea de control.

- **Día 3:** Se realizó la primera sesión experimental que consistió en la solución de las tareas BSK o MR con o sin Slicing y aplicando la técnica ITLD. Posteriormente, se realizó un receso de 15 minutos y se concluyó la sesión con un feedback para resolver las inquietudes surgidas.

- **Día 4:** Se realizó la segunda fase de capacitación en la técnica TDD. De igual forma, se tuvo un receso de 15 minutos y se concluyó con una tarea de control para reforzar los conocimientos adquiridos.

- **Día 5:** Finalmente se realizó la segunda tarea experimental, en este caso solucionar BSK o MR con o sin Slicing y aplicando la estrategia TDD. Se cerró la sesión con un receso de 15 minutos y un feedback para resolver las inquietudes de los participantes.

La capacitación estuvo a cargo de Geovanny Raura con la colaboración de Rodrigo Fonseca y se utilizaron los materiales de entrenamiento preparados por Oscar Dieste.

5.4.1.4. Desviaciones

No existieron desviaciones en cuanto al protocolo establecido. En general se cumplieron las horas de inicio y culminación de las actividades experimentales y de la capacitación con un promedio de cinco minutos de retraso, aunque se observó que unos pocos estudiantes no llegaron puntuales o abandonaron antes de concluir el tiempo previsto en razón de que pudieron resolver con mayor rapidez las tareas.

5.4.1.5. Reducción del conjunto de datos

Las variaciones en el número de sujetos que asistieron al experimento a lo largo de las tres semanas de duración del mismo fueron mínimas en este caso. Se presentaron 35 y de ellos todos realizaron la primera tarea experimental, es decir la aplicación de la estrategia ITLD. La segunda tarea experimental fue entregada por 33 sujetos quienes aplicaron la estrategia TDD. Se pudo notar que dos sujetos ya no asistieron a partir del tercer día de capacitación, aunque no se pudo determinar las razones.

5.4.2. Estadísticos descriptivos

Se describen por separado cada una de las variables respuestas de Calidad y Productividad obtenidas.

5.4.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	80.59	21.63	-1.66	3.23
TDD	77.63	24.67	-1.81	3.22

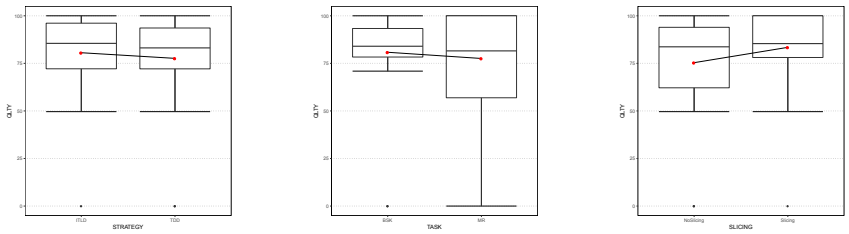
Tabla 5.57: Estadísticos descriptivos para QLTY

En la tabla 5.169 podemos notar que los promedios para las estrategias ITLD y TDD son bastante parecidos, con apenas un 3 % de diferencia de ITLD sobre TDD. En ambos casos, las desviaciones estándar son algo elevadas. Las asimetrías son negativas en ambos casos, en tanto que la curtosis es positiva con valores no tan elevados.

En los gráficos de box-plot mostrado en la figura 5.157 apreciamos la similitud en el valor de las medianas para las dos estrategias. Podemos ver que existe una distribución de puntos bastante homogénea tanto para ITLD como para TDD (representados por los puntos azules). En este caso no parece haber diferencias significativas.

Con respecto a la tarea, BSK nuevamente obtiene mayores valores de calidad que MR, tal y como indica la figura 5.157b, aunque la diferencia no es tan marcada como ha ocurrido en otras instancias experimentales. Es muy probable que no encontremos diferencias significativas para la tarea.

El efecto del factor SLICING puede observarse en la figura 5.157c. No está claro si este factor influye significativamente en los resultados obtenidos, aunque el gráfico permite apreciar que los sujetos que aplican slicing obtienen un ligero mayor nivel de calidad.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.51: Box plots para la variable respuesta QLTY

5.4.2.2. Productividad

En la tabla 5.170 se muestran los promedios de la variable Productividad. Podemos notar que las dos estrategias bordean el 50 % aunque con apenas 5 puntos de diferencia para TDD sobre ITLD. Las desviaciones estándar son

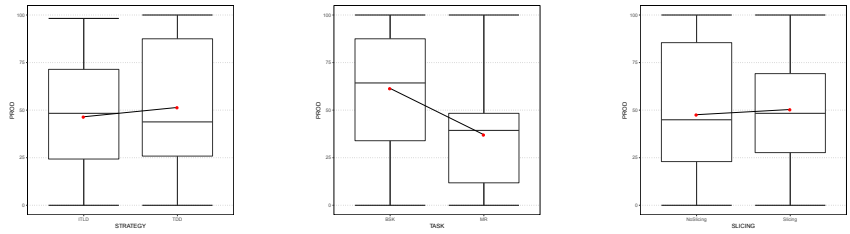
Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	46.51	29.64	0.14	-1.19
TDD	51.41	32.88	0.15	-1.32

Tabla 5.58: Estadísticos descriptivos para PROD

más elevadas que las obtenidas para la Calidad y son relativamente altas, con asimetrías positivas y curtosis negativas relativamente bajas.

En cuanto a la estrategia de desarrollo, la figura 5.158a es parecida a la de Calidad (5.157a), aunque la estrategia ITLD tiene una mediana ligeramente superior (aunque su promedio es ligeramente inferior como ya habíamos señalado). Por otro lado, en la figura 5.158b se aprecia que existe un impacto de la tarea; siendo una vez más la tarea BSK donde los sujetos obtienen un mayor valor de Productividad. Finalmente, el nivel de definición de la tarea (slicing) no parece influir en esta variable (5.158c).

En síntesis, en esta instancia experimental, al parecer no existen efectos significativos para la Calidad y un posible efecto de la tarea para la Productividad.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.52: Box plots para la variable respuesta PROD

5.4.3. Prueba de hipótesis

5.4.3.1. Análisis estadístico

El análisis estadístico se ha realizado de igual modo que en experimentos anteriores (a excepción de Quito2016). Los resultados de la ANOVA respecto a las variables respuesta Calidad y Productividad se muestran en la tabla 5.171. La tarea arroja resultados significativos para la Productividad, como ya lo habíamos indicado en la sección 5.11.2. En cuanto a la variable Slicing, resulta que es significativo para la Calidad. Este es un resultado que no lo veíamos tan claro en los gráficos de box plot de la sección anterior.

Tabla 5.59: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>
STRATEGY	162939929844133.69	162939929844133.69	1.00	299592.53	0.28	0.59
TASK	43626182056077.64	43626182056077.64	1.00	283455.99	0.08	0.78
SLICING	3618305112631899.00	3618305112631899.00	1.00	606.28	6.27	0.01
GROUP	12478565907476.67	12478565907476.67	1.00	31.19	0.02	0.88

(b) Productividad

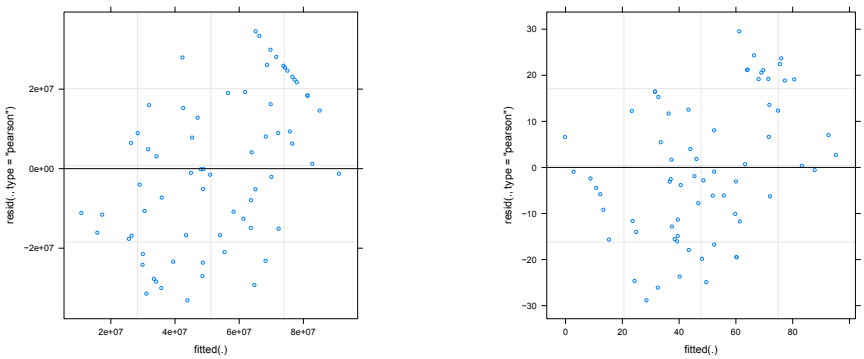
	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	389.62	389.62	1.00	30.83	1.05	0.3145
TASK	9491.55	9491.55	1.00	30.86	25.47	0.0000
SLICING	584.52	584.52	1.00	44.79	1.57	0.2169
GROUP	200.18	200.18	1.00	32.74	0.54	0.4688

5.4.3.2. Chequeo del análisis estadístico

Realizamos el chequeo del análisis estadístico siguiendo el mismo patrón de todos lo experimentos anteriores (excepto Quito2016 que tiene ciertas diferencias). Comprobaremos la homogeneidad de varianzas y normalidad de los residuos ya que los factores tienen solo dos niveles por lo que la linealidad siempre se cumple.

Homogeneidad de varianzas

En la figura 5.159 se muestran los gráficos de valores predichos vs. residuos estandarizados. Al parecer se cumple la homogeneidad de varianzas ya que no se observan patrones de embudo.



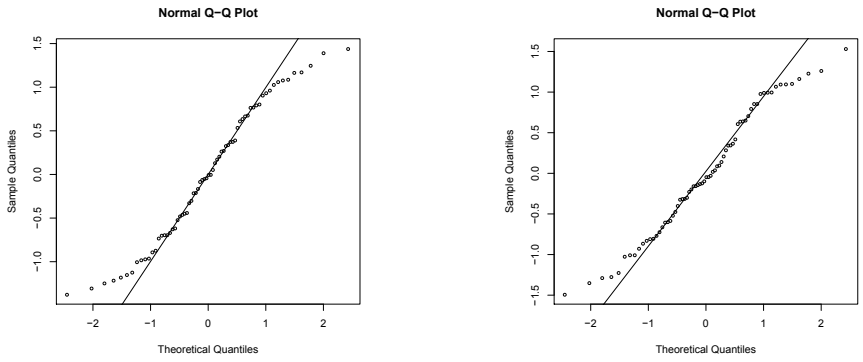
(a) Calidad (b) Productividad

Figura 5.53: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

Normalidad de residuos

Analizamos los factores fijos para determinar la normalidad de los residuos. En lo que respecta a los factores fijos, podemos apreciar en la figura 5.160, que la distribución de los residuos de la variable respuesta PROD se solapan razonablemente con la distribución normal, representada en la línea recta diagonal, con desviaciones ligeras en los extremos. El test de Shapiro-Wilks confirma la normalidad de los residuos para la productividad ($W = 0{,}97$, $p - value = 0{,}11$)

La distribución de los residuos de la variable QLTY se parece mucho a la variable PROD, pero el test Shapiro-Wilks rechaza la normalidad de los residuos. Con la finalidad de conseguir normalidad, hemos aplicado una transformación Box-Cox $\lambda = 4$, un tanto forzada, pero que mejora ostensiblemente la normalidad de los datos. El test de Shapiro-Wilks se sitúa en el límite de la no normalidad ($W = 0{,}96$, $p - value = 0{,}04$), pero este resultado es mucho mejor que el de la variable sin transformar. Nos damos por satisfechos.



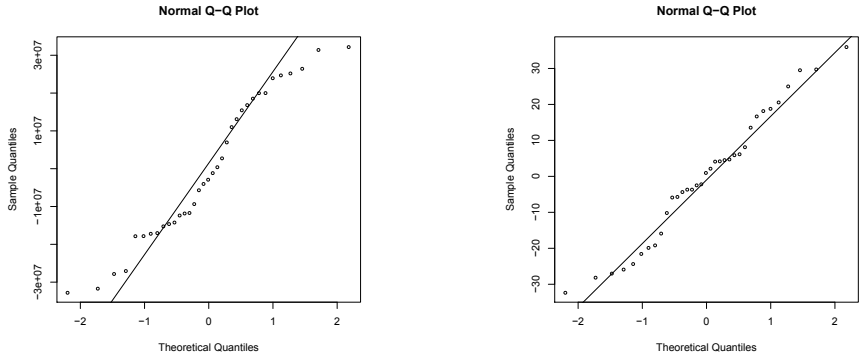
(a) Calidad (b) Productividad

Figura 5.54: Gráficos QQ para los factores fijos

En lo que respecta a los factores aleatorios, el gráfico QQ de la figura 5.161 muestra una situación aparentemente peor que la anterior. Sin embargo, el test de Shapiro-Wilks confirma la normalidad de los efectos aleatorios tanto para la Calidad ($W = 0{,}95$, $p - value = 0{,}09$) como para la Productividad ($W = 0{,}97$, $p - value = 0{,}5$).

5.4.3.3. Influencia de factores instrumentales

Como hemos procedido en otros experimentos, en esta sección analizamos la dificultad de la tarea experimental percibida por los sujetos para descartar una posible influencia de la tarea ocasionada por el diseño del experimento.

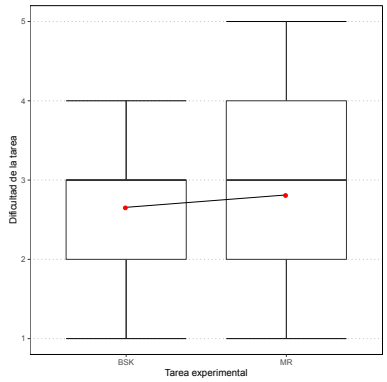


(a) Calidad (b) Productividad

Figura 5.55: Gráficos QQ para los factores aleatorios

Tarea experimental

En la figura 5.162a, apreciamos que la complejidad percibida para las dos tareas es muy similar, aunque la dispersión de datos para la tarea MR es mucho mayor. El chequeo del análisis estadístico arrojó diferencias significativas de la tarea en la Productividad. Sin embargo, este análisis indica que no fue percibido así por los sujetos. El test Kruskal-Wallis entre las tarea experimental y la dificultad percibida arroja un p-valor = 0,48, que confirma lo observado en la figura 5.162a.



(a) Dificultad percibida de la tarea

Figura 5.56: Efecto de la dificultad de la tarea percibida por los sujetos

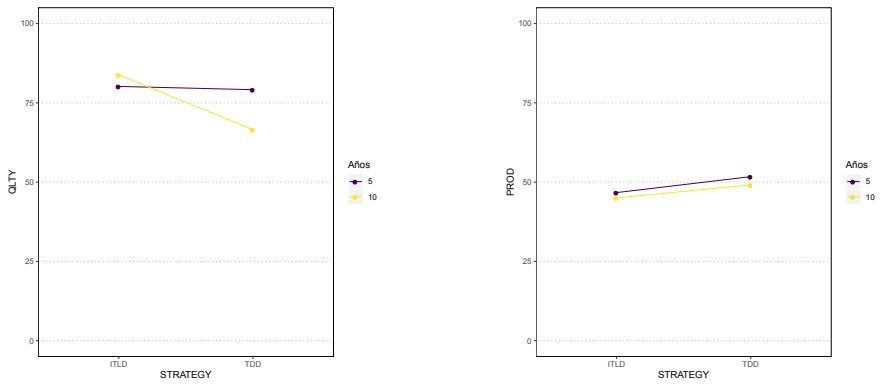
5.4.4. Análisis de subgrupos

En este experimento hemos considerado los siguientes aspectos que creemos son relevantes conforme a los datos demográficos de los participantes

(véase sección 5.11.1.1):

- Experiencia en programación.
- Uso de herramientas de pruebas.
- Conocimiento del entorno Eclipse.
- Grado de estudios de los participantes.

5.4.4.1. Experiencia en programación

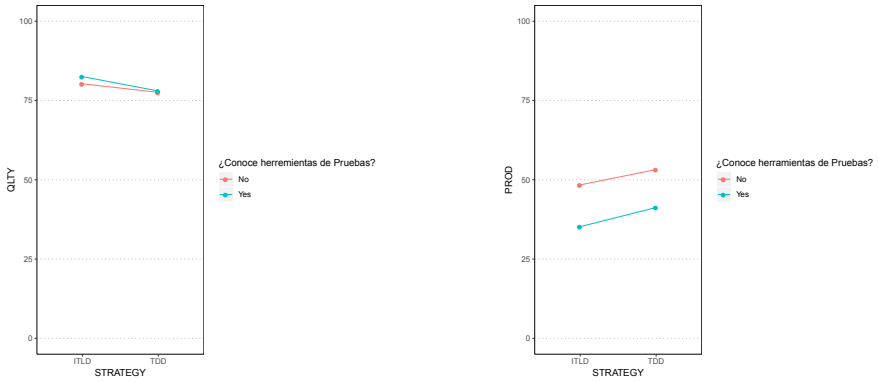


(a) Calidad (b) Productividad

Figura 5.57: Efecto de la experiencia en programación. La experiencia se reporta redondeada a lustros.

En lo que respecta a la experiencia en programación, el análisis muestra una influencia positiva para la Calidad ($B = 42.59$) y para la Productividad ($B = 1.54$). En los dos casos los efectos no son significativos ($p - value = 0.65$ y $p - value = 0.96$) respectivamente. No incluímos la tabla de análisis en razón de que los resultados no son significativos. Procederemos de igual forma en los siguientes análisis.

Como puede obeservarse en la figura 5.163, los desarrolladores más experimentados obtienen mejores resultados de Calidad al aplicar la estrategia ITDD en comparación con TDD. En contraste, los desarrolladores menos experimentados prácticamente no presentan diferencias de Calidad al aplicar ITLD o TDD. La Productividad de los desarrolladores tiene una cierta mejora al aplicar TDD, independientemente de los años de experiencia en programación (nótese que las líneas se encuentran prácticamente adyacentes y son paralelas).



(a) Calidad (b) Productividad

Figura 5.58: Efecto del uso de herramientas de pruebas.

5.4.4.2. Uso de herramientas de pruebas

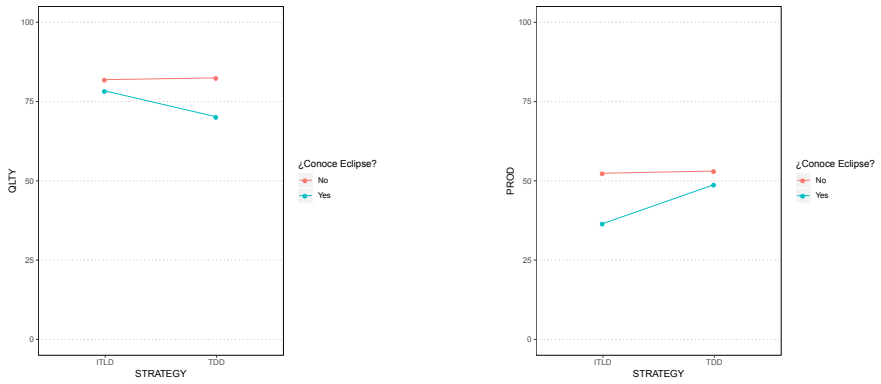
En cuanto al uso de herremientas de pruebas, el análisis muestra un efecto positivo para las variable Calidad (B =40.62) y negativo para la Productividad (B =-10.96). Sin embargo los efectos no son significativos en ambos casos ($P-valor$ =0.76 y $P-valor$ =0.3). La figura 5.164, confirma que no existe influencia del conocimiento del uso de herramientas de prueba tanto en la Calidad como en la Productividad al aplicar TDD o ITLD. Como se puede notar, las líneas son prácticamente paralelas en ambos casos, aunque para la Productividad, los que no conocen de herramientas de prueba, tienen un mejor desempeño.

5.4.4.3. Conocimiento del entorno Eclipse

El análisis del conocimiento del entorno de desarrollo Eclipse indica una influencia negativa para la Calidad y Productividad (B =-47.82 y B =-7.3). Los efectos tampoco son significativos ($P-value$ =0.7 y $P-value$ =0.43). En la figura 5.165, se puede observar que la Calidad y Productividad alcanzada por los desarrolladores que no conocían Eclipse, es independiente de la técnica utilizada (nótese que la línea roja prácticamete no tiene pendiente). En cambio, los desarrolladores que sí conocían Eclipse, tienen la tendencia a disminuir la Calidad al aplicar TDD, mientras que mejoran su Productividad y están por encima de los que si conocen el IDE Eclipse.

5.4.4.4. Grado de estudios de los participantes

En este experimento resulta de interés analizar el grado de estudios de los participantes y su influencia en la Calidad y Productividad. Como hemos indicado en secciones anteriores, los participantes fueron estudiantes que se



(a) Calidad (b) Productividad

Figura 5.59: Efecto del conocimiento del entorno Eclipse.

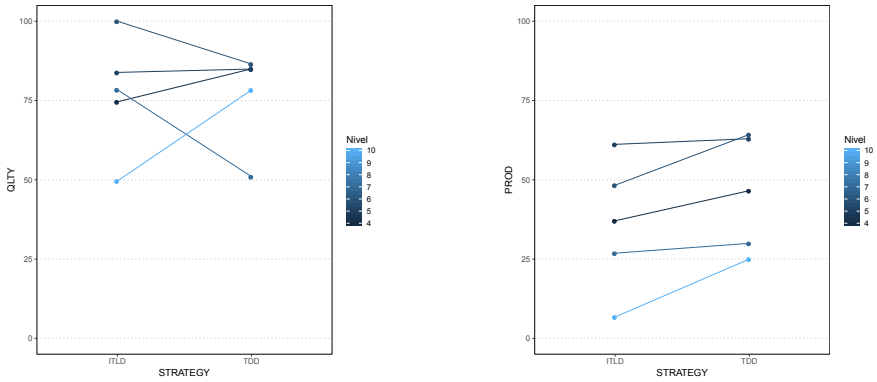
Tabla 5.60: Resultados del análisis estadístico para nivel de estudios de los participantes

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	44177142964770.48	44177142964770.48	1.00	30800.70	0.07	0.7875
Seniority	2196939485300452.00	2196939485300452.00	1.00	29.59	3.61	0.0670
TASK	89799451459131.64	89799451459131.64	1.00	116819.23	0.15	0.7007
SLICING	3402708782554296.50	3402708782554296.50	1.00	413.79	5.60	0.0184
GROUP	16571686749292.16	16571686749292.16	1.00	29.81	0.03	0.8700
STRATEGY:Seniority	90238820063996.69	90238820063996.69	1.00	36215.06	0.15	0.7000

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	159.75	159.75	1.00	30.76	0.41	0.5260
Seniority	1601.21	1601.21	1.00	31.35	4.12	0.0509
TASK	9678.24	9678.24	1.00	29.88	24.92	0.0000
SLICING	534.13	534.13	1.00	46.11	1.38	0.2469
GROUP	69.75	69.75	1.00	31.57	0.18	0.6746
STRATEGY:Seniority	65.80	65.80	1.00	30.62	0.17	0.6835



(a) Calidad (b) Productividad

Figura 5.60: Efecto del nivel de estudios de los participantes.

encontraban cursando distintos grados de la Carrera en Licienciatura en Informática de la UADY. Específicamente fueron estudiantes matriculados de cuarto a séptimo nivel de formación de la Carrera, con excepción de un único participante que se encontraba cursando una maestría. La tabla 5.173 muestra la distribución de sujetos participantes de acuerdo al nivel de estudios; 4S indica que el estudiante se encontraba cursando el cuarto semestre, 5S en el quinto semestre y así sucesivamente:

Tabla 5.61: Distribución de sujetos por nivel de estudios

	Nivel de estudios				
	4S	5S	6S	7S	MSc
Número de sujetos	8	17	2	7	1

El análisis muestra una influencia negativa del nivel de estudios de los participantes tanto para la Calidad como para la Productividad (B =-49.94 y B =-5.83). En este caso, los efectos se acercan al nivel de significación para la Calidad ($P - value$ =0.07 y son significativos para la Productividad $P - value$ =0.05).

Aunque el efecto del nivel de estudios en términos prácticos es estadísticamente significativo, sin embargo, el efecto es opuesto a lo esperado. Como se aprecia en la figura 5.166, los estudiantes de los más altos niveles obtienen una menor Calidad y Productividad. Nótese que las líneas de los estudiantes de niveles inferiores se encuentran en el cuadrante superior al 75 % para la Calidad y sobre el 35 % para la Productividad, por sobre los estudiantes de últimos niveles que apenas superan el 75 % en la Calidad y no sobrepasan el 30 % de Productividad. La tendencia en general es que los estudiantes independientemente del nivel mejoran la Productividad al aplicar TDD. En lo que respecta a la Calidad, el patrón no es el mismo, en algunos casos existe

mejora al aplicar ITLD y en otros al aplicar TDD.

Las tendencias observadas deben tomarse con cautela, dado que los grupos de sujetos no están balanceados en cada uno de los niveles. Por ejemplo, tenemos un único sujeto que se encontraba cursando el nivel más alto (maestría), y la mayor porción de sujetos se encontraban cursando el quinto semestre de la carrera.

5.4.4.5. Resumen general

Los resultados del análisis de subgrupos se muestran en las tablas 5.174 y 5.175. No se observan diferencias significativas entre las variables demográficas estudiadas con respecto a la Calidad y Productividad. El *Nivel de estudios de los participantes* sin embargo, ejerce influencia estadísticamente significativa para la Productividad y se acerca al nivel de significación para el caso de la Calidad.

Tabla 5.62: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	42.59	0.65	-45.31	0.15
Uso de herramientas de pruebas	40.62	0.76	-61.8	0.38
Conocimiento del Entorno Eclipse	-47.82	0.7	36.03	0.9
Grado de estudios de los participantes	**-49.94**	**0.07**	-36.91	0.7

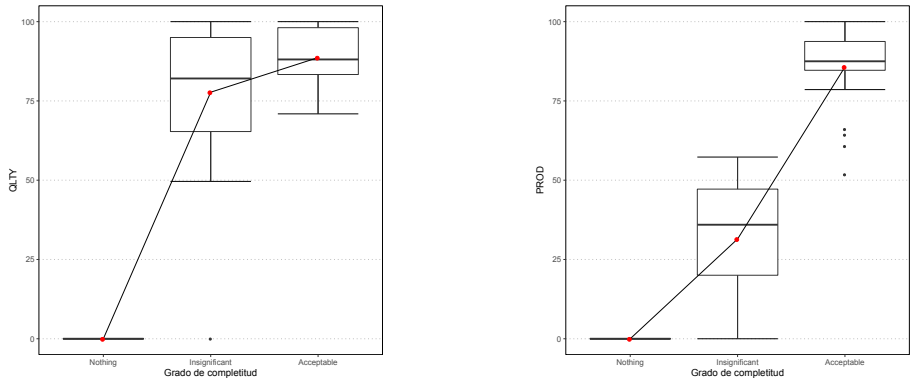
Tabla 5.63: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	1.54	0.96	-3.33	0.17
Uso de herramientas de pruebas	-10.96	0.3	-4.39	0.75
Conocimiento del Entorno Eclipse	-7.3	0.43	-0.63	0.95
Grado de estudios de los participantes	**-5.83**	**0.05**	-1.59	0.68

5.4.5. Grado de completitud

5.4.5.1. Aspectos generales

Siguiendo el mismo protocolo de análisis de experimentos anteriores, analizamos manualmente el código entregado por los sujetos. En este caso sólo un par de sujetos no hicieron ningún código (*Nothing*), la mayoría realizaron unas pocas líneas de código (*Insignificant*); y algunos lo hicieron de forma *Aceptable*.



(a) Calidad (b) Productividad

Figura 5.61: Relación entre el grado de completitud y la calidad y productividad

En la tabla 5.176 podemos apreciar que de un total de 68 tareas recogidas, 2 sujetos no hicieron nada, 43 hicieron una mínima cantidad de trabajo y 23 sujetos realizaron el trabajo adecuadamente.

	Número de tareas entregadas
Nothing	2
Insignificant	43
Acceptable	23

Tabla 5.64: Grado de completitud

En la figura 5.167 mostramos la relación entre el grado de completitud y las variables Calidad y Productividad. Comparamos la QLTY de la primera y segunda sesión incluyendo y excluyendo al grupo de sujetos que hizo las tareas calificadas como *Nothing* e *Insignificant* y obtenemos los siguientes promedios: Sesión ITLD: 80.59 %. Sesión TDD: 77.63 % Si tomamos en cuenta únicamente a los sujetos que hicieron un trabajo calificado como *Acceptable*, obtenemos los siguientes promedios: Sesión ITLD: 85.94 %. Sesión TDD: 91.15 %.

Como ha ocurrido en experimentos anteriores, el hecho de incluir a los sujetos que no hicieron nada y los sujetos que tampoco hicieron un trabajo aceptable, influye en los resultados del análisis de manera sustancial.

5.4.5.2. Impacto de la estrategia de programación y la tarea experimental

Cuando analizamos la cantidad de trabajo realizado por los sujetos en relación a la estrategia de programación, al parecer no existen diferencias significativas; el porcentaje de sujetos se distribuye casi de manera equitativa como se muestra en la tabla 5.177).

	Nothing	Insignificant	Acceptable
ITLD	1	23	11
TDD	1	20	12

Tabla 5.65: Estrategia de programación

Este hecho se comprueba mediante la realización de un test χ^2 que arroja resultados no significativos: $\chi^2 = 0{,}19$, $df = 2$, $p - value = 0{,}91$.

	Nothing	Insignificant	Acceptable
BSK	1	13	19
MR	1	30	4

Tabla 5.66: Grado de completitud por tarea experimental (BSK, MR)

En cuanto a la relación entre el grado de completitud y la tarea, en la tabla 5.178 podemos apreciar que existen diferencias marcadas para los sujetos calificados como *Nothing* y *Acceptable*. El test χ^2 muestra diferencias significativas entre las tareas MR y BSK ($\chi^2 = 16{,}46$, $df = 2$, $p - value = 0$).

5.4.5.3. Impacto de las variables demográficas

Analizamos si el grado de completitud de las tareas se encuentra reacionado con las variables demográficas obtenidas de los sujetos experimentales. Consideramos las mismas variables de la sección 5.11.4:

- Experiencia en programación.
- Uso de herramientas de pruebas.
- Conocimiento del entorno Eclipse.
- Grado de estudios de los participantes.

Experiencia en Programación

La experiencia en programación no influye en el grado de presentación de las tareas (véase la figura 5.168). Podemos notar que aunque la mayoría de sujetos indicó tener experiencia menor a cinco años, la tendencia es que los más experimentados entregaron las tareas como se esperaba, aunque los resultados no son significativos como demuestra el test Kruskal-Wallis entre la experiencia en programación y el grado de completitud: p-valor = 0,66.

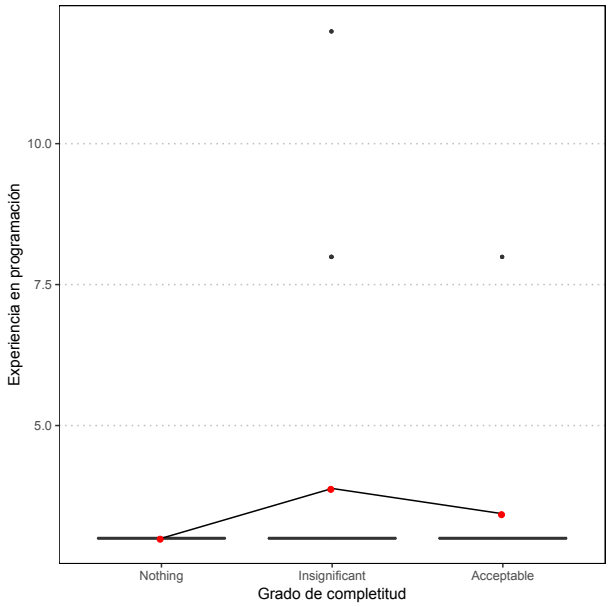


Figura 5.62: programmingExperience

Uso de herramientas de pruebas

	Nothing	Insignificant	Acceptable
No	2	35	21
Yes	0	8	2

Tabla 5.67: Uso de herramientas de prueba

Como se aprecia en la tabla 5.179, unos pocos sujetos que si tenían conocimiento de herramientas de pruebas entregaron las tareas de forma aceptable, pero la mayor parte que indicó no haber usado herramientas de pruebas también entregó las tareas de manera aceptable. Al parecer no existe influencia del grado de completitud de las tareas y el uso de herrmientas de pruebas. El test χ^2 confirma este hecho: $\chi^2 = 1{,}53$, $df = 2$, $p - value = 0{,}47$.

Conocimiento del entorno Eclipse

	Nothing	Insignificant	Acceptable
No	1	25	16
Yes	1	18	7

Tabla 5.68: Conocimiento del entorno eclipse

El conocimiento del entorno de desarrollo Eclipse tampoco tiene influencia en el grado de completitud de las tareas experimentales. El test χ^2 arroja resultados no significativos ($\chi^2 = 0{,}95$, $df = 2$, $p - value = 0{,}62$).

Grado de estudios de los participantes

El grado de estudios de los participantes al parecer tiene influencia en el grado de completitud de las tareas. En la Fig. 5.169 podemos apreciar que los sujetos con un nivel de estudios inferior, realizaron las tareas experimentales de manera aceptable. El test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,01 arroja resultados significativos.

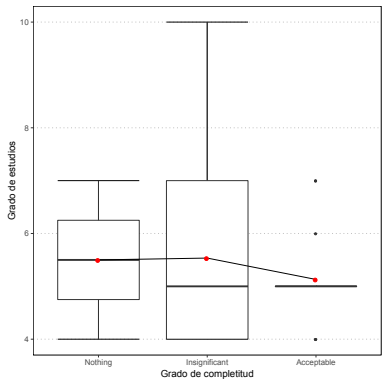


Figura 5.63: Seniority

5.4.5.4. Resumen general impacto variables demográficas

La tabla 5.181 resume los resultados de los análisis de la influencia de las variables demográficas estudiadas frente al grado de completitud de las tareas realizadas. Como se puede ver, el *Grado de estudios* arroja resultados significativos. Los pocos sujetos que declararon tener mayor experiencia (entre 5 y 8 años) entregaron las tareas de manera aceptable, al parecer influyen en los resultados.

Tabla 5.69: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Experiencia en programación	0.66
Conocimiento del entorno Eclipse	0.62
Uso de herramientas de pruebas	0.47
Grado de estudios	**0.01**

5.5. Experimento EcuadorESPE2016

5.5.1. Ejecución

El experimento ESPE2016, al igual que el experimento ESPE2015, fue realizado en la Universidad de las Fuerzas Armadas ESPE de Ecuador con estudiantes de pregrado. El experimento se lo ejecutó durante el mes de febrero del 2016 dentro del periodo académico regular de clases. Esta replicación puede ser catalogada como literal (ya que se asemeja tanto como ha sido posible a ESPE2015), conjunta (ya que participaron los mismos investigadores) e interna (ya que fue realizada en el mismo sitio).

5.5.1.1. Muestra

El experimento ESPE2016 es de tipo académico y participaron 15 estudiantes del quinto y sexto grado de la Carrera de Ingeniería de Sistemas e Informática en la Universidad de las Fuerzas Armadas ESPE. En la tabla 5.182 se muestran los datos demográficos de los sujetos. La totalidad de sujetos son jóvenes menores de 30 años, sin mayor experiencia profesional. Los únicos aspectos que los podrían diferenciar son:

- Un 80 % de los sujetos indica tener entre 2 y 5 años de experiencia en programación; dos sujetos indicaron tener entre seis y 10 años de experiencia en programación y un solo sujeto indicó no tener experiencia programando.
- Un 20 % de los sujetos indica tener menos de dos años de experiencia en Java; un 74 % se considera novato y un único sujeto indicó tener entre 6 y 10 años de experiencia en Java.
- Aproximadamente el 20 % de sujetos ha usado la técnica TDD, conoce el entorno de desarrollo Eclipse y ha utilizado herramientas de pruebas.

5.5.1.2. Preparación

Para el experimento ESPE2016, al igual que para el experimento ESPE2015, se utilizaron los laboratorios de computación de la Universidad

Tabla 5.70: Características demográficas de los sujetos del experimento ESPE2016

EXPERIMENTO: ESPE2016		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	15
	30 >= Edad < 40 años	15
	40 >= Edad < 50 años	15
	Edad >= 50 años	15
Nivel de Educación	Other	0
	Undergraduate	15
	Bachelor	0
Experiencia profesional	Sin experiencia (<2 años)	15
	Novato (2 - 5 años)	15
	Intermedio (6 - 10 años)	15
	Experto (>10 años)	15
Experiencia en programación	Sin experiencia (<2 años)	1
	Novato (2 - 5 años)	12
	Intermedio (6 - 10 años)	2
	Experto (>10 años)	0
Uso de herramientas de pruebas	Yes	3
	No	12
Experiencia en lenguaje Java	Sin experiencia (<2 años)	3
	Novato (2 - 5 años)	11
	Intermedio (6 - 10 años)	1
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	15
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	3
	No	12
Experiencia en TDD	Sin experiencia (<2 años)	15
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	0
	No	15
Conocimiento del entorno Eclipse	Yes	3
	No	12
Función actual en la organización	Tester	0
	Developer	0
	Student	15

configurados con las mismas herramientas de software usadas en todos los experimentos.

5.5.1.3. Realización

El experimento se realizó como parte de la asignatura de Desarrollo de software que se dicta en el quinto nivel de la Carrera de Ingeniería en Sistemas e Informática. El experimento se realizó en similares condiciones que ESPE2015, esto es, en dos días a la semana durante tres semanas dentro del horario de clase de las asignaturas, con un horario de dos horas para el entrenamiento y dos horas con treinta minutos para la realización de las tareas experimentales. El entrenamiento estuvo a cargo de Geovanny Raura con la colaboración de Rodrigo Fonseca y bajo la supervisión de Oscar Dieste. El protocolo seguido es el mismo empleado para el experimento ESPE2015 por lo que no lo hemos incluido en esta sección.

5.5.1.4. Desviaciones

No existieron desviaciones sustanciales en el protocolo. Las sesiones se cumplieron de acuerdo al cronograma establecido dentro del horario de clases de los estudiantes con no más de cinco minutos de retraso.

5.5.1.5. Reducción del conjunto de datos

En este caso no existieron variaciones en cuanto al número de sujetos. Se presentaron 15estudiantes y todos realizaron tanto la primera tarea experimental utilizando la técnica ITLD, como la segunda tarea utilizando la técnica TDD.

5.5.2. Estadísticos descriptivos

Las variables respuestas de Calidad y Productividad obtenidas en este experimento se describen a continuación:

5.5.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	59.29	42.47	-0.48	-1.70
TDD	59.43	42.29	-0.55	-1.67

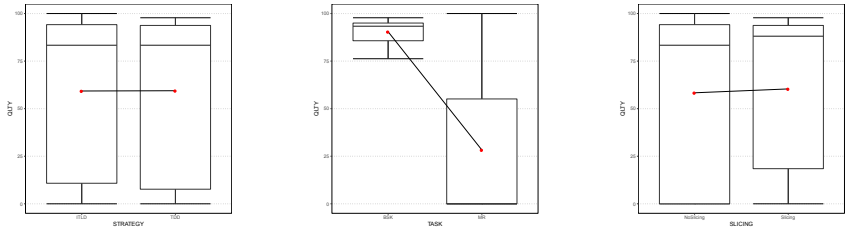
Tabla 5.71: Estadísticos descriptivos para QLTY

Como se indica en la tabla 5.183 los promedios así como la desviación estándar son prácticamente idénticas, con cierta diferencia en las cifras decimales. La asimetría y la curtosis para ITLD y TDD son negativas y no

excesivamente altas, lo que indica que la cola de la distribución se alarga para valores inferiores a la media aunque no muy cercanos a la misma.

En los gráficos de box-plot de la figura 5.170 podemos observar también similitud tanto en las medianas como en el tamaño de las cajas para ITLD y TDD.

En cuanto a la tarea experiemental, una vez más observamos en la figura 5.170b que BSK obtiene resultados superiores en Calidad que MR y al parecer con diferencias significativas. Por otra parte, el factor SLICING, no parece influir en la Calidad obtenida, como puede observarse en la figura 5.170c, aunque existe una mayor dispersión para la tarea con grado de definición No-Slicing.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.64: Box plots para la variable respuesta QLTY

5.5.2.2. Productividad

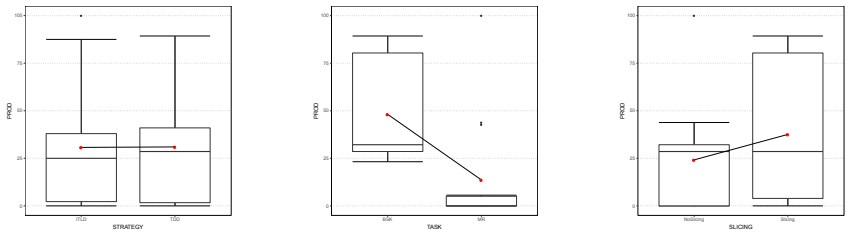
Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	30.74	33.65	0.86	-0.71
TDD	30.99	32.25	0.68	-1.00

Tabla 5.72: Estadísticos descriptivos para PROD

En lo que respecta a la Productividad, obtenemos promedios y desviaciones estándar similares para las dos estrategias (ver tabla 5.184). Podemos notar también que los promedios son menores que los de la variable Calidad con casi el doble. Esto implica que solo unas pocas historias de usuario fueron correctamente implementadas. La asimetría en este caso es positiva y la curtosis es negativa (platicúrtica) para ambas estrategias, lo que indica que la cola de la distribución se alarga (hacia la derecha) para valores superiores a la media y hay una menor concentración de datos entorno a la media.

Estos resultados se aprecian con mayor detalle en la figura 5.171. Podemos observar una gran similitud entre los box plot de Productividad para ITLD y para TDD. Una vez más se aprecia que existe un impacto de la

tarea para la Productividad, estando BSK por sobre MR. El nivel de definición de la tarea (Slicing) al parecer tampoco es significativo, aunque existen diferencias sustanciales en cuanto a la dispersión de datos.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.65: Box plots para la variable respuesta PROD

5.5.3. Prueba de hipótesis

5.5.3.1. Análisis estadístico

Mantenemos la misma estrategia de análisis estadístico que en los experimentos anteriores. En tabla **??** se muestran los resultados de la ANOVA tanto para la Calidad como para la Productividad. Nuevamente los resultados son significativos para la Tarea, pero no para ningún otro factor, como ya anticipábamos en los estadísticos descriptivos.

Tabla 5.73: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	130.15	130.15	1.00	12.02	0.16	0.6968
TASK	29071.90	29071.90	1.00	12.02	35.57	0.0001
SLICING	43.39	43.39	1.00	12.02	0.05	0.8216
GROUP	107.19	107.19	1.00	13.01	0.13	0.7230

(b) Productividad

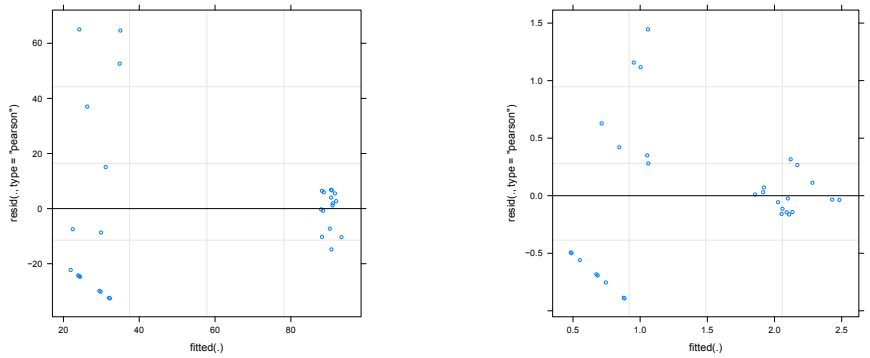
	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.04	0.04	1.00	12.00	0.09	0.7736
TASK	12.62	12.62	1.00	12.00	28.47	0.0002
SLICING	0.28	0.28	1.00	12.00	0.63	0.4431
GROUP	0.15	0.15	1.00	13.00	0.33	0.5739

5.5.3.2. Chequeo del análisis estadístico

Analizamos la homogeneidad de varianzas y la distribución normal de los residuos bajo las mismas consideraciones que en los anteriores experimentos. Remarcamos que los factores tienen solo dos niveles por lo que la linealidad siempre se cumple.

Homogeneidad de varianzas

En los gráficos de valores predichos vs. residuos estandarizados (5.172a) no se aprecia la existencia de ninguna figura en embudo que sugiera heterogeneidad entre varianzas.



(a) Calidad (b) Productividad

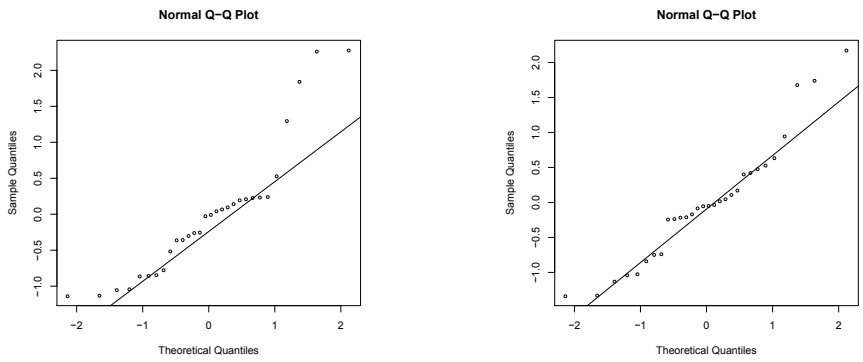
Figura 5.66: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

Normalidad de residuos

Los datos sin transformar resultan no normales al aplicar el test Shapiro-Wilks a los residuos de los efectos fijos, así como también a los residuos de los efectos aleatorios. Después de probar varias transformaciones, hemos conseguido normalizar la variable PROD usando una transformación Box-Cox $\lambda = -5$. Tal y como podemos observar en las figuras 5.173 y 5.174, los puntos se aproximan a la línea diagonal, aunque la coincidencia dista de ser perfecta. El test de Shapiro-Wilks confirma la impresión visual para los residuos fijos ($W = 0{,}94$, $p - value = 0{,}09$) y aleatorios ($W = 0{,}89$, $p - value = 0{,}06$).

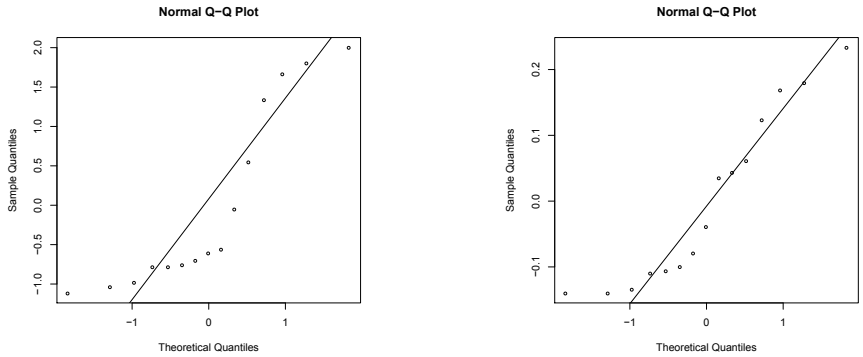
En el caso de la Calidad, ha sido imposible alcanzar algo parecido a la normalidad independientemente de la transformación aplicada. Nótese como los puntos se separan de la línea diagonal en las figuras 5.173 y 5.174. En términos del test Shapiro-Wilks, tanto los residuos ($W = 0{,}87$, $p-value = 0$) como los factores aleatorios ($W = 0{,}81$, $p-value = 0$) no resultan normales.

Tampoco podemos apoyarnos en los valores de asimetría y curtosis, los cuales son bastante elevados. En consecuencia, los resultados de los análisis de la Calidad deben tomarse con cautela.



(a) Calidad (b) Productividad

Figura 5.67: Gráficos QQ para los factores fijos



(a) Calidad (b) Productividad

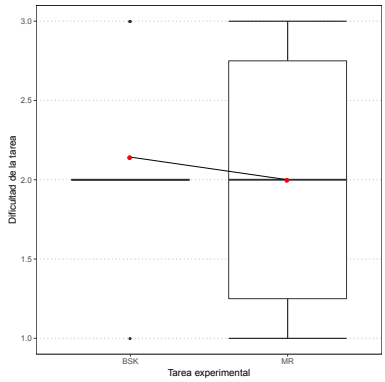
Figura 5.68: Gráficos QQ para los factores aleatorios

5.5.3.3. Influencia de factores instrumentales

En esta sección analizamos la dificultad de la tarea experimental percibida por los sujetos. Como hemos comentado en anteriores experimentos, las tareas fueron seleccionadas ad-hoc y por ende los resultados obtenidos podrían deberse al diseño experimental.

Tarea experimental

La figura 5.175a sugiere que la dificultad de las tareas experimentales MR y BSK es prácticamente similar. El test Kruskal-Wallis entre la tarea experimental y la dificultad percibida arroja un p-valor = 0,61, el cual no es significativo. En consecuencia, no parece que la dificultad de la tarea produzca las diferencias significativas entre BSK-MR para QLTY y PROD reportadas en la tabla 5.185.



(a) Dificultad percibida de la tarea

Figura 5.69: Efecto de la dificultad de la tarea percibida por los sujetos

5.5.4. Análisis de subgrupos

De acuerdo a los datos demográficos de los participantes mostrados en la sección 5.12.1.1, los aspectos que se pueden considerar en el análisis son la experiencia en Programación y la experiencia en Java. En cuanto a otros factores como el conocimiento del entorno Eclipse (que se incluye en este análisis), el uso de la técnica TDD y el uso de herramientas de pruebas, apenas un 20 % de sujetos presentan diferencias entre subgrupos. En el resto de aspectos preguntados, no existen diferencias entre sujetos que puedan ser usadas para el análisis.

5.5.4.1. Experiencia en programación

La experiencia en programación tiene una influencia negativa para la Calidad y es casi nula para la Productividad (B =-2.64 B =0). Los efectos no son significativos ($P-value$ =0.6 y $P-value$ =0.18). Existe una tendencia a mejorar notablemente tanto la Calidad como la Productividad cuando aplicaron TDD los desarrolladores menos experimentados (con menos de dos años de experiencia en programación). Los desarrolladores más experimentados prácticamente mantienen la misma Calidad y Productividad, como se aprecia en la figura 5.176.

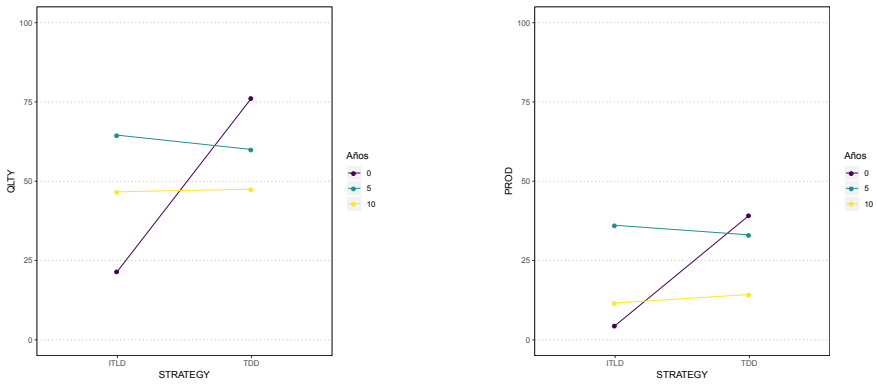
Tabla 5.74: Resultados del análisis estadístico para la Experiencia en Programación

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	196.28	196.28	1.00	11.00	0.22	0.6464
programmingExperience	251.32	251.32	1.00	12.00	0.28	0.6033
TASK	28784.16	28784.16	1.00	11.00	32.62	0.0001
SLICING	112.83	112.83	1.00	11.00	0.13	0.7274
GROUP	71.88	71.88	1.00	12.00	0.08	0.7802
STRATEGY:programmingExperience	101.79	101.79	1.00	11.00	0.12	0.7405

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.01	0.01	1.00	11.00	0.02	0.8958
programmingExperience	1.00	1.00	1.00	12.00	2.06	0.1768
TASK	12.25	12.25	1.00	11.00	25.33	0.0004
SLICING	0.21	0.21	1.00	11.00	0.43	0.5233
GROUP	0.09	0.09	1.00	12.00	0.18	0.6798
STRATEGY:programmingExperience	0.00	0.00	1.00	11.00	0.00	0.9888



(a) Calidad (b) Productividad

Figura 5.70: Efecto de la experiencia en programación. La experiencia se reporta redondeada a lustros.

5.5.4.2. Experiencia en programación Java

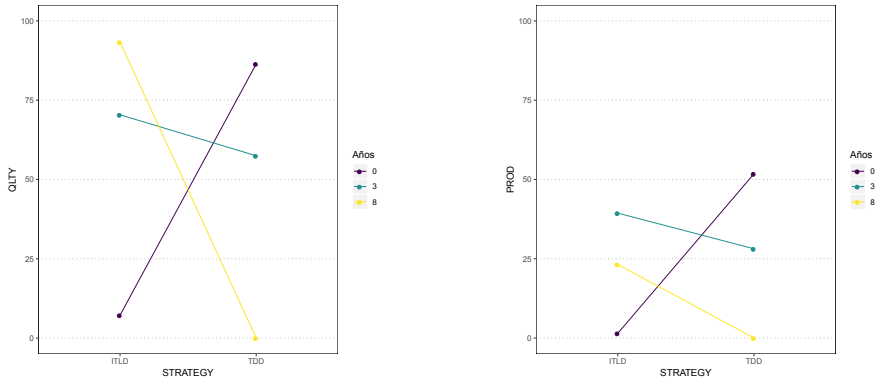
Tabla 5.75: Resultados del análisis estadístico para la Experiencia en Java

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	673.76	673.76	1.00	11.00	0.87	0.3701
javaExperience	255.95	255.95	1.00	12.00	0.33	0.5753
TASK	16602.94	16602.94	1.00	11.00	21.52	0.0007
SLICING	5.81	5.81	1.00	11.00	0.01	0.9324
GROUP	271.08	271.08	1.00	12.00	0.35	0.5644
STRATEGY:javaExperience	1320.28	1320.28	1.00	11.00	1.71	0.2175

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.39	0.39	1.00	11.00	0.92	0.3570
javaExperience	0.00	0.00	1.00	12.00	0.00	0.9804
TASK	7.01	7.01	1.00	11.00	16.65	0.0018
SLICING	0.38	0.38	1.00	11.00	0.91	0.3606
GROUP	0.09	0.09	1.00	12.00	0.22	0.6481
STRATEGY:javaExperience	0.69	0.69	1.00	11.00	1.63	0.2278



(a) Calidad (b) Productividad

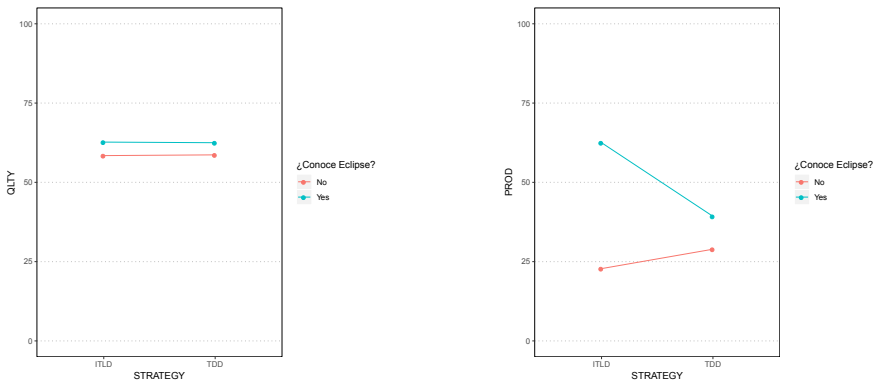
Figura 5.71: Efecto de la experiencia en programación Java. La experiencia se reporta redondeada en años.

El uso del lenguaje de programación Java presenta una influencia positiva para la Calidad y neutra para la Productividad (B =6.17 B =0). Los efectos tampoco son significativos en ambos casos ($P - value$ =0.58 y $P - value$ =0.98).

Los programadores con menor experiencia en Java, al igual que aquellos que tenían menor experiencia en programación, mejoraron notablemente al aplicar TDD. Al contrario, aquellos con mayor experiencia en Java son los

que obtuvieron menor Calidad y Productividad al aplicar TDD.

5.5.4.3. Conocimiento del entorno Eclipse



(a) Calidad (b) Productividad

Figura 5.72: Efecto del conocimiento del entorno Eclipse.

El conocimiento del entorno de desarrollo Eclipse presenta una influencia positiva para las variables Calidad y Productividad (B =14.94, B =0.01). Los efectos sin embargo, no son significativos ($P - value$ =0.81 y $P - value$ =0.8).

5.5.4.4. Resumen general

En las tablas 5.188 y 5.189 mostramos un resumen de los resultados de los análisis de subgrupos realizados. Ninguno de los factores analizados tiene un efecto que sea estadísticamente significativo.

Tabla 5.76: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	-2.64	0.6	2.19	0.74
Experiencia en Java	6.17	0.58	-8.36	0.22
Conocimiento del Entorno Eclipse	14.94	0.81	-22.97	0.41

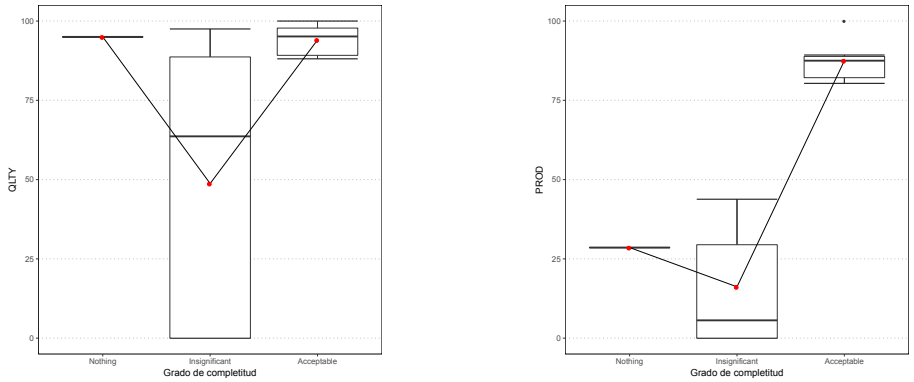
Tabla 5.77: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	0	0.18	0	0.99
Experiencia en Java	0	0.98	0	0.23
Conocimiento del Entorno Eclipse	0.01	0.8	-0.09	0.34

5.5.5. Grado de completitud

5.5.5.1. Aspectos generales

Como hemos venido realizando en otros experimentos, en esta sección analizamos manualmente el código entregado por los sujetos. La relación entre el grado de completitud frente a las variables QLTY y PROD se presenta en la figura 5.179. Adicionalmente, en la Tabla 5.190 se cuantifica el grado de completitud del trabajo realizado.



(a) Calidad (b) Productividad

Figura 5.73: Relación entre el grado de completitud y la calidad y productividad

Se entregaron un total de 30 tareas, de las cuales revisamos que 1 sujetos no hicieron nada. Por otro lado, 23 hicieron una mínima cantidad de trabajo y 6 sujetos realizaron el trabajo como era esperado.

Como ha ocurrido en experimentos anteriores, al incluir las tareas vacías los resultados difieren significativamente. Por ejemplo, podemos ver los siguientes datos: Promedio de QLTY para la estrategia TDD incluidos los sujetos que no realizaron un trabajo sustancial: 59.43 %. Promedio de QLTY para la estrategia TDD incluyendo únicamente a los sujetos que hicieron un

	Número de tareas entregadas
Nothing	1
Insignificant	23
Acceptable	6

Tabla 5.78: Grado de completitud

trabajo aceptable: 92.79 %.

5.5.5.2. Impacto de la estrategia de programación y la tarea experimental

Como se muestra en la tabla tabla 5.191, la estrategia de programación al parecer no tiene impacto en la distribución entre las tareas entregadas con los grados de finalización analizados. La diferencia es con una sola tarea que tiene calificación *Nothing* para TDD y que produce el desbalance al restarse de las tareas calificadas como *Insignificant* para esta estrategia.

	Nothing	Insignificant	Acceptable
ITLD	0	12	3
TDD	1	11	3

Tabla 5.79: Estrategia de programación

El test χ^2 que arroja los siguientes resultados: $\chi^2 = 1{,}04$, $df = 2$, $p - value = 0{,}59$, lo cual comprueba la inexistencia de una relación estadísticamente significativa entre la estrategia de programación y el grado de finalización de la tarea.

	Nothing	Insignificant	Acceptable
BSK	1	9	5
MR	0	14	1

Tabla 5.80: Grado de completitud por tarea experimental (BSK, MR)

Adicionalmente, la tabla 5.192 muestra que la mayor parte de tareas calificadas como *Acceptable* fueron para BSK. Sin embargo, las diferencias entre BSK y MR no significativas como muestra el test χ^2 ($\chi^2 = 4{,}75$, $df = 2$, $p - value = 0{,}09$).

5.5.5.3. Impacto de las variables demográficas

En las siguintes secciones analizaremos si el grado de completitud de las tareas, se encuentra reacionado con las variables demográficas obtenidas

de los sujetos experimentales. Vamos a analizar las mismas variables de la sección 5.12.4, esto es:

- Experiencia en Programación.
- Experiencia en Java.
- Conocimiento del entorno Eclipse.

Como podemos apreciar en la figura 5.180, la experiencia en programación al parecer no influye en el grado de completitud de las tareas. El test Kruskal-Wallis entre la experiencia en programación y el grado de completitud muestra un p-valor = 0,57 que no es significativo y nos confirma lo mostrado en la figura.

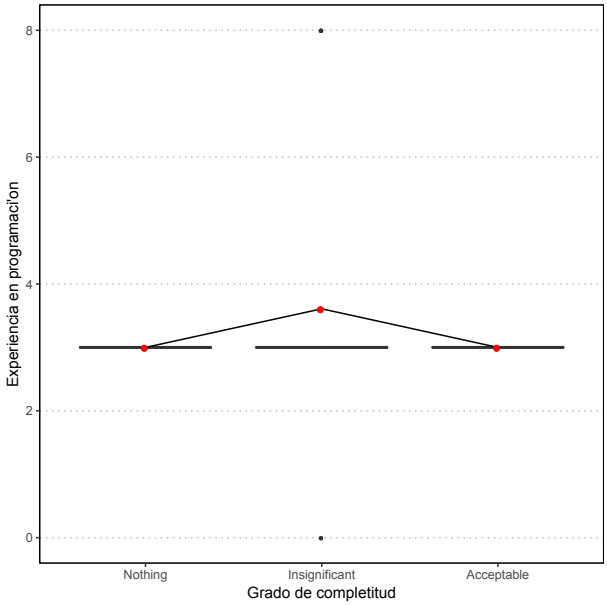


Figura 5.74: programmingExperience

Experiencia en programación Java

La experiencia en desarrollo con el lenguaje Java no parece influir en el grado de presentación de las tareas como podemos apreciar en la figura 5.181. Al realizar el test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud obtenemos un p-valor = 0,85 que no es significativo y nos confirma la impresión visual de la figura 5.181.

Conocimiento del entorno Eclipse

El conocimiento del entorno de desarrollo Eclipse si tiene influencia en el grado de completitud de las tareas experimentales. El test χ^2 nos indica

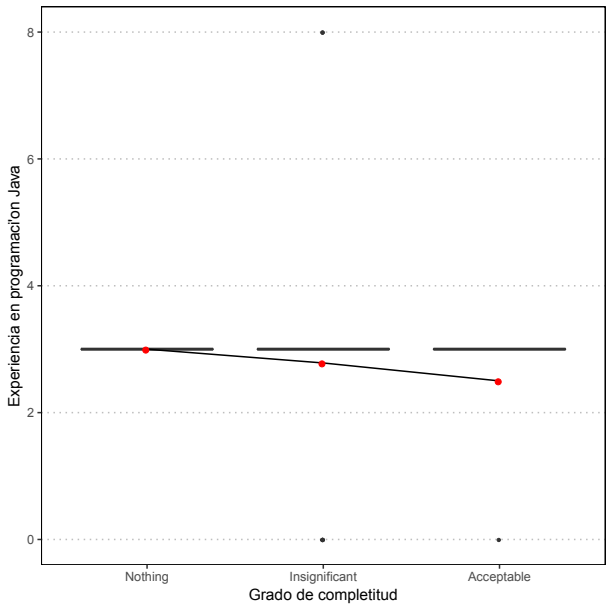


Figura 5.75: javaExperience

	Nothing	Insignificant	Acceptable
No	0	21	3
Yes	1	2	3

Tabla 5.81: Conocimiento del entorno eclipse

este hecho ($\chi^2 = 9{,}21$, $df = 2$, $p-value = 0{,}01$). En la tabla 5.193, podemos apreciar la distribución de los sujetos que indicaron tener o no conocimiento del entorno Eclipse, frente al grado de completitud de las tareas. Existe un elevado número de tareas donde el trabajo realizado fue catalogado como *Insignificant* y donde los sujetos indicaron que no tenían conocimiento previo del entorno Eclipse. Este hecho podría ser la causa de las diferencias significativas.

5.5.5.4. Resumen general impacto variables demográficas

En la tabla 5.194 se resume los resultados de los análisis de la influencia de las variables demográficas estudiadas. Existen diferencias significativas entre el *Conocimiento del entono Eclipse* y el grado de completitud de las tareas entregadas.

Tabla 5.82: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Experiencia en programación	0.57
Experiencia en Java	0.85
Conocimiento del entorno Eclipse	0.01

5.6. Experimento UNLP2015

5.6.1. Ejecución

El experimento UNLP2015 es el primero de los experimentos realizados con estudiantes de postgrado, y es una replicación del experimento ESPE2015 en el ámbito académico. Este experimento se realizó con un grupo de estudiantes de doctorado de la Universidad Nacional de La Plata de Argentina (UNLP) mientras realizaban sus actividades de investigación.

5.6.1.1. Muestra

En el experimento UNLP2015 participaron 9 estudiantes de post-grado de la Universidad Nacional de La Plata de Argentina. En general los estudiantes fueron jóvenes menores de 30 años sin mayor experiencia profesional, como se indica en la tabla 5.195. Las diferencias más significativas que encontramos entre grupos de sujetos son:

- La experiencia en programación se encuentra distribuida en niveles intermedio, novato y sin experiencia.
- Aproximadamente el 45 % no tiene experiencia en lenguaje Java e igual porcentaje indicó tener entre 2 y 5 años de experiencia. Un estudiante que representa el 10 % restante indicó tener entre 6 y 10 años de experiencia.
- Existe un 44 % de estudiantes que tienen entre 2 y 5 años de experiencia en JUnit. El resto indica no tener mayor experiencia.

5.6.1.2. Preparación

Este experimento se lo realizó en uno de los laboratorios de computación de la UNLP y, como en anteriores experimentos, se instaló la máquina virtual de Java, el framework Junit, el plugin para empaquetar y realizar mediciones y el IDE de desarrollo Eclipse.

Tabla 5.83: Características demográficas de los sujetos del experimento UNLP2015

EXPERIMENTO: UNLP2015		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	9
	30 >= Edad < 40 años	9
	40 >= Edad < 50 años	9
	Edad >= 50 años	9
Nivel de Educación	Other	0
	Undergraduate	0
	Bachelor	0
Experiencia profesional	Sin experiencia (<2 años)	9
	Novato (2 - 5 años)	9
	Intermedio (6 - 10 años)	9
	Experto (>10 años)	9
Experiencia en programación	Sin experiencia (<2 años)	2
	Novato (2 - 5 años)	3
	Intermedio (6 - 10 años)	4
	Experto (>10 años)	0
Uso de herramientas de pruebas	Yes	8
	No	1
Experiencia en lenguaje Java	Sin experiencia (<2 años)	4
	Novato (2 - 5 años)	4
	Intermedio (6 - 10 años)	1
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	6
	Novato (2 - 5 años)	3
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	1
	No	8
Experiencia en TDD	Sin experiencia (<2 años)	9
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	1
	No	8
Conocimiento del entorno Eclipse	Yes	0
	Yes	9
Función actual en la organización	Tester	0
	Developer	0
	Student	9

5.6.1.3. Realización

El experimento se realizó como una actividad colaborativa de investigación organizada por Claudia Pons con alumnos doctorandos del Laboratorio de Investigación y Formación en Informática Avanzada (Lifia). El experimento se realizó siguiendo el mismo protocolo utilizado en el experimento UADY2015, esto es, mediante un seminario taller de 5 días con un horario de cuatro horas por día y con la participación de los mismos investigadores de UADY2015. Por esta razón, hemos omitido la descripción en detalle del protocolo en este experimento.

5.6.1.4. Desviaciones

No existieron desviaciones que se consideren importantes durante la realización de experimento. En general se siguió el cronograma establecido y los estudiantes asistieron puntualmente a cada una de las sesiones, excepto uno de ellos que no asistió a la sesión final.

5.6.1.5. Reducción del conjunto de datos

A este experimento asistieron 9 sujetos, y de ellos un único sujeto no asistió a la segunda sesión experimental y por tanto no realizó la tarea aplicando la técnica de TDD. No pudimos conocer la causa de este abandono.

5.6.2. Estadísticos descriptivos

En la siguiente sección, describimos los datos estadísticos obtenidos para las variables respuestas de Calidad y Productividad.

5.6.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	83.57	17.10	-0.61	-0.94
TDD	83.86	16.99	-1.10	0.10

Tabla 5.84: Estadísticos descriptivos para QLTY

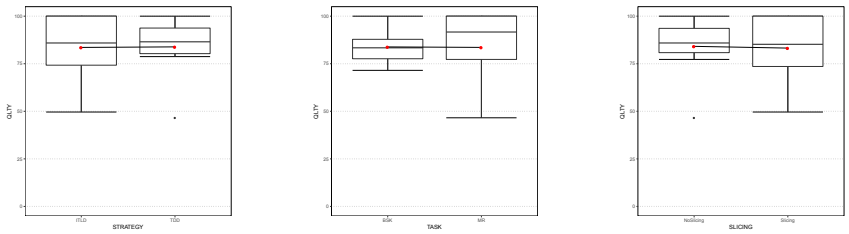
La calidad obtenida por los participantes es relativamente alta. Las medias obtenidas por los grupos ITLD y TDD son similares, tal y como se indica en la tabla 5.196. La asimetría y la curtosis para ITLD es negativa, en tanto que para TDD la asimetría es negativa y la curtosis positiva. Esto implica que la cola de distribución se alarga para valores inferiores a la media existiendo una menor concentración de datos entorno a la misma.

En lo gráficos de box-plot de la figura 5.182 podemos apreciar que las medianas para las dos estrategias (ITLD y TDD) son bastante coincidentes.

Además, ambas cajas se sitúan sobre el primer cuadrante por lo que no prevemos la existencia de significación estadística.

En cuanto a la tarea, al contrario de la mayoría de experimentos antes analizados, el promedio de MR es ligeramente superior al de BSK, como indica la figura 5.182b. No podemos apreciar claramente si los efectos de la tarea son significativos para la Calidad.

El efecto del nivel de descripción de la tarea (slicing) tiene una mediana similar y al parecer esta factor no ejerce efectos significativos sobre la calidad como puede observarse en la figura 5.182c.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.76: Box plots para la variable respuesta QLTY

5.6.2.2. Productividad

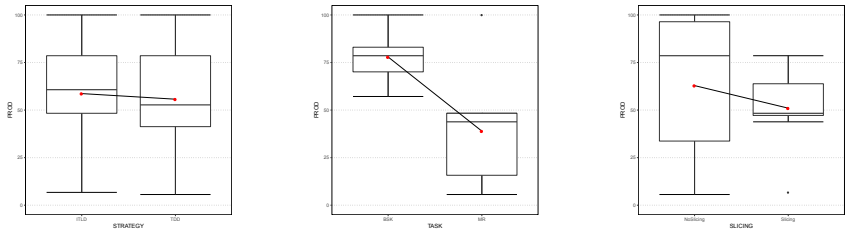
Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	58.67	32.60	-0.28	-1.44
TDD	55.72	29.76	-0.13	-1.25

Tabla 5.85: Estadísticos descriptivos para PROD

La Productividad tiene promedios menores que la Calidad, como se muestra en la tabla 5.197, con desviaciones estándar relativamente elevadas para las dos estrategias. Tanto la asimetría como la curtosis en este caso son negativas. En la figura 5.183a, se aprecia gráficamente que no existen diferencias sustanciales entre ITLD y TDD. La mediana de ITLD es ligeramente superior a la de TDD, y la mayoría de datos se agrupan en el tercer cuartil.

En la figura 5.183b se nota claramente el impacto de la tarea para la Productividad. Al contrario de la Calidad, en este caso BSK obtiene valores considerablemente superiores que MR, y el efecto parece significativo.

En cuanto al nivel de definición de la tarea (Slicing), al parecer no tiene influencia sobre la productividad, aunque la mediana se ubica por sobre el 75 % para el nivel No-Slicing y por debajo del 50 % para el nivel slicing.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.77: Box plots para la variable respuesta PROD

5.6.3. Prueba de hipótesis

5.6.3.1. Análisis estadístico

La estrategia de análisis estadístico en este caso, sigue las mismas pautas de los experimentos hechos con estudiantes. Los resultados de la ANOVA respecto a las variables respuesta Calidad y Productividad se muestran en la tabla 5.198. Los resultados no son significativos en casi todos los casos. La única excepción es la Tarea que obtiene resultados estadísticamente significativos para el caso de la Productividad (aunque BSK sigue siendo mayor que MR como ha sido habitual).

Tabla 5.86: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	5499459605.56	5499459605.56	1.00	8059.80	0.12	0.7248
TASK	5710811418.35	5710811418.35	1.00	8059.80	0.13	0.7198
SLICING	2198160713.15	2198160713.15	1.00	4307.17	0.05	0.8239
GROUP	15172439169.26	15172439169.26	1.00	6.83	0.34	0.5775

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	32.27	32.27	1.00	4.72	0.55	0.4948
TASK	8404.24	8404.24	1.00	4.72	142.36	0.0001
SLICING	82.63	82.63	1.00	4.74	1.40	0.2927
GROUP	27.54	27.54	1.00	6.63	0.47	0.5177

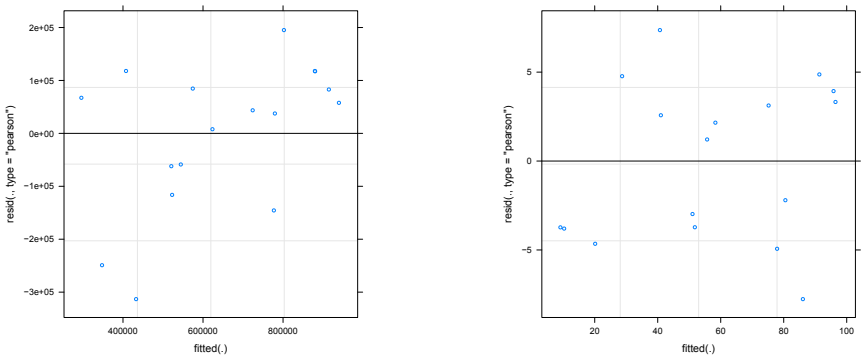
5.6.3.2. Chequeo del análisis estadístico

El chequeo del análisis estadístico sigue el mismo patrón que el experimento base ESPE2015. Como se ha mencionado, de las tres asunciones

dadas para los modelos mixtos (linealidad entre factores, homogeneidad de varianzas y distribución normal de residuos) comprobaremos la segunda y tercera condición ya que en este experimento al igual que en los restantes, los factores tienen solo dos niveles, por lo tanto, la linealidad siempre se cumple.

Homogeneidad de varianzas

Los gráficos de valores predichos vs. residuos estandarizados se muestran en la figura 5.184a. No se aprecia la existencia de ninguna figura de embudo que sugiera heterogeneidad entre varianzas.



(a) Calidad (b) Productividad

Figura 5.78: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

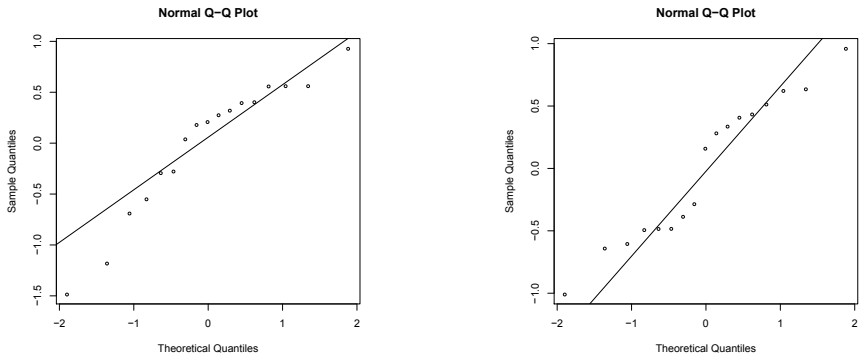
Normalidad de residuos

Estudiamos los residuos tanto de los factores fijos como de los factores aleatorios. En lo que respecta a los factores fijos, en la figura 5.185 podemos apreciar la distribución de los residuos para las variables respuesta QLTY y PROD. Notamos desviaciones a lo largo de la recta, por lo que la gráfica no nos permite apreciar la existencia de normalidad de residuos.

El test de Shapiro-Wilks indica que existe normalidad de los residuos para la Productividad ($W = 0{,}93$, $p - value = 0{,}23$). Sin embargo, en el caso de la Calidad, el test indica que los residuos no están distribuidos normalmente. Por ello, hemos aplicado una transformación Box-Cox con $\lambda = 3$, lo que produce que los residuos sean normales ($W = 0{,}91$, $p - value = 0{,}11$).

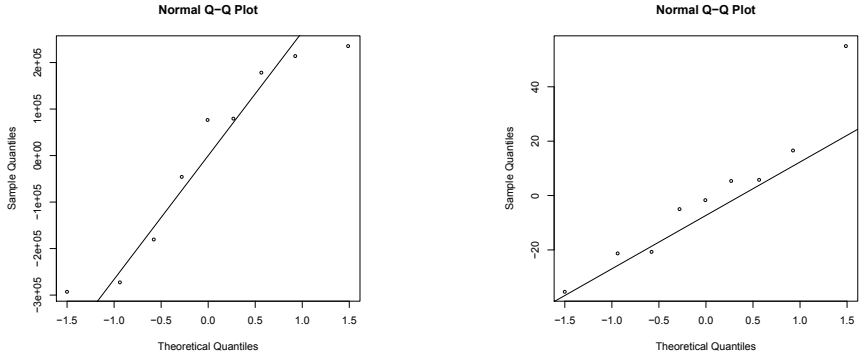
En lo que respecta a los factores aleatorios, la figura 5.186 muestra el gráfico QQ de los residuos para las variables QLTY y PROD. Excepto por los extremos, los datos se ajustan razonablemente a la distribución normal.

El test de Shapiro-Wilks nos permite confirmar la existencia de normalidad para la Calidad ($W = 0{,}9$, $p - value = 0{,}23$) y para la Productividad ($W = 0{,}93$, $p - value = 0{,}49$).



(a) Calidad (b) Productividad

Figura 5.79: Gráficos QQ para los factores fijos



(a) Calidad (b) Productividad

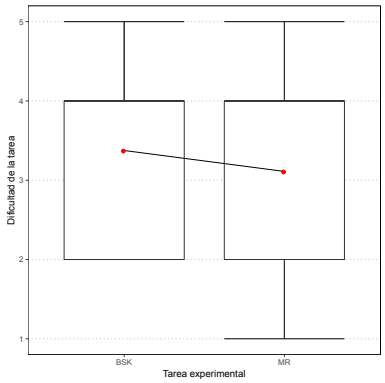
Figura 5.80: Gráficos QQ para los factores aleatorios

5.6.3.3. Influencia de factores instrumentales

En esta sección analizamos la dificultad de la tarea experimental percibida por los sujetos. Como en todas las instancias experimentales, las tareas fueron seleccionadas ad-hoc. Nuestro interés con este análisis es determinar si los resultados obtenidos podrían deberse al diseño experimental.

Tarea experimental

Como podemos observar en la figura 5.187a, la complejidad percibida por los sujetos es similar para MR y BSK. Esto lo podemos confirmar mediante el test Kruskal-Wallis entre la tarea experimental y la dificultad percibida, que arroja un p-valor = 0,68. Por lo tanto, aunque la tarea mostró diferencias significativas para la Productividad (ver sección **??**), los sujetos consideraron que el grado de dificultad no es un factor que probablemente incida en las variables estudiadas.



(a) Dificultad percibida de la tarea

Figura 5.81: Efecto de la dificultad de la tarea percibida por los sujetos

5.6.4. Análisis de subgrupos

Conforme las diferencias encontradas en los datos demográficos de la sección 5.13.1.1, en esta para el análisis de subgrupos consideramos los siguientes aspectos:

- Experiencia en programación.
- Experiencia en Java.
- Experiencia de el framework de pruebas JUnit.

5.6.4.1. Experiencia en programación

La experiencia en programación muestra una influencia negativa para las variable Calidad (B =-33.73) y Productividad (B =-1.67). El análisis también nos indica que los efectos no son significativos en ambos casos ($P-valor$ =0.82 y $P-valor$ =0.77). En el gráfico 5.188 se aprecia que los desarrolladores con un nivel de experiencia en programación intermedia (en el rango de 6 a 10 años de experiencia) mejoran su Calidad y Productividad al aplicar TDD. En contraste, aquellos programadores menos experimentados (considerados como novatos, entre 0 y menos de 2 años de experiencia),

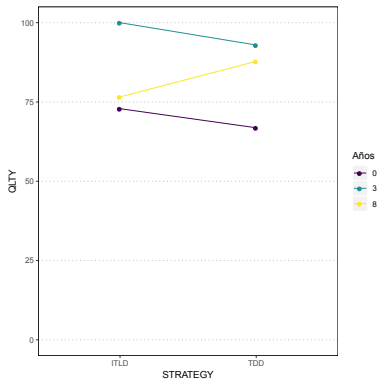
Tabla 5.87: Resultados del análisis estadístico para la experiencia en programación

(a) Calidad

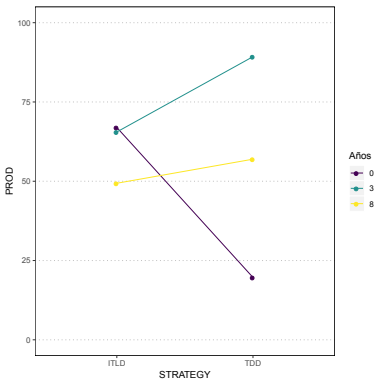
	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	95163037862.58	95163037862.58	1.00	250.02	3.31	0.0700
programmingExperience	1687455975.30	1687455975.30	1.00	6.02	0.06	0.8166
TASK	65430893996.31	65430893996.31	1.00	533.85	2.28	0.1320
SLICING	65965231464.74	65965231464.74	1.00	246.18	2.29	0.1311
GROUP	2518272071.17	2518272071.17	1.00	6.23	0.09	0.7769
STRATEGY:programmingExperience	130757753215.11	130757753215.11	1.00	332.57	4.55	0.0337

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	10.04	10.04	1.00	3.81	0.13	0.7384
programmingExperience	7.58	7.58	1.00	5.12	0.10	0.7671
TASK	4830.67	4830.67	1.00	3.66	62.16	0.0020
SLICING	0.26	0.26	1.00	3.81	0.00	0.9568
GROUP	33.03	33.03	1.00	5.22	0.43	0.5421
STRATEGY:programmingExperience	27.58	27.58	1.00	3.74	0.35	0.5855



(a) Calidad

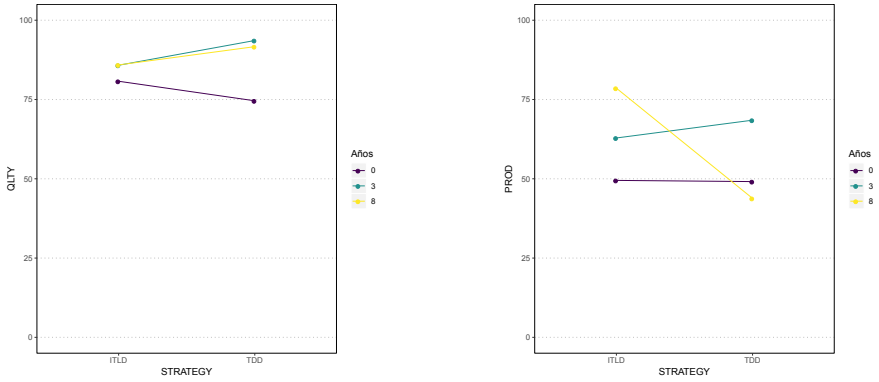


(b) Productividad

Figura 5.82: Efecto de la experiencia en programación. La experiencia se reporta redondeada a años.

disminuyeron su Calidad y Productividad. Por otra parte, los programadores dentro del rango de 2 a 5 años de experiencia, disminuyeron la Calidad al aplicar TDD y mejoraron su Productividad al usar esta misma técnica. La interacción *STRATEGY*programmingExperience*, en este caso resulta ser significativa para la Calidad.

5.6.4.2. Experiencia en programación Java



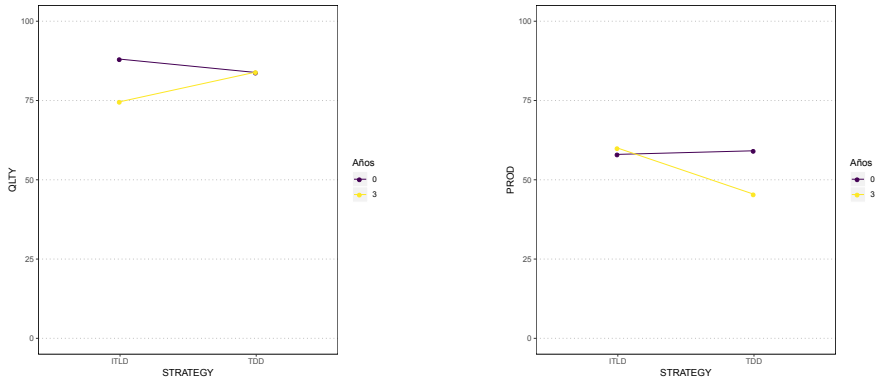
(a) Calidad (b) Productividad

Figura 5.83: Efecto de la experiencia en programación Java. La experiencia se reporta redondeada en años.

La experiencia que indicaron tener los sujetos en el lenguaje de programación Java tiene una influencia positiva para la Calidad y para la Productividad (B =28.59 B =2.95). Los efectos en este caso tampoco son significativos ($P - value$ =0.35 y $P - value$ =0.44) por lo que omitimos la tabla de análisis. En la figura 5.189 se observa el patrón de compartamiento al aplicar TDD e ITLD tanto para la Calidad como para la Productividad. Los programadores con mayor experiencia en Java mejoraron la Calidad al aplicar TDD, no así para la Productividad, donde la tendencia es decreciente al aplicar TDD. Los desarrolladores menos experimentados también muestran una ligera tendencia decreciente al aplicar TDD para la Calidad. Para la Productividad no existen deiferencias notables cuando este grupo de desarrolladores aplicaron las dos técnicas. El grupo de desarrolladores con una experiencia intermedia en Java, muestran una cierta mejora tanto de Calidad como de Productividad al aplicar TDD (existe cierto paralelismo entre lineas, como en casos anteriores).

5.6.4.3. Experiencia en el Framework JUnit

La experiencia de los sujetos en el framework JUnit presenta efectos negativos para la Calidad (B =-44.16) y positivos para la Productividad



(a) Calidad (b) Productividad

Figura 5.84: Efecto de la experiencia en el Framework JUnit. La experiencia se reporta redondeada en años

(B =2.2). Una vez más, los efetos no son significativos ($P-value$ =0.98 y $P-value$ =0.64). En este caso, la gráfica 5.190 muestra que los desarrolladores con más experiencia en el Framework JUnit mejoran la Calidad, pero empeoran su Productividad al aplicar TDD. Los desarrolladores sin experiencia en el Framework JUnit prácticamente mantienen constante su Calidad y Productividad con ambas técnicas. La interacción *STRATEGY*jUnitExperience* también ha resultado significativa en este caso.

5.6.4.4. Resumen general

En las tablas 5.200 y 5.201 mostramos un resumen de los resultados de los análisis de subgrupos realizados. En este experimento, no existen efectos significativos para ninguna de las variables consideradas. Sin embargo, se observa que las interacciones *STRATEGY * Variable* para la experiencia en programación y para la experiencia en el framework Junit resultan significativas para la Calidad. La limitada cantidad de sujetos no nos permite hacer afirmaciones que sean relevantes.

Tabla 5.88: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	-33.73	0.82	45.15	**0.03**
Experiencia en Java	28.59	0.35	33.02	0.45
Experiencia en el Framework JUnit	-44.16	0.98	55.18	**0**

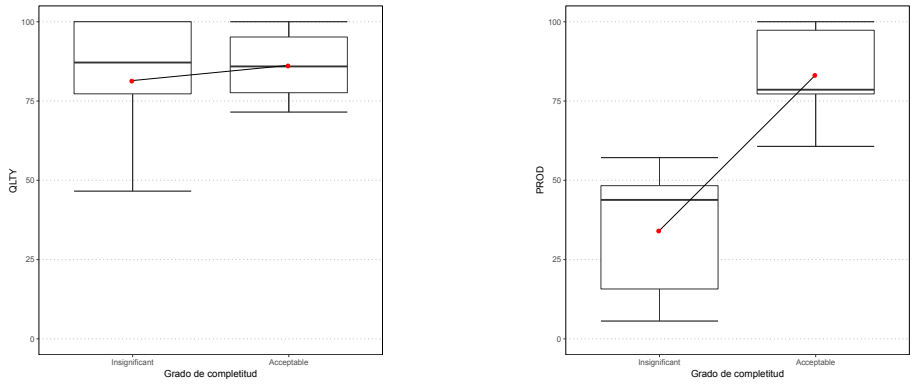
Tabla 5.89: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	-1.67	0.77	1.42	0.59
Experiencia en Java	2.95	0.35	0.81	0.45
Experiencia en el Framework JUnit	2.2	0.64	3.48	0.32

5.6.5. Grado de completitud

5.6.5.1. Aspectos generales

En este experimento, observamos que aproximadamente la mitad de sujetos hicieron las tareas de acuerdo a como se esperaba, por lo que se les calificó como *Aceptable*. La otra mitad realizó unas pocas líneas de código y se calificó como *Insignificant*. Ninguno de los participantes entregó tareas vacías como puede verse en la tabla 5.202. Por otro lado, en la figura 5.191, se muestra la relación entre el grado de completitud frente a las variables QLTY y PROD.



(a) Calidad (b) Productividad

Figura 5.85: Relación entre el grado de completitud y la calidad y productividad

De un total de 17 tareas recogidas, 0 sujetos no hicieron nada. Por otro lado, 9 hicieron una mínima cantidad de trabajo y 8 sujetos realizaron el trabajo como era esperado.

Al no tener tareas vacías, en este caso el análisis no se ve afectado significativamente para la Productividad. Al comparar el promedio de la QLTY

	Número de tareas entregadas
Nothing	0
Insignificant	9
Acceptable	8

Tabla 5.90: Grado de completitud

de la segunda sesión, incluidos los sujetos que no realizaron un trabajo sustancial, es del 83.86 %; cuando se excluyen estos sujetos, la QLTY es del 88.22 %.

Al realizar el mismo análisis para la Productividad, por el contrario, obtenemos que el promedio de PROD de la segunda sesión, incluidos los particpantes que no realizaron un trabajo sustancial es del 55.72 %; sin considerar estos sujetos, la PROD es del 85.71 %.

5.6.5.2. Impacto de la estrategia de programación y la tarea experimental

La tabla 5.203, nos induce a pensar que la *estrategia de programación* no parece estar relacionada con el grado de finalización de las tareas. Existe el mismo número de sujetos que no hicieron nada tanto para TDD como para ITLD. Existe una mínima diferencia entre el número de sujetos que hicieron las tareas de manera aceptable y la estrategia de programación.

	Nothing	Insignificant	Acceptable
ITLD	0	4	5
TDD	0	5	3

Tabla 5.91: Estrategia de programación

Podemos indicar que no existe relación entre la estrategia de programación y el grado de finalización de las tareas. Esto se comprueba mediante la realización de un test χ^2 (hemos excluído de la tabla los valores nulos (*Insignificant*) para que se pueda realizar el test). El test arroja los siguientes resultados: $\chi^2 = 0{,}07$, $df = 1$, $p - value = 0{,}8$.

	Nothing	Insignificant	Acceptable
BSK	0	1	7
MR	0	8	1

Tabla 5.92: Grado de completitud por tarea experimental (BSK, MR)

La tabla 5.204 muestra cierta relación entre el grado de completitud y las tareas. En este caso, el doble de sujetos no hicieron nada en la tarea MR, aunque casi el mismo número de sujetos entregaron las dos tareas de

forma aceptable. Al excluir las tareas nulas (*Nothing*), las diferencias entre BSK y MR son significativas como muestra el test χ^2 ($\chi^2 = 7{,}09$, $df = 1$, $p - value = 0{,}01$). Esto indica que MR es probablemente más complejo que BSK.

5.6.5.3. Impacto de las variables demográficas

Siguiendo con la misma metodología de análisis utilizado en todos los experimentos, analizamos si el grado de completitud de las tareas se encuentra reacionado con las variables demográficas obtenidas de los sujetos experimentales. Analizaremos las mismas variables de la sección 5.13.4, esto es:

- Experiencia en programación.
- Experiencia en Java.
- Experiencia en el framework de pruebas JUnit.

Experiencia en programación

Como se aprecia en la figura 5.192, la experiencia en programación al parecer no tiene impacto sobre el grado de completitud de las tareas. Los máximos y mínimos que se observan en el gráfico tienen cierta diferencia en el nivel inferior. Algunos sujetos que entregaron la tarea de manera aceptable son programadores menos experimentados. El test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,74 el mismo que no es significativo.

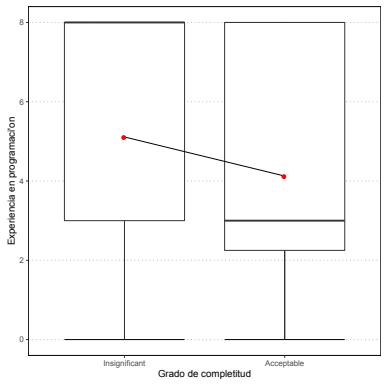


Figura 5.86: programmingExperience

Experiencia en programación Java

En cuanto a la experiencia en desarrollo con el lenguaje Java, al parecer tampoco influye en el grado de presentación de las tareas tal como se aprecia en la figura 5.193. El test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud no es significativo, con un p-valor = 0,96.

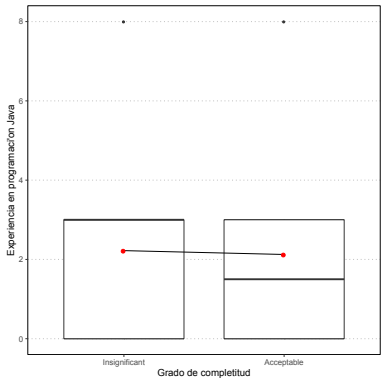


Figura 5.87: javaExperience

Experiencia en el framework de pruebas JUnit

En cuanto a la experiencia en el Framework de pruebas JUnit, en la figura 5.194. El test Kruskal-Wallis entre la experiencia en JUnit y el grado de completitud no es significativo, con un p-valor = 0,72.

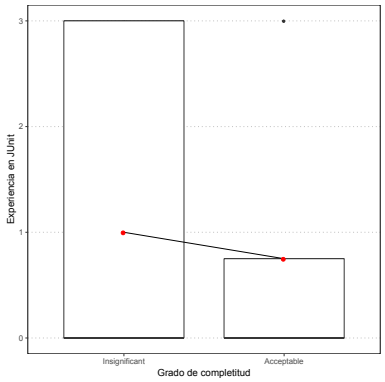


Figura 5.88: jUnitExperience

5.6.5.4. Resumen general impacto variables demográficas

La tabla 5.205 resume los resultados de los análisis de la influencia de las variables demográficas estudiadas frente al grado de completitud de las

tareas realizadas. No existen diferencias que sean estadísticamente significativas.

Tabla 5.93: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Experiencia en programación	0.74
Experiencia en Java	0.96
Experiencia en el framework JUNit	0.72

5.7. Experimento EcuadorUTN2017

5.7.1. Ejecución

El experimento UTN2017 es el último de la serie de experimentos realizados en el ámbito académico. Este experimento se llevó a cabo en la Universidad Técnica del Norte de Ecuador (UTN) con estudiantes de postgrado, todos ellos profesionales en el área de Ciencias de la Computación. La mayoría se encontraba desempeñando funciones relacionadas con el desarrollo de software. UTN2017 es una replicación del experimento ESPE2015.

5.7.1.1. Muestra

En el experimento UTN2017 participaron 27 estudiantes de la maestría en Ingeniería de Software perteneciente a la Universidad Técnica del Norte (UTN). En la tabla 5.206 podemos observar que ningún participante indicó tener experiencia previa en TDD. La mayoría reportó que no tenía experiencia en el framework JUnit (apenas un 7 % indicó tener entre 2 y 5 años de experiencia). El 85 % indicó que no había utilizado la técnica TDD y la mayoría de participantes (aproximadamente el 81 %) señaló tener conocimiento del entorno de desarrollo eclipse. Como puede notarse, las diferencias más notables se dan en los siguientes aspectos:

- La mayor parte de los participantes (70 %) tenían edades comprendidas entre 30 y 40 años. Un 11 % eran más jóvenes y un 19 % superaba los 40 años de edad.
- Existen participantes con diferentes niveles de experiencia en programación, siendo la mayor parte (un 40 %) programadores con nivel intermedio.
- La mayoría indicó no tener experiencia en Java (un 56 %). Los restantes programadores se caracterizaron como expertos (un 7 %), intermedios (un 11 %) y novatos (un 26 %).

- Aproximadamente un 37 % indicó haber utilizado herramientas de pruebas.
- Aproximadamente un 34 % indicó tener entrenamiento previo en pruebas unitarias.
- Los participantes desempeñaban diferentes funciones en la organización. La mayor parte eran desarrolladores (33 %), seguidos por analistas (19 %) y gestores (15 %). El 33 % restante, se encontraba desempeñando otras funciones.

5.7.1.2. Preparación

Este experimento fue realizado en las aulas de clase de la Universidad. Los participantes instalaron en sus computadores personales las herramientas necesarias. Se les solicitó instalar la máquina virtual de Java y el framework Junit. En este caso, los participantes entregaron el código realizado en un archivo empaquetado, y posteriormente se hizo las mediciones utilizando el plugin desarrollado para el efecto.

5.7.1.3. Realización

El experimento se realizó como parte del módulo de Gestión de proyectos de software, de la Maestría en Ingeniería de Software ofertada por la Universidad Técnica del Norte. No obstante, los estudiantes eran profesionales en activo, 18 de ellos se encontraban desempeñando funciones relacionadas con el desarrollo de software. El experimento se realizó en dos fines de semana consecutivos dentro del horario de clase del módulo que fue de 4 horas diarias. El experimento se realizó en cuatro días con el siguiente protocolo:

- **Día 1:** Los sujetos llenaran el cuestionario demográfico previo al inicio del entrenamiento. Durante el primer día se realizó una sesión de introducción al desarrollo ágil y formación en pruebas unitarias utilizando el framework Junit.
- **Día 2:** En el segundo día de la primera semana, se capacitó en las técnicas ITLD y slicing. Posteriormente se realizó la primera sesión experimental que consistió en la solución de las atreas BSK o MR con o sin slicing y aplicando la técnica ITLD. Posteriormente.
- **Día 3:** En el tercer día de la segunda semana, se realizó una sesión de entrenamiento en la técnica TDD.
- **Día 4:** Finalmente, en el cuarto día de la segunda semana se realizó la segunda tarea experimental, en este caso solucionar BSK o MR con o sin Slicing y aplicando la estrategia TDD.

Tabla 5.94: Características demográficas de los sujetos del experimento UTN2017

EXPERIMENTO: UTN2017		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	3
	30 >= Edad < 40 años	19
	40 >= Edad < 50 años	5
	Edad >= 50 años	0
Nivel de Educación	Other	0
	Undergraduate	0
	Bachelor	0
Experiencia profesional	Sin experiencia (<2 años)	1
	Novato (2 - 5 años)	15
	Intermedio (6 - 10 años)	8
	Experto (>10 años)	3
Experiencia en programación	Sin experiencia (<2 años)	1
	Novato (2 - 5 años)	8
	Intermedio (6 - 10 años)	11
	Experto (>10 años)	7
Uso de herramientas de pruebas	Yes	10
	No	17
Experiencia en lenguaje Java	Sin experiencia (<2 años)	15
	Novato (2 - 5 años)	7
	Intermedio (6 - 10 años)	3
	Experto (>10 años)	2
Experiencia en framework JUnit	Sin experiencia (<2 años)	25
	Novato (2 - 5 años)	2
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	4
	No	23
Experiencia en TDD	Sin experiencia (<2 años)	27
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	9
	No	18
Conocimiento del entorno Eclipse	Yes	22
	No	5
Función actual en la organización	Manager	4
	Developer	9
	Analyst	5

Como en la mayoría de la serie de experimentos realizados, la capacitación estuvo a cargo de Geovanny Raura con la colaboración de Rodrigo Fonseca y se utilizaron los materiales de entrenamiento preparados por Oscar Dieste.

5.7.1.4. Desviaciones

El protocolo se cumplió de acuerdo a lo establecido. Las horas de inicio y culminación de las actividades experimentales y de la capacitación se cumplieron dentro de lo establecido sin retrasos.

5.7.1.5. Reducción del conjunto de datos

Prácticamente no existieron variaciones en el número de sujetos que asistieron al experimento, excepto por un único participante que por razones de fuerza mayor no asistió la primera semana de ejecución del experimento. Se presentaron 27 participantes y de ellos 26 realizaron la primera tarea experimental, es decir la aplicación de la estrategia ITLD. El total de participantes realizaron la segunda tarea utilizando la estrategia TDD.

5.7.2. Estadísticos descriptivos

Se describen por separado cada una de las variables respuestas de Calidad y Productividad obtenidas en esta instancia experimental.

5.7.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	62.58	35.02	-0.86	-0.79
TDD	70.77	24.70	-0.53	-1.20

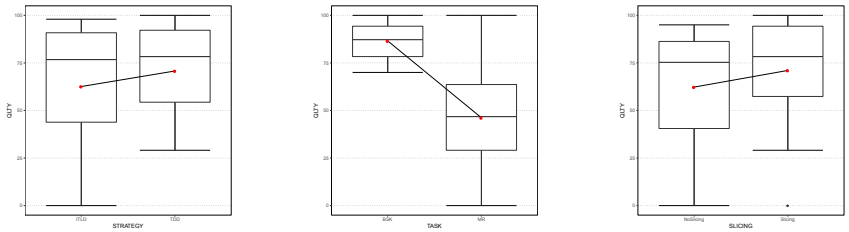
Tabla 5.95: Estadísticos descriptivos para QLTY

La tabla 5.207 muestra que la Calidad obtenida con la estrategia TDD es algo superior a lo obtenido para ITLD y superan el sesenta por ciento. Sin embargo, las desviaciones estándar son relativamente elevadas con asimetría y curtosis negativas en ambos casos y no tan altas.

De acuerdo al box-plot mostrado en la figura 5.195 las medianas son casi coincidentes para ambas estrategias como se muestra la tabla 5.207). No se visualiza la presencia de valores nulos para la estrategia TDD y la dispersión de puntos es menor que para ITLD.

En relación a la tarea, BSK sigue obteniendo mayores valores de Calidad que MR, tal y como indica la figura 5.195b. Al parecer los resultados son significativos para la tarea.

Finalmente, el efecto del factor Slicing presenta medianas similares con la mayor concentración de datos por encima del 50 %, como se aprecia en la figura: 5.195c.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.89: Box plots para la variable respuesta QLTY

5.7.2.2. Productividad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	26.82	30.19	1.31	0.53
TDD	37.40	31.34	0.58	-1.07

Tabla 5.96: Estadísticos descriptivos para PROD

La Productividad también es algo superior para la estrategia TDD sobre ITLD, aunque los promedios obtenidos son significativamente menores que para la Calidad. Como se observa en la tabla 5.208, los promedios están por debajo del 50 por ciento. La asimetría es positiva para las dos estrategias y la curtosis es opuesta con valores no excesivamente altos.

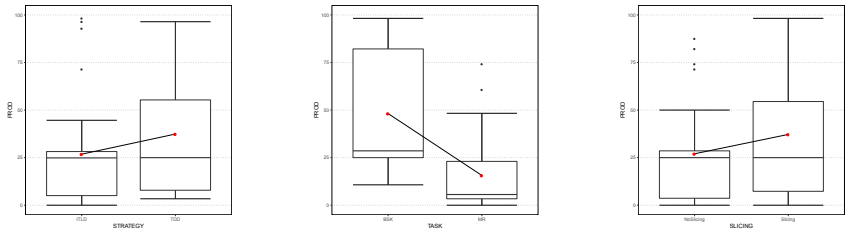
Esta tendencia podemos visualizarla de mejor manera en la figura 5.196. Se aprecia que los valores de las medianas son coincidentes para ITLD y TDD con similar cantidad de datos dispersos en los cuatro cuadrantes. Una vez más se aprecia que existe un impacto de la tarea para la Productividad: La tarea BSK obtiene una mediana superior en comparación con MR.

Finalmente, el nivel de definición de la tarea (slicing) presenta medianas similares con la mayor concentración de datos por debajo del 50 %.

5.7.3. Prueba de hipótesis

5.7.3.1. Análisis estadístico

Siguiendo la misma estrategia de análisis estadístico realizado con el experimento base ESPE2015, se realiza una ANOVA respecto a las variables



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.90: Box plots para la variable respuesta PROD

respuesta Calidad y Productividad. Como se muestran en la tabla 5.209, los resultados son significativos en la mayoría de casos:

- La estrategia de programación ha resultado significativa para la variable respuesta Productividad y se acerca al nivel de significación para la Calidad. No es necesario realizar un análisis post-hoc; la tabla 5.208 indica claramente que la Productividad obtenida con TDD es mayor que la obtenida con ITLD.
- Por otra parte, tal y como anticipábamos en la sección: 5.14.2, la tarea arroja resultados significativos tanto para Calidad como Productividad. De nuevo, la tarea BSK obtiene mejores valores que MR.
- Finalmente, el factor slicing también ha resultado significativo tanto para la Calidad como para la Productividad, aunque esto no se pudo visualizar claramente en los gráficos box plot analizados en la sección 5.14.2.

5.7.3.2. Chequeo del análisis estadístico

El chequeo del análisis estadístico sigue el mismo patrón que todos los experimentos en el ámbito académico. De las tres asunciones dadas para los modelos mixtos (linealidad entre factores, homogeneidad de varianzas y distribución normal de residuos) comprobaremos la segunda y tercera condición ya que la linealidad siempre se cumple (los factores tienen sólo dos niveles).

Homogeneidad de varianzas

No se aprecia la existencia de ninguna figura en embudo que sugiera heterogeneidad entre varianzas para la Productividad, como puede comprobarse en el gráfico de valores predichos vs. residuos estandarizados de la figura 5.197b. En el caso de la Calidad, la figura 5.197a sugiere un patrón

Tabla 5.97: Resultados del análisis estadístico

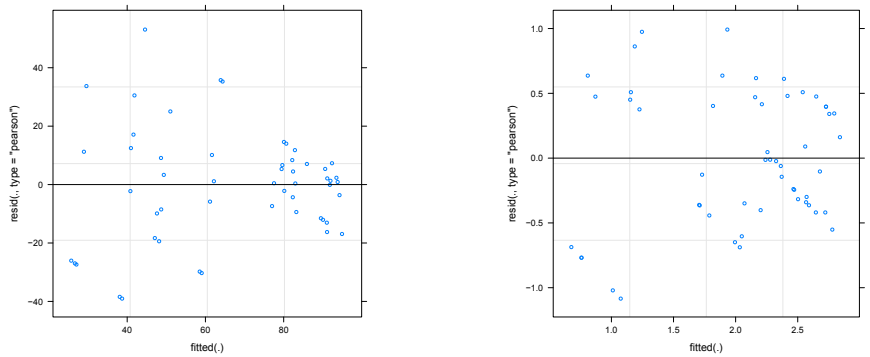
(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1718.01	1718.01	1.00	24.29	4.13	0.0532
TASK	23054.71	23054.71	1.00	24.29	55.43	0.0000
SLICING	1883.12	1883.12	1.00	24.29	4.53	0.0437
GROUP	998.90	998.90	1.00	25.29	2.40	0.1336

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	4.02	4.02	1.00	23.95	12.46	0.0017
TASK	15.51	15.51	1.00	23.95	48.03	0.0000
SLICING	1.40	1.40	1.00	23.95	4.34	0.0480
GROUP	2.00	2.00	1.00	24.98	6.20	0.0198

de embudo (la varianza decrece a medida que aumenta el tamaño de efecto). De todas formas, el patrón está lejos de ser concluyente.



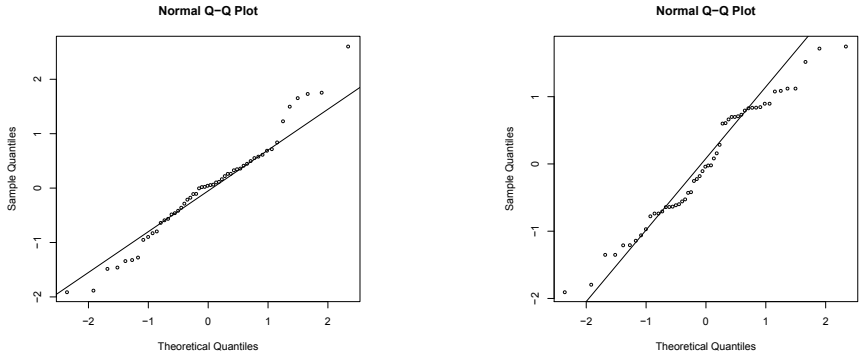
(a) Calidad (b) Productividad

Figura 5.91: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

Normalidad de residuos

Se estudian tanto de los factores fijos como de los factores aleatorios siguiendo la estrategia de análisis de todos los experimentos. En lo que respecta a los factores fijos, podemos apreciar en la figura 5.198, que la distribución de los residuos de la variable respuesta PROD se solapan razonablemente con la distribución normal (representada en la línea recta diagonal) con desviaciones en los extremos. Para el caso de la Productividad las desviaciones parecen más pronunciadas.

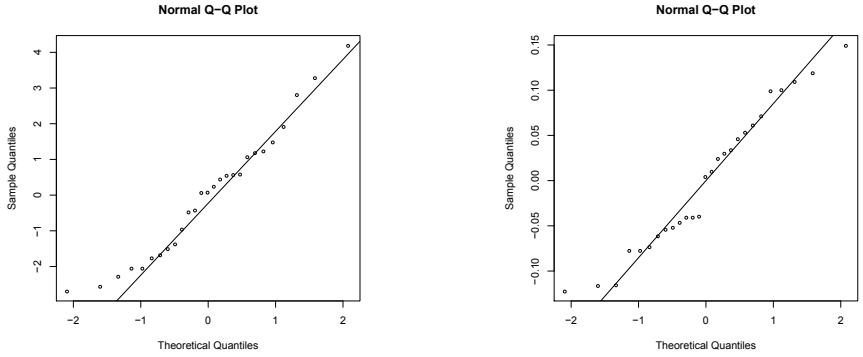
El test de Shapiro-Wilks indica que los residuos no se distribuyen normalmente para la Productividad, ($W = 0{,}97$, $p - value = 0{,}14$). Para el caso de la Calidad el test confirma la normalidad de residuos ($W = 0{,}98$, $p - value = 0{,}39$).



(a) Calidad (b) Productividad

Figura 5.92: Gráficos QQ para los factores fijos

En lo que respecta a los factores aleatorios, el test de Shapiro-Wilks una vez más confirma la normalidad de residuos para la Calidad ($W = 0{,}96$, $p - value = 0{,}37$) y no para la Productividad ($W = 0{,}95$, $p - value = 0{,}26$).



(a) Calidad (b) Productividad

Figura 5.93: Gráficos QQ para los factores aleatorios

```
> lmpt <- lmer(PROD^(1/4) ~  1 + STRATEGY +
+                              TASK +
+                              SLICING +
+                              GROUP +
```

```
+                                  (1 | subjectID),
+              data = all_expData)
> npt <- shapiro.test(resid(lmpt, scaled = TRUE))
> npat <- shapiro.test(ranef(lmpt)$subjectID$`(Intercept)`)
```

Debido a la falta de normalidad de residuos, aplicamos la transformación Box-Cox con $\lambda = \frac{1}{4}$, para la Productividad, con lo cual alcanzamos la normalidad para las dos variables respuesta (Calidad Efectos fijos: $W = 0{,}98$, $p-value = 0{,}39$, Efectos aleatorios: $W = 0{,}96$, $p-value = 0{,}37$; Productividad Efectos fijos: $W = 0{,}97$, $p-value = 0{,}14$, Efectos aleatorios: $W = 0{,}95$, $p - value = 0{,}26$). Los análisis realizados en la sección 5.14.3 ya incluyen esta transformación.

5.7.3.3. Influencia de factores instrumentales

En esta sección analizamos la dificultad de la tarea experimental percibida por los sujetos y su posible influencia debido al diseño experimental (como ha sido habitual en todos los experimentos).

Tarea experimental

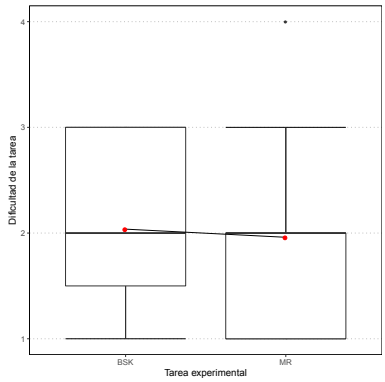
En la figura 5.200a, observamos que la complejidad percibida para MR es prácticamente similar a la complejidad de BSK (utilizando una escala de Likert de 1 a 5). El test Kruskal-Wallis entre la tarea experimental y la dificultad percibida arroja un p-valor = 0,62, lo cual confirma la impresión visual obtenida en la figura 5.200a.

Aunque el análisis estadítico arrojó resultados significativos para la tarea experimental (con mayores porecentajes de Calidad y Productividad para BSK sobre MR), sin embargo, de acuerdo a la percepción de los participantes la dificultad de la tarea no juega un papel relevante en la realización del experimento.

5.7.4. Análisis de subgrupos

Conforme a los datos demográficos de los participantes (véase sección 5.14.1.1) vamos a analizar los aspectos que contienen diferencias entre sujetos que merecen considerarse, estos son:

- Edad.
- Experiencia profesional
- Experiencia en programación.
- Experiencia en Java.
- Uso de herramientas de pruebas.



(a) Dificultad percibida de la tarea

Figura 5.94: Efecto de la dificultad de la tarea percibida por los sujetos

- Entrenamiento previo en desarrollo de pruebas unitarias.
- Función actual en la organización.

5.7.4.1. Edad

```
> lmq <- lmer(QLTY ~ 1 +
+                     STRATEGY * age +
+                     TASK +
+                     SLICING +
+                     GROUP +
+                     (1 | subjectID),
+             data = all_expData)
> lmp <- lmer(PROD^(1/4) ~  1 +
+                           STRATEGY * age +
+                           TASK  +
+                           SLICING +
+                           GROUP +
+                           (1 | subjectID),
+             data = all_expData)
```

El análisis realizado para determinar el impacto de la edad de los sujetos en la calidad del código y en la productividad de los programadores posee una influencia negativa (B = -2.4) para la Calidad y neutra ($B = 0$) para la Productividad. Los efectos no son significativos para la Calidad ($p-value =$ 0.12) pero si lo son para la Productividad $p-value = 0.02$). La introducción de la **edad** produce cambios en el nivel de significación de las estrategias de programación $ITLD/TDD$, tanto para la Calidad como la Productividad. Esto sugiere que los efectos observados para la estrategia de programación se deben en parte a la edad, y no únicamente a la estrategia de programación.

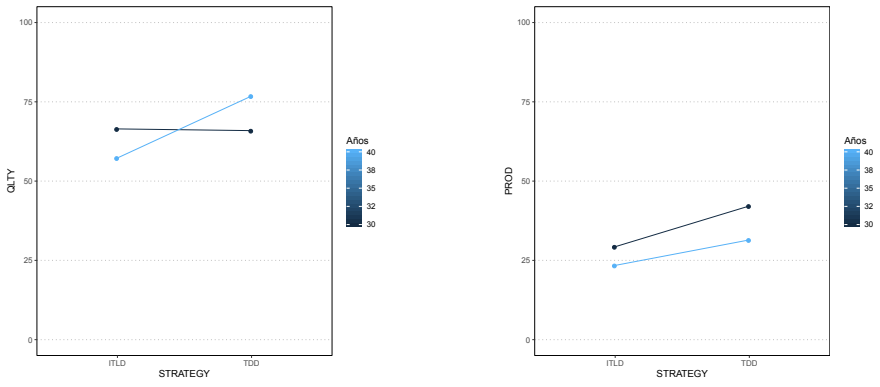
Tabla 5.98: Resultados del análisis estadístico para la Edad

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1074.92	1074.92	1.00	22.79	2.90	0.1024
age	965.46	965.46	1.00	23.95	2.60	0.1198
TASK	24091.19	24091.19	1.00	23.30	64.92	0.0000
SLICING	1733.03	1733.03	1.00	23.22	4.67	0.0412
GROUP	1172.10	1172.10	1.00	24.34	3.16	0.0880
STRATEGY:age	1442.48	1442.48	1.00	22.84	3.89	0.0609

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.24	0.24	1.00	22.55	0.78	0.3855
age	1.89	1.89	1.00	23.65	6.09	0.0212
TASK	16.07	16.07	1.00	23.07	51.78	0.0000
SLICING	1.34	1.34	1.00	22.99	4.30	0.0494
GROUP	2.73	2.73	1.00	24.04	8.79	0.0067
STRATEGY:age	0.54	0.54	1.00	22.61	1.75	0.1996

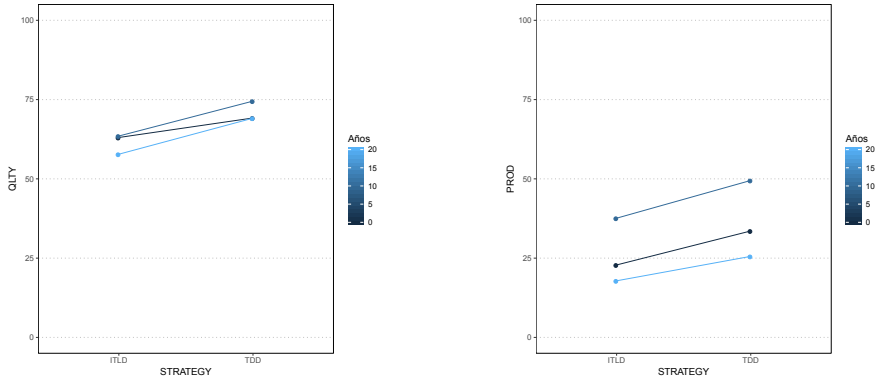


(a) Calidad (b) Productividad

Figura 5.95: Efecto de la Edad. La edad se reporta en décadas. Por ejemplo, el valor 20 representa el rango de 20-29 años de experiencia. El valor 30 representa el rango 30-39, etc.

La influencia de la edad en la estrategia de programación se muestra en la Fig. 5.201. Puede observarse que, a mayor edad, mayor es la Calidad y Productividad alcanzadas con TDD. Los participantes más jóvenes mantuvieron constante la Calidad tanto con $ITLD$ como con TDD, aunque mejoraron su Productividad al aplicar TDD.

5.7.4.2. Experiencia profesional



(a) Calidad (b) Productividad

Figura 5.96: Efecto de la experiencia profesional. La experiencia se reporta en décadas. Por ejemplo, el valor 0 representa 0-9 años de experiencia. El valor 10 representa 10-19 años de experiencia, etc.

En cuanto a la experiencia profesional, el análisis muestra una influencia negativa (B = -0.37) para la Calidad y neutra ($B = 0$) para la Productividad. Los efectos no son significativos ($p-value = 0.9$ y $p-value = 0.47$) en los dos casos, por esta razón se omiten las tablas de resultados. En este caso se observa en la figura 5.202, una marcada tendencia creciente a mejorar la Calidad y Productividad al aplicar TDD.

5.7.4.3. Experiencia en programación

La experiencia en programación muestra una influencia negativa para las variable calidad (B =-0.31) y neutra para la productividad (B =0). Los efectos no son significativos en ambos casos ($P-valor$ =0.54 y $P-valor$ =0.75). En lo que respecta a la aplicación de las estrategias de programación, en la figura 5.203 se observa una tendencia de los participantes con mayor experiencia en programación a mejorar la Calidad y Productividad cuando aplican TDD. Los participantes menos experimentados en programación disminuyeron ligeramente la Calidad con la estrategia TDD, aunque su Productividad se incrementó con esta estrategia.

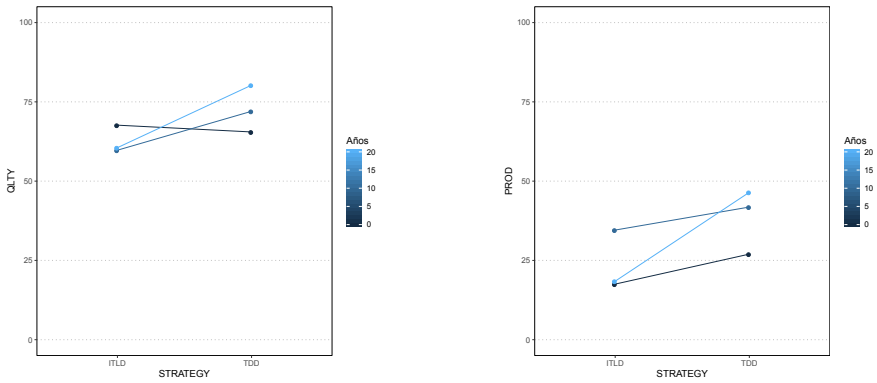
Tabla 5.99: Resultados del análisis estadístico para la experiencia en programación

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.61	0.61	1.00	22.88	0.00	0.9696
programmingExperience	155.32	155.32	1.00	23.79	0.38	0.5447
TASK	23253.66	23253.66	1.00	23.19	56.54	0.0000
SLICING	2018.68	2018.68	1.00	23.24	4.91	0.0368
GROUP	710.24	710.24	1.00	24.19	1.73	0.2011
STRATEGY:programmingExperience	523.47	523.47	1.00	22.72	1.27	0.2710

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.19	0.19	1.00	22.56	0.58	0.4556
programmingExperience	0.03	0.03	1.00	23.51	0.10	0.7546
TASK	15.62	15.62	1.00	22.87	48.42	0.0000
SLICING	1.50	1.50	1.00	22.92	4.64	0.0419
GROUP	1.67	1.67	1.00	23.91	5.18	0.0321
STRATEGY:programmingExperience	0.33	0.33	1.00	22.40	1.03	0.3207



(a) Calidad (b) Productividad

Figura 5.97: Efecto de la experiencia en programación. La experiencia se reporta en décadas. Por ejemplo, el valor 0 representa 0-9 años de experiencia. El valor 10 representa 10-19 años de experiencia, etc.

5.7.4.4. Experiencia en programación Java

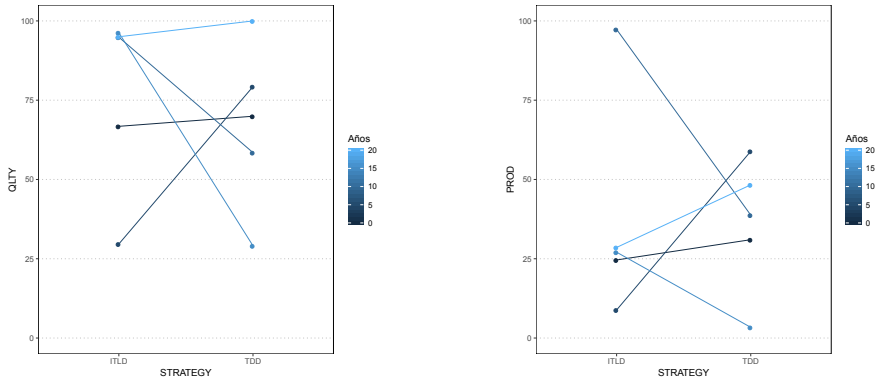
Tabla 5.100: Resultados del análisis estadístico para la Experiencia en Java

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	823.23	823.23	1.00	23.21	1.91	0.1802
javaExperience	252.04	252.04	1.00	23.79	0.58	0.4521
TASK	22586.73	22586.73	1.00	23.19	52.39	0.0000
SLICING	1904.96	1904.96	1.00	23.26	4.42	0.0466
GROUP	771.27	771.27	1.00	24.18	1.79	0.1935
STRATEGY:javaExperience	71.57	71.57	1.00	22.75	0.17	0.6875

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	2.09	2.09	1.00	22.90	6.27	0.0199
javaExperience	0.04	0.04	1.00	23.53	0.13	0.7238
TASK	15.34	15.34	1.00	22.88	46.03	0.0000
SLICING	1.42	1.42	1.00	22.95	4.26	0.0505
GROUP	1.78	1.78	1.00	23.92	5.34	0.0298
STRATEGY:javaExperience	0.10	0.10	1.00	22.45	0.29	0.5977



(a) Calidad (b) Productividad

Figura 5.98: Efecto de la experiencia en programación Java. La experiencia se reporta en lustros (5 años). Por ejemplo, el valor 0 representa 0-4 años de experiencia. El valor 5 representa 5-9 años de experiencia, etc.

La experiencia en el lenguaje de programación Java presenta una influencia positiva para la Calidad y nuetra para la Productividad (B =0.27 B =0). Los efectos no alcanzan el nivel de significación ($P - value$ =0.45 y $P - value$ =0.72).

En este caso la figura 5.204b muestra que algunos grupos de sujetos, principalmente los que más experiencia tienen en lenguaje Java, mejoraron

la Calidad y Productividad al aplicar TDD. Aunque también existen casos donde la Calidad y Productividad disminuyó drásticamente al aplicar TDD.

5.7.4.5. Uso de herramientas de pruebas

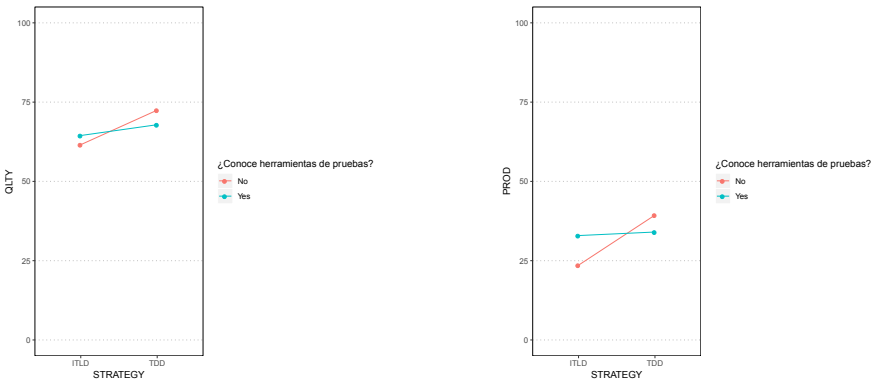
Tabla 5.101: Resultados del análisis estadístico para el uso de herramientas de pruebas

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1321.19	1321.19	1.00	23.61	3.09	0.0916
testingToolUsage	56.94	56.94	1.00	24.55	0.13	0.7182
TASK	23098.62	23098.62	1.00	23.24	54.07	0.0000
SLICING	1899.98	1899.98	1.00	23.24	4.45	0.0459
GROUP	966.90	966.90	1.00	24.24	2.26	0.1454
STRATEGY:testingToolUsage	156.45	156.45	1.00	23.46	0.37	0.5509

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3.44	3.44	1.00	23.35	10.24	0.0039
testingToolUsage	0.00	0.00	1.00	24.32	0.00	0.9604
TASK	15.51	15.51	1.00	22.98	46.20	0.0000
SLICING	1.40	1.40	1.00	22.98	4.18	0.0525
GROUP	1.98	1.98	1.00	24.01	5.90	0.0230
STRATEGY:testingToolUsage	0.06	0.06	1.00	23.20	0.17	0.6853



(a) Calidad (b) Productividad

Figura 5.99: Efecto del uso de herramientas de pruebas.

El uso de herramientas de pruebas presenta una influencia positiva (B =1.27) para la Calidad y neutra para la Productividad (B =0). Se puede notar también que los efectos una vez más no son significativos ($P - value$ =0.72 y $P - value$ =0.96).

Loa participantes que indicaron no conocer de herramientas de pruebas, mejoraron ligeramente la Calidad y Productividad al aplicar TDD, como se observa en la figura 5.205. Aquellos participantes que si conocían el uso de herramientas de prueba, prácticamente mantuvieron la misma Calidad y Productividad con las dos estrategias.

5.7.4.6. Función actual en la organización

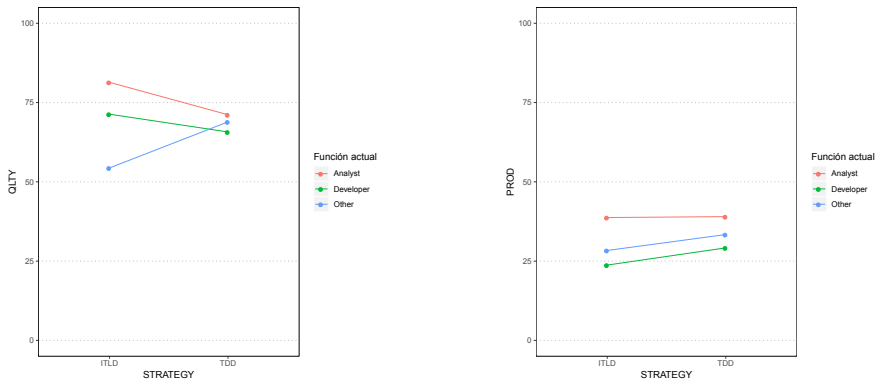
Tabla 5.102: Resultados del análisis estadístico para la Función Actual en la Organización

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	389.41	389.41	1.00	19.01	1.40	0.2506
currentFunction	479.20	239.60	2.00	19.70	0.86	0.4368
TASK	18453.48	18453.48	1.00	18.46	66.55	0.0000
SLICING	1939.21	1939.21	1.00	18.46	6.99	0.0162
GROUP	146.02	146.02	1.00	19.51	0.53	0.4766
STRATEGY:currentFunction	2345.72	1172.86	2.00	18.61	4.23	0.0306

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1.81	1.81	1.00	17.83	9.11	0.0074
currentFunction	0.38	0.19	2.00	18.78	0.96	0.4011
TASK	11.64	11.64	1.00	17.34	58.58	0.0000
SLICING	1.60	1.60	1.00	17.34	8.04	0.0113
GROUP	0.65	0.65	1.00	18.60	3.29	0.0860
STRATEGY:currentFunction	1.25	0.62	2.00	17.47	3.14	0.0682



(a) Calidad (b) Productividad

Figura 5.100: Efecto de la función actual en la organización

La función que se encontraban los sujetos desempeñando en la organi-

zación, presenta una influencia positiva para la Calidad y Productividad (B =8.64, B =0.01). Los efectos no son significativos en ambos casos ($p-value = 0.44$ y $p-value = 0.4$). La interacción *STRATEGY*currentFuntion*, resulta significativa para la Calidad y se aproxima al nivel de significación para la Productividad.

La figura 5.206 indica que los analistas y desarrolladores disminuyeron ligeramente la Calidad al aplicar TDD, aunque su productividad prácticamente es la misma con ambas estrategias. Los participantes con otras funciones, mejoraron ligeramente tanto en Calidad como en Productividad al aplicar TDD.

5.7.4.7. Entrenamiento previo en desarrollo de pruebas unitarias

Tabla 5.103: Resultados del análisis estadístico para el conocimiento de Pruebas Unitarias

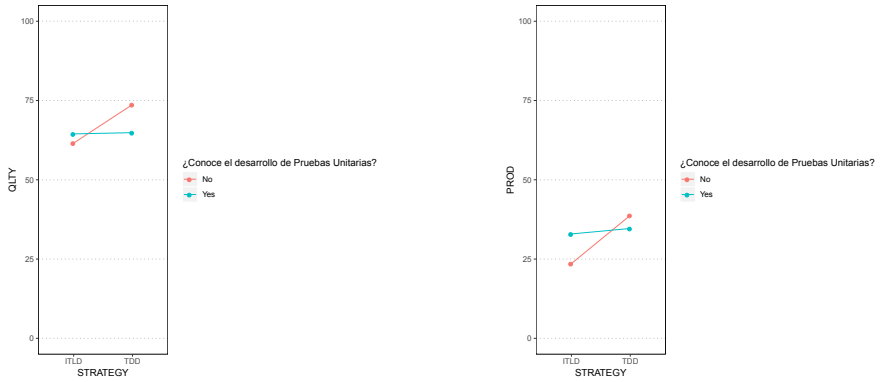
(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1215.54	1215.54	1.00	23.04	2.85	0.1050
UTTraining	68.75	68.75	1.00	23.97	0.16	0.6917
TASK	22814.60	22814.60	1.00	23.24	53.46	0.0000
GROUP	1019.50	1019.50	1.00	24.25	2.39	0.1351
SLICING	1823.79	1823.79	1.00	23.24	4.27	0.0500
STRATEGY:UTTraining	173.29	173.29	1.00	22.89	0.41	0.5303

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3.46	3.46	1.00	22.70	10.30	0.0039
UTTraining	0.01	0.01	1.00	23.67	0.03	0.8689
TASK	15.46	15.46	1.00	22.90	45.97	0.0000
SLICING	1.39	1.39	1.00	22.90	4.14	0.0536
GROUP	1.99	1.99	1.00	23.94	5.92	0.0229
STRATEGY:UTTraining	0.01	0.01	1.00	22.56	0.04	0.8354

El entrenamiento previo en desarrollo de pruebas unitarias presenta una influencia positiva para la Calidad(B =1.25) y neutra para la Productividad (B =0). Los efectos no son significativos en ambos casos (B =0.69 y B =0.87). Como se aprecia en la figura 5.207, los participantes que no tenían entrenamiento previo en desarrollo de pruebas unitarias, mejoraron ligeramente la Calidad y Productividad con la estrategia TDD. Aquellos participantes que indicaron tener entrenamiento previo en desarrollo de pruebas unitarias, no mejoraron en Calidad y Productividad al aplicar tanto $ITLD$ como TDD.



(a) Calidad (b) Productividad

Figura 5.101: Efecto del entrenamiento previo en desarrollo de Pruebas Unitarias.

5.7.4.8. Resumen general

Tabla 5.104: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Edad	-2.4	0.12	2.53	0.06
Experiencia profesional	-0.37	0.9	0.92	0.52
Experiencia en Programación	-0.31	0.54	1.5	0.27
Experiencia en Programación Java	0.27	0.45	0.54	0.69
Uso de herramientas de pruebas	1.27	0.72	-7.18	0.55
Conocimiento de Pruebas Unitarias	1.25	0.69	-7.65	0.53
Función actual en la organización	8.64	0.44	**-4.6**	**0.03**

Las tablas 5.216 y 5.217 muestran el resumen de los análisis de subgrupos realizados.

5.7.5. Grado de completitud

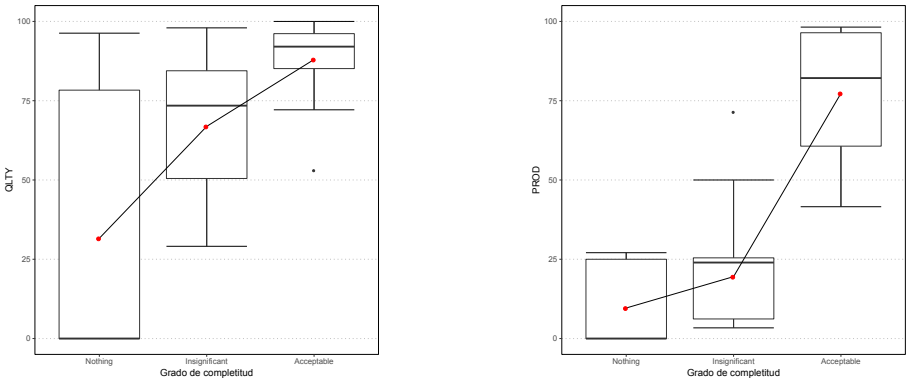
5.7.5.1. Aspectos generales

En la Tabla 5.218 se muestra el total de sujetos que no hicieron nada de la tarea experimental (*Nothing*), los que hicieron una mínima cantidad de trabajo (*Insignificant*) y los que realizaron el trabajo como se esperaba (*Acceptable*). Como en todos los experimentos, esta clasificación obedece a

Tabla 5.105: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Edad	**0**	**0.02**	0	0.2
Experiencia profesional	0	0.47	0	0.44
Experiencia en Programación	0	0.75	0	0.32
Experiencia en Programación Java	0	0.72	0	0.6
Uso de herramientas de pruebas	0	0.96	0	0.69
Conocimiento de Pruebas Unitarias	0	0.87	0	0.84
Función actual en la organización	0.01	0.4	0.02	0.07

una revisión manual realizada al código entregado. Adicionalmente, en la figura 5.208 se presenta la relación entre el grado de completitud frente a las variables QLTY y PROD.



(a) Calidad (b) Productividad

Figura 5.102: Relación entre el grado de completitud y la calidad y productividad

De un total de 53 tareas que fueron entregadas, 8 sujetos no hicieron nada. Por otro lado, 32 participantes hicieron una mínima cantidad de trabajo y 13 sujetos realizaron el trabajo de la forma esperada.

En este caso un 15 % de sujetos entregaron las tareas vacías (*Nothing*), lo cual podría comprometer el análisis aunque en menor medida que en otros experimentos.

	Número de tareas entregadas
Nothing	8
Insignificant	32
Acceptable	13

Tabla 5.106: Grado de completitud

5.7.5.2. Impacto de la estrategia de programación y la tarea experimental

En la tabla 5.219, podemos ver que la totalidad de sujetos que no hicieron nada de las tareas fueron aquellos que utilizaron la técnica ITLD. Por otro lado, un 19 % del total de tareas entregadas fueron calificadas como *Acceptable* con la estrategia TDD, en contraste con el 6 % de tareas calificadas de igual forma al utilizar la estrategia ITLD. Al parecer la *Estrategia de programación* tendría influencia en el grado de completitud de las tareas.

	Nothing	Insignificant	Acceptable
ITLD	8	15	3
TDD	0	17	10

Tabla 5.107: Estrategia de programación

Para verificar la existencia de una posible relación entre la estrategia de programación y el grado de completitud de las tareas, realizamos un test χ^2 que arroja los siguientes resultados: $\chi^2 = 11{,}88$, $df = 2$, $p - value = 0$. Por lo tanto, se comprueba la existencia de una relación estadísticamente significativa entre estos factores.

	Nothing	Insignificant	Acceptable
BSK	3	16	8
MR	5	16	5

Tabla 5.108: Grado de completitud por tarea experimental (BSK, MR)

Por otra parte, la tabla 5.220 muestra el grado de completitud en función de la tarea. En este caso, la distribución es más o menos uniforme, por lo que no se esperaría la existencia de una relación estadíticamente significativa, como lo demuestra el test χ^2 ($\chi^2 = 1{,}17$, $df = 2$, $p - value = 0{,}56$).

5.7.5.3. Impacto de las variables demográficas

En las siguientes secciones analizaremos si el grado de completitud de las tarea se encuentra relacionado con las variables demográficas obtenidas

de los sujetos experimentales. Vamos a analizar las mismas variables de la sección 5.14.4, esto es:

- Edad.
- Experiencia profesional.
- Experiencia en programación.
- Experiencia en Java.
- Uso de herramientas de pruebas.
- Función actual en la organización.
- Entrenamiento previo en desarrollo de pruebas unitarias.

Edad

En la figura 5.209, podemos observar cierta tendencia a que los sujetos de mayor edad entreguen tareas incompletas. Sin embargo, el test Kruskal-Wallis entre la edad y el grado de completitud arroja un p-valor = 0,21, lo cual es insuficiente para confirmar esta tendencia.

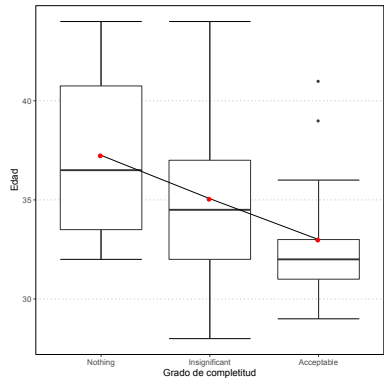


Figura 5.103: Relación entre la edad y el grado de completitud de las tareas

Experiencia profesional

Los años de experiencia al parecer no tienen influencia en el grado de completitud de las tareas, aunque la tendencia es que los sujetos que no hicieron nada fueron los que indicaron tener más años de experiencia (véase la figura 5.210). El test Kruskal-Wallis entre los años de experiencia y el grado de completitud arroja un p-valor = 0,65, el mismo que no es significativo y confirma la impresión visual de la figura 5.210.

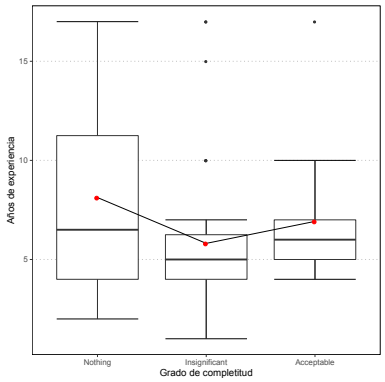


Figura 5.104: Relación entre los años de experiencia profesional y grado de completitud de las tareas

Experiencia en programación

La experiencia programación al parecer no tiene relación con el grado de completitud de las tareas, como se muestra en la Fig. 5.211. Como era de esperarse, los sujetos con más experiencia en programación son los que mejor hicieron las tareas, aunque el test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,47 no significativo.

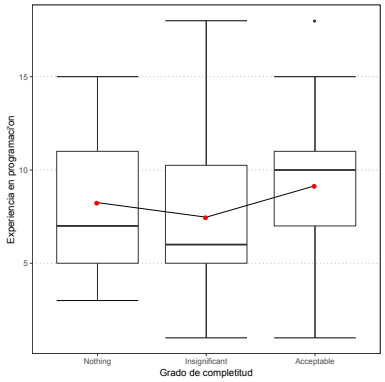


Figura 5.105: programmingExperience

Experiencia en programación Java

La experiencia en desarrollo con el lenguaje Java al igual que la experiencia en programación al parecer influye en el grado de presentación de las tareas (véase la figura 5.212). El test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud nos da un p-valor = 0,45 el cual no es significativo. Aunque los resultados del test no son significativos, sin embargo, podemos notar un mejor desempeño de los estudiantes

con mayor experiencia en Java, lo cual es un resultado previsible.

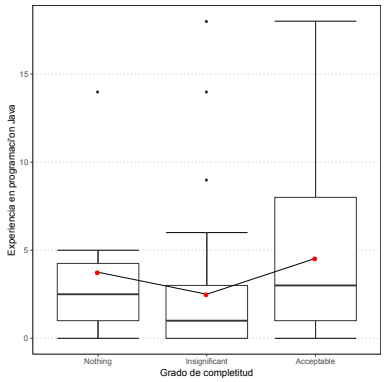


Figura 5.106: javaExperience

Uso de herramientas de prueba

	Nothing	Insignificant	Acceptable
No	6	20	8
Yes	2	12	5

Tabla 5.109: Conocimiento del entorno eclipse

El uso de herramientas de prueba no tiene influencia en el grado de completitud de las tareas experimentales. El test χ^2 nos confirma este hecho ($\chi^2 = 0{,}49$, $df = 2$, $p - value = 0{,}78$).

Función actual en la organización

	Nothing	Insignificant	Acceptable
Analyst	1	5	3
Developer	0	15	3
Manager	2	3	3
Other	5	9	4

Tabla 5.110: Funcion actual en la organización

La función que se encontraban desempeñando los sujetos en la organización tampoc tiene influencia en el grado de completitud de las tareas como podemos comprobar mediante el test χ^2 que arroja resultados no significativos ($\chi^2 = 9{,}2$, $df = 6$, $p - value = 0{,}16$).

Entrenamiento previo en desarrollo de pruebas unitarias

	Nothing	Insignificant	Acceptable
No	6	21	8
Yes	2	11	5

Tabla 5.111: Entrenamiento previo en pruebas unitarias

El entrenamiento previo en el desarrollo de pruebas unitarias tampoco presenta influencia en el grado de completitud de las tareas. El test χ^2 confirma este hecho ($\chi^2 = 0{,}41$, $df = 2$, $p - value = 0{,}82$). Por lo tanto no es significativo.

5.7.5.4. Resumen general impacto variables demográficas

La tabla 5.224 resume los resultados de los análisis de la influencia de las variables demográficas estudiadas frente al grado de completitud de las tareas realizadas. Como se observa, las variables no ejercen un impacto que sea estadísticamente significativo, aunque hemos observado tendencias como la influencia de la edad en el grado de terminación de las tareas: Los sujetos de mayor edad entregaron la mayor parte de tareas vacías o realizaron un trabajo insignificante.

Tabla 5.112: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Edad	0.21
Experiencia profesional	0.65
Experiencia en programación	0.47
Experiencia en Java	0.45
Uso de herramientas de prueba	0.78
Función actual en la organización	0.16
Entrenamiento previo en desarrollo de pruebas unitarias	0.82

5.8. Experimento Quito2016

5.8.1. Ejecución

Este experimento fue realizado en la Universidad de las Fuerzas Armadas ESPE en Sangolquí (Quito - Ecuador) en Mayo de 2016. Los sujetos experimentales fueron profesionales en activo, reclutados mediante invitaciones realizadas en coordinación con representantes de las instituciones participantes.

5.8.1.1. Muestra

En el experimento Quito2016 participaron 23 profesionales, pertenecientes a:

1. Grupos de desarrollo de aplicaciones de software del Comando Conjunto de las Fuerzas Armadas del Ecuador (FF.AA).
2. Desarrolladores de empresas afiliadas a la Asociación Ecuatoriana de software (AESOFT).

Tal y como se observa en la tabla 5.113, la mayoría de los sujetos son desarrolladores menores de 40 años. Casi todos poseen estudios superiores, normalmente en informática. La mayor parte de los sujetos (con sólo 3 excepciones) poseen más de dos años de experiencia profesional y más de 2 años de experiencia en programación. El 65 % ha programado en Java, que es el lenguaje de programación utilizando en esta investigación.

Por otro lado, la mayor parte de sujetos (excepto en tres casos) no han utilizado pruebas unitarias, y tampoco conocen el uso de herramientas de pruebas y del framework Junit. Como era esperable dado lo anterior, los sujetos tampoco tienen experiencia en TDD. Precisamente por este motivo, de forma previa a las sesiones experimentales, hemos formado a los sujetos específicamente en pruebas unitarias y TDD.

5.8.1.2. Preparación

El día previo a la realización del experimento se prepararon los computadores con las herramientas de software necesarias, esto es: la máquina virtual de Java, el framework Junit, el código para empaquetar y realizar mediciones de los experimentos y el IDE de desarrollo Eclipse. A cada participante se le dotó con un computador personal con similares características.

5.8.1.3. Realización

El experimento se realizó durante 4 días, en sesiones de mañana (desarrolladores FF.AA) y tarde (desarrolladores AESOF). Cada sesión tuvo una duración de cuatro horas. Durante las sesiones se impartió formación en TDD y se realizaron las tareas experimentales, conforme al siguiente protocolo:

- **Sesión 1:** En primera instancia, se solicitó que los sujetos llenaran el cuestionario demográfico. A continuación, se llevó a cabo la primera tarea experimental (solucionar BSK, MR o SS aplicando la estrategia YW). Finalmente, se realizó una primera sesión de formación en pruebas unitarias utilizando el framework Junit.
- **Sesión 2:** Se impartió formación en la técnica de ITLD.

Tabla 5.113: Características demográficas de los sujetos del experimento Quito2016

EXPERIMENTO: Quito2016		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	5
	30 >= Edad < 40 años	11
	40 >= Edad < 50 años	6
	Edad >= 50 años	1
Nivel de Educación	Other	1
	Undergraduate	0
	Bachelor	20
Experiencia profesional	Sin experiencia (<2 años)	3
	Novato (2 - 5 años)	8
	Intermedio (6 - 10 años)	6
	Experto (>10 años)	6
Experiencia en programación	Sin experiencia (<2 años)	3
	Novato (2 - 5 años)	11
	Intermedio (6 - 10 años)	7
	Experto (>10 años)	2
Experiencia en lenguaje Java	Sin experiencia (<2 años)	8
	Novato (2 - 5 años)	15
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	22
	Novato (2 - 5 años)	1
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de herramientas de pruebas	Yes	3
	No	20
Uso de la técnica TDD	Yes	1
	No	22
Experiencia en TDD	Sin experiencia (<2 años)	23
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	3
	No	20
Conocimiento del entorno Eclipse	Yes	13
	No	10
Función actual en la organización	Manager	2
	Developer	15
	Analyst	6

- **Sesión 3:** Se realizó la segunda tarea experimental experimental (solucionar BSK, MR o SS aplicando la estrategia ITLD). Posteriormente, se realizó una primera etapa de capacitación en TDD.

- **Sesión 4:** Se realizó una segunda etapa de capacitación en TDD, seguida por la realización de la tercera tarea experimental (solucionar BSK, MR o SS aplicando la estrategia TDD).

Cada tarea experimental duró unas dos horas y media, con leves desviaciones. Para evitar cansancio, cada sesión tuvo un pausa establecida de 20 minutos aproximadamente. Se pidió a los sujetos que no compartiesen información sobre las tareas experimentales durante los recesos. La capacitación estuvo a cargo de Oscar Dieste y Geovanny Raura (proponente de esta tesis).

5.8.1.4. Desviaciones

El experimento se llevó a cabo de acuerdo al protocolo establecido salvo el primer día, donde la sesión de la mañana tuvo que retrasarse unos 30 minutos aproximadamente. La razón fue que que algunos sujetos tuvieron dificultad para encontrar el lugar indicado para la realización del experimento.

5.8.1.5. Reducción del conjunto de datos

Se produjo una reducción progresiva en el número de sujetos que asistieron al experimento a lo largo de los cuatro días de duración del mismo. El primer día se presentaron 23 sujetos, pero sólo 22 sujetos entregaron la primera tarea experimental (que implicaba la aplicación de la estrategia YW). La segunda y tercera tarea experimentales fueron entregadas por 19 y 15 sujetos, respectivamente. Sólo 14 sujetos realizaron en su integridad las tres tareas experimentales de que consta el experimento.

5.8.2. Estadísticos descriptivos

Describiremos por separado cada una de las variables respuestas obtenidas: Calidad y Productividad. Para una mejor compresión representaremos en el mismo gráfico un diagrama box-plot y un diagrama de perfil, donde los puntos rojos representan las medias.

5.8.2.1. Calidad

Como se observa en la tabla 5.114, las calidades son intermedias con desviaciones estándar altas. Ello significa que existen marcadas diferencias entre sujetos (algunos sujetos realizaron la tarea de manera satisfactoria,

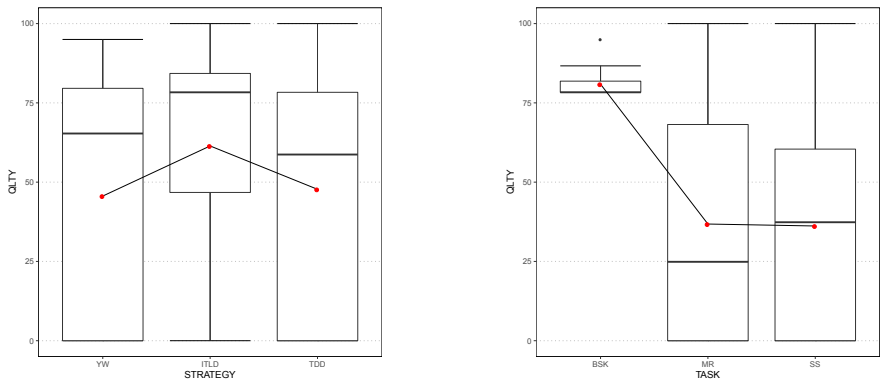
Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
YW	45.62	40.06	-0.20	-1.92
ITLD	61.42	35.88	-0.79	-0.93
TDD	47.74	37.72	-0.28	-1.71

Tabla 5.114: Estadísticos descriptivos para QLTY

pero otros sólo consiguieron resultados deficientes). En promedio, ITLD obtiene un 61 % de Calidad y supera a YW y TDD, aunque no en gran medida (15 y 13 puntos, respectivamente). La figura 5.107a muestra el box-plot para el factor estrategia. La impresión que proporciona este gráfico corrobora lo indicado anteriormente: ITLD parece comportarse mejor que YW y TDD, pero en escasa medida.

La curtosis (o apuntamiento de la curva de distribución) es de tipo platicúrtica o negativa (< 0), e indica que los datos se agrupan poco respecto de la media, lo que refrenda la interpretación inicial de las diferencias entre sujetos. La asimetría es negativa, lo que indica una mayor densidad de puntos a la derecha de la media. Los valores de curtosis son bastante altos, lo que sugiere que pueden existir problemas respecto a la normalidad de los datos.

Las tareas experimentales no tienen un excesivo interés a efectos de investigación, por lo que no adjuntamos una tabla con los estadísticos descriptivos, pero si los box-plot en la figura 5.107b. MR y SS muestran unas medias, medianas y dispersión muy parecidas. En el caso de BSK, los sujetos obtienen valores de calidad muy uniformes. La media de BSK es superior a las de MR y SS. En otras palabras: el box-plot anticipa un posible efecto significativo para la tarea.



(a) Estrategia de Programación (b) Tarea

Figura 5.107: Box plots para la variable respuesta QLTY

5.8.2.2. Productividad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
YW	15.45	17.09	0.81	-0.29
ITLD	13.40	11.30	0.14	-1.86
TDD	14.19	12.97	0.25	-1.39

Tabla 5.115: Estadísticos descriptivos para PROD

En cuanto a la productividad, la tabla 5.115 muestra claramente que los promedios son menores que los de Calidad. Esto indica que, si bien los sujetos han conseguido implementar de forma razonablemente correcta (en promedio) la funcionalidad requerida, lo han hecho sobre un subconjunto reducido de la funcionalidad total (esto es, unas pocas historias de usuario). Ninguna estrategia de programación (YW, ITLD, TDD) destaca sobre las restantes, tal y como se aprecia en la figura 5.108a.

En cuanto a la distribución, se producen dispersiones menores que en el caso de la Calidad, aunque ello era esperable debido a los también menores valores promedio de la productividad. La curtosis es de tipo platicúrtica al igual que para la Calidad. Sin embargo, la Productividad presenta asimetría positiva (aunque no muy marcada), lo que indica que un número considerable de sujetos tienen productividades inferiores al promedio. Los valores son, en todos los casos, cercanos o menores a 1, lo cual sugiere que los valores están normalmente distribuidos.

El impacto de la tarea (BSK, MR o SS) es claramente visible en la figura 5.108b, poseyendo las mismas características que en el caso de la Calidad, esto es, los mejores valores están siempre asociados a la tarea BSK.

En general, los estadísticos descriptivos y box-plot para la Productividad sugieren las mismas conclusiones que para el caso de la Calidad: nulo efecto de la estrategia de programación, y un posible efecto de tarea.

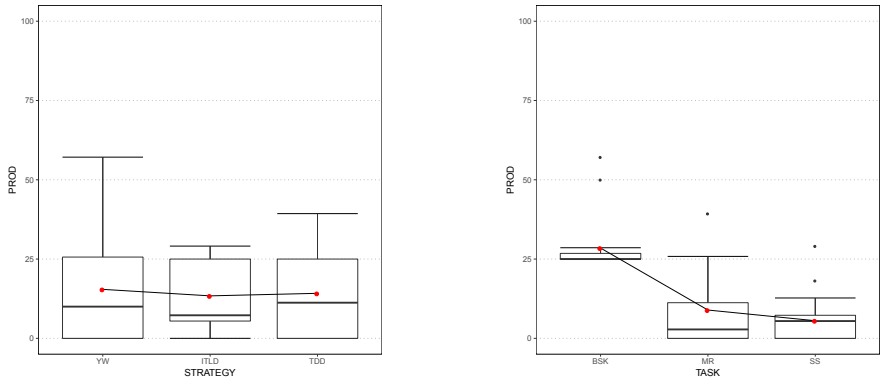
Vale mencionar que en este experimento también queríamos comprobar si el nivel de especificación de la tarea experimental (slicing) tuvo algún efecto sobre la Calidad y Productividad, sin embargo, esto no fue posible porque había pocos sujetos como para ejecutar el experimento con 3 niveles en la estrategia de programación.

5.8.3. Prueba de hipótesis

5.8.3.1. Análisis estadístico

El análisis de los datos experimentales se ha realizado de la forma siguiente (el análisis de la Productividad es similar:

```
# Variable respuesta Calidad
lmq <- lmer(QLTY ~  1 + STRATEGY +
                        TASK +
```



(a) Estrategia de Programación (b) Tarea

Figura 5.108: Box plots para la variable respuesta PROD

```
        GROUP + # Esta variable
            representa la secuencia en que
             las tareas han sido
            realizadas
        (1 | subjectID),
    data = all_expData)
```

GROUP representa la secuencia en que los sujetos realizaron las tareas experimentales. Dicha secuencia no es relevante a efectos de responder a las preguntas de investigación de la presente tesis, pero es necesaria para realizar un análisis correcto de los datos experimentales, tal y como indica Vegas et al. [145]

La tabla 5.116 muestra los resultados del análisis respecto a las variables respuesta Calidad y Productividad en formato de tabla ANOVA. Tal y como se anticipó en la sección 5.8.2, la Tarea arroja resultados significativos, tanto para Calidad (tabla 5.116a) como Productividad (tabla 5.116b). Por el contrario, la Estrategia de Programación también resulta significativa para Calidad (no así para la Productividad). Esto se debe a los mejores resultados de la estrategia ITLD, como ya mencionamos en la sección 5.8.2 y lo comprobaremos seguidamente mediante análisis post-hoc.

La variable GROUP, aunque está acotada a pequeños p-valores, no resulta estadísticamente significativa. Esto implica que la secuencia en que las tareas han sido implementadas por los sujetos experimentales no resulta relevante.

5.8.3.2. Análisis post-hoc

La existencia de resultados significativos en el análisis estadístico exige realizar un análisis post-hoc. Este análisis se muestra, para la calidad, en la

Tabla 5.116: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	5072.89	2536.44	2.00	22.87	3.88	0.0355
TASK	16587.08	8293.54	2.00	23.41	12.67	0.0002
GROUP	4667.11	2333.56	2.00	11.82	3.57	0.0614

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	45.42	22.71	2.00	19.41	0.70	0.5078
TASK	3258.72	1629.36	2.00	19.60	50.36	0.0000
GROUP	152.65	76.33	2.00	20.12	2.36	0.1201

tabla 5.117, mientras que para la Productividad aparece en la tabla 5.118.

El análisis post-hoc confirma que la estrategia ITLD produce código con mayor calidad que YW (unos 27 puntos porcentuales), pero ninguna otra diferencia (por ejemplo: TDD vs. ITLD resulta estadísticamente no significativa). En lo referente a las tareas, las tablas 5.117 y 5.118 muestran claramente que la tarea BSK obtiene mayores valores de Calidad y Productividad que las tareas MR y SS. Esto se apreciaba en los box-plot de la sección 5.8.2. Las diferencias son notables, llegando a producirse hasta 48 puntos de diferencia entre BSK y SS para la Calidad. Las tareas MR y SS se comportan de forma parecida, aunque para la Productividad su diferencia resulta estadísticamente significativa.

Tabla 5.117: Resultados del análisis post-hoc para QLTY

(a) STRATEGY

	Estimate	Std. Error	z value	Pr(>|z|)
ITLD - YW	27.07	9.72	2.78	0.01
TDD - YW	11.03	9.99	1.10	0.51
TDD - ITLD	-16.05	10.80	-1.49	0.30

(b) TASK

	Estimate	Std. Error	z value	Pr(>|z|)
MR - BSK	-34.53	9.85	-3.51	0.00
SS - BSK	-48.11	10.03	-4.79	0.00
SS - MR	-13.58	10.48	-1.30	0.40

Tabla 5.118: Resultados del análisis post-hoc para PROD

	Estimate	Std. Error	z value	Pr(>|z|)
MR - BSK	-15.73	2.48	-6.33	0.00
SS - BSK	-24.15	2.48	-9.73	0.00
SS - MR	-8.41	2.67	-3.15	0.00

5.8.3.3. Chequeo del análisis estadístico

Existen tres condiciones [2] que deben cumplirse para tener la seguridad de que los resultados del análisis estadístico mediante el modelo mixto sean fiables[146]:

1) Existe una relación lineal entre el modelo completo (y cada uno de los factores) y la variable respuesta (para esto seguiremos las recomendaciones de[147]),
2) Homogeneidad de varianzas entre los niveles de los factores, y
3) Los residuos del modelo están normalmente distribuidos.

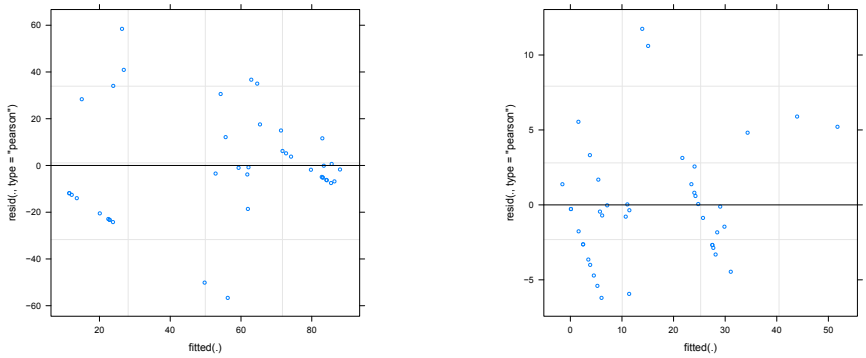
Tal y como describiremos en los apartados siguientes, las condiciones 1) y 2) se cumplen razonablemente, en tanto que la 3) no. Esto pone en cuestión la fiabilidad del análisis mostrado en la tabla 5.116, aunque la significación estadística de la Tarea puede observarse claramente en los box-plot de las figuras 5.107 y 5.108. En cualquier caso, los resultados del análisis deben tomarse con la debida cautela.

Linealidad

La relación lineal entre el modelo y las variables respuesta puede comprobarse mediante un gráfico de dispersión. En el eje X se representa los valores ajustados del modelo, mientras que en el eje Y se muestran los residuos de Pearson de cada variable respuesta (Calidad o Productividad). R proporciona directamente este gráfico mediante el comando *plot*. Los gráficos resultantes se muestran en la figura 5.109.

Cuando existe una relación lineal entre el modelo y la variable respuesta, el gráfico muestra los puntos aleatoriamente distribuidos alrededor del valor $y = 0$, y a lo largo de todos los valores de x. De existir otra relación, por ejemplo cuadrática, los puntos tenderían a formar una gráfica $y = x^2$. En el caso de la figura 5.109, si bien no se observa una nube de puntos, tampoco se observa otro patrón discernible. Por lo tanto, podemos aceptar (o mejor dicho, no rechazar) que las variables respuesta pueden representarse como una función lineal de los factores.

[2] A estas tres condiciones podría añadírsele la condición de independencia. No estudiamos esta condición ya que los datos provienen de un experimento aleatorizado, por lo que independencia debería garantizarse por diseño.

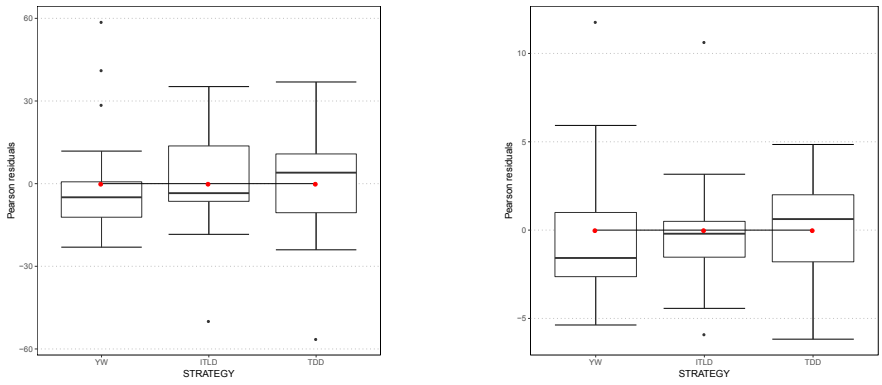


(a) Calidad (b) Productividad

Figura 5.109: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

Una comprobación similar debe realizarse para cada factor del modelo. En este caso, debemos extraer a mano los residuos de Pearson de cada factor antes de generar el gráfico. El código correspondiente es (para el factor STRATEGY):

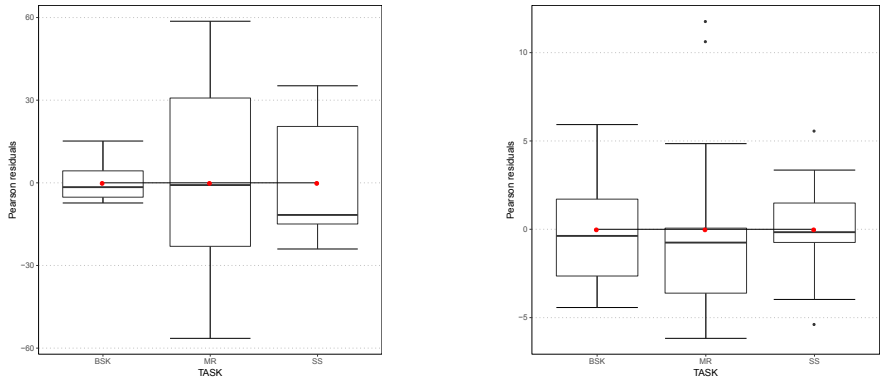
```
tmp <- data.frame(lmq@frame$STRATEGY,
                  resid(lmq, type="pearson"))
```



(a) Calidad (b) Productividad

Figura 5.110: Chequeo de la relación lineal entre el factor Estrategia y los residuos de Pearson

Si las variables independientes hubieran sido predictores continuos, la linealidad podría ponerse de manifiesto mediante una figura similar a la figura 5.109. Al tratarse de factores, no es posible crear un gráfico de dispersión y tendremos que acudir a gráficos de box-plot. Para determinar la linealidad,



(a) Calidad (b) Productividad

Figura 5.111: Chequeo de la relación lineal entre el factor Tarea y los residuos de Pearson

deberemos observar la relación entre las medias/medianas de cada uno de los niveles de los factores y el valor $y = 0$, así como las dispersiones de los mismos (que hacen el papel de "nube de puntos"). Los gráficos resultantes se muestran así:

- Para el factor Estrategia, en la figura 5.110a para Calidad y en la figura 5.110b para la Productividad.

  Podemos observar que, en todos los casos, las medias (puntos rojos) y las medianas (línea central de cada caja) se localizan razonablemente en una línea recta horizontal que pasa por cero. Se observa que a medida que pasamos del factor YW a ITLD y a TDD, la media de los residuos se incrementa. No obstante, observando las dispersiones de los niveles de los factores, puede comprobarse que están razonablemente enfrentadas, sin que aparezcan patrones ascendentes o descendentes que sugieran no linealidad.

- Para el factor Tarea, en las figuras 5.111a y 5.111b para Calidad y Productividad, respectivamente.

  Las medias y medianas están alineadas y cercanas a cero en todos los casos. Las dispersiones son parecidas, con la excepción de ITLD para la Calidad, que es mucho más amplia que las restantes. No obstante, no se observa ningún patrón de no linealidad.

Homogeneidad de varianzas

La homogeneidad de varianzas se estudia utilizando el mismo gráfico mostrado en la figura 5.109, pero esta vez buscando patrones de embudo.

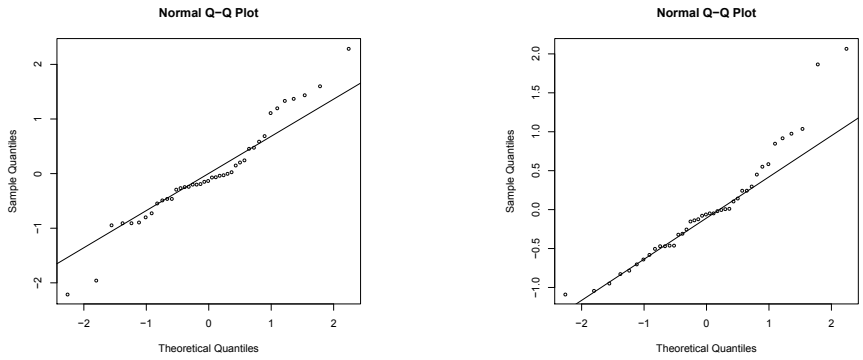
El reducido tamaño muestral perjudica la identificación de patrones claros, sin embargo, no se aprecia la existencia de ninguna figura en embudo que sugiera heterogeneidad de varianzas.

Normalidad de residuos

En los modelos mixtos, es necesario estudiar la normalidad tanto de los factores fijos como de los factores aleatorios (nótese que hay dos fuentes de varianza).

En lo atinente a los factores fijos, tal y como puede apreciarse en los gráficos QQ mostrados en la figura 5.112, la distribución de los residuos de las variables respuesta QLTY y PROD se solapan razonablemente con la distribución normal, representada en la línea recta diagonal, produciéndose desviaciones sólo en los extremos, lo cual es un hecho bastante corriente.

El test de Shapiro-Wilks confirma la normalidad de los residuos para la calidad ($W = 0{,}96$, $p - value = 0{,}12$). Sin embargo, en el caso de la productividad, el test indica que los residuos no están distribuidos normalmente ($W = 0{,}93$, $p - value = 0{,}01$).

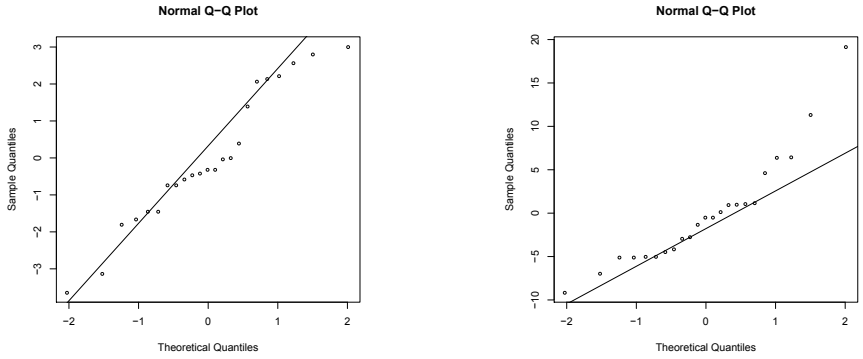


(a) Calidad (b) Productividad

Figura 5.112: Gráficos QQ para los factores fijos

En lo que respecta a los factores aleatorios, la situación es muy parecida a la anterior. Tal y como puede observarse en la figura 5.113, el gráfico QQ muestra que los residuos se separan notablemente de la linea diagonal, pero el test de Shapiro-Wilks confirma la normalidad de la Calidad ($W = 0{,}95$, $p - value = 0{,}26$). Ello no ocurre, al igual que en los factores fijos, para la Productividad ($W = 0{,}89$, $p - value = 0{,}02$).

La falta de normalidad de residuos para la variable respuesta PROD produce que los análisis estadísticos no sean fiables. Cuando esto sucede, lo habitual es repetir el análisis utilizando métodos no paramétricos o aplicar transformaciones a los datos. Hemos aplicado esta última estrategia. Dado



(a) Calidad (b) Productividad

Figura 5.113: Gráficos QQ para los factores aleatorios

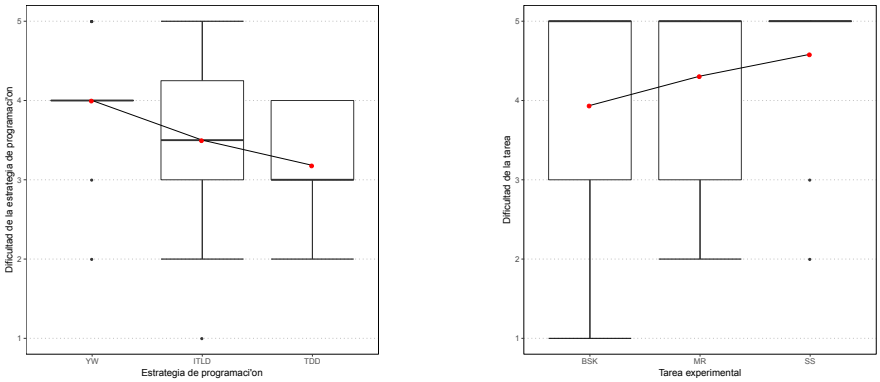
que existen varios ceros en la variable respuesta PROD, hemos optado por usar la transformación "raíz cuadrada" en lugar del logaritmo, para no perder datos. Procediendo de esta forma, conseguimos que todos los residuos de los factores fijos sean normales ($W = 0{,}96$, $p - value = 0{,}16$), como también los aleatorios ($W = 0{,}95$, $p - value = 0{,}26$). Los resultados del análisis no cambian. La única excepción es la diferencia entre las tareas SS y MR, que pasan a ser no significativas, tal como se muestra en la tabla 5.119. De esta forma, los resultados respecto al factor tarea coinciden completamente para Calidad y Productividad.

Tabla 5.119: Resultados del análisis post-hoc para PROD una vez aplicada la transformación *raíz cuadrada*

	Estimate	Std. Error	z value	Pr(>|z|)
MR - BSK	-2.66	0.51	-5.23	0.00
SS - BSK	-3.83	0.51	-7.44	0.00
SS - MR	-1.17	0.54	-2.15	0.08

5.8.3.4. Influencia de factores instrumentales

Los resultados obtenidos en las secciones anteriores podrían deberse a factores instrumentales (no debidos al fenómeno en sí, sino al modo en que se ha investigado, esto es, al diseño experimental). Por una parte, la estrategia de programación no es algo que los sujetos experimentales conozcan de antemano, sino que fue enseñada mediante cursos de formación específicos. Del mismo modo, las tareas fueron seleccionadas específicamente para el experimento. Lo lógico sería esperar que los resultados obtenidos dependan, al menos parcialmente, de la complejidad de las estrategias de programación y



(a) Dificultad percibida de la estrategia de programación

(b) Dificultad percibida de la tarea

Figura 5.114: Efecto de la dificultad percibida por los sujetos

tareas experimentales.

Dificultad de la estrategia de programación

Los sujetos informaron, mediante una encuesta post-experimental, su punto de vista sobre la complejidad de las diferentes estrategias de programación usando una escala de Likert de 1 a 5 puntos (1 = más compleja, 5 = más sencilla). Las medianas y las medias difieren entre categorías, tal y como puede comprobarse en la figura 5.114a. YW es la estrategia considerada más sencilla, mientras que TDD es vista como la más difícil. El test Kruskal-Wallis entre las estrategias de programación y la dificultad percibida arroja un p-valor = 0,03 estadísticamente significativo, lo cual confirma la impresión visual obtenida en la la figura 5.114a.

Dado que existen tres estrategias de programación (YW, ITLD, TDD), la diferencia significativa en la dificultad percibida pueden darse en todas las combinaciones posibles, o sólo en algunas de ellas. A continuación mostramos los resultados de los test de Wilcoxon para cada par de estrategias:

- $p - valor_{YW,ITLD} = 0{,}23$
- $p - valor_{YW,TDD} = 0{,}005$
- $p - valor_{ITLD,TDD} = 0{,}42$

La única diferencia estadísticamente significativa surge entre YW y TDD. En los casos YW-ITLD y ITLD-TDD, las diferencias son menores y no significativas. Estos resultados sugieren que las diferencias entre YW y TDD podrían estar infladas tanto para la Calidad como la Productividad. No obstante, en el caso particular de YW-TDD, las diferencias en Calidad y Productividad no son significativas. Esto impica que es poco probable que

el factor instrumental *Dificultad de la estrategia de programación* esté actuando.

Dificultad de la Tarea experimental

La dificultad de la tarea posee características muy simulares a las indicadas anteriormente para la estrategia de programación, tal y como puede observarse en la figura 5.114b: las medianas son similares y las disperciones relativamente parecidas. Las medias divergen ligeramente, aunque esto debe tomarse con cautela al tratarse de una escala ordinal.

El test Kruskal-Wallis entre las tareas experimentales y la dificultad percibida arroja un p-valor = 0,35, lo cual sugiere que todas las tareas experimentales poseen una complejidad comparable. Esto, al contrario a lo que pueda parecer, no es un resultado sencillo de comprender. Si la dificultad de la tarea no explica las diferencias entre BSK y MR/SS que señalan los análisis estadísticos de la tabla 5.116, entonces otro aspecto relacionado con la tarea está causando dichas diferencias. No es el momento de indagar las causas subyacentes, pero es un aspecto interesante que merece investigar en el futuro.

5.8.4. Análisis de subgrupos

La demografía de los participantes (véase sección 5.8.1.1) indica que existen diferencias sustanciales entre sujetos en lo que respecta a:

- Edad.
- Experiencia profesional.
- Experiencia en programación.
- Experiencia en Java.
- Conocimiento del entorno Eclipse.
- Función actual en la organización.

En las secciones siguientes estudiaremos si la calidad del código o la productividad de los programadores están relacionadas con algunas de las características anteriores. Sería posible estudiar otras variables (las restantes que aparecen en la tabla 5.113). Sin embargo, los sujetos no presentan diferencias sustanciales en dichas características (nivel de educación, uso de herramientas de pruebas, experiencia en el framework jUnit, uso de TDD, experiencia en TDD y entrenamiento en pruebas unitarias), por lo que su análisis resultaría estéril.

5.8.4.1. Edad

Hemos calculado la influencia de la edad sobre la Calidad y la Productividad de la forma siguiente (para la Productividad el análisis es similar):

```
# Variable respuesta Calidad
lmq <- lmer(QLTY ~ 1 +
                   STRATEGY * age +
                   TASK +
                   GROUP +
                   (1 | subjectID),
            data = all_expData)
```

Tabla 5.120: Resultados del análisis estadístico para la Edad

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3616.29	1808.14	2.00	26.02	3.27	0.0539
age	195.66	195.66	1.00	22.03	0.35	0.5578
TASK	7918.95	3959.47	2.00	21.10	7.17	0.0042
GROUP	5417.30	2708.65	2.00	12.13	4.90	0.0275
STRATEGY:age	3498.24	1749.12	2.00	25.78	3.17	0.0589

(b) Productividad

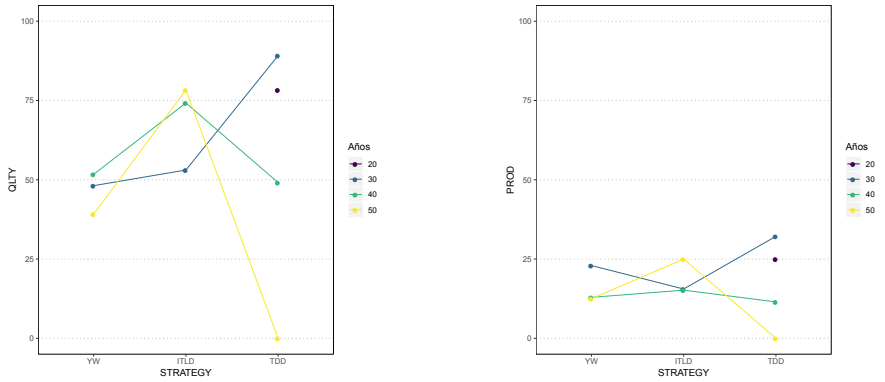
	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	6.09	3.05	2.00	24.43	2.05	0.1501
age	2.09	2.09	1.00	24.39	1.40	0.2473
TASK	59.22	29.61	2.00	20.29	19.95	0.0000
GROUP	10.77	5.38	2.00	16.97	3.63	0.0488
STRATEGY:age	5.65	2.83	2.00	24.00	1.90	0.1708

El resultado del análisis indica que la Edad no posee un efecto significativo ni en solitario ni en interacción con la Estrategia de Programación, tal y como puede observarse en la tabla 5.120. Cabe una pequeña excepción: La interacción $STRATEGY * age$ es casi significativa (p-valor=0.059).

En la tabla 5.120 también puede observarse que el efecto de la Estrategia de Programación pasa a no ser significativo para la Calidad (sin incluir la edad, el resultado era estadísticamente significativo), lo que sugiere la existencia de algún tipo de relación entre la Estrategia de Programación y la Edad (de no existir, lo normal sería que los p-valores disminuyeran al introducir nuevas variables). Creemos que esta relación es precisamente lo que denota la interacción $STARTEGY * age$ indicada más arriba.

Dicha relación también puede observarse gráficamente, aunque con dificultades, en la figura 5.115. Dado que el rango de edades es muy amplio (24 - 52), hemos discretizado la edad en decadas de la forma siguiente:

```
all_expData$ageInDecades <- round(all_expData$age/10)*10
```



(a) Calidad (b) Productividad

Figura 5.115: Efecto de la Edad. La edad se reporta redondeado a décadas. Por ejemplo, el valor 20 representa el rango de 15-24 años de experiencia. El valor 30 representa el rango 25-34, etc. El valor redondeado se escoge en el centro del intervalo para facilitar la lectura de los gráficos y, al mismo tiempo, evitar cambios súbitos.

lo cual produce un gráfico de perfil, más sencillo de interpretar que un scatter plot:

- Si la Edad (sin interacción con la Estrategia de Programación) ejerciese un efecto en la variable respuesta (Calidad o Productividad), entonces deberíamos observar líneas (de colores, cada una correspondiente a una década) en paralelo, apiladas unas encima de otras.
- Si la interacción *STRATEGY * age* ejerciese un efecto en la variable respuesta, deberíamos ver dientes de sierra definidos por un patrón ascendente o descendente de edad.

Las líneas de la figura 5.115 se superponen, no se apilan. Esto implica, tal y como indica la tabla 5.120, que el efecto de la Edad es no significativo tanto para la Calidad como la Productividad.

Sin embargo, si son claramente visibles los dientes de sierra, especialmente para la Calidad. Los sujetos se comportan de manera similar para YW e ITLD, aunque de forma bastante sorpresiva, los resultados para ITLD parecen un poco mejores que los resultados de YW. La clave estriba en la estrategia TDD. Los sujetos de 30 años parecen incrementar su efectividad al usar TDD, mientras que los sujetos de 40 y 50 años muestran eficacias claramente decrecientes. En otras palabras: TDD funciona peor a medida que aumenta la edad de los sujetos. No obstante, al tratarse de efectos no significativos al nivel $\alpha = 0{,}05$, esta tendencia deberá confirmarse en futuros experimentos.

5.8.4.2. Experiencia profesional

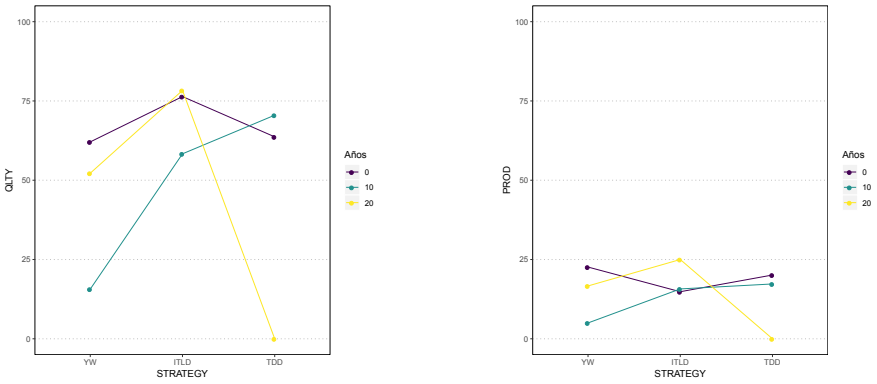
Tabla 5.121: Resultados del análisis estadístico para la Experiencia Profesional

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	2909.15	1454.57	2.00	31.00	2.39	0.1082
experienceYears	2643.65	2643.65	1.00	31.00	4.35	0.0454
TASK	13497.38	6748.69	2.00	31.00	11.09	0.0002
GROUP	5829.75	2914.88	2.00	31.00	4.79	0.0154
STRATEGY:experienceYears	1556.57	778.28	2.00	31.00	1.28	0.2925

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	4.39	2.19	2.00	24.76	1.21	0.3164
experienceYears	7.00	7.00	1.00	17.11	3.85	0.0664
TASK	82.11	41.06	2.00	24.05	22.56	0.0000
GROUP	18.09	9.05	2.00	14.07	4.97	0.0233
STRATEGY:experienceYears	4.20	2.10	2.00	24.55	1.15	0.3317



(a) Calidad (b) Productividad

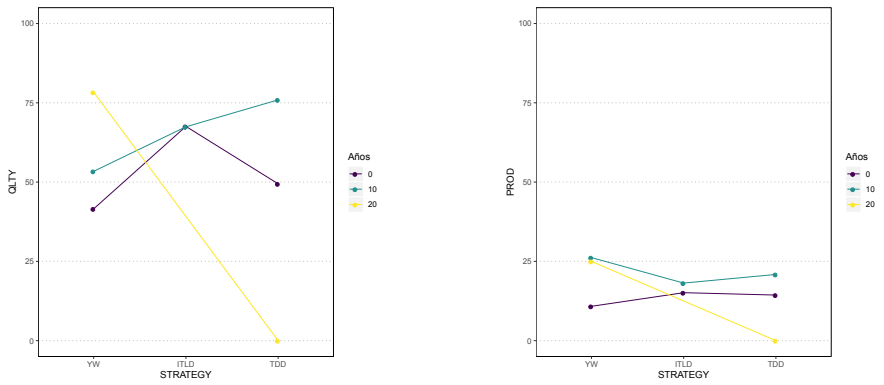
Figura 5.116: Efecto de la experiencia profesional. La experiencia se redondea a décadas, como en el gráfico de perfil anterior.

De forma similar a la Edad, hemos calculado la influencia de la Experiencia Profesional de los sujetos sobre la Calidad y la Productividad. La tabla 5.121 muestra los resultados del análisis. La Experiencia Profesional resulta estadísticamente significativa para la Calidad (nótese que la línea verde es "paralela" a las líneas morada y amarilla), aunque no para la Productividad (las tres líneas se superponen, aunque por muy poco), en la figura 5.116a. Al igual que en el caso de la Edad, una vez introducido el

aspecto personal en el análisis, la Estrategia de Programación deja de ser estadísticamente significativa.

Como la variable *experienceYears* es numérica, el modelo mixto proporciona directamente una estimación de su efecto, equivalente al parámetro B de un modelo de regresión. La Experiencia Profesional posee una influencia negativa ($B = -0{,}79$ puntos/año) para la Calidad

5.8.4.3. Experiencia en programación



(a) Calidad (b) Productividad

Figura 5.117: Efecto de la experiencia en programación. La experiencia se reporta redondeada a décadas.

En el caso de la experiencia en programación, el análisis no proporciona ningún resultado significativo. Por brevedad, no mostraremos la tabla de análisis. Si mostraremos, sin embargo, el gráfico de perfil en la figura 5.117. El gráfico de perfil sugiere que la Experiencia en Programación ejerce un efecto en la Calidad y Productividad similar al de la Experiencia Profesional y la Edad (aunque, es conveniente indicarlo nuevamente, los resultados no son estadísticamente significativos).

De nuevo, una vez introducido el aspecto personal en el análisis, la Estrategia de Programación deja de ser estadísticamente significativa.

5.8.4.4. Experiencia en programación Java

El análisis estadístico mostrado en la tabla 5.122 indica que la Experiencia en el Lenguaje de Programación Java muestra una influencia positiva, con p-valores (a efectos prácticos) estadísticamente significativos ($p - value = 0.01$ y $p - value = 0.06$) tanto para la Calidad como para la Productividad, respectivamente.

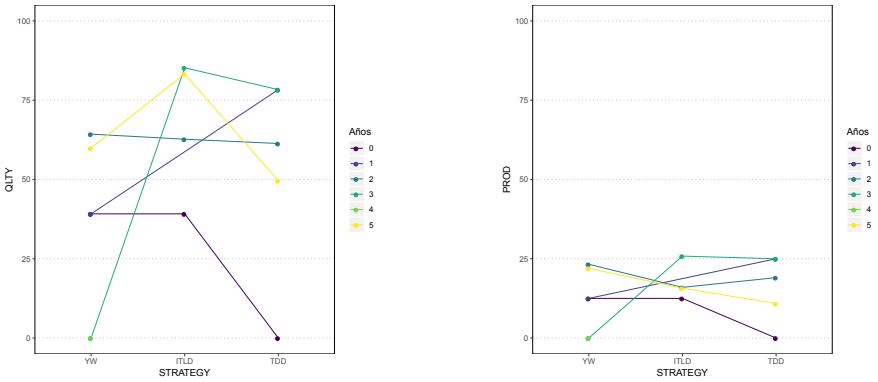
Tabla 5.122: Resultados del análisis estadístico para la Experiencia en Java

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	798.16	399.08	2.00	31.00	0.66	0.5230
javaExperience	4664.94	4664.94	1.00	31.00	7.74	0.0091
TASK	17640.89	8820.45	2.00	31.00	14.63	0.0000
GROUP	6658.24	3329.12	2.00	31.00	5.52	0.0089
STRATEGY:javaExperience	169.91	84.95	2.00	31.00	0.14	0.8691

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	6.78	3.39	2.00	24.21	2.44	0.1081
javaExperience	5.52	5.52	1.00	19.44	3.98	0.0603
TASK	98.16	49.08	2.00	20.93	35.37	0.0000
GROUP	12.21	6.11	2.00	16.71	4.40	0.0291
STRATEGY:javaExperience	2.11	1.05	2.00	23.08	0.76	0.4790



(a) Calidad (b) Productividad

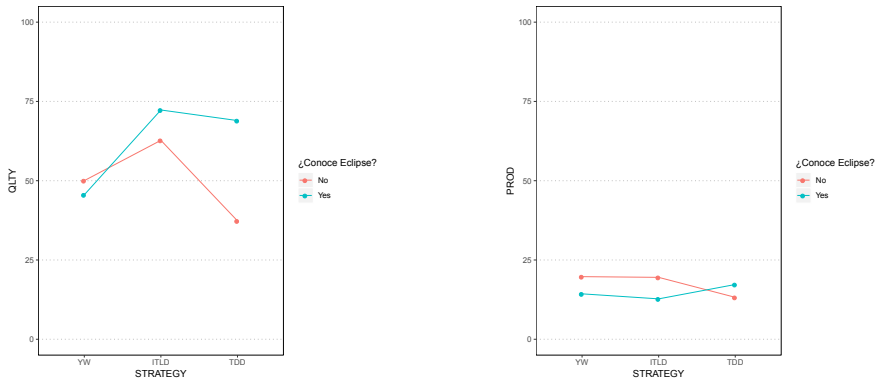
Figura 5.118: Efecto de la experiencia en programación Java. La experiencia se reporta redondeada a años, ya que los sujetos reportan experiencias únicamente entre 0-5 años.

De nuevo, una vez introducido el aspecto personal en el análisis, la Estrategia de Programación deja de ser estadísticamente significativa. Como ya hemos indicado anteriormente, de no existir relación entre la Estrategia de Programación y la Experiencia en el Uso del Lenguaje de Programación Java, dicho p-valor no debería sufrir alteraciones sustanciales y, normalmente, su valor disminuye, no aumenta. Es razonable suponer, por lo tanto, que si existe dicha relación.

En la figura 5.118 puede observarse con una relativa claridad (menor de la que nos gustaría) que a mayor experiencia, mayor Calidad. La productividad está muy igualada. El efecto es considerable para la Calidad ($B = 4.99$, aunque no para la Productividad ($B = 0.18$). Por poner los efectos en contexto, una persona con 10 años de experiencia en Java produciría un código con 49.9 puntos más de calidad que una persona sin experiencia. Se trata sin duda de un efecto muy notable que explica por qué la la Estrategia de Programación deja de ser estadísticamente significativa.

Aunque se observan dientes de sierra sugiriendo una interacción *STRATEGY * javaExperience*, las diferencias no son suficientes para obtener resultados estadísticamente significativos.

5.8.4.5. Conocimiento del entorno Eclipse



(a) Calidad (b) Productividad

Figura 5.119: Efecto del conocimiento del entorno Eclipse.

Otro aspecto analizado es el conocimiento del entorno de desarrollo Eclipse. No adjuntamos la tabla de análisis ya que todos los resultados son no significativos. La figura 5.119 sugiere que el Conocimiento del Entorno Eclipse podría mejorar la Calidad, pero las líneas están bastante superpuestas, lo que resulta plenamente coherente con el análisis estadístico. Adicionalmente, la Estrategia de Programación sigue siendo estadísticamente significativa, lo que pone de manifiesto que el Conocimiento del Entorno Eclipse no tiene

impacto sustancial en la Calidad.

5.8.4.6. Función actual en la organización

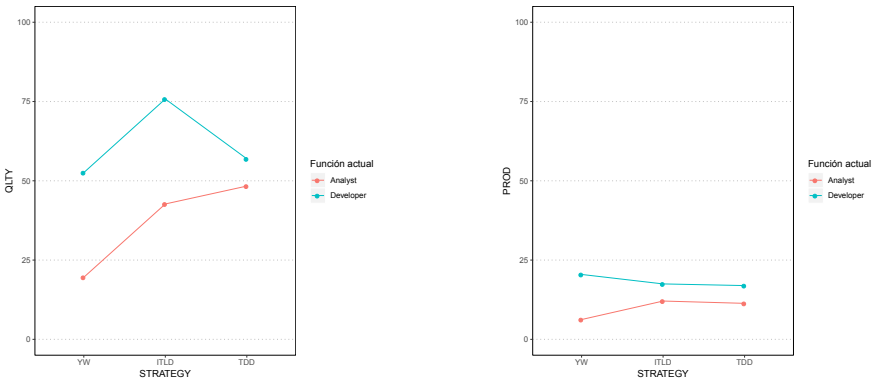
Tabla 5.123: Resultados del análisis estadístico para la Función Actual en la Organización

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3504.07	1752.03	2.00	29.00	2.74	0.0812
currentFunction	2079.52	2079.52	1.00	29.00	3.25	0.0817
TASK	16984.28	8492.14	2.00	29.00	13.29	0.0001
GROUP	4399.93	2199.96	2.00	29.00	3.44	0.0456
STRATEGY:currentFunction	754.39	377.20	2.00	29.00	0.59	0.5607

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	5.36	2.68	2.00	22.04	1.42	0.2623
currentFunction	3.42	3.42	1.00	15.40	1.82	0.1971
TASK	90.97	45.48	2.00	20.26	24.13	0.0000
GROUP	10.19	5.09	2.00	14.01	2.70	0.1017
STRATEGY:currentFunction	0.31	0.16	2.00	21.71	0.08	0.9208



(a) Calidad (b) Productividad

Figura 5.120: Efecto de la función actual en la organización

Existen tres tipos de perfiles: Gestores, analistas y desarrolladores. Como sólo había dos gestores en la muestra de sujetos experimentales, los hemos eliminado del conjunto de datos y hemos analizado únicamente el comportamiento de los analistas y desarrolladores.

Tal y como se observa en la tabla 5.123, la función que se encontraban desempeñando los sujetos en la organización ejerce un efecto (a efectos

prácticos) estadísticamente significativo en la Calidad, aunque no en la Productividad. Al igual que en otros casos, el p-valor de la Estrategia de Programacion supera, al introducir la característica personal, el nivel $\alpha = 0{,}05$.

Este efecto se muestra en la figura 5.120, donde se observa claramente que los desarrolladores (línea verde) consiguen mejores resultados que los analistas (línea roja). Las interacciones no son estadísticamente significativas. En general, los resultados se parecen mucho y, probablemente, están relacionados, con la Experiencia en Programación Java.

5.8.4.7. Resumen general

Tabla 5.124: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Variable		STRATEGY * Variable			
	B	p-value	B ITLD	p-value ITLD	B TDD	p-value TDD
Edad	0.52	0.56	0.72	0.68	**-3.64**	**0.02**
Experiencia Profesional	**-0.79**	**0.05**	0.32	0.86	-2.13	0.16
Experiencia en Programación	-0.58	0.92	3.54	0.21	-2.13	0.31
Experiencia en Programación Java	**4.99**	**0.01**	2.66	0.6	1.4	0.8
Conocimiento del Entorno Eclipse	6.98	0.1	13.68	0.49	13.97	0.5
Función Actual en la Organización	**16.38**	**0.08**	15.44	0.52	-11.93	0.6

Las tablas 5.124 y 5.125 muestran el resumen de los análisis de subgrupos realizados. Estrictamente hablando, la mayor parte de las variables estudiadas no ejercen ningún impacto en la Calidad o la Productividad, con excepción de la Experiencia Profesional y la Experiencia en Programación Java. Relajando un poco la condición $\alpha = 0{,}05$, otras variables (resaltadas en negrita en las tablas) pueden considerarse influyentes. Los resultados para la Calidad y Productividad son coherentes entre sí.

Todos los aspectos personales estudiados tienen una cierta relación entre sí. La Edad, y las distintas experiencias, comparten la dimensión *tiempo*. Esto produce que exista cierta correlación entre las mismas. Por ejemplo, la existencia de efectos significativos relacionados con la Experiencia Profesional (negativos) y con la Experiencia en Java (positivos) se podría explicar porque los sujetos más jóvenes y con menor experiencia profesional, generalmente tienen mayor conocimiento del lenguaje Java.

Tabla 5.125: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Variable		STRATEGY * Variable			
	B	p-value	B ITLD	p-value ITLD	B TDD	p-value TDD
Edad	0	0.25	0	0.78	**0.03**	**0.06**
Experiencia Profesional	**0**	**0.07**	0	0.59	0.01	0.27
Experiencia en Programación	0	0.8	0.03	0.23	0.02	0.21
Experiencia en Programación Java	**0.18**	**0.06**	0.1	0.24	0	0.84
Conocimiento del Entorno Eclipse	0.04	0.33	0.1	0.76	0.88	0.39
Función Actual en la Organización	1.11	0.2	0	1	0.24	0.71

Del mismo modo, se podría explicar el menor desempeño de los analistas por la misma razón anterior, aunque invertida: los analistas son habitualmente de mayor edad que los desarrolladores, y por lo tanto con menor experiencia profesional.

De todas formas, un único experimento proporciona evidencia limitada acerca del fenómeno de estudio. Deberemos aguardar hasta el meta-análisis del Capítulo 6 para determinar si los efectos observados son consistentes a lo largo del conjunto de experimentos realizados.

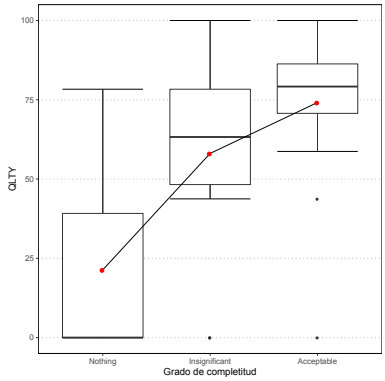
5.8.5. Grado de completitud

5.8.5.1. Aspectos generales

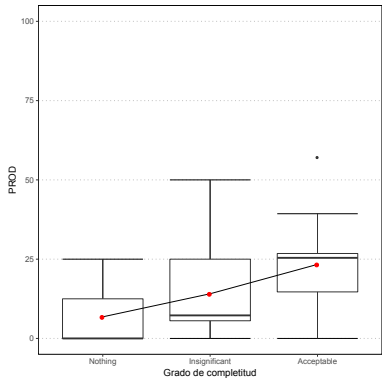
Además de la medición de las variables respuesta, el código que entregaron los sujetos fue revisado manualmente, independientemente de las puntuaciones obtenidas para las variables Calidad y Productividad. Inmediatamente pudimos percibir que algunos sujetos habían trabajado considerablemente, mientras que otros apenas habían tocado el código. En algunos casos, los sujetos devolvieron exactamente el mismo código que habían recibido al inicio de la tarea experimental.

Para estudiar este fenómeno, a cada tarea entregada por los sujetos se le asignó una etiqueta: *Nothing*, cuando el código entregado fue *prácticamente* el mismo código que el descargado de *github*; *Insignificant*, cuando los cambios realizados estaban restringidos a unas pocas líneas de código; y *Aceptable* en los casos restantes. Observe que dichos grados de completitud no implican puntuaciones concretas de QLTY o PROD, sino que en realidad

incluye códigos que exhiben un amplio rango de valores, tal y como puede observarse con claridad en la figura 5.121.



(a) Calidad



(b) Productividad

Figura 5.121: Relación entre el grado de completitud y la calidad y productividad

El número de sujetos que no realizó ningún trabajo sustancial en las tareas experimentales es bastante alto. La Tabla 5.126 muestra los detalles. De un total de 41 tareas recogidas, 11 sujetos no hicieron nada. En 12 casos, los sujetos hicieron una mínima cantidad de trabajo. Únicamente en 18 casos, los sujetos realizaron el trabajo esperado.

5.8.5.2. Impacto de la estrategia de programación y la tarea experimental

Un primer aspecto que deberíamos comprobar es si el grado de completitud de las tareas está relacionado con la estrategia de programación o la tarea experimental. Esto podría sugerir, por ejemplo, si alguna tarea fue especialmente compleja, o si los sujetos dejaron de trabajar a medida que

	Número de tareas entregadas
Nothing	11
Insignificant	12
Acceptable	18

Tabla 5.126: Grado de completitud

progresaban las sesiones experimentales, por citar dos ejemplos.

La *estrategia de programación* no parece estar relacionada con el grado de completitud de las tareas, como se muestra en la tabla 5.127. La estrategia YW muestra un mayor número de tareas vacías, pero esto podría ser explicado por las deserciones que se dieron después de la primera sesión experimental, lo que redujo el número de tareas vacías en la 2^{da} (ITLD) y 3^{ra} sesión (TDD).

	Nothing	Insignificant	Acceptable
YW	6	5	7
ITLD	2	6	4
TDD	3	1	7

Tabla 5.127: Estrategia de programación

Este juicio informal puede corroborarse mediante la realización de un test χ^2, el cual no indica ninguna relación entre la estrategia de programación y el grado de finalización de las tareas ($\chi^2 = 5{,}44$, $df = 4$, $p - value = 0{,}25$).

	Nothing	Insignificant	Acceptable
BSK	3	4	9
MR	3	3	7
SS	5	5	2

Tabla 5.128: Grado de completitud por tarea experimental (BSK, MR, SS)

Por el contrario, grado de completitud si parece estar relacionado con las tareas. SS (que es la tarea considerada más compleja, tal y como puede apreciarse en la figura 5.114b) es la tarea entregada más veces con completitudes de tipo *Nothing* e *Insignificant*. BSK y MR tienen muchas más entregas de tipo *Acceptable*. No obstante, las diferencias entre SS y BSK/MR no son tan elevadas como para que el test χ^2 muestre diferencias significativas ($\chi^2 = 5{,}22$, $df = 4$, $p - value = 0{,}27$).

5.8.5.3. Impacto de las variables demográficas

La sección anterior ha permitido poner de manifiesto que las causas subyacentes al grado de completitud no se deben a los factores experimentales,

sino que deben tener relación con otros aspectos como, por ejemplo, las características personales de los participantes, que es lo que interesa a esta investigación. Esta suposición se basa en el comportamiento mostrado por los sujetos durante la realización de las sesiones experimentales. Varios sujetos abandonaron el experimento de forma temprana, indicando que no estaban interesados en el entrenamiento asociado. Los que permanecieron podrían clasificarse en dos grupos: Aquellos que mostraron interés en los temas enseñados, o los que comenzaron a distraerse en lugar de prestar atención al entrenamiento. Muchos de este último grupo eran profesionales experimentados. Los primeros eran principalmente jóvenes profesionales. Este comportamiento puede relacionarse fácilmente con los resultados de la sección 5.8.4.

En las secciones siguientes estudiaremos si las variables demográficas que hemos recogido, explican el grado de completitud de las tareas entregadas por los sujetos experimentales. Las variables que examinaremos serán las mismas que en la sección 5.8.4, esto es:

- Edad.
- Experiencia profesional.
- Experiencia en programación.
- Experiencia en Java.
- Conocimiento del entorno Eclipse.
- Función actual en la organización.

Edad

La edad parece afectar al grado de terminación de las tareas. Los sujetos de mayor edad mostraron mayor probabilidad de presentar tareas incompletas, lo cual es claramente visible en el gráfico tipo box-plot mostrado en la figura 5.122. No obstante, el test Kruskal-Wallis entre la edad y el grado de completitud arroja un p-valor = 0,16, lo cual es insuficiente para confirmar la impresión visual obtenida en la figura 5.122.

Experiencia profesional

Los años de experiencia (véase la figura 5.123) exhiben un patrón similar al mostrado para la edad, probablemente debido a que la edad y los años de experiencia están moderadamente correlacionados (véase la figura 5.122). No obstante, las diferencias son más acentuadas: el test Kruskal-Wallis entre los años de experiencia y el grado de completitud arroja un p-valor = 0,02, lo cual confirma el impacto (negativo) de la experiencia profesional en el

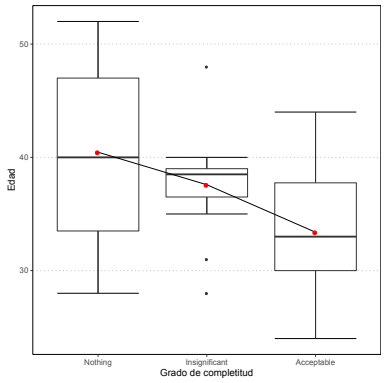


Figura 5.122: Relación entre la edad y el grado de completitud de las tareas

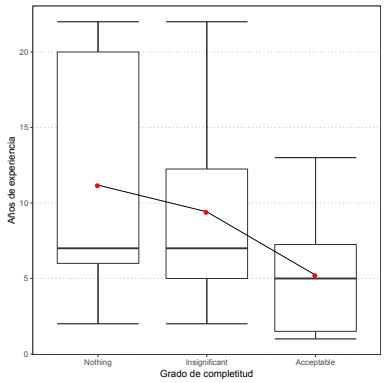


Figura 5.123: Relación entre los años de experiencia profesional y grado de completitud de las tareas

grado de completitud de las tareas: los profesionales con mayor experiencia entregan más frecuentemente tareas con completitud *Nothing* e *Insignificant*.

Experiencia en programación

La experiencia en programación no está fuertemente relacionada con la presentación de tareas vacías, como se muestra en la Fig. 5.124. La mayoría de los sujetos tenían entre 5 y 8 años de experiencia, y la distribución de experiencias fue bastante similar en todas las categorías de terminación de tareas. El test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,38, lo cual confirma la impresión visual obtenida en la figura 5.124.

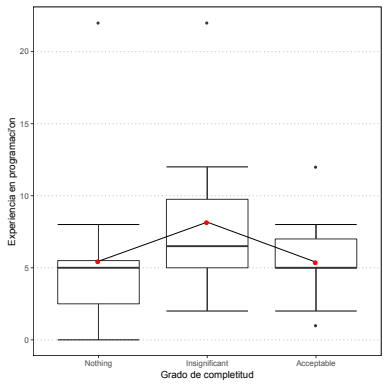


Figura 5.124: Relación entre la experiencia en programación y el grado de completitud de las tareas

Experiencia en programación Java

Por el contrario, la experiencia de programación con el lenguaje Java si influyó en la presentación de tareas incompletas (véase la figura 5.125). El test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud arroja un p-valor = 0,01.

Conocimiento del entorno Eclipse

	Nothing	Insignificant	Acceptable
No	8	5	6
Yes	3	7	12

Tabla 5.129: Conocimiento del entorno eclipse

En cuanto al conocimiento del entorno de desarrollo Eclipse y el grado de completitud de las tareas, los resultados del test χ^2 son no-significativos ($\chi^2 = 4{,}41$, $df = 2$, $p - value = 0{,}11$).

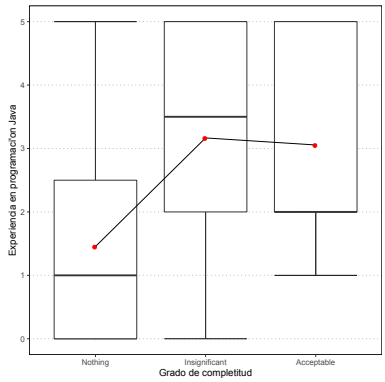


Figura 5.125: Relación entre la experiencia en Java y el grado de completitud de las tareas

Función actual en la organización

	Nothing	Insignificant	Acceptable
Analyst	5	4	1
Developer	5	8	16
Manager	1	0	1

Tabla 5.130: Funcion actual en la organización

De forma similar a lo obtenido para el conocimiento del entorno de desarrollo Eclipse, los resultados del test χ^2 son no significativos ($\chi^2 = 7{,}88$, $df = 4$, $p - value = 0{,}1$) para la función que se encontraban desempeñando los sujetos en la organización.

5.8.5.4. Resumen general impacto variables demográficas

Los resultados del análisis de la influencia de las variables demográficas estudiadas frente al grado de completitud de las tareas realizadas se sintetizan en la tabla 5.131. Como hemos advertido, las variables: *Experiencia profesional* y *Experiencia en Java* tienen influencia significativa en el grado de completitud de las tareas. Estos resultados concuerdan con el análisis realizado en la sección 5.8.4, es decir, se confirma el hecho de que los sujetos menos experimentados son los que más se esfuerzan en completar las tareas experimentales. Además, es razonable suponer que los profesionales más jóvenes tienen un mayor conocimiento del lenguaje Java. En síntesis, los resultados obtenidos de esta replicación nos conducen a pensar que la calidad y productividad de los desarrolladores se han visto mayormente afectadas por las características personales de los programadores (particularmente la

experiencia profesional y la experiencia en Java), antes que por las estrategias de programación ensayadas (YW, ITLD, TDD).

Tabla 5.131: Resumen de la influencia del grado de completitud de las tareas

	p-value
Edad	0.16
Experiencia Profesional	**0.02**
Experiencia en Programación	0.38
Experiencia en Java	**0.01**
Conocimiento del Entorno Eclipse	0.11
Función Actual en la Organización	0.1

5.9. Experimento CostaRicaBabel2016

5.9.1. Ejecución

El experimento Babel2016, al igual que el experimento Quito2016, corresponde a una replicación del experimento base ESPE2015 como se describió en el capítulo de Metodología (véase sección 4.6). El experimento fue realizado entre el 5 de diciembre hasta el 9 de diciembre de 2016 en el Grupo Empresarial Babel Software de San José de Costa Rica. Los sujetos experimentales fueron desarrolladores profesionales empleados de la empresa, quienes participaron como parte de los programas de capacitación continua que se realizan de manera permanente.

5.9.1.1. Muestra

En el experimento Babel2016 participaron 10 profesionales, pertenecientes a la empresa Grupo Babel. En la tabla 5.132, se observa que la mayoría de los sujetos tienen menos de 30 años de edad y poseen estudios superiores (Bachelor) generalmente relacionados al campo de la informática. Sin embargo, la experiencia profesional de la mayoría es inferior a 2 años. Además los sujetos indican tener entre 2 y 5 años de experiencia en programación. En cuanto a la experiencia en el lenguaje Java, el 50 % son novatos (entre 2 y 5 años) y el 50 % restante no tiene experiencia (menos de 2 años) en este lenguaje.

Ningún sujeto indica conocer herramientas de prueba aunque dos sujetos dicen conocer la técnica TDD; sin embargo, ninguno posee experiencia en desarrollo con esta técnica. Tres sujetos indican haber recibido algún tipo de entrenamiento previo en desarrollo con pruebas unitarias y la mayoría conocen el entorno de desarrollo Eclipse que fue utilizado durante la ejecución de los experimentos. La mayoría de sujetos se encontraba desempeñando las

funciones de desarrollador dentro de la empresa. Uno de ellos era Tester y tres ocupaban otras funciones.

Tabla 5.132: Características demográficas de los sujetos del experimento Babel2016

EXPERIMENTO: Babel2016		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	7
	30 >= Edad < 40 años	3
	40 >= Edad < 50 años	0
	Edad >= 50 años	0
Nivel de Educación	Other	1
	Undergraduate	0
	Bachelor	8
Experiencia profesional	Sin experiencia (<2 años)	8
	Novato (2 - 5 años)	2
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Experiencia en programación	Sin experiencia (<2 años)	2
	Novato (2 - 5 años)	7
	Intermedio (6 - 10 años)	1
	Experto (>10 años)	0
Uso de herramientas de pruebas	Yes	0
	No	10
Experiencia en lenguaje Java	Sin experiencia (<2 años)	5
	Novato (2 - 5 años)	5
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	10
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	2
	No	8
Experiencia en TDD	Sin experiencia (<2 años)	10
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	3
	No	7
Conocimiento del entorno Eclipse	Yes	7
	No	3
Función actual en la organización	Tester	1
	Developer	6
	Another	3

5.9.1.2. Preparación

Antes de la realización del experimento, los participantes instalaron en su computador personal las herramientas de software con las mismas características y versiones utilizados en todos los experimentos; esto es: la máquina virtual de Java, el framework Junit, el plugin para realizar mediciones de los experimentos y el IDE de desarrollo Eclipse.

5.9.1.3. Realización

La ejecución del experimento se realizó durante 4 días en sesiones de cuatro horas diarias a partir de las 5 de la tarde en las instalaciones de la empresa. Durante las sesiones se impartió formación en TDD y se realizaron las tareas experimentales, conforme al siguiente protocolo:

- **Sesión 1:** Previo al inicio del entrenamiento los sujetos llenaran el cuestionario demográfico. A continuación, se realizó una primera sesión de introducción al desarrollo ágil y formación en pruebas unitarias utilizando el framework Junit. A diferencia del experimento Quito2016, no se realizó una sesión experimental aplicando la técnica YW.
- **Sesión 2:** Se impartió formación en la técnica de ITLD.
- **Sesión 3:** Se realizó la primera tarea experimental que consistió en solucionar BSK o MR con slicing o sin slicing aplicando la estrategia ITLD. Posteriormente, se realizó una primera etapa de capacitación en TDD.
- **Sesión 4:** Durante esta sesión se realizó una segunda etapa de capacitación en TDD, seguida por la realización de la segunda tarea experimental (la tarea no realizada en la sesión 3).

De forma similar al experimento Quito2016, cada sesión experimental duró unas dos horas y media, con leves desviaciones. Cada sesión tuvo un pausa establecida de 20 minutos aproximadamente para evitar cansancio de los participantes. La capacitación estuvo a cargo de Geovanny Raura con la colaboración de otro investigador (Rodrigo Fonseca). Se utilizaron los materiales de entrenamiento preparados por Oscar Dieste quien también participó como capacitador en el experimento original.

5.9.1.4. Desviaciones

El protocolo se llevó a cabo de acuerdo a lo establecido. Sin embargo, durante las sesiones de entrenamiento, no todos los participantes llegaron puntuales (con un promedio de 15 minutos de retraso) debido principalmente a la densidad de tráfico y la dificultad que existía para movilizarse hacia el lugar de la capacitación en las horas programadas.

5.9.1.5. Reducción del conjunto de datos

Se produjeron variaciones en el número de sujetos que asistieron al experimento a lo largo de los cuatro días de duración del mismo. Se presentaron inicialmente 10 sujetos, pero sólo 9 realizaron la primera tarea experimental (que implicaba la aplicación de la estrategia ITLD). La segunda tarea experimental (aplicación de la estrategia TDD) fue entregada por 7 sujetos. Sólo 6 sujetos realizaron en su integridad las dos tareas experimentales planificadas para este experimento. Este comportamiento se explica de cierta forma ya que algunos profesionales invitados tenían tareas pendientes por resolver propias de su trabajo y por tanto le dieron prioridad a estas actividades antes que a la participación en las sesiones experimentales.

5.9.2. Estadísticos descriptivos

Se describen por separado los estadísticos descriptivos para cada una de las variables respuesta: Calidad y Productividad. En los gráficos mostrados, utilizamos puntos rojos para representar las medias.

5.9.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	54.69	43.79	-0.34	-1.92
TDD	39.73	46.71	0.25	-2.13

Tabla 5.133: Estadísticos descriptivos para QLTY

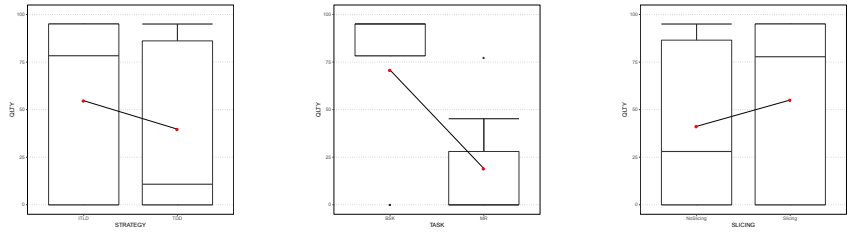
En la tabla 5.133 se observa que la Calidad es intermedia para el caso de la estrategia ITLD y un poco menor para la estrategia TDD, con desviaciones estándar altas en ambos casos. Esto implica que no todos los sujetos hicieron las tareas de forma razonablemente satisfactoria, existiendo marcadas diferencias entre ellos. El solapamiento de las cajas de la figura 5.126a) permite visualizar que es poco probable que la prueba de hipótesis nos de un resultado significativo para la estrategia de programación.

En cuanto a la distribución, la curtosis es de tipo Platicúrtica o negativa (< 0), e indica que los datos se agrupan poco respecto de la media, lo que explica las diferencias entre sujetos. La asimetría en este caso es negativa para ITLD y positiva para TDD, lo que confirma que existe una mayor densidad en la distribución a la izquierda de la media para el primer caso, y hacia la derecha para el segundo. Esto implica que más sujetos estuvieron por debajo del promedio en el caso de ITLD, en tanto que para TDD más sujetos estuvieron por encima del promedio. Como se puede observar en los datos demográficos de la tabla 5.132, un par de participantes afirmaron tener conocimiento previo en TDD, sin embargo dada la limitada cantidad

de sujetos, no podemos afirmar si las diferencias en la QLTY se relacionan con este hecho.

La figura 5.126b muestra el impacto que tiene la tarea (BSK o MR) en la dispersión de resultados. De acuerdo a la gráfica, se advierte que existen diferencias significativas con un alto rendimiento para los sujetos que realizaron BSK, y con un pobre rendimiento de los sujetos que hicieron MR (posteriormente complementaremos el análisis de este hecho, al indagar sobre la dificultad de la tarea percibida por los sujetos).

Por otro lado, como se aprecia en la figura 5.126c, no se observan diferencias significativas para el nivel de definición de la tarea, lo que nos hace suponer que las tareas presentadas con un grado de definición detallado (Slicing), no aportó mayormente en la calidad del código desarrollado por lo sujetos.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.126: Box plots para la variable respuesta QLTY

5.9.2.2. Productividad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	18.86	17.52	0.25	-1.40
TDD	9.61	13.25	0.65	-1.67

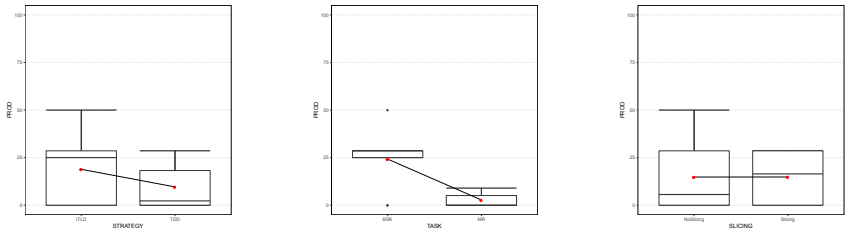
Tabla 5.134: Estadísticos descriptivos para PROD

En cuanto a la Productividad, la tabla 5.134 muestra que los promedios son menores que los de la Calidad. Esto se explica de manera análoga a lo indicado en el experimento Quito2016, donde sólo unas pocas historias de usuario fueron correctamente implementadas. Sin embargo, ITLD destaca en promedio sobre TDD con aproximadamente el doble de diferencia.

Las dispersiones obtenidas para la Productividad son ligeramente menores comparadas con la Calidad. La curtosis es negativa en ambos casos. En cuanto a la asimetría, esta vuelve a tener una valor muy bajo, cercano a cero. La interpretación de éstos estadísticos es similar a lo realizado para la Calidad.

En la figura 5.127, se aprecia el solapamiento entre las distribuciones de los datos de ITLD y TDD. Ello podría producir resultados no significativos durante el análisis, no así para la tarea (BSK o MR), donde se aprecian diferencias en las medias y no existe solapamiento entre las distribuciones, lo que sugiere la existencia de resultados significativos para la tarea (al igual que para la Calidad, BSK produce mejores resultados de Productividad que MR). Por otra parte, tampoco se aprecia una influencia del nivel de definición de la tarea sobre la Productividad aunque el grado de dispersión es mucho menor que para la Calidad.

En síntesis podemos anotar que existe un nulo efecto de la Estrategia de Programación tanto para la QLTY como para la PROD, y un posible efecto de la Tarea (aunque no se ve un efecto significativo de su nivel de definición *slicing*).



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.127: Box plots para la variable respuesta PROD

5.9.3. Prueba de hipótesis

5.9.3.1. Análisis estadístico

En este y los siguientes experimentos se consideró para el análisis, el factor *Slicing* o nivel de definición de la tarea.

```
> lmq <- lmer(QLTY ~  1 +
+                   STRATEGY +
+                   TASK +
+                            SLICING + # Esta variable representa el nivel de d
+                   GROUP +
+                   (1 | subjectID),
+                   data = all_expData
+ )
> lmp <- lmer(PROD ~  1 +
+                   STRATEGY +
+                   TASK +
```

```
+                    SLICING +
+                             GROUP +
+                    (1 | subjectID),
+                    data = all_expData
+ )
```

Los resultados del análisis estadístico respecto a las variables respuesta Calidad y Productividad se muestran en la tabla 5.135. Como ya se anticipó en la sección 5.9.2, sólo la Tarea arroja resultados significativos, tanto para la Calidad (tabla 5.135a) como para la Productividad (tabla 5.135b). Ninguna de las Estrategias de Programación estudiadas mejora la Calidad del código o la Productividad de los programadores. Es notable haber obtenido algún resultado significativo (en este caso la Tarea), dado el escaso número de sujetos experimentales involucrados.

En virtud al diseño experimental considerado para esta replicación, esto es, con dos tareas experimentales (MR y BSK), no es pertinente la realización de un análisis post-hoc; los estadísticos descriptivos muestran claramente que la Calidad y Productividad obtenidos para BSK supera a los de MR.

Tabla 5.135: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	677.88	677.88	1.00	4.60	1.10	0.3466
TASK	8811.96	8811.96	1.00	5.06	14.27	0.0126
SLICING	1951.42	1951.42	1.00	5.10	3.16	0.1344
GROUP	1120.03	1120.03	1.00	7.69	1.81	0.2163

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	298.90	298.90	1.00	11.00	2.60	0.1350
TASK	1661.71	1661.71	1.00	11.00	14.47	0.0029
SLICING	54.26	54.26	1.00	11.00	0.47	0.5062
GROUP	339.23	339.23	1.00	11.00	2.95	0.1137

5.9.3.2. Chequeo del análisis estadístico

Las asunciones del análisis estadístico son prácticamente las mismas que para el experimento Quito2016. En la sección 5.8.3.3, indicábamos que las asunciones de los modelos mixtos eran:

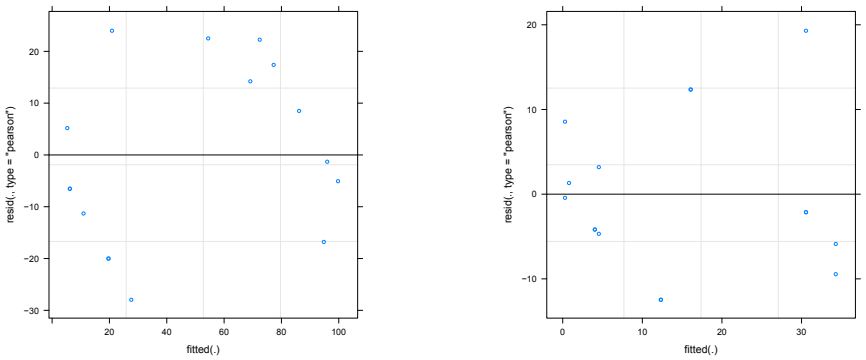
1) Existe una relación lineal entre los factores y la variable respuesta,
2) Homogeneidad de varianzas entre los niveles de los factores, y
3) Los residuos del modelo están normalmente distribuidos.

En el caso de este experimento (y en los restantes experimentos de la tesis) no es necesario comprobar la condición 1) de linealidad. Esto se debe a que los factores tienen únicamente dos niveles, lo que produce que la relación entre ambos sea siempre lineal.

Analizaremos, por tanto, únicamente las condiciones 2) y 3). En este caso, ambas condiciones se cumplen como se demuestra en las siguientes secciones. De ello se deduce que el análisis presentado en la tabla 5.135 es fiable. Al igual que en el experimento Quito2016, se mantiene la significación estadística de la Tarea como sugerían los box-plot de las figuras 5.126 y 5.127.

Homogeneidad de varianzas

El estudio de la homogeneidad de varianzas se realiza de la misma forma que en el experimento Quito2016. En los gráficos presentados en 5.128 no se aprecian patrones de embudo que nos sugiera heterogeneidad entre varianzas tanto para QLTY como para PROD, aunque el reducido tamaño muestral no permite la identificación de patrones claros.



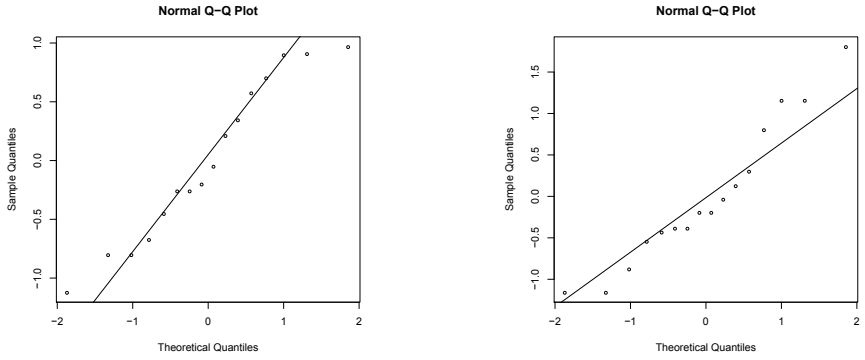
(a) Calidad (b) Productividad

Figura 5.128: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

Normalidad de residuos

Al igual que en el experimento Quito2016, estudiamos la normalidad tanto de los factores fijos como de los factores aleatorios. En lo que respecta a los factores fijos, podemos notar en la figura 5.129 que la distribución de los residuos de las variables respuesta QLTY y PROD se solapan razonablemente con la distribución normal, representada en la línea recta diagonal, con ciertas desviaciones en los extremos, lo cual es un hecho frecuente como se ha comentado anteriormente. Al realizar el test de Shapiro-Wilks confirmamos la normalidad de los residuos tanto para la Calidad ($W = 0{,}94$,

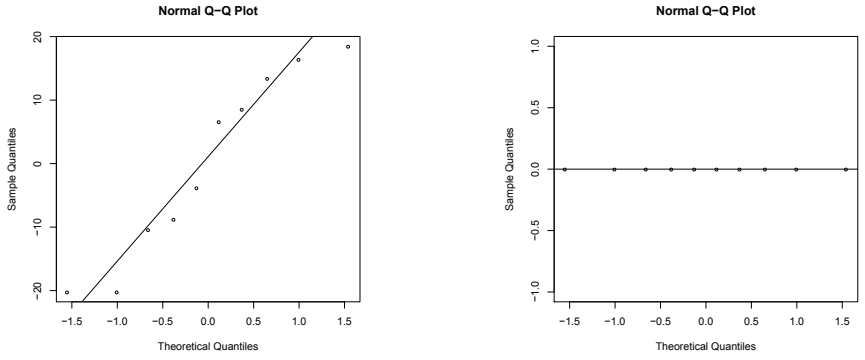
$p-value = 0{,}36$) como para la Productividad ($W = 0{,}94$, $p-value = 0{,}33$).



(a) Calidad (b) Productividad

Figura 5.129: Gráficos QQ para los factores fijos

En lo que respecta a los factores aleatorios, la situación es muy parecida a la anterior. Tal y como puede observarse en la figura 5.130, el gráfico QQ muestra que los residuos se ajustan razonablemente a la distribución normal para QLTY. El test de Shapiro-Wilks nos permite confirmar esta impresión visual ($W = 0{,}91$, $p - value = 0{,}3$). Para la productividad, no es posible realizar tal análisis, ya que los efectos aleatorios son muy pequeños e iguales a cero, tal y como se puede apreciar en la figura 5.129b. Estos ceros se deben a una simplificación que realiza el análisis *lmer* mas no a un problema asociado con los datos experimentales.



(a) Calidad (b) Productividad

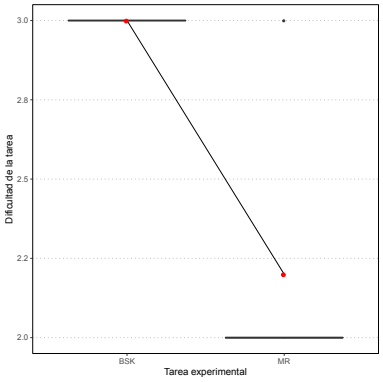
Figura 5.130: Gráficos QQ para los factores aleatorios

5.9.3.3. Influencia de factores instrumentales

Como sucedió en el experimento Quito2016, las tareas fueron seleccionadas ad-hoc siendo razonable pensar que las diferencias entre las tasas de Calidad y Productividad obtenidas al aplicar las estrategias ITLD y TDD pudieran depender en parte de la complejidad de la tarea.

Dificultad de la Tarea experimental

La figura 5.131a muestra la existencia de valores nulos para la estrategia TDD, por tal motivo, no es factible el análisis de la dificultad percibida de la tarea. El test Kruskal-Wallis entre la tarea experimental y la dificultad percibida arroja un p-valor = 0,16.



(a) Dificultad percibida de la tarea

Figura 5.131: Efecto de la dificultad de la tarea percibida por los sujetos

5.9.4. Análisis de subgrupos

De acuerdo a los datos demográficos de los participantes (véase sección 5.9.1.1) podemos establecer que existen diferencias que merecen considerarse en los siguientes aspectos:

- Edad.
- Experiencia profesional.
- Experiencia en programación.
- Experiencia en Java.
- Conocimiento del entorno Eclipse.
- Función actual en la organización.
- Entrenamiento previo en desarrollo de pruebas unitarias.

En lo referente al Uso de la Técnica de TDD, si bien dos personas indicaron que han utilizado esta técnica antes de participar en el experimento, sin embargo en este análisis no se lo ha considerado dada la poca cantidad de sujetos en los grupos. A diferencia del experimento Quito2016, en esta replicación si se incluye el análisis del entrenamiento previo en desarrollo de pruebas unitarias. En las secciones siguientes vamos a tratar de establecer si alguna de estas características guarda relación con la calidad del código o con la productividad de los desarrolladores.

5.9.4.1. Edad

De forma análoga a lo realizado en el experimento Quito2016, hemos analizado la interacción $STRATEGY * age$ y confeccionado un gráfico de perfil para determinar el impacto de la edad de los sujetos en la Calidad y Productividad. El código de R utilizado es el siguiente:

```
> lmq <- lmer(QLTY ~ 1 +
+                     STRATEGY * age +
+                     TASK +
+                     SLICING +
+                     GROUP +
+                     (1 | subjectID),
+              data = all_expData)
> lmp <- lmer(PROD ~  1 +
+                      STRATEGY * age +
+                      TASK  +
+                      SLICING +
+                      GROUP +
+                      (1 | subjectID),
+              data = all_expData)
```

El resultado muestra que la edad posee una influencia negativa para la calidad (B =-1.1) y una influencia positiva para la productividad (B =1.41). Los efectos no son significativos en los dos casos ($P - value$ =0.33 y $P - value$ =0.77).

Sin embargo, es más probable para el caso de la Calidad, que estemos en presencia de un error tipo II provocado por el reducido número de sujetos experimentales. En la figura 5.132 se puede apreciar que la Calidad obtenida por los sujetos dentro del rango de 20 a 29 años de edad supera ampliamente a la del grupo de 30 a 39 años. Este efecto no se observa para la Productividad.

En síntesis, existe cierta tendencia a mejorar la Calidad cuando las personas de mayor edad (de 30 a 39 años) aplican TDD. Por otro lado, la Productividad de todos los participantes tiende a decrecer al aplicar la es-

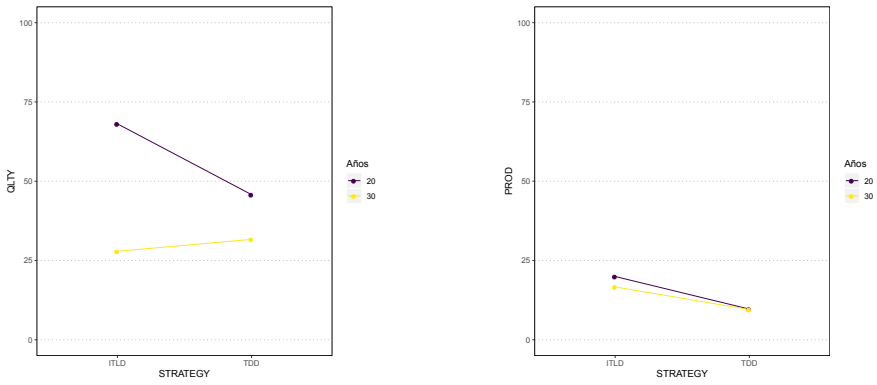
Tabla 5.136: Resultados del análisis estadístico para la Edad

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	338.69	338.69	1.00	5.21	0.60	0.4713
age	627.10	627.10	1.00	6.41	1.12	0.3290
TASK	8266.83	8266.83	1.00	4.02	14.71	0.0184
SLICING	2022.89	2022.89	1.00	4.28	3.60	0.1261
GROUP	982.10	982.10	1.00	6.43	1.75	0.2313
STRATEGY:age	464.14	464.14	1.00	4.86	0.83	0.4064

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	190.03	190.03	1.00	5.73	1.77	0.2340
age	10.09	10.09	1.00	5.95	0.09	0.7696
TASK	1799.22	1799.22	1.00	4.86	16.75	0.0100
SLICING	107.87	107.87	1.00	4.88	1.00	0.3633
GROUP	409.38	409.38	1.00	5.49	3.81	0.1032
STRATEGY:age	263.07	263.07	1.00	5.30	2.45	0.1751



(a) Calidad (b) Productividad

Figura 5.132: Efecto de la Edad. La edad se reporta redondeada a décadas. Por ejemplo, el valor 20 representa el rango de 20 a 29 años de experiencia.

trategia TDD. No obstante, estos resultados no son fiables debido a que no alcanzan la significación estadística.

5.9.4.2. Experiencia profesional

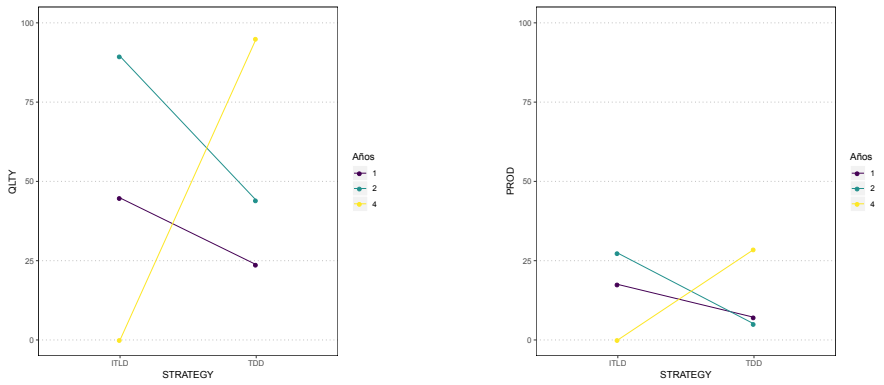
Tabla 5.137: Resultados del análisis estadístico para la Experiencia Profesional

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	2381.65	2381.65	1.00	2.35	31.40	0.0209
experienceYears	65.80	65.80	1.00	6.58	0.87	0.3845
TASK	4487.98	4487.98	1.00	2.30	59.17	0.0110
SLICING	31.71	31.71	1.00	2.44	0.42	0.5736
GROUP	89.72	89.72	1.00	6.81	1.18	0.3138
STRATEGY:experienceYears	1892.06	1892.06	1.00	2.28	24.94	0.0287

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	719.55	719.55	1.00	1.54	61.99	0.0317
experienceYears	1.19	1.19	1.00	5.48	0.10	0.7609
TASK	692.16	692.16	1.00	1.50	59.63	0.0348
SLICING	74.99	74.99	1.00	1.61	6.46	0.1559
GROUP	5.86	5.86	1.00	5.77	0.50	0.5051
STRATEGY:experienceYears	433.01	433.01	1.00	1.49	37.30	0.0499



(a) Calidad (b) Productividad

Figura 5.133: Efecto de la experiencia profesional. La experiencia se reporta redondeada a años. Por ejemplo, 1.5 años de experiencia se redondea a 2. No realizamos una discretación más amplia (lustros, décadas) debido a que todos los valores de ExperienceYears oscilan entre 0 y 4.

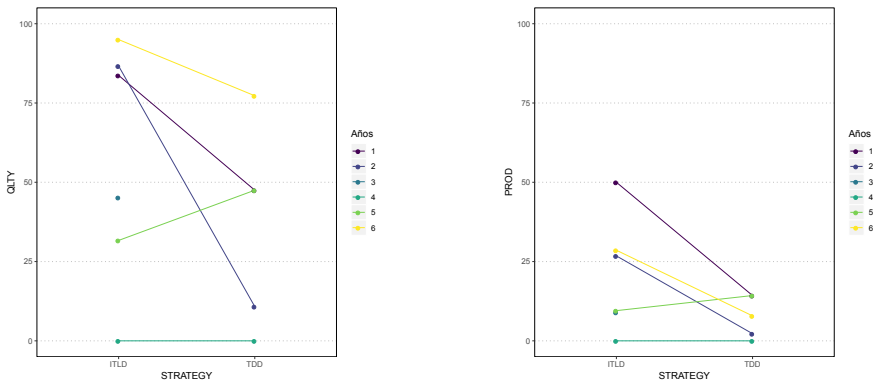
La experiencia profesional posee una influencia negativa tanto para la

Calidad como para la Productividad (B =-5.09, B =-5.73). También se observa que los efectos no son significativos ($p-value$ =0.38 y $p-value$ =0.76) en los dos casos, aunque al observar la figura 5.133 podríamos sugerir respecto a la Calidad, que se está produciendo un error tipo II, esto es, un falso negativo.

Además de los efectos principales, la tabla 5.137 muestra interacciones significativas $STRATEGY * experienceYears$ tanto para la Calidad como para la Productividad. Siendo $STRATEGY$ un factor con dos niveles (ITLD y TDD), y $experienceYears$ una variable tipo ratio, la interacción debe entenderse como el efecto que 1 año de experiencia tiene en TDD, comparado con ITLD. Por ejemplo, el efecto de la interacción es B =29.2 lo que significa que los programadores que usan TDD obtienen 29.2 puntos más que los que usan ITLD. Esta cifra depende de los años de experiencia profesional, esto es, para 2 años de experiencia profesional, el efecto sería $2 \times 29{,}2 = 58{,}4$. Para la productividad, el efecto es $B = 13{,}95$. En ambos casos, los resultados son estadísticamente significativos ($p - value$ =0.03 y $p - value$ =0.05 para la Calidad y Productividad, respectivamente).

Debemos indicar que la interacción $STRATEGY * experienceYears$ antes mencionada, es también visible en el gráfico 5.133. La interacción se manifiesta como el cruce de la línea correspondiente a 4 años de experiencia con las demás. En general, una interacción se manifiesta siempre como un cruce de líneas, pero el análisis puede resultar estadísticamente no significativo (como es el caso, por ejemplo, de la figura 5.135 relacionada con la experiencia en programación Java).

5.9.4.3. Experiencia en programación



(a) Calidad (b) Productividad

Figura 5.134: Efecto de la experiencia en programación. La experiencia se reporta redondeada a años, como en el caso anterior.

Tabla 5.138: Resultados del análisis estadístico para la Experiencia en programación

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	703.20	703.20	1.00	4.08	1.16	0.3413
programmingExperience	18.68	18.68	1.00	6.45	0.03	0.8661
TASK	6191.81	6191.81	1.00	4.05	10.20	0.0325
SLICING	1660.71	1660.71	1.00	3.44	2.74	0.1849
GROUP	822.68	822.68	1.00	6.56	1.36	0.2850
STRATEGY:programmingExperience	320.60	320.60	1.00	3.78	0.53	0.5098

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	153.12	153.12	1.00	9.00	1.21	0.3001
programmingExperience	107.22	107.22	1.00	9.00	0.85	0.3816
TASK	1382.47	1382.47	1.00	9.00	10.91	0.0092
SLICING	23.35	23.35	1.00	9.00	0.18	0.6778
GROUP	225.34	225.34	1.00	9.00	1.78	0.2151
STRATEGY:programmingExperience	24.57	24.57	1.00	9.00	0.19	0.6700

El análisis muestra una influencia negativa para las variables Calidad y Productividad (B = -1.87, B = -2.29). Los efectos no son significativos en ambos casos (p-valor = 0.87 y p-valor = 0.38). En la figura 5.134, se aprecia que en general la Calidad y Productividad decrecen al aplicar TDD, excepto para el grupo de sujetos en el rango de 5 años de experiencia en programación, donde las variables respuesta obtienen mejores resultados al aplicar TDD. No obstante, la falta de un patrón claro en los gráficos de perfil hace que los efectos no se puedan interpretar con claridad.

5.9.4.4. Experiencia en programación Java

La influencia del uso del lenguaje de programación Java es positiva para la Calidad (B =4.61) y negativa para la Productividad (B =-1.23). Vale subrayar que para el caso de la variable Calidad, el efecto se acerca al nivel de significación estadística ($P-value$ =0.06), no así para el caso de la Productividad donde el efecto no es estadísticamente significativo ($P-value$ =0.2). Los gráficos de la figura: 5.135, muestran que los desarrolladores con más experiencia en Java (con 5 o más años), no presentan diferencias sustanciales de Calidad o Productividad al aplicar TDD o ITLD, aunque la Calidad para este grupo de sujetos bordea el 100 %, en contraste con la Productividad que apenas supera el 25 %. Los desarrolladores en el rango de 2 años de experiencia en Java, tienden a mejorar la Calidad y Productividad al aplicar TDD, en cambio los desarrolladores con menos experiencia en Java obtienen peores resultados con TDD. Si bien la Calidad se aproxima al nivel de sig-

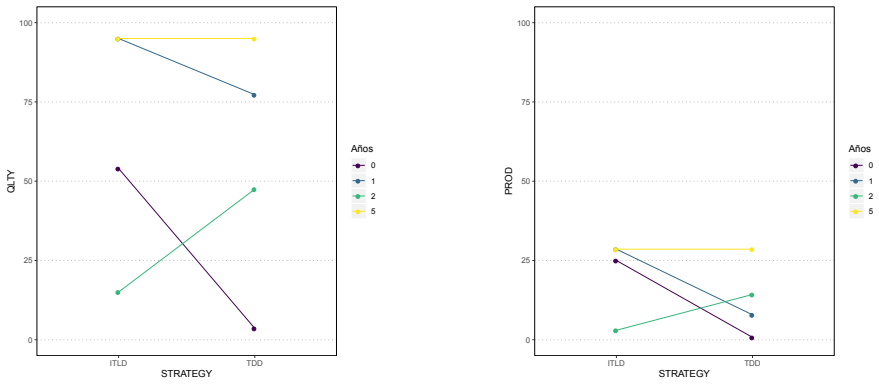
Tabla 5.139: Resultados del análisis estadístico para la Experiencia en Java

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1639.08	1639.08	1.00	4.01	2.24	0.2090
javaExperience	3504.99	3504.99	1.00	8.53	4.78	0.0581
TASK	2903.43	2903.43	1.00	5.02	3.96	0.1030
SLICING	460.52	460.52	1.00	4.20	0.63	0.4703
GROUP	5011.09	5011.09	1.00	5.50	6.84	0.0432
STRATEGY:javaExperience	999.95	999.95	1.00	7.92	1.36	0.2767

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	592.19	592.19	1.00	9.00	6.19	0.0345
javaExperience	179.57	179.57	1.00	9.00	1.88	0.2038
TASK	564.52	564.52	1.00	9.00	5.90	0.0380
SLICING	6.70	6.70	1.00	9.00	0.07	0.7973
GROUP	670.62	670.62	1.00	9.00	7.01	0.0266
STRATEGY:javaExperience	311.95	311.95	1.00	9.00	3.26	0.1044



(a) Calidad (b) Productividad

Figura 5.135: Efecto de la experiencia en programación Java. La experiencia se reporta redondeada a años.

nificación estadística, la poca cantidad de sujetos hace que éstos resultados deban tomarse con cautela.

5.9.4.5. Conocimiento del entorno Eclipse

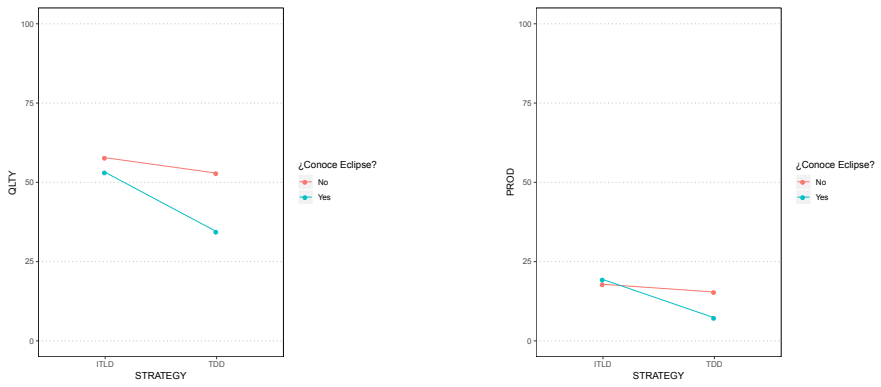
Tabla 5.140: Resultados del análisis estadístico para el conocimiento de entorno Eclipse

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	125.67	125.67	1.00	3.18	0.28	0.6286
knowEclipse	0.88	0.88	1.00	6.79	0.00	0.9657
TASK	8862.56	8862.56	1.00	3.47	20.09	0.0151
SLICING	2312.31	2312.31	1.00	3.42	5.24	0.0952
GROUP	692.17	692.17	1.00	6.84	1.57	0.2515
STRATEGY:knowEclipse	1147.22	1147.22	1.00	3.09	2.60	0.2026

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	109.41	109.41	1.00	3.89	1.30	0.3194
knowEclipse	0.32	0.32	1.00	5.79	0.00	0.9528
TASK	1731.78	1731.78	1.00	4.20	20.59	0.0094
SLICING	106.46	106.46	1.00	3.95	1.27	0.3243
GROUP	198.29	198.29	1.00	5.97	2.36	0.1758
STRATEGY:knowEclipse	268.99	268.99	1.00	3.56	3.20	0.1571



(a) Calidad (b) Productividad

Figura 5.136: Efecto del conocimiento del entorno Eclipse.

El conocimiento del entorno de desarrollo Eclipse presenta una influencia negativa tanto para la Calidad como para la Productividad (B =20.98 y B =9.03 respectivamente). Tal y como se aprecia en la figura 5.136, los

sujetos que tenían conocimiento del entorno Eclipse obtienen peores resultados de Calidad tanto al aplicar ITLD como TDD. De hecho, los sujetos que manejan Eclipse empeoran al usar TDD.

A efectos prácticos, ocurre lo mismo con la Productividad. No obstante, los efectos no son significativos ($p - value$ =0.97 y $p - value$ =0.95).

5.9.4.6. Función actual en la organización

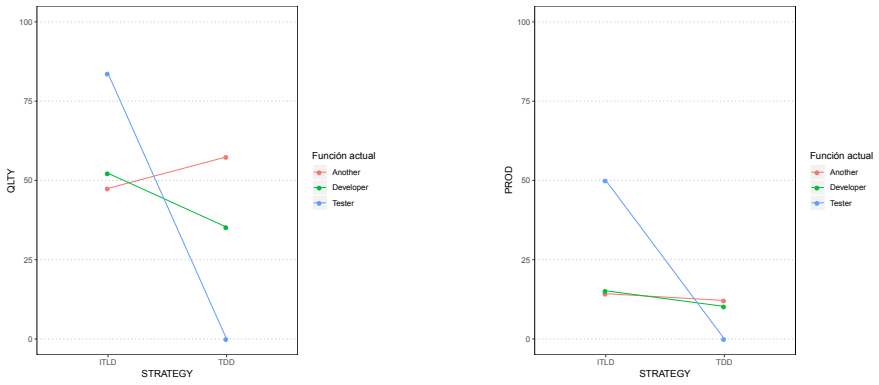
Tabla 5.141: Resultados del análisis estadístico para la Función Actual en la Organización

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1636.87	1636.87	1.00	1.02	12.87	0.1684
currentFunction	68.03	34.01	2.00	5.76	0.27	0.7743
TASK	4199.57	4199.57	1.00	1.09	33.02	0.0954
SLICING	3322.14	3322.14	1.00	1.11	26.12	0.1056
GROUP	224.46	224.46	1.00	5.86	1.76	0.2334
STRATEGY:currentFunction	1852.40	926.20	2.00	2.00	7.28	0.1207

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	697.80	697.80	1.00	1.35	18.00	0.0970
currentFunction	23.68	11.84	2.00	4.97	0.31	0.7498
TASK	752.73	752.73	1.00	1.68	19.41	0.0651
SLICING	189.20	189.20	1.00	1.74	4.88	0.1762
GROUP	47.28	47.28	1.00	5.48	1.22	0.3156
STRATEGY:currentFunction	411.80	205.90	2.00	2.00	5.31	0.1585



(a) Calidad (b) Productividad

Figura 5.137: Efecto de la función actual en la organización

El efecto de la función de los participantes en la organización sobre la Calidad y la Productividad es negativo en los dos casos (B =-5.17 y B =-2.2). Los efectos tampoco son significativos ($p-value$ =0.77 y $p-value$ =0.75).

De acuerdo a los gráficos presentados en 5.137, llama la atención que el único sujeto que indicó cumplir funciones como Tester dentro de la empresa (como se desprende de la tabla **??**, obtuvo los peores resultados tanto en la Calidad como en la Productividad al aplicar TDD. Sin embargo, como se visualiza en los gráficos, el obtener un cero por ciento de efectividad al aplicar TDD en ambos casos nos induce a pensar que el participante posiblemente entregó la tarea en blanco al realizar la segunda sesión experimental (este comportamiento no es nuevo y ya fue identificado en el experimento Quito2016. En la sección 5.9.5 se analizará con más detalle este aspecto).

5.9.4.7. Entrenamiento previo en desarrollo de pruebas unitarias

Tabla 5.142: Resultados del análisis estadístico para el conocimiento de Pruebas Unitarias

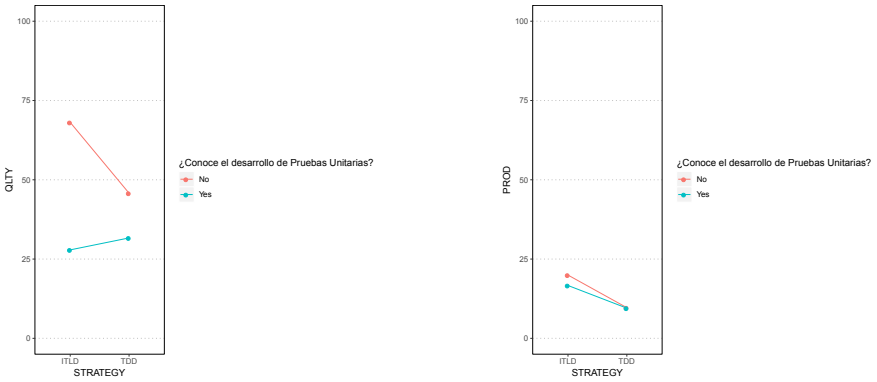
(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	699.49	699.49	1.00	3.46	1.10	0.3614
UTTraining	550.90	550.90	1.00	6.09	0.87	0.3868
TASK	8348.44	8348.44	1.00	4.63	13.16	0.0173
SLICING	2121.03	2121.03	1.00	4.79	3.34	0.1296
GROUP	709.16	709.16	1.00	6.70	1.12	0.3270
STRATEGY:UTTraining	198.02	198.02	1.00	4.38	0.31	0.6037

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	391.34	391.34	1.00	9.00	3.14	0.1104
UTTraining	26.26	26.26	1.00	9.00	0.21	0.6574
TASK	1795.05	1795.05	1.00	9.00	14.38	0.0043
SLICING	88.56	88.56	1.00	9.00	0.71	0.4214
GROUP	389.80	389.80	1.00	9.00	3.12	0.1110
STRATEGY:UTTraining	110.08	110.08	1.00	9.00	0.88	0.3722

El análisis muestra que el entrenamiento previo en desarrollo de pruebas unitarias presenta una influencia negativa para la Calidad(B =-11.88) y positiva para la Productividad (B =8.51). Sin embargo los efectos no son significativos en ambos casos (B =0.39 y B =0.66). En los gráficos presentados en 5.138, se aprecia que aquellos sujetos que indicaron conocer el desarrollo de pruebas unitarias, mejoraron la Calidad al aplicar TDD, al contrario de aquellos que indicaron no conocer el desarrollo de pruebas unitarias. Sin embargo, se aprecia una disminución de la Productividad al aplicar TDD en ambos casos.



(a) Calidad (b) Productividad

Figura 5.138: Efecto del conocimiento de Pruebas Unitarias.

5.9.4.8. Resumen general

Las tablas 5.143 y 5.144 sintetizan los resultados de los análisis de subgrupos realizados. En este caso, únicamente la *Experiencia en Java* se aproxima al nivel de significación para la variable Calidad, no así para la Productividad. Se pueden observar valores notablemente superiores de B para la Calidad con respecto a la Productividad en cuanto al *Conocimiento del Entorno Eclipse* y el *Entrenamiento previo en desarrollo de pruebas unitarias*. Al parecer existe una fuerte correlación entre estas dos variables y la Calidad del código desarrollado. Sin embargo, la limitada cantidad de sujetos no permite alcanzar el poder estadístico suficiente como para que podamos hacer afirmaciones razonables.

Tabla 5.143: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Edad	-1.1	0.33	-3.53	0.41
Experiencia profesional	-5.09	0.38	**29.2**	**0.03**
Experiencia en Programación	-1.87	0.87	5.92	0.51
Experiencia en Programación Java	**4.61**	**0.06**	13.59	0.28
Conocimiento del Entorno Eclipse	20.98	0.97	-39.96	0.2
Conocimiento de Pruebas Unitarias	-11.88	0.39	-16.65	0.6

Tabla 5.144: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Edad	1.41	0.77	-2.36	0.18
Experiencia profesional	-5.73	0.76	**13.95**	**0.05**
Experiencia en Programación	-2.29	0.38	1.47	0.67
Experiencia en Programación Java	-1.23	0.2	7.21	0.1
Conocimiento del Entorno Eclipse	9.03	0.95	-18.82	0.16
Conocimiento de Pruebas Unitarias	8.51	0.66	-11.51	0.37

5.9.5. Grado de completitud

5.9.5.1. Aspectos generales

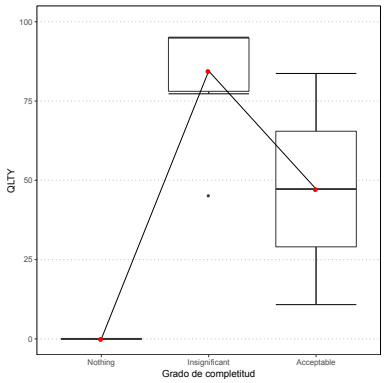
En esta sección analizamos manualmente el código entregado por los sujetos. Como en el caso de Quito2016, se utilizó una escala de Likert designando la calificación de *Nothing* a los sujetos que no hicieron nada de código, textitInsignificant para los que realizaron unas pocas líneas, y *Aceptable* para los que hicieron el trabajo como se esperaba.

En Tabla 5.145 se cuantifica el grado de completitud del trabajo realizado. De un total de 16 tareas recogidas, 6 sujetos no hicieron nada, 8 sujetos hicieron una mínima cantidad de trabajo y 2 sujeto realizó el trabajo como era esperado.

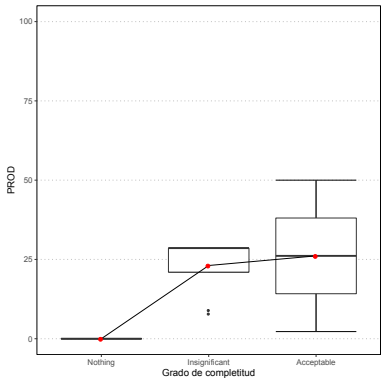
	Número de tareas entregadas
Nothing	6
Insignificant	8
Acceptable	2

Tabla 5.145: Grado de completitud

Podemos advertir que el comportamiento de los sujetos influye en la QLTY y PROD. A modo de ejemplo, el promedio de la QLTY de la primera sesión, incluidos los sujetos que no realizaron un trabajo sustancial, es del 54.69 %; cuando se excluyen estos sujetos, la QLTY aumenta a 83.7 %. El incluir las tareas vacías en el análisis de datos puede comprometer significativamente la interpretación de los resultados.



(a) Calidad



(b) Productividad

Figura 5.139: Relación entre el grado de completitud y la calidad y productividad

5.9.5.2. Impacto de la estrategia de programación y la tarea experimental

Podemos ver que la *estrategia de programación* no parece estar relacionada con el grado de finalización de las tareas, como se muestra en la tabla 5.146. Tanto la estrategia ITLD como TDD muestran el mismo número de tareas vacías y poco trabajadas. Solo un sujeto hizo la tarea de manera aceptable en la primera sesión experimental (ITLD).

	Nothing	Insignificant	Acceptable
ITLD	3	5	1
TDD	3	3	1

Tabla 5.146: Estrategia de programación

Mediante la realización de un test χ^2, podemos confirmar que no existe una relación entre la estrategia de programación y el grado de finalización de las tareas ($\chi^2 = 0{,}25$, $df = 2$, $p - value = 0{,}88$).

	Nothing	Insignificant	Acceptable
BSK	2	6	1
MR	4	2	1

Tabla 5.147: Grado de completitud por tarea experimental (BSK, MR)

El grado de completitud, por otra parte, parece tener cierta relación con las tareas. BSK es la tarea donde menos sujetos tienen un grado de completitud tipo *Nothing*. Sin embargo las diferencias entre BSK y MR no significativas como muestra el test χ^2 ($\chi^2 = 2{,}46$, $df = 2$, $p - value = 0{,}29$).

5.9.5.3. Impacto de las variables demográficas

En base al análisis realizado en la sección anterior, una vez más hemos podido notar que el grado de completitud de las tareas experimentales aparentemente no está directamente relacionado con los factores experimentales sino más bien con otros aspectos como las características personales de los sujetos. En las siguientes secciones analizaremos si el grado de completitud de las tareas se encuentra relacionado con las variables demográficas. Vamos a analizar las mismas variables demográficas de la sección 5.9.4, esto es:

- Edad.
- Experiencia profesional.
- Experiencia en programación.
- Experiencia en Java.

- Conocimiento del entorno Eclipse.
- Función actual en la organización.
- Entrenamiento previo en desarrollo de pruebas unitarias.

Edad

El gráfico tipo box-plot mostrado en la figura 5.140. muestra que la Edad tiene cierta relación con el grado de completitud de las tareas. Los sujetos de mayor edad fueron los que más presentaron tareas incompletas, con excepción del único sujeto que entregó la tarea de tipo *Acceptable*. El test Kruskal-Wallis entre la edad y el grado de completitud no arroja resultados significativos (p-valor = 0,45).

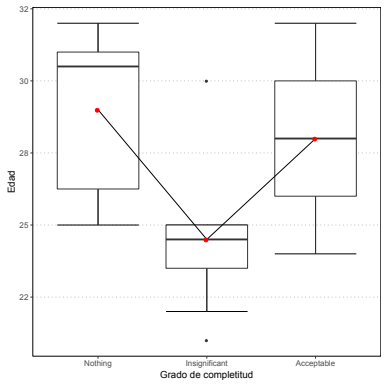


Figura 5.140: age

Experiencia profesional

La experiencia profesional no indica tener influencia en la presentación de las tareas. La figura 5.141 muestra una mayor dispersión para el grado de completitud *Insignificant*, con una mediana inferior a los dos años de experiencia. El test Kruskal-Wallis entre los años de experiencia y el grado de completitud arroja un p-valor = 0,41 no significativo y confirma lo observado en la gráfica 5.141.

Experiencia en programación

En la Fig. 5.142, se aprecia que los sujetos con mayor experiencia en programación son los que menos grado de completitud obtienen. Sin embargo, el test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,22, por lo que no podemos confirmar desde un punto de vista estadístico esta tendencia.

Experiencia en programación Java

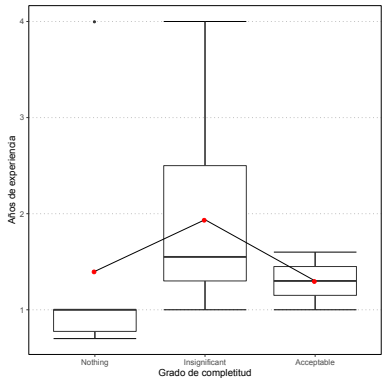


Figura 5.141: ExperienceYears

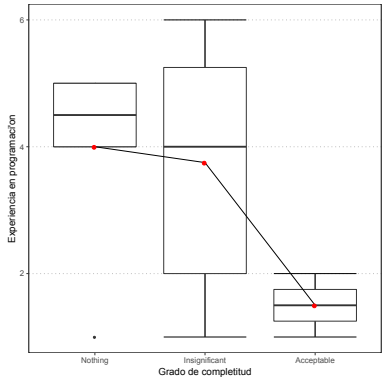


Figura 5.142: programmingExperience

La experiencia de programación con el lenguaje Java no influyó en la presentación de tareas incompletas (véase la figura 5.143). El test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud arroja un p-valor = 0,5, lo cual confirma la impresión visual obtenida.

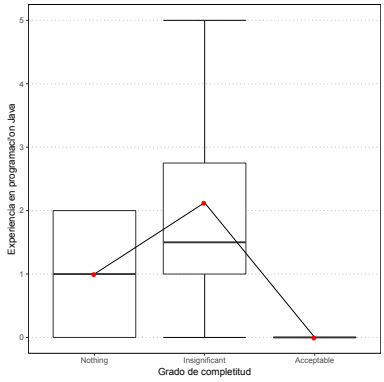


Figura 5.143: javaExperience

Conocimiento del entorno Eclipse

	Nothing	Insignificant	Acceptable
No	1	3	1
Yes	5	5	1

Tabla 5.148: Conocimiento del entorno eclipse

El conocimiento del entorno de desarrollo Eclipse no tiene influencia en el grado de completitud de las tareas experimentales. Los resultados del test χ^2 confirman este hecho ($\chi^2 = 1{,}07$, $df = 2$, $p - value = 0{,}59$).

Función actual en la organización

	Nothing	Insignificant	Acceptable
Another	2	3	0
Developer	3	5	1
Tester	1	0	1

Tabla 5.149: Funcion actual en la organización

En esta instancia experimental, la función que indicaron desempeñar los sujetos dentro de la organización tampoco presenta influencia en el grado de completitud de las tareas. Los resultados del test χ^2 no son significativos ($\chi^2 = 4{,}18$, $df = 4$, $p - value = 0{,}38$). Podemos advertir que el único

sujeto que obtuvo un grado de completitud *Acceptable*, se encontraba desempeñando la función de *Tester* dentro de la empresa, aunque este grado de completitud sólo lo alcanzó en una de las tareas experimentales. El limitado núemero de sujetos participantes no nos permite examinar de manera más objetiva estos resultados.

Entrenamiento previo en desarrollo de pruebas unitarias

	Nothing	Insignificant	Acceptable
No	2	7	1
Yes	4	1	1

Tabla 5.150: Entrenamiento previo en pruebas unitarias

El entrenamiento previo que algunos sujetos indicaron tener en el desarrollo de pruebas unitarias tampoco presenta influencia estadísticamente significativa en el grado de completitud de las tareas. El test χ^2 confirma este hecho ($\chi^2 = 4{,}44$, $df = 2$, $p - value = 0{,}11$).

5.9.5.4. Resumen general impacto variables demográficas

La tabla 5.151 resume los resultados de la influencia de las variables demográficas estudiadas frente al grado de completitud de las tareas realizadas. Como se puede observar, en todos los casos, los p-valores están lejos del nivel de significación. No obstante, el reducido tamaño muestral de esta instancia experimental no permite obtener mayores conclusiones al respecto.

Tabla 5.151: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Edad	0.45
Experiencia profesional	0.41
Experiencia en programación	0.22
Experiencia en Java	0.5
Conocimiento del entorno Eclipse	0.59
Función actual en la organización	0.38
Entrenamiento previo en desarrollo de pruebas unitarias	0.11

5.10. Experimento EcuadorESPE2015

5.10.1. Ejecución

ESPE2015 es el experimento base de la familia de experimentos realizado durante la presente investigación. El experimento fue realizado en la Universidad de las Fuerzas Armadas ESPE de Ecuador con estudiantes de pregrado durante los periodos académicos regulares de clases comprendidos entre abril-agosto del 2015. Este experimento es el primero realizado en el ámbito académico.

5.10.1.1. Muestra

En el experimento ESPE2015 participaron 43 estudiantes del quinto y sexto grado de la Carrera de Ingeniería de Sistemas e Informática en la Universidad de las Fuerzas Armadas ESPE. La tabla 5.152 muestra que la totalidad de los sujetos son jóvenes (<30 años) y con poca experiencia en los diferentes aspectos considerados para este estudio. Como únicas excepciones, podemos indicar que:

- Aproximadamente el 70 % de los sujetos ha utilizado herramientas de prueba.
- Aproximadamente el 50 % de los mismos tiene experiencia en el entorno de desarrollo Eclipse.

5.10.1.2. Preparación

Para la realización de los experimentos se utilizaron los laboratorios de computación de la Universidad, donde se instalaron las herramientas de software necesarias para la realización de los experimentos; esto es: la máquina virtual de Java, el framework Junit, el plugin para empaquetar y realizar mediciones de los experimentos y el IDE de desarrollo Eclipse.

5.10.1.3. Realización

El experimento se realizó como parte de las asignaturas de Desarrollo de software y Programación Avanzada correspondientes al quinto y sexto nivel de la Carrera de Ingeniería en Sistemas e Informática respectivamente. El experimento se realizó en dos días a la semana durante tres semanas dentro del horario de clase de las asignaturas, con un horario de dos horas para el entrenamiento y dos horas con treinta minutos para la realización de las tareas experimentales. Se definió el siguiente protocolo de ejecución:

Tabla 5.152: Características demográficas de los sujetos del experimento ESPE2015

EXPERIMENTO: ESPE2015		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	43
	30 >= Edad < 40 años	43
	40 >= Edad < 50 años	43
	Edad >= 50 años	43
Nivel de Educación	Other	0
	Undergraduate	43
	Bachelor	0
Experiencia profesional	Sin experiencia (<2 años)	43
	Novato (2 - 5 años)	43
	Intermedio (6 - 10 años)	43
	Experto (>10 años)	43
Experiencia en programación	Sin experiencia (<2 años)	3
	Novato (2 - 5 años)	36
	Intermedio (6 - 10 años)	4
	Experto (>10 años)	0
Uso de herramientas de pruebas	Yes	30
	No	13
Experiencia en lenguaje Java	Sin experiencia (<2 años)	5
	Novato (2 - 5 años)	36
	Intermedio (6 - 10 años)	2
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	38
	Novato (2 - 5 años)	5
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	5
	No	38
Experiencia en TDD	Sin experiencia (<2 años)	40
	Novato (2 - 5 años)	3
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	1
	No	42
Conocimiento del entorno Eclipse	Yes	20
	No	23
Función actual en la organización	Tester	0
	Developer	0
	Student	43

- **Semana 1:** Previo al inicio del entrenamiento los sujetos llenaron el cuestionario demográfico. En el día uno de la primera semana se realizó una sesión de introducción al desarrollo ágil y formación en pruebas unitarias utilizando el framework Junit. En el segundo día de la primera semana se capacitó en las técnicas ITLD y slicing.

- **Semana 2:** En el primer día de la segunda semana se realizó la primera sesión experimental, que consistió en la solución de las tareas BSK o MR con o sin slicing y aplicando la técnica ITLD. Posteriormente, en el segundo día de la segunda semana, se realizó una primera sesión de entrenamiento en la técnica TDD.

- **Semana 3:** Durante el primer día de la tercera semana se realizó la segunda fase de capacitación en la técnica TDD. Finalmente, en el segundo día de la tercera semana, se realizó la segunda tarea experimental, en este caso, solucionar BSK o MR con o sin Slicing y aplicando la estrategia TDD.

La capacitación estuvo a cargo de Geovanny Raura con la colaboración de Rodrigo Fonseca, y se utilizaron los materiales de entrenamiento preparados por Oscar Dieste.

5.10.1.4. Desviaciones

En general, el protocolo se cumplió de acuerdo a lo establecido. Al utilizarse los horarios regulares de clases de los estudiantes, las horas de inicio y culminación de las actividades experimentales y de la capacitación se cumplieron dentro de lo establecido con un promedio de cinco minutos de retraso.

5.10.1.5. Reducción del conjunto de datos

Existieron ciertas variaciones en el número de sujetos que asistieron al experimento a lo largo de las tres semanas de duración del mismo. Se presentaron 43, y de ellos 42 sujetos realizaron la primera tarea experimental, es decir la aplicación de la estrategia ITLD. La segunda tarea experimental fue entregada por 33 sujetos quienes aplicaron la estrategia TDD. La reducción de algunos sujetos en la segunda tarea experimental puede explicarse en razón de que esta actividad fue optativa para quienes querían mejorar sus promedios en las asignaturas antes indicadas.

5.10.2. Estadísticos descriptivos

Se describen por separado cada una de las variables respuestas de Calidad y Productividad obtenidas.

5.10.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	60.36	39.27	-0.64	-1.26
TDD	67.00	33.47	-1.31	0.04

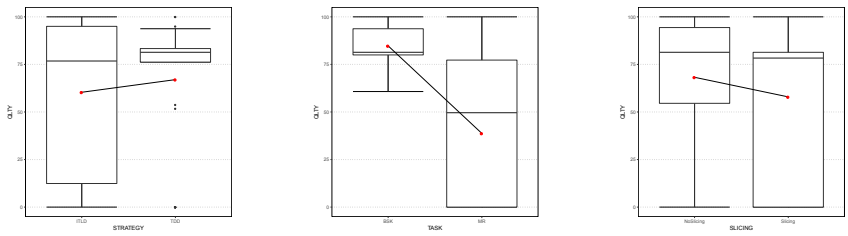
Tabla 5.153: Estadísticos descriptivos para QLTY

La tabla 5.153 muestra que la calidad es intermedia, con desviaciones estándar relativamente elevadas. La asimetría y la curtosis para ITLD y TDD son opuestas y no excesivamente altas; esto ayuda a que los residuos del modelo de análisis se aproximen más fácilmente a la normalidad.

El box-plot mostrado en la figura 5.144 es probablemente más sencillo de comprender. Las medianas (y las medias, tal y como muestra la tabla 5.153), son muy parecidas, pero una mayor cantidad de sujetos obtiene calidades más altas usando TDD. Este hecho podría tener algún impacto en el análisis a favor de TDD.

Por su parte, la tarea BSK obtiene mayores valores de calidad que MR, tal y como indica la figura 5.144b. Esto sugiere que los resultados del análisis pueden resultar significativos para la tarea.

En este experimento ensayamos, además de los factores STRATEGY y TASK, el factor SLICING. El efecto de este factor puede observarse en la figura 5.144c. Como se aprecia, los sujetos que aplican slicing parece que obtienen mejores valores de Calidad que los sujetos que no lo aplican, aunque las diferencias no serán probablemente estadísticamente significativas.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.144: Box plots para la variable respuesta QLTY

5.10.2.2. Productividad

Podemos observar en la tabla 5.154 que los promedios de la variable Productividad son menores que los de Calidad. Esto implica que solo unas pocas historias de usuario fueron correctamente implementadas. La estrategia

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	20.39	24.26	1.42	1.53
TDD	45.05	30.02	-0.27	-1.48

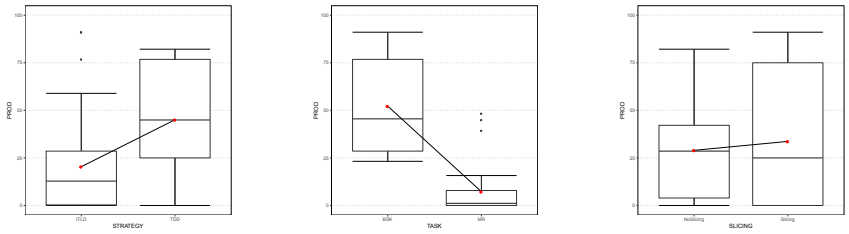
Tabla 5.154: Estadísticos descriptivos para PROD

TDD es superior a ITLD con más del doble de diferencia. Las desviaciones estándar son elevadas.

La asimetría y curtosis es positiva para el caso de ITLD, y negativa para el caso de TDD. Las dispersiones obtenidas son menores que en el caso de la Calidad, aunque no en gran medida al ser comparadas.

En la figura 5.145 se aprecia que, al igual que las medias, las medianas también difieren considerablemente, obteniendo un valor más alto para la estrategia TDD. Se aprecia que existe un impacto de la tarea para la Productividad: la media se encuentra muy superior para BSK en comparación con MR. Por otra parte, el nivel de definición de la tarea (slicing) parece no tener influencia sobre la Productividad.

En conclusión, se puede apreciar un posible efecto de la estrategia de programación, así como un posible efecto de la tarea para la Productividad.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.145: Box plots para la variable respuesta PROD

5.10.3. Prueba de hipótesis

5.10.3.1. Análisis estadístico

El análisis estadístico se ha realizado de igual modo que lo realizado en el experimento Babel2016. Los resultados de la ANOVA respecto a las variables respuesta Calidad y Productividad se muestran en la tabla 5.155. Los resultados son, en general, no significativos con dos excepciones:

- La estrategia de programación ha resultado significativa para la variable respuesta Productividad. No es necesario realizar un análisis post-hoc; la tabla 5.154 indica claramente que la Productividad obtenida con TDD es mayor que la obtenida con ITLD.

- Por otra parte, tal y como anticipábamos en la sección 5.10.2, la tarea arroja resultados significativos tanto para Calidad como Productividad. De nuevo, la tarea BSK obtiene mejores valores que MR.

Tabla 5.155: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	604.80	604.80	1.00	70.00	0.74	0.3914
TASK	39061.50	39061.50	1.00	70.00	48.03	0.0000
SLICING	3088.74	3088.74	1.00	70.00	3.80	0.0553
GROUP	7.22	7.22	1.00	70.00	0.01	0.9252

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3355.23	3355.23	1.00	36.70	13.61	0.0007
TASK	29252.75	29252.75	1.00	36.78	118.65	0.0000
SLICING	395.75	395.75	1.00	37.18	1.61	0.2130
GROUP	191.23	191.23	1.00	40.94	0.78	0.3836

5.10.3.2. Chequeo del análisis estadístico

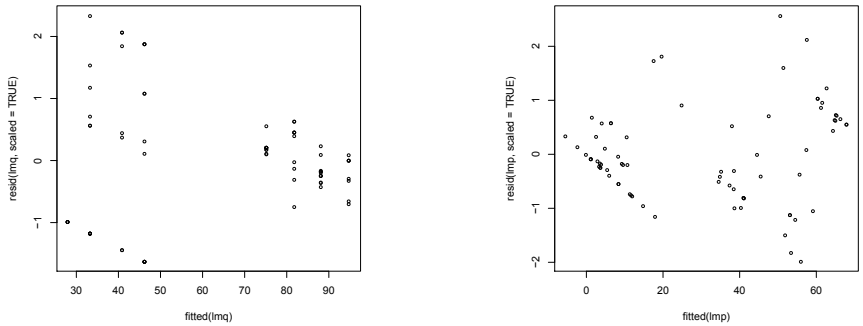
El chequeo del análisis estadístico sigue el mismo patrón que el experimento Babel 2016. Como se ha mencionado, de las tres asunciones dadas para los modelos mixtos (linealidad entre factores, homogeneidad de varianzas y distribución normal de residuos) comprobaremos la segunda y tercera condición ya que en este experimento al igual que en los restantes, los factores tienen solo dos niveles, por lo que la linealidad siempre se cumple.

Homogeneidad de varianzas

Los gráficos de valores predichos vs. residuos estandarizados se muestran en la figura **??**. No se aprecia la existencia de ninguna figura en embudo que sugiera heterogeneidad entre varianzas.

Normalidad de residuos

Estudiamos tanto los factores fijos como los factores aleatorios. En lo que respecta a los factores fijos, podemos apreciar en la figura 5.147, que la distribución de los residuos de la variable respuesta PROD se solapan razonablemente con la distribución normal, representada en la línea recta diagonal. Como ha ocurrido en los experimentos anteriores, notamos que se producen desviaciones sólo en los extremos. En cuanto a la variable QLTY, podemos notar una gran desviación en los extremos.

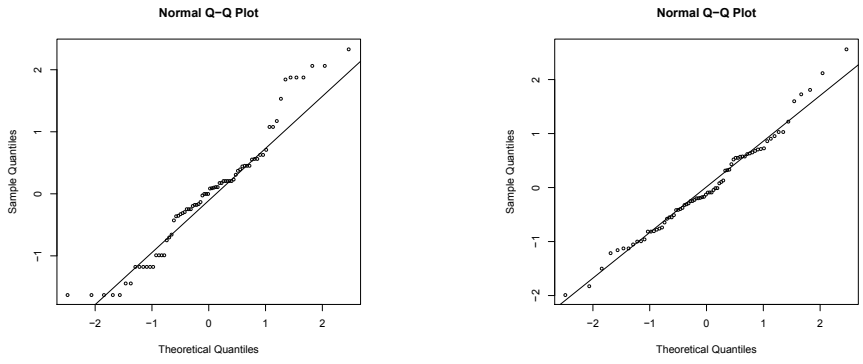


(a) Calidad (b) Productividad

Figura 5.146: Chequeo de la homogeneidad de varianzas

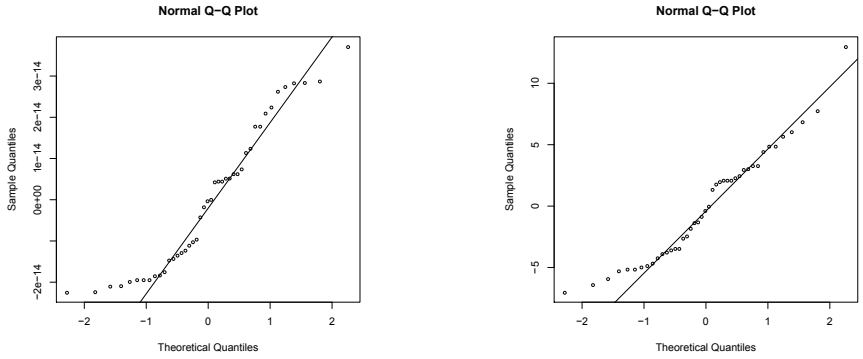
El test de Shapiro-Wilks confirma la normalidad de los residuos para la Productividad ($W = 0{,}98$, $p - value = 0{,}37$). Sin embargo, en el caso de la Calidad, el test indica que los residuos no están distribuidos normalmente como ya lo habíamos advertido ($W = 0{,}96$, $p - value = 0{,}01$). Ninguna de las transformaciones aplicadas ha logrado alcanzar la normalidad. Adicionalmente, la asimetría y curtosis de los datos son en general mayor que 1. En consecuencia, los análisis referidos a la Calidad deben tomarse con cautela.

En relación a los efectos aleatorios, la situación es la misma que en el caso de los factores fijos: Normalidad para la Productividad ($W = 0{,}95$, $p - value = 0{,}07$) pero no para la Calidad ($W = 0{,}92$, $p - value = 0{,}01$).



(a) Calidad (b) Productividad

Figura 5.147: Gráficos QQ para los factores fijos



(a) Calidad (b) Productividad

Figura 5.148: Gráficos QQ para los factores aleatorios

5.10.3.3. Influencia de factores instrumentales

En esta sección analizamos la dificultad de la tarea experimental de acuerdo a la percepción de los participantes. Como hemos advertido, las tareas fueron seleccionadas ad-hoc y por ende los resultados obtenidos podrían deberse al diseño experimental.

Tarea experimental

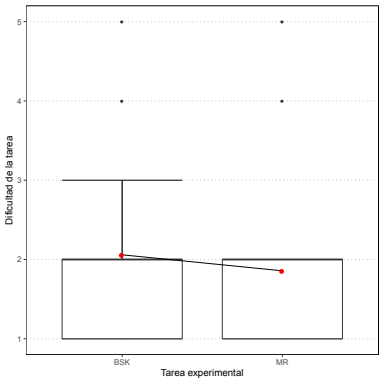
Utilizamos una escala de Likert de 1 a 5 puntos para calificar la complejidad de la tarea, podemos observar en la figura 5.149a, que la complejidad percibida para MR es prácticamente similar a la complejidad de BSK, aunque el análisis estadístico indicó la existencia de diferencias significativas entre BSK-MR tanto para la variable QLTY como para la variable PROD. El test Kruskal-Wallis entre las tarea experimental y la dificultad percibida arroja un p-valor = 0,57, lo cual confirma la impresión visual obtenida en la figura 5.149a.

De acuerdo a la percepción de los sujetos, la dificultad de la tarea no juega un papel relevante, aun cuando estadísticamente existen diferencias significativas. En esta replicación los sujetos llenaron únicamente una de las dos encuestas post experimentales lo cual no nos permite explicar de forma categórica estos resultados.

5.10.4. Análisis de subgrupos

Conforme a los datos demográficos de los participantes (véase sección 5.10.1.1), podemos indicar que existen diferencias que merecen considerarse en los siguientes aspectos:

- Experiencia en programación.



(a) Dificultad percibida de la tarea

Figura 5.149: Efecto de la dificultad de la tarea percibida por los sujetos

- Experiencia en Java.
- Conocimiento del entorno Eclipse.
- Experiencia en el Framework JUnit.

5.10.4.1. Experiencia en programación

Tabla 5.156: Resultados del análisis estadístico para la Experiencia en programación

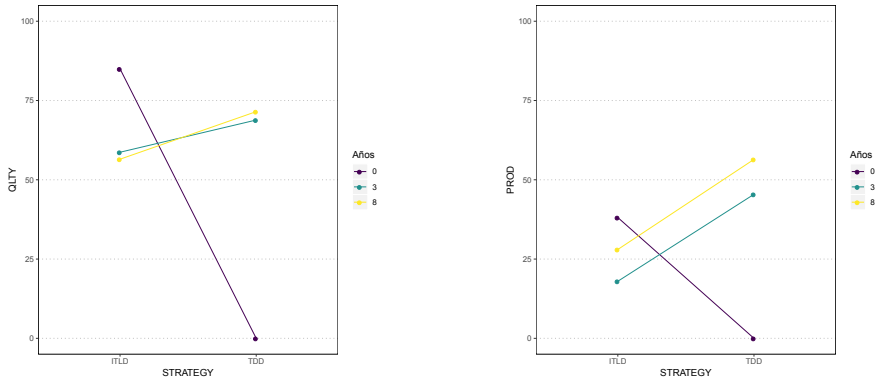
(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	724.71	724.71	1.00	68.00	0.87	0.3541
programmingExperience	36.25	36.25	1.00	68.00	0.04	0.8353
TASK	38024.53	38024.53	1.00	68.00	45.68	0.0000
SLICING	2891.07	2891.07	1.00	68.00	3.47	0.0667
GROUP	9.23	9.23	1.00	68.00	0.01	0.9165
STRATEGY:programmingExperience	309.95	309.95	1.00	68.00	0.37	0.5438

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	167.81	167.81	1.00	36.41	0.70	0.4077
programmingExperience	571.43	571.43	1.00	40.77	2.39	0.1299
TASK	28582.79	28582.79	1.00	35.33	119.48	0.0000
SLICING	456.39	456.39	1.00	35.52	1.91	0.1758
GROUP	130.77	130.77	1.00	39.76	0.55	0.4640
STRATEGY:programmingExperience	217.39	217.39	1.00	35.36	0.91	0.3469

El análisis de la experiencia en programación muestra una influencia negativa para las variable Calidad (B =-0.84) y positiva para la Productividad



(a) Calidad (b) Productividad

Figura 5.150: Efecto de la experiencia en programación. La experiencia se reporta redondeada en años. Por ejemplo, el valor 0.5 representa 1 años de experiencia, mientras que 2.3 representa dos años de experiencia.

(B =1.03). Los efectos no son significativos en ambos casos ($P-valor$ =0.84 y $P-valor$ =0.13). No obstante, los gráficos 5.150a y 5.150b) sugieren que los programadores más experimentados mejoran al aplicar TDD tanto en la Calidad como en la Productividad (nótese las líneas amarillas y verdes). Los programadores nóveles fallan estrepitosamente al aplicar TDD. Hay que recalcar que con la introducción de la experiencia en programación, las variables respuesta dejan de ser significativas. Este es un hecho ya ocurrió en Quito2016 y nos permite advertir que ciertos fatores personales, puden insidir en los resutados obtenidos. Esta tendencia la iremos corroborando durante el análisis de las siguientes instancias experimentales.

5.10.4.2. Experiencia en programación Java

El uso del lenguaje de programación Java presenta una influencia negativa para la Calidad y positiva para la Productividad (B =-1.77 y B =0.66). Los efectos no son significativos ($P-value$ =0.87 y $P-value$ =0.25). En este caso, el gráfico 5.151 muestra que los programadores con más experiencia en Java obtienen mejores resultados en cuanto a la Calidad y Productvidad al aplicar TDD. Los programadores nóveles fallan estrepitosamente al aplicar TDD.

5.10.4.3. Conocimiento del entorno Eclipse

El conocimiento del entorno de desarrollo Eclipse presenta una influencia estadísticamente significativa para la Calidad ($P-value$ =0.01), con un tamaño de efecto fuertemente negativo (B =-25.08). Como se aprecia en

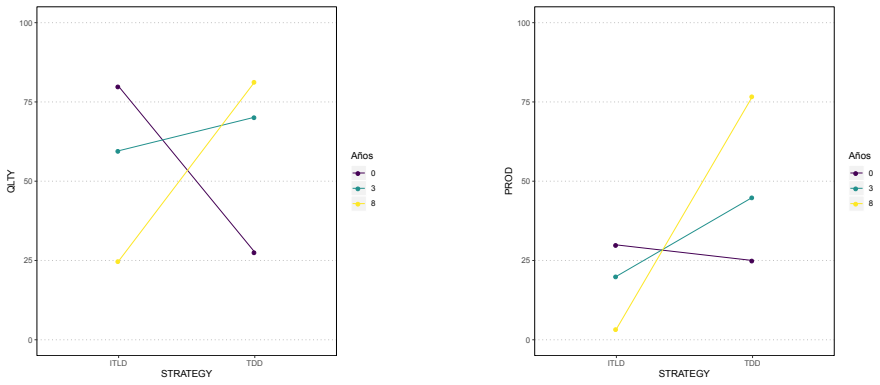
Tabla 5.157: Resultados del análisis estadístico para la Experiencia en Java

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1240.14	1240.14	1.00	68.00	1.50	0.2248
javaExperience	22.88	22.88	1.00	68.00	0.03	0.8684
TASK	32801.42	32801.42	1.00	68.00	39.69	0.0000
SLICING	2845.06	2845.06	1.00	68.00	3.44	0.0679
GROUP	12.87	12.87	1.00	68.00	0.02	0.9011
STRATEGY:javaExperience	716.86	716.86	1.00	68.00	0.87	0.3550

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	228.88	228.88	1.00	32.87	0.91	0.3471
javaExperience	338.85	338.85	1.00	35.03	1.35	0.2537
TASK	25641.17	25641.17	1.00	35.26	101.93	0.0000
SLICING	426.57	426.57	1.00	36.11	1.70	0.2011
GROUP	67.71	67.71	1.00	39.26	0.27	0.6068
STRATEGY:javaExperience	191.54	191.54	1.00	31.59	0.76	0.3895



(a) Calidad (b) Productividad

Figura 5.151: Efecto de la experiencia en programación Java. La experiencia se reporta redondeada a años como en el caso anterior.

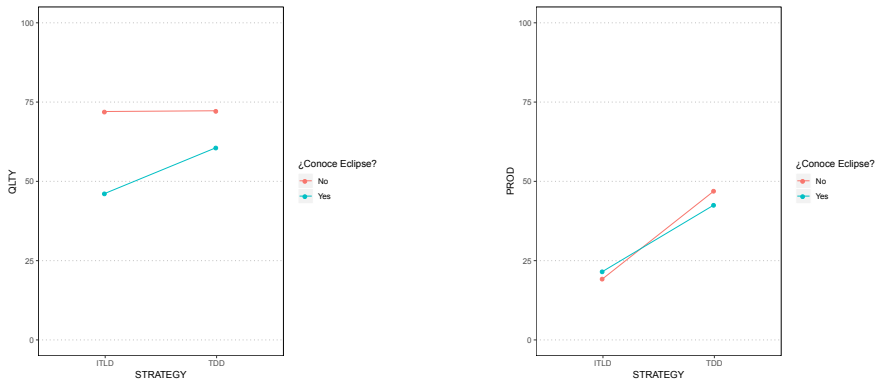
Tabla 5.158: Resultados del análisis estadístico para el conocimiento de entorno Eclipse

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	452.32	452.32	1.00	68.00	0.62	0.4331
knowEclipse	5819.64	5819.64	1.00	68.00	8.00	0.0061
TASK	38721.58	38721.58	1.00	68.00	53.23	0.0000
SLICING	2873.89	2873.89	1.00	68.00	3.95	0.0509
GROUP	4.00	4.00	1.00	68.00	0.01	0.9411
STRATEGY:knowEclipse	958.37	958.37	1.00	68.00	1.32	0.2551

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3134.28	3134.28	1.00	35.50	12.74	0.0011
knowEclipse	3.05	3.05	1.00	37.41	0.01	0.9119
TASK	29271.71	29271.71	1.00	35.35	118.97	0.0000
SLICING	423.01	423.01	1.00	35.63	1.72	0.1982
GROUP	184.61	184.61	1.00	39.64	0.75	0.3916
STRATEGY:knowEclipse	252.33	252.33	1.00	33.49	1.03	0.3185



(a) Calidad (b) Productividad

Figura 5.152: Efecto del conocimiento del entorno Eclipse.

la figura 5.152a, los participantes que indicaron conocer el IDE Eclipse alcanzan menores calidades que los sujetos que no conocen Eclipse. En el caso particular de TDD, los sujetos que conocen Eclipse mejoran ostensiblemente la Calidad.

En lo que respecta a la variable Productividad, el conocimiento del IDE Eclipse posee una influencia positiva (B =3.27). Sin embargo, los efectos no son signficativos ($P - value$ =0.01). En este caso, todos los participantes obtuvieron una mejor Productividad al aplicar TDD, como se observa en la figura 5.152b.

5.10.4.4. Experiencia en el Framework JUnit

Tabla 5.159: Resultados del análisis estadístico para la Experiencia en el Framework JUnit

(a) Calidad

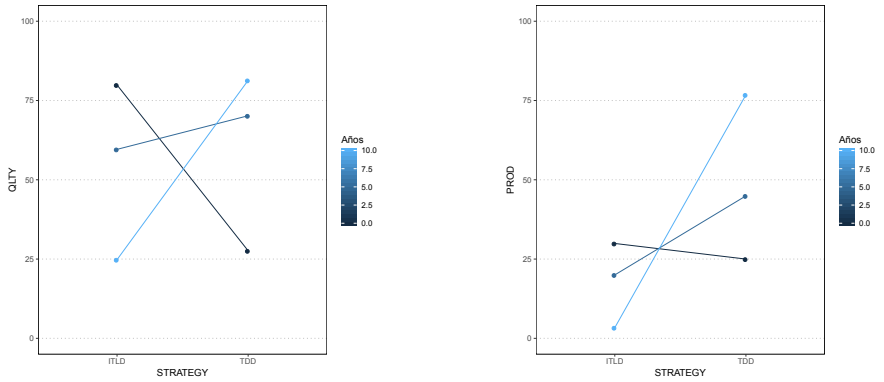
	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	831.44	831.44	1.00	68.00	1.01	0.3177
jUnitExperience	749.05	749.05	1.00	68.00	0.91	0.3428
TASK	36103.54	36103.54	1.00	68.00	43.99	0.0000
SLICING	2961.00	2961.00	1.00	68.00	3.61	0.0617
GROUP	6.24	6.24	1.00	68.00	0.01	0.9307
STRATEGY:jUnitExperience	234.38	234.38	1.00	68.00	0.29	0.5948

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	2194.25	2194.25	1.00	34.07	9.84	0.0035
jUnitExperience	356.77	356.77	1.00	36.64	1.60	0.2139
TASK	24647.95	24647.95	1.00	35.22	110.51	0.0000
SLICING	599.03	599.03	1.00	35.55	2.69	0.1101
GROUP	54.70	54.70	1.00	40.38	0.25	0.6231
STRATEGY:jUnitExperience	1050.57	1050.57	1.00	32.60	4.71	0.0374

La experiencia en el Framework Junit presenta una influencia negativa tanto para la Calidad, como para la Productividad (B =-5.24 y B =-1.1). Sin embargo, los efectos no son significativos en ambos casos ($P - value$ =0.34 y $P - value$ =0.21). La interacción *STRATEGY*jUnitExperience* resulta significativa para la Productividad, no así para la Calidad. Esto indica que los programadores que aplican TDD obtienen B =8.13 puntos más que los que usan ITLD.

En las figuras 5.153a y 5.153b podemos notar que los participantes más experimentados con el Framework Junit, mejoran tanto en la Calidad como en la Productividad al utilizar TDD, al contrario de los menos experimentados.



(a) Calidad (b) Productividad

Figura 5.153: Efecto de la experiencia en el Framework JUnit. La experiencia se reporta en lustros (5 años). Por ejemplo, el valor 0 representa 0-4 años de experiencia. El valor 5 representa 5-9 años de experiencia, etc.

5.10.4.5. Resumen general

En las tablas 5.160 y 5.161 mostramos un resumen de los resultados de los análisis de subgrupos realizados. Como se puede observar, el *Conocimiento del entorno de desarrollo Eclipse* es la única variable que tiene un impacto significativo sobre la Calidad, no así para la Productividad.

Tabla 5.160: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	-0.84	0.84	2.55	0.54
Experiencia en Programación Java	-1.77	0.87	4.31	0.35
Conocimiento del Entorno Eclipse	**-25.08**	**0.01**	14.48	0.26
Experiencia en Framework JUnit	-5.24	0.34	3.77	0.59

5.10.5. Grado de completitud

5.10.5.1. Aspectos generales

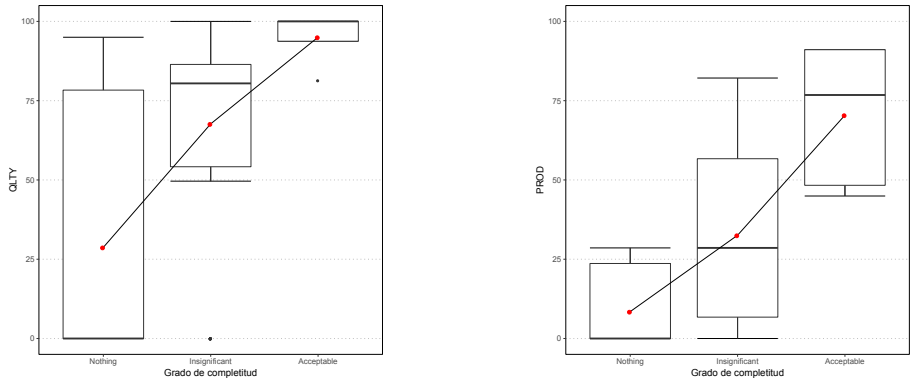
Analizamos manualmente el código entregado por los sujetos con las mismas consideraciones hechas para los anteriores experimentos. Observamos que algunos sujetos no hicieron ningún código (*Nothing*), la mayoría realizaron unas pocas líneas de código (*Insignificant*); y la menor parte obtuvo

Tabla 5.161: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	1.03	0.13	2.18	0.35
Experiencia en Programación Java	0.66	0.25	2.25	0.39
Conocimiento del Entorno Eclipse	3.27	0.91	-7.56	0.32
Experiencia en Framework Junit	-1.1	0.21	**8.13**	**0.04**

una calificación de *Aceptable*.

La relación entre el grado de completitud frente a las variables QLTY y PROD, se presentan en la figura: 5.154. Adicionalmente, en la tabla En Tabla 5.162 se cuantifica el grado de completitud del trabajo realizado.



(a) Calidad (b) Productividad

Figura 5.154: Relación entre el grado de completitud y la calidad y productividad

De un total de 75 tareas recogidas, 12 sujetos no hicieron nada. Por otro lado, 58 hicieron una mínima cantidad de trabajo y 5 sujetos realizaron el trabajo como era esperado.

Si consideramos las tareas calificadas como *Nothing* en la medición del estudio, la QLTY y PROD disminuyen en promedio, como hemos venido observando en experimentos anteriores. A modo de ejemplo comparamos que el promedio de la QLTY de la segunda sesión, incluidos los sujetos que no realizaron un trabajo sustancial, es del 67 %; cuando se excluyen estos sujetos, la QLTY aumenta a 91.7 %. El hecho de incluir las tareas vacías compromete nuestro análisis significativamente.

	Número de tareas entregadas
Nothing	12
Insignificant	58
Acceptable	5

Tabla 5.162: Grado de completitud

5.10.5.2. Impacto de la estrategia de programación y la tarea experimental

La tabla 5.163, nos permite ver que al parecer la *estrategia de programación* no parece estar relacionada con el grado de finalización de las tareas. Existe el mismo número de sujetos que no hicieron nada tanto para TDD como para ITLD. Existe una mínima diferencia entre el núemero de sujetos que hicieron las tareas de manera aceptable y la estrategia de programación.

	Nothing	Insignificant	Acceptable
ITLD	6	34	2
TDD	6	24	3

Tabla 5.163: Estrategia de programación

Podemos indicar que no existe relación entre la estrategia de programación y el grado de finalización de las tareas. Esto se comprueba mediante la realización de un test χ^2 que arroja los siguientes resultados: $\chi^2 = 0{,}86$, $df = 2$, $p - value = 0{,}65$.

	Nothing	Insignificant	Acceptable
BSK	4	33	3
MR	8	25	2

Tabla 5.164: Grado de completitud por tarea experimental (BSK, MR)

La tabla 5.164 muestra cierta relación entre el grado de completitud y las tareas. En este caso, el doble de sujetos no hicieron nada en la tarea MR, aunque casi el mismo número de sujetos entregaron las dos tareas de forma aceptable. Sin embargo, las diferencias entre BSK y MR no significativas como muestra el test χ^2 ($\chi^2 = 2{,}31$, $df = 2$, $p - value = 0{,}31$).

5.10.5.3. Impacto de las variables demográficas

En las siguintes secciones analizaremos si el grado de completitud de las tareas, se encuentra reacionado con las variables demográficas obtenidas de los sujetos experimentales. Vamos a analizar las mismas variables de la sección 5.10.4, esto es:

- Experiencia en programación.
- Experiencia en Java.
- Conocimiento del entorno Eclipse.
- Entrenamiento previo en desarrollo de pruebas unitarias.

Experiencia en programación

La experiencia en programación parece tener alguna relación con la presentación de tareas vacías, como se muestra en la Fig. 5.211. Podemos notar que los sujetos más experimentados, en promedio hicieron un trabajo más aceptable. El test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,32, que no resulta estadísticamente significativo.

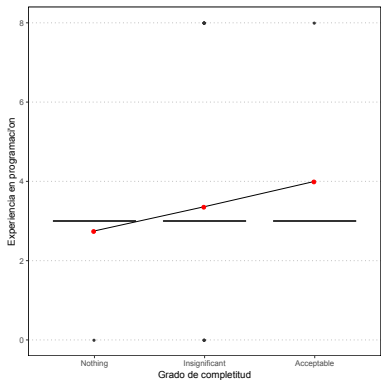


Figura 5.155: Experiencia en programación

Experiencia en programación Java

La experiencia en desarrollo con el lenguaje Java, al igual que la experiencia en programación, al parecer influye en el grado de presentación de las tareas (véase la figura 5.212). Aunque el test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud arroja un p-valor = 0,23 el cual no resulta estadísticamente significativo.

Conocimiento del entorno Eclipse

El conocimiento del entorno de desarrollo Eclipse no tiene influencia en el grado de completitud de las tareas experimentales. El test χ^2 nos indica este hecho ($\chi^2 = 0{,}51$, $df = 2$, $p - value = 0{,}78$).

Función actual en la organización

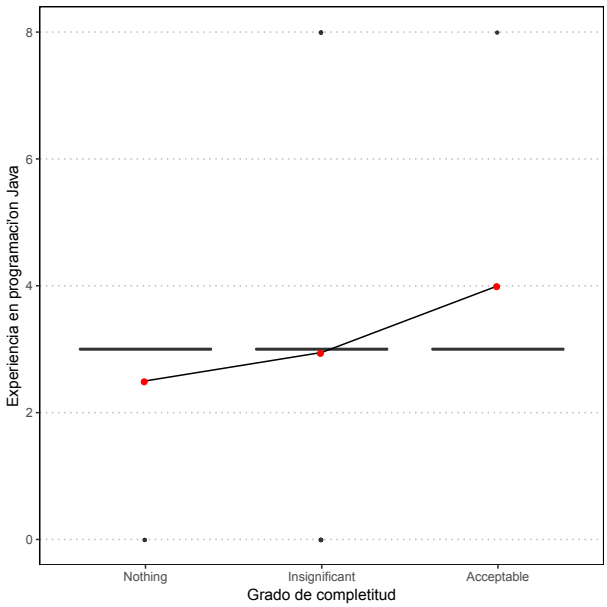


Figura 5.156: javaExperience

	Nothing	Insignificant	Acceptable
No	7	32	2
Yes	5	26	3

Tabla 5.165: Conocimiento del entorno eclipse

El entrenamiento previo en el desarrollo de pruebas unitarias tampoco presentan influencia en el grado de completitud de las tareas. El test χ^2 confirma este hecho ($\chi^2 = 1{,}82$, $df = 2$, $p - value = 0{,}4$). Por lo tanto no es significativo.

5.10.5.4. Resumen general impacto variables demográficas

La tabla 5.167 resume los resultados de los análisis de la influencia de las variables demográficas estudiadas frente al grado de completitud de las tareas realizadas. Ninguna de las variables ejercen un impacto que sea estadísticamente significativo.

	Nothing	Insignificant	Acceptable
No	11	57	5
Yes	1	1	0

Tabla 5.166: Entrenamiento previo en pruebas unitarias

Tabla 5.167: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Experiencia en programación	0.32
Experiencia en Java	0.23
Conocimiento del entorno Eclipse	0.78
Entrenamiento previo en desarrollo de pruebas unitarias	0.4

5.11. Experimento MexicoUADY2015

5.11.1. Ejecución

El experimento UADY2015 es una replicación del experimento base. El experimento fue realizado en la Universidad Autónoma de Yucatán de México con estudiantes de pregrado como parte de un evento académico organizado por la Universidad. Este es el segundo experimento realizado en el ámbito académico con similares características al experimento ESPE2015.

5.11.1.1. Muestra

En el experimento UADY2015 participaron 35 estudiantes que se encontraban cursando (excepto por un estudiante de maestría) el cuarto, quinto, sexto y séptimo grados en la Facultad de Licenciatura en Ciencias de la Computación de la Universidad Autónoma de Yucatán de México (UADY). En la tabla 5.168 se observa que todos los participantes son jóvenes menores de 30 años sin experiencia profesional. Las diferencias mas notables de este grupo de sujetos que podemos destacar es que aproximadamente el 40 % de los mismos tiene experiencia en el entorno de desarrollo Eclipse. Casi todos los participantes indicaron tener experiencia tanto en programación como en lenguaje Java en el rango de 2 a 5 años. En los otros aspectos, el grupo es bastante homogéneo excepto por unos pocos estudiantes que no superan el 10 % en promedio.

5.11.1.2. Preparación

De forma similar a los anteriores experimentos realizados, se instalaron previamente las herramientas de software necesarias; esto es: la máquina virtual de Java, el framework Junit, el plugin para empaquetar y realizar

Tabla 5.168: Características demográficas de los sujetos del experimento UADY2015

EXPERIMENTO: UADY2015		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	35
	30 >= Edad < 40 años	35
	40 >= Edad < 50 años	35
	Edad >= 50 años	35
Nivel de Educación	Other	0
	Undergraduate	34
	Bachelor	0
Experiencia profesional	Sin experiencia (<2 años)	35
	Novato (2 - 5 años)	35
	Intermedio (6 - 10 años)	35
	Experto (>10 años)	35
Experiencia en programación	Sin experiencia (<2 años)	0
	Novato (2 - 5 años)	31
	Intermedio (6 - 10 años)	3
	Experto (>10 años)	1
Uso de herramientas de pruebas	Yes	5
	No	30
Experiencia en lenguaje Java	Sin experiencia (<2 años)	2
	Novato (2 - 5 años)	33
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	33
	Novato (2 - 5 años)	2
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	2
	No	33
Experiencia en TDD	Sin experiencia (<2 años)	35
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	3
	No	32
Conocimiento del entorno Eclipse	Yes	13
	No	22
Función actual en la organización	Tester	0
	Developer	0
	Student	35

mediciones de los experimentos y el IDE de desarrollo Eclipse. Se utilizó uno de los laboratorios de la Universidad para la realización de las sesiones de entrenamiento y de los experimentos.

5.11.1.3. Realización

Esta instancia experimental se realizó como parte de un evento académico denominado Jornadas de Ingeniería de Software FMAT2015, organizado por la Facultad de Ciencias de la Computación de la UADY. Tanto las sesiones de entrenamiento como las sesiones experimentales se realizaron como un curso taller de 5 días con un horario de cuatro horas por día. El protocolo seguido se detalla a continuación:

- **Día 1:** Previo al inicio del entrenamiento los sujetos llenaran el cuestionario demográfico. Posteriormente se realizó una sesión de introducción al desarrollo ágil y formación en pruebas unitarias utilizando el framework Junit. Los estudiantes tuvieron un receso de 15 minutos, luego de lo cual se les pidió realizar un ejercicio práctico como tarea de control.
- **Día 2:** Durante el segundo día se realizó una fase de capacitación en la técnica de slicing e Incremental Test Last Development (ITLD). Posteriormente hubo un receso de 15 minutos seguido por la realización de una segunda tarea de control.
- **Día 3:** Se realizó la primera sesión experimental que consistió en la solución de las tareas BSK o MR con o sin Slicing y aplicando la técnica ITLD. Posteriormente, se realizó un receso de 15 minutos y se concluyó la sesión con un feedback para resolver las inquietudes surgidas.
- **Día 4:** Se realizó la segunda fase de capacitación en la técnica TDD. De igual forma, se tuvo un receso de 15 minutos y se concluyó con una tarea de control para reforzar los conocimientos adquiridos.
- **Día 5:** Finalmente se realizó la segunda tarea experimental, en este caso solucionar BSK o MR con o sin Slicing y aplicando la estrategia TDD. Se cerró la sesión con un receso de 15 minutos y un feedback para resolver las inquietudes de los participantes.

La capacitación estuvo a cargo de Geovanny Raura con la colaboración de Rodrigo Fonseca y se utilizaron los materiales de entrenamiento preparados por Oscar Dieste.

5.11.1.4. Desviaciones

No existieron desviaciones en cuanto al protocolo establecido. En general se cumplieron las horas de inicio y culminación de las actividades

experimentales y de la capacitación con un promedio de cinco minutos de retraso, aunque se observó que unos pocos estudiantes no llegaron puntuales o abandonaron antes de concluir el tiempo previsto en razón de que pudieron resolver con mayor rapidez las tareas.

5.11.1.5. Reducción del conjunto de datos

Las variaciones en el número de sujetos que asistieron al experimento a lo largo de las tres semanas de duración del mismo fueron mínimas en este caso. Se presentaron 35 y de ellos todos realizaron la primera tarea experimental, es decir la aplicación de la estrategia ITLD. La segunda tarea experimental fue entregada por 33 sujetos quienes aplicaron la estrategia TDD. Se pudo notar que dos sujetos ya no asistieron a partir del tercer día de capacitación, aunque no se pudo determinar las razones.

5.11.2. Estadísticos descriptivos

Se describen por separado cada una de las variables respuestas de Calidad y Productividad obtenidas.

5.11.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	80.59	21.63	-1.66	3.23
TDD	77.63	24.67	-1.81	3.22

Tabla 5.169: Estadísticos descriptivos para QLTY

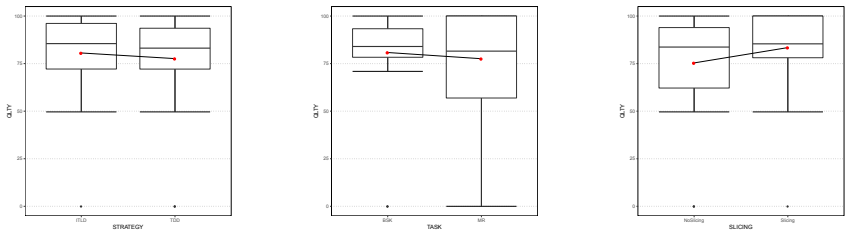
En la tabla 5.169 podemos notar que los promedios para las estrategias ITLD y TDD son bastante parecidos, con apenas un 3 % de diferencia de ITLD sobre TDD. En ambos casos, las desviaciones estándar son algo elevadas. Las asimetrías son negativas en ambos casos, en tanto que la curtosis es positiva con valores no tan elevados.

En los gráficos de box-plot mostrado en la figura 5.157 apreciamos la similitud en el valor de las medianas para las dos estrategias. Podemos ver que existe una distribución de puntos bastante homogénea tanto para ITLD como para TDD (representados por los puntos azules). En este caso no parece haber diferencias significativas.

Con respecto a la tarea, BSK nuevamente obtiene mayores valores de calidad que MR, tal y como indica la figura 5.157b, aunque la diferencia no es tan marcada como ha ocurrido en otras instancias experimentales. Es muy probable que no encontremos diferencias significativas para la tarea.

El efecto del factor SLICING puede observarse en la figura 5.157c. No está claro si este factor influye significativamente en los resultados obtenidos,

aunque el gráfico permite apreciar que los sujetos que aplican slicing obtienen un ligero mayor nivel de calidad.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.157: Box plots para la variable respuesta QLTY

5.11.2.2. Productividad

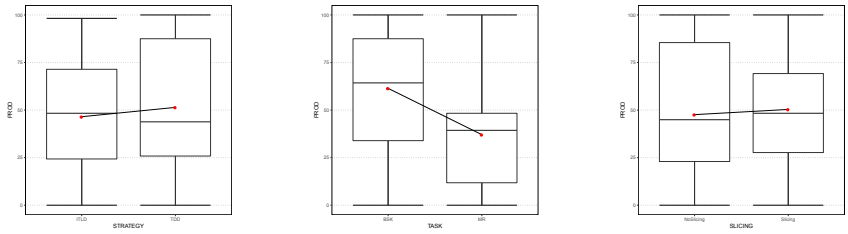
Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	46.51	29.64	0.14	-1.19
TDD	51.41	32.88	0.15	-1.32

Tabla 5.170: Estadísticos descriptivos para PROD

En la tabla 5.170 se muestran los promedios de la variable Productividad. Podemos notar que las dos estrategias bordean el 50 % aunque con apenas 5 puntos de diferencia para TDD sobre ITLD. Las desviaciones estándar son más elevadas que las obtenidas para la Calidad y son relativamente altas, con asimetrías positivas y curtosis negativas relativamente bajas.

En cuanto a la estrategia de desarrollo, la figura 5.158a es parecida a la de Calidad (5.157a), aunque la estrategia ITLD tiene una mediana ligeramente superior (aunque su promedio es ligeramente inferior como ya habíamos señalado). Por otro lado, en la figura 5.158b se aprecia que existe un impacto de la tarea; siendo una vez más la tarea BSK donde los sujetos obtienen un mayor valor de Productividad. Finalmente, el nivel de definición de la tarea (slicing) no parece influir en esta variable (5.158c).

En síntesis, en esta instancia experimental, al parecer no existen efectos significativos para la Calidad y un posible efecto de la tarea para la Productividad.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.158: Box plots para la variable respuesta PROD

5.11.3. Prueba de hipótesis

5.11.3.1. Análisis estadístico

El análisis estadístico se ha realizado de igual modo que en experimentos anteriores (a excepción de Quito2016). Los resultados de la ANOVA respecto a las variables respuesta Calidad y Productividad se muestran en la tabla 5.171. La tarea arroja resultados significativos para la Productividad, como ya lo habíamos indicado en la sección 5.11.2. En cuanto a la variable Slicing, resulta que es significativo para la Calidad. Este es un resultado que no lo veíamos tan claro en los gráficos de box plot de la sección anterior.

Tabla 5.171: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	162939929844133.69	162939929844133.69	1.00	299592.53	0.28	0.5952
TASK	43626182056077.64	43626182056077.64	1.00	283455.99	0.08	0.7834
SLICING	3618305112631899.00	3618305112631899.00	1.00	606.28	6.27	0.0125
GROUP	12478565907476.67	12478565907476.67	1.00	31.19	0.02	0.8840

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	389.62	389.62	1.00	30.83	1.05	0.3145
TASK	9491.55	9491.55	1.00	30.86	25.47	0.0000
SLICING	584.52	584.52	1.00	44.79	1.57	0.2169
GROUP	200.18	200.18	1.00	32.74	0.54	0.4688

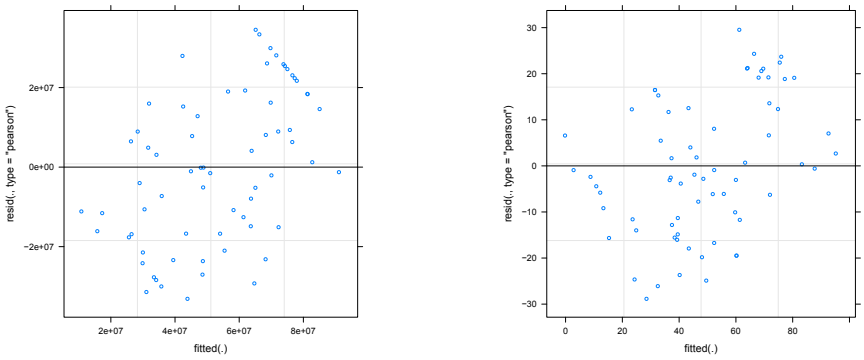
5.11.3.2. Chequeo del análisis estadístico

Realizamos el chequeo del análisis estadístico siguiendo el mismo patrón de todos lo experimentos anteriores (excepto Quito2016 que tiene ciertas

diferencias). Comprobaremos la homogeneidad de varianzas y normalidad de los residuos ya que los factores tienen solo dos niveles por lo que la linealidad siempre se cumple.

Homogeneidad de varianzas

En la figura 5.159 se muestran los gráficos de valores predichos vs. residuos estandarizados. Al parecer se cumple la homogeneidad de varianzas ya que no se observan patrones de embudo.



(a) Calidad (b) Productividad

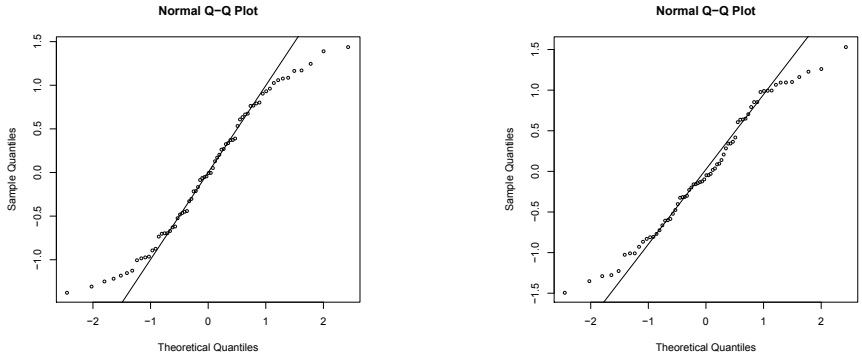
Figura 5.159: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

Normalidad de residuos

Analizamos los factores fijos para determinar la normalidad de los residuos. En lo que respecta a los factores fijos, podemos apreciar en la figura 5.160, que la distribución de los residuos de la variable respuesta PROD se solapan razonablemente con la distribución normal, representada en la línea recta diagonal, con desviaciones ligeras en los extremos. El test de Shapiro-Wilks confirma la normalidad de los residuos para la productividad ($W = 0{,}97$, $p - value = 0{,}11$)

La distribución de los residuos de la variable QLTY se parece mucho a la variable PROD, pero el test Shapiro-Wilks rechaza la normalidad de los residuos. Con la finalidad de conseguir normalidad, hemos aplicado una transformación Box-Cox $\lambda = 4$, un tanto forzada, pero que mejora ostensiblemente la normalidad de los datos. El test de Shapiro-Wilks se sitúa en el límite de la no normalidad ($W = 0{,}96$, $p - value = 0{,}04$), pero este resultado es mucho mejor que el de la variable sin transformar. Nos damos por satisfechos.

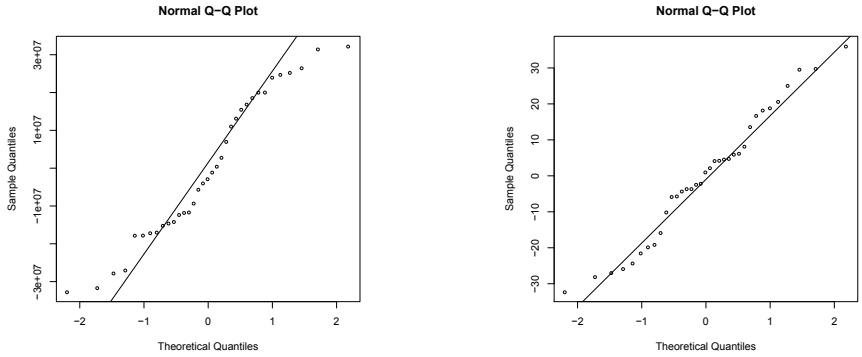
En lo que respecta a los factores aleatorios, el gráfico QQ de la figura



(a) Calidad (b) Productividad

Figura 5.160: Gráficos QQ para los factores fijos

5.161 muestra una situación aparentemente peor que la anterior. Sin embargo, el test de Shapiro-Wilks confirma la normalidad de los efectos aleatorios tanto para la Calidad ($W = 0{,}95$, $p - value = 0{,}09$) como para la Productividad ($W = 0{,}97$, $p - value = 0{,}5$).



(a) Calidad (b) Productividad

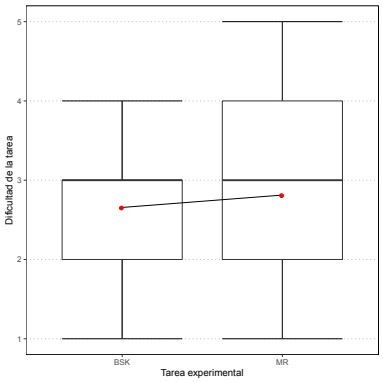
Figura 5.161: Gráficos QQ para los factores aleatorios

5.11.3.3. Influencia de factores instrumentales

Como hemos procedido en otros experimentos, en esta sección analizamos la dificultad de la tarea experimental percibida por los sujetos para descartar una posible influencia de la tarea ocasionada por el diseño del experimento.

Tarea experimental

En la figura 5.162a, apreciamos que la complejidad percibida para las dos tareas es muy similar, aunque la dispersión de datos para la tarea MR es mucho mayor. El chequeo del análisis estadístico arrojó diferencias significativas de la tarea en la Productividad. Sin embargo, este análisis indica que no fue percibido así por los sujetos. El test Kruskal-Wallis entre las tarea experimental y la dificultad percibida arroja un p-valor = 0,48, que confirma lo observado en la figura 5.162a.



(a) Dificultad percibida de la tarea

Figura 5.162: Efecto de la dificultad de la tarea percibida por los sujetos

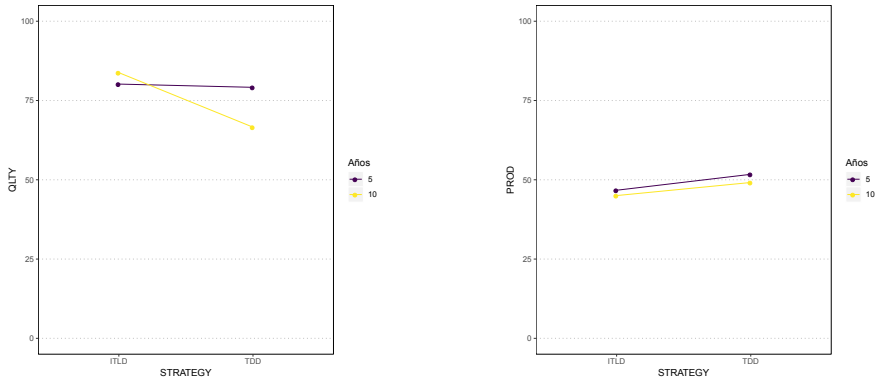
5.11.4. Análisis de subgrupos

En este experimento hemos considerado los siguientes aspectos que creemos son relevantes conforme a los datos demográficos de los participantes (véase sección 5.11.1.1):

- Experiencia en programación.
- Uso de herramientas de pruebas.
- Conocimiento del entorno Eclipse.
- Grado de estudios de los participantes.

5.11.4.1. Experiencia en programación

En lo que respecta a la experiencia en programación, el análisis muestra una influencia positiva para la Calidad ($B = 42.59$) y para la Productividad ($B = 1.54$). En los dos casos los efectos no son significativos ($p - value = 0.65$ y $p - value = 0.96$) respectivamente. No incluímos la tabla de análisis en razón de que los resultados no son significativos. Procederemos de igual forma en los siguientes análisis.

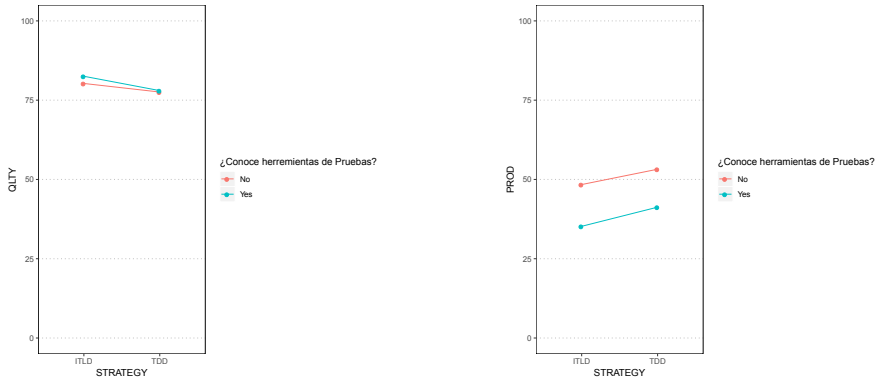


(a) Calidad (b) Productividad

Figura 5.163: Efecto de la experiencia en programación. La experiencia se reporta redondeada a lustros.

Como puede obeservarse en la figura 5.163, los desarrolladores más experimentados obtienen mejores resultados de Calidad al aplicar la estrategia ITDD en comparación con TDD. En contraste, los desarrolladores menos experimentados prácticamente no presentan diferencias de Calidad al aplicar ITLD o TDD. La Productividad de los desarrolladores tiene una cierta mejora al aplicar TDD, independientemente de los años de experiencia en programación (nótese que las líneas se encuentran prácticamente adyacentes y son paralelas).

5.11.4.2. Uso de herramientas de pruebas



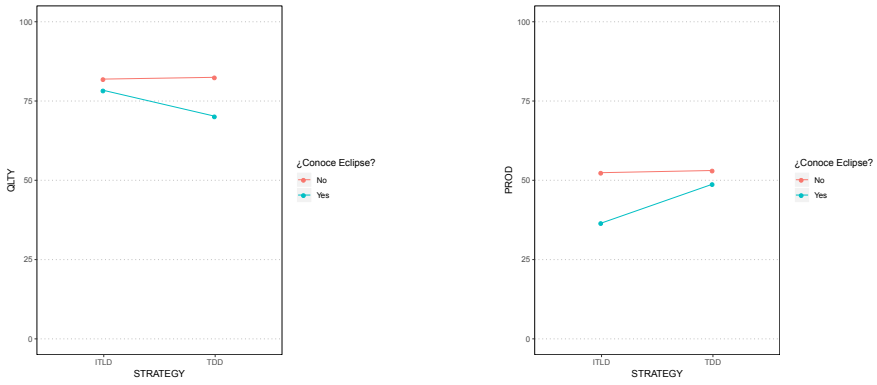
(a) Calidad (b) Productividad

Figura 5.164: Efecto del uso de herramientas de pruebas.

En cuanto al uso de herremientas de pruebas, el análisis muestra un efecto

positivo para las variable Calidad (B =40.62) y negativo para la Productividad (B =-10.96). Sin embargo los efectos no son significativos en ambos casos ($P-valor$ =0.76 y $P-valor$ =0.3). La figura 5.164, confirma que no existe influencia del conocimiento del uso de herramientas de prueba tanto en la Calidad como en la Productividad al aplicar TDD o ITLD. Como se puede notar, las líneas son prácticamente paralelas en ambos casos, aunque para la Productividad, los que no conocen de herramientas de prueba, tienen un mejor desempeño.

5.11.4.3. Conocimiento del entorno Eclipse



(a) Calidad (b) Productividad

Figura 5.165: Efecto del conocimiento del entorno Eclipse.

El análisis del conocimiento del entorno de desarrollo Eclipse indica una influencia negativa para la Calidad y Productividad (B =-47.82 y B =-7.3). Los efectos tampoco son significativos ($P-value$ =0.7 y $P-value$ =0.43). En la figura 5.165, se puede observar que la Calidad y Productividad alcanzada por los desarrolladores que no conocían Eclipse, es independiente de la técnica utilizada (nótese que la línea roja prácticamete no tiene pendiente). En cambio, los desarrolladores que sí conocían Eclipse, tienen la tendencia a disminuir la Calidad al aplicar TDD, mientras que mejoran su Productividad y están por encima de los que si conocen el IDE Eclipse.

5.11.4.4. Grado de estudios de los participantes

En este experimento resulta de interés analizar el grado de estudios de los participantes y su influencia en la Calidad y Productividad. Como hemos indicado en secciones anteriores, los participantes fueron estudiantes que se encontraban cursando distintos grados de la Carrera en Licienciatura en Informática de la UADY. Específicamente fueron estudiantes matriculados

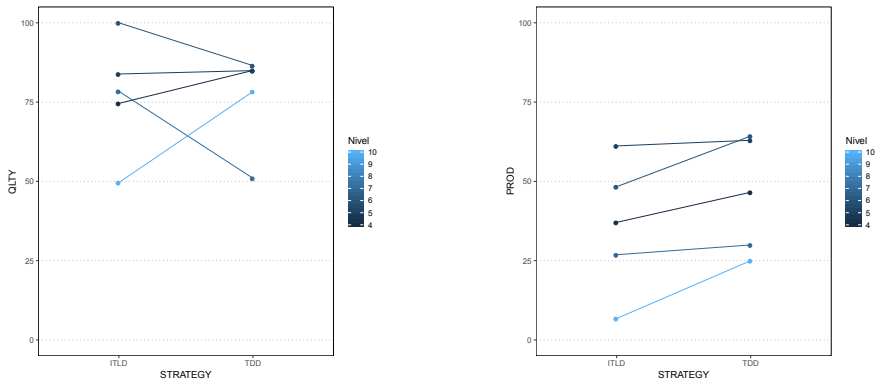
Tabla 5.172: Resultados del análisis estadístico para nivel de estudios de los participantes

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	44177142964770.48	44177142964770.48	1.00	30800.70	0.07	0.7875
Seniority	2196939485300452.00	2196939485300452.00	1.00	29.59	3.61	0.0670
TASK	89799451459131.64	89799451459131.64	1.00	116819.23	0.15	0.7007
SLICING	3402708782554296.50	3402708782554296.50	1.00	413.79	5.60	0.0184
GROUP	16571686749292.16	16571686749292.16	1.00	29.81	0.03	0.8700
STRATEGY:Seniority	90238820063996.69	90238820063996.69	1.00	36215.06	0.15	0.7000

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	159.75	159.75	1.00	30.76	0.41	0.5260
Seniority	1601.21	1601.21	1.00	31.35	4.12	0.0509
TASK	9678.24	9678.24	1.00	29.88	24.92	0.0000
SLICING	534.13	534.13	1.00	46.11	1.38	0.2469
GROUP	69.75	69.75	1.00	31.57	0.18	0.6746
STRATEGY:Seniority	65.80	65.80	1.00	30.62	0.17	0.6835



(a) Calidad (b) Productividad

Figura 5.166: Efecto del nivel de estudios de los participantes.

de cuarto a séptimo nivel de formación de la Carrera, con excepción de un único participante que se encontraba cursando una maestría. La tabla 5.173 muestra la distribución de sujetos participantes de acuerdo al nivel de estudios; 4S indica que el estudiante se encontraba cursando el cuarto semestre, 5S en el quinto semestre y así sucesivamente:

Tabla 5.173: Distribución de sujetos por nivel de estudios

	Nivel de estudios				
	4S	5S	6S	7S	MSc
Número de sujetos	8	17	2	7	1

El análisis muestra una influencia negativa del nivel de estudios de los participantes tanto para la Calidad como para la Productividad (B =-49.94 y B =-5.83). En este caso, los efectos se acercan al nivel de significación para la Calidad ($P - value$ =0.07 y son significativos para la Productividad $P - value$ =0.05).

Aunque el efecto del nivel de estudios en términos prácticos es estadísticamente significativo, sin embargo, el efecto es opuesto a lo esperado. Como se aprecia en la figura 5.166, los estudiantes de los más altos niveles obtienen una menor Calidad y Productividad. Nótese que las líneas de los estudiantes de niveles inferiores se encuentran en el cuadrante superior al 75 % para la Calidad y sobre el 35 % para la Productividad, por sobre los estudiantes de últimos niveles que apenas superan el 75 % en la Calidad y no sobrepasan el 30 % de Productividad. La tendencia en general es que los estudiantes independientemente del nivel mejoran la Productividad al aplicar TDD. En lo que respecta a la Calidad, el patrón no es el mismo, en algunos casos existe mejora al aplicar ITLD y en otros al aplicar TDD.

Las tendencias observadas deben tomarse con cautela, dado que los grupos de sujetos no están balanceados en cada uno de los niveles. Por ejemplo, tenemos un único sujeto que se encontraba cursando el nivel más alto (maestría), y la mayor porción de sujetos se encontraban cursando el quinto semestre de la carrera.

5.11.4.5. Resumen general

Los resultados del análisis de subgrupos se muestran en las tablas 5.174 y 5.175. No se observan diferencias significativas entre las variables demográficas estudiadas con respecto a la Calidad y Productividad. El *Nivel de estudios de los participantes* sin embargo, ejerce influencia estadísticamente significativa para la Productividad y se acerca al nivel de significación para el caso de la Calidad.

Tabla 5.174: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	42.59	0.65	-45.31	0.15
Uso de herramientas de pruebas	40.62	0.76	-61.8	0.38
Conocimiento del Entorno Eclipse	-47.82	0.7	36.03	0.9
Grado de estudios de los participantes	**-49.94**	**0.07**	-36.91	0.7

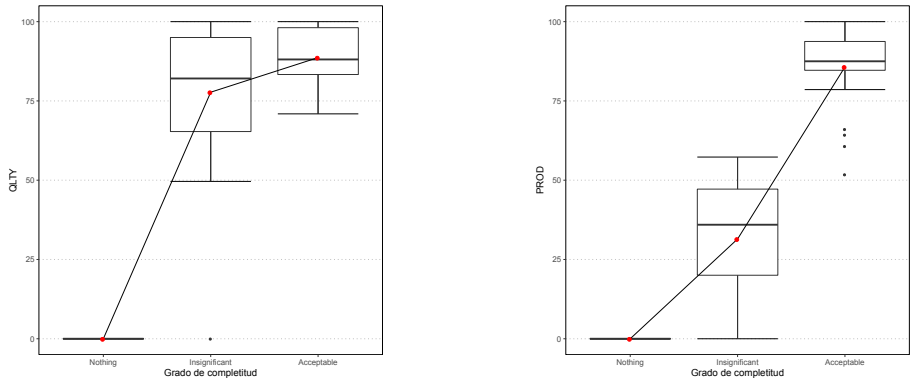
Tabla 5.175: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	1.54	0.96	-3.33	0.17
Uso de herramientas de pruebas	-10.96	0.3	-4.39	0.75
Conocimiento del Entorno Eclipse	-7.3	0.43	-0.63	0.95
Grado de estudios de los participantes	**-5.83**	**0.05**	-1.59	0.68

5.11.5. Grado de completitud

5.11.5.1. Aspectos generales

Siguiendo el mismo protocolo de análisis de experimentos anteriores, analizamos manualmente el código entregado por los sujetos. En este caso sólo un par de sujetos no hicieron ningún código (*Nothing*), la mayoría realizaron unas pocas líneas de código (*Insignificant*); y algunos lo hicieron de forma *Aceptable*.



(a) Calidad (b) Productividad

Figura 5.167: Relación entre el grado de completitud y la calidad y productividad

En la tabla 5.176 podemos apreciar que de un total de 68 tareas recogidas, 2 sujetos no hicieron nada, 43 hicieron una mínima cantidad de trabajo y 23 sujetos realizaron el trabajo adecuadamente.

	Número de tareas entregadas
Nothing	2
Insignificant	43
Acceptable	23

Tabla 5.176: Grado de completitud

En la figura 5.167 mostramos la relación entre el grado de completitud y las variables Calidad y Productividad. Comparamos la QLTY de la primera y segunda sesión incluyendo y excluyendo al grupo de sujetos que hizo las tareas calificadas como *Nothing* e *Insignificant* y obtenemos los siguientes promedios: Sesión ITLD: 80.59 %. Sesión TDD: 77.63 % Si tomamos en cuenta únicamente a los sujetos que hicieron un trabajo calificado como *Acceptable*, obtenemos los siguientes promedios: Sesión ITLD: 85.94 %. Sesión TDD: 91.15 %.

Como ha ocurrido en experimentos anteriores, el hecho de incluir a los sujetos que no hicieron nada y los sujetos que tampoco hicieron un trabajo aceptable, influye en los resultados del análisis de manera sustancial.

5.11.5.2. Impacto de la estrategia de programación y la tarea experimental

Cuando analizamos la cantidad de trabajo realizado por los sujetos en relación a la estrategia de programación, al parecer no existen diferencias significativas; el porcentaje de sujetos se distribuye casi de manera equitativa como se muestra en la tabla 5.177).

	Nothing	Insignificant	Acceptable
ITLD	1	23	11
TDD	1	20	12

Tabla 5.177: Estrategia de programación

Este hecho se comprueba mediante la realización de un test χ^2 que arroja resultados no significativos: $\chi^2 = 0{,}19$, $df = 2$, $p - value = 0{,}91$.

	Nothing	Insignificant	Acceptable
BSK	1	13	19
MR	1	30	4

Tabla 5.178: Grado de completitud por tarea experimental (BSK, MR)

En cuanto a la relación entre el grado de completitud y la tarea, en la tabla 5.178 podemos apreciar que existen diferencias marcadas para los sujetos calificados como *Nothing* y *Acceptable*. El test χ^2 muestra diferencias significativas entre las tareas MR y BSK ($\chi^2 = 16{,}46$, $df = 2$, $p - value = 0$).

5.11.5.3. Impacto de las variables demográficas

Analizamos si el grado de completitud de las tareas se encuentra reacionado con las variables demográficas obtenidas de los sujetos experimentales. Consideramos las mismas variables de la sección 5.11.4:

- Experiencia en programación.
- Uso de herramientas de pruebas.
- Conocimiento del entorno Eclipse.
- Grado de estudios de los participantes.

Experiencia en Programación

La experiencia en programación no influye en el grado de presentación de las tareas (véase la figura 5.168). Podemos notar que aunque la mayoría de sujetos indicó tener experiencia menor a cinco años, la tendencia es que los más experimentados entregaron las tareas como se esperaba, aunque los resultados no son significativos como demuestra el test Kruskal-Wallis entre la experiencia en programación y el grado de completitud: p-valor = 0,66.

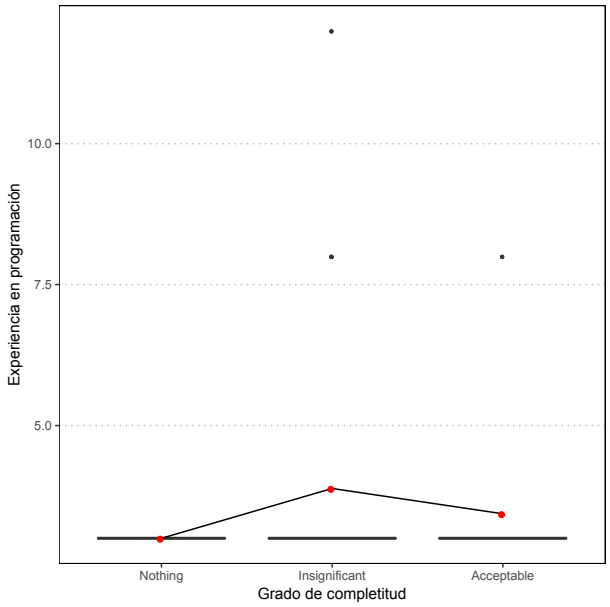


Figura 5.168: programmingExperience

Uso de herramientas de pruebas

	Nothing	Insignificant	Acceptable
No	2	35	21
Yes	0	8	2

Tabla 5.179: Uso de herramientas de prueba

Como se aprecia en la tabla 5.179, unos pocos sujetos que si tenían conocimiento de herramientas de pruebas entregaron las tareas de forma aceptable, pero la mayor parte que indicó no haber usado herramientas de pruebas también entregó las tareas de manera aceptable. Al parecer no existe influencia del grado de completitud de las tareas y el uso de herrmientas de pruebas. El test χ^2 confirma este hecho: $\chi^2 = 1{,}53$, $df = 2$, $p - value = 0{,}47$.

Conocimiento del entorno Eclipse

	Nothing	Insignificant	Acceptable
No	1	25	16
Yes	1	18	7

Tabla 5.180: Conocimiento del entorno eclipse

El conocimiento del entorno de desarrollo Eclipse tampoco tiene influencia en el grado de completitud de las tareas experimentales. El test χ^2 arroja resultados no significativos ($\chi^2 = 0{,}95$, $df = 2$, $p - value = 0{,}62$).

Grado de estudios de los participantes

El grado de estudios de los participantes al parecer tiene influencia en el grado de completitud de las tareas. En la Fig. 5.169 podemos apreciar que los sujetos con un nivel de estudios inferior, realizaron las tareas experimentales de manera aceptable. El test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,01 arroja resultados significativos.

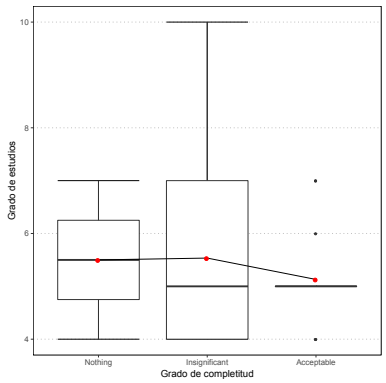


Figura 5.169: Seniority

5.11.5.4. Resumen general impacto variables demográficas

La tabla 5.181 resume los resultados de los análisis de la influencia de las variables demográficas estudiadas frente al grado de completitud de las tareas realizadas. Como se puede ver, el *Grado de estudios* arroja resultados significativos. Los pocos sujetos que declararon tener mayor experiencia (entre 5 y 8 años) entregaron las tareas de manera aceptable, al parecer influyen en los resultados.

Tabla 5.181: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Experiencia en programación	0.66
Conocimiento del entorno Eclipse	0.62
Uso de herramientas de pruebas	0.47
Grado de estudios	**0.01**

5.12. Experimento EcuadorESPE2016

5.12.1. Ejecución

El experimento ESPE2016, al igual que el experimento ESPE2015, fue realizado en la Universidad de las Fuerzas Armadas ESPE de Ecuador con estudiantes de pregrado. El experimento se lo ejecutó durante el mes de febrero del 2016 dentro del periodo académico regular de clases. Esta replicación puede ser catalogada como literal (ya que se asemeja tanto como ha sido posible a ESPE2015), conjunta (ya que participaron los mismos investigadores) e interna (ya que fue realizada en el mismo sitio).

5.12.1.1. Muestra

El experimento ESPE2016 es de tipo académico y participaron 15 estudiantes del quinto y sexto grado de la Carrera de Ingeniería de Sistemas e Informática en la Universidad de las Fuerzas Armadas ESPE. En la tabla 5.182 se muestran los datos demográficos de los sujetos. La totalidad de sujetos son jóvenes menores de 30 años, sin mayor experiencia profesional. Los únicos aspectos que los podrían diferenciar son:

- Un 80 % de los sujetos indica tener entre 2 y 5 años de experiencia en programación; dos sujetos indicaron tener entre seis y 10 años de experiencia en programación y un solo sujeto indicó no tener experiencia programando.
- Un 20 % de los sujetos indica tener menos de dos años de experiencia en Java; un 74 % se considera novato y un único sujeto indicó tener entre 6 y 10 años de experiencia en Java.
- Aproximadamente el 20 % de sujetos ha usado la técnica TDD, conoce el entorno de desarrollo Eclipse y ha utilizado herramientas de pruebas.

5.12.1.2. Preparación

Para el experimento ESPE2016, al igual que para el experimento ESPE2015, se utilizaron los laboratorios de computación de la Universidad

Tabla 5.182: Características demográficas de los sujetos del experimento ESPE2016

EXPERIMENTO: ESPE2016		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	15
	30 >= Edad < 40 años	15
	40 >= Edad < 50 años	15
	Edad >= 50 años	15
Nivel de Educación	Other	0
	Undergraduate	15
	Bachelor	0
Experiencia profesional	Sin experiencia (<2 años)	15
	Novato (2 - 5 años)	15
	Intermedio (6 - 10 años)	15
	Experto (>10 años)	15
Experiencia en programación	Sin experiencia (<2 años)	1
	Novato (2 - 5 años)	12
	Intermedio (6 - 10 años)	2
	Experto (>10 años)	0
Uso de herramientas de pruebas	Yes	3
	No	12
Experiencia en lenguaje Java	Sin experiencia (<2 años)	3
	Novato (2 - 5 años)	11
	Intermedio (6 - 10 años)	1
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	15
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	3
	No	12
Experiencia en TDD	Sin experiencia (<2 años)	15
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	0
	No	15
Conocimiento del entorno Eclipse	Yes	3
	No	12
Función actual en la organización	Tester	0
	Developer	0
	Student	15

configurados con las mismas herramientas de software usadas en todos los experimentos.

5.12.1.3. Realización

El experimento se realizó como parte de la asignatura de Desarrollo de software que se dicta en el quinto nivel de la Carrera de Ingeniería en Sistemas e Informática. El experimento se realizó en similares condiciones que ESPE2015, esto es, en dos días a la semana durante tres semanas dentro del horario de clase de las asignaturas, con un horario de dos horas para el entrenamiento y dos horas con treinta minutos para la realización de las tareas experimentales. El entrenamiento estuvo a cargo de Geovanny Raura con la colaboración de Rodrigo Fonseca y bajo la supervisión de Oscar Dieste. El protocolo seguido es el mismo empleado para el experimento ESPE2015 por lo que no lo hemos incluido en esta sección.

5.12.1.4. Desviaciones

No existieron desviaciones sustanciales en el protocolo. Las sesiones se cumplieron de acuerdo al cronograma establecido dentro del horario de clases de los estudiantes con no más de cinco minutos de retraso.

5.12.1.5. Reducción del conjunto de datos

En este caso no existieron variaciones en cuanto al número de sujetos. Se presentaron 15estudiantes y todos realizaron tanto la primera tarea experimental utilizando la técnica ITLD, como la segunda tarea utilizando la técnica TDD.

5.12.2. Estadísticos descriptivos

Las variables respuestas de Calidad y Productividad obtenidas en este experimento se describen a continuación:

5.12.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	59.29	42.47	-0.48	-1.70
TDD	59.43	42.29	-0.55	-1.67

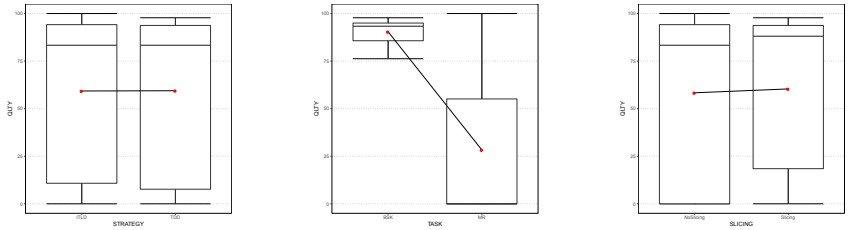
Tabla 5.183: Estadísticos descriptivos para QLTY

Como se indica en la tabla 5.183 los promedios así como la desviación estándar son prácticamente idénticas, con cierta diferencia en las cifras decimales. La asimetría y la curtosis para ITLD y TDD son negativas y no

excesivamente altas, lo que indica que la cola de la distribución se alarga para valores inferiores a la media aunque no muy cercanos a la misma.

En los gráficos de box-plot de la figura 5.170 podemos observar también similitud tanto en las medianas como en el tamaño de las cajas para ITLD y TDD.

En cuanto a la tarea experiemental, una vez más observamos en la figura 5.170b que BSK obtiene resultados superiores en Calidad que MR y al parecer con diferencias significativas. Por otra parte, el factor SLICING, no parece influir en la Calidad obtenida, como puede observarse en la figura 5.170c, aunque existe una mayor dispersión para la tarea con grado de definición No-Slicing.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.170: Box plots para la variable respuesta QLTY

5.12.2.2. Productividad

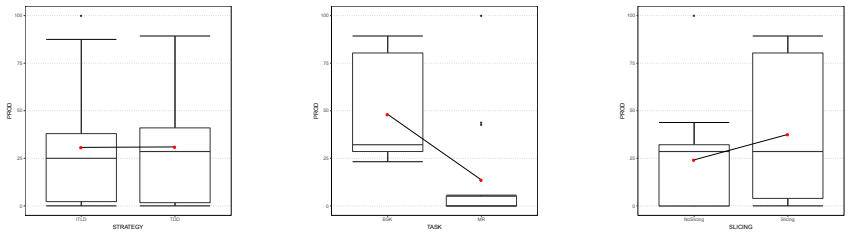
Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	30.74	33.65	0.86	-0.71
TDD	30.99	32.25	0.68	-1.00

Tabla 5.184: Estadísticos descriptivos para PROD

En lo que respecta a la Productividad, obtenemos promedios y desviaciones estándar similares para las dos estrategias (ver tabla 5.184). Podemos notar también que los promedios son menores que los de la variable Calidad con casi el doble. Esto implica que solo unas pocas historias de usuario fueron correctamente implementadas. La asimetría en este caso es positiva y la curtosis es negativa (platicúrtica) para ambas estrategias, lo que indica que la cola de la distribución se alarga (hacia la derecha) para valores superiores a la media y hay una menor concentración de datos entorno a la media.

Estos resultados se aprecian con mayor detalle en la figura 5.171. Podemos observar una gran similitud entre los box plot de Productividad para ITLD y para TDD. Una vez más se aprecia que existe un impacto de la

tarea para la Productividad, estando BSK por sobre MR. El nivel de definición de la tarea (Slicing) al parecer tampoco es significativo, aunque existen diferencias sustanciales en cuanto a la dispersión de datos.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.171: Box plots para la variable respuesta PROD

5.12.3. Prueba de hipótesis

5.12.3.1. Análisis estadístico

Mantenemos la misma estrategia de análisis estadístico que en los experimentos anteriores. En tabla **??** se muestran los resultados de la ANOVA tanto para la Calidad como para la Productividad. Nuevamente los resultados son significativos para la Tarea, pero no para ningún otro factor, como ya anticipábamos en los estadísticos descriptivos.

Tabla 5.185: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	130.15	130.15	1.00	12.02	0.16	0.6968
TASK	29071.90	29071.90	1.00	12.02	35.57	0.0001
SLICING	43.39	43.39	1.00	12.02	0.05	0.8216
GROUP	107.19	107.19	1.00	13.01	0.13	0.7230

(b) Productividad

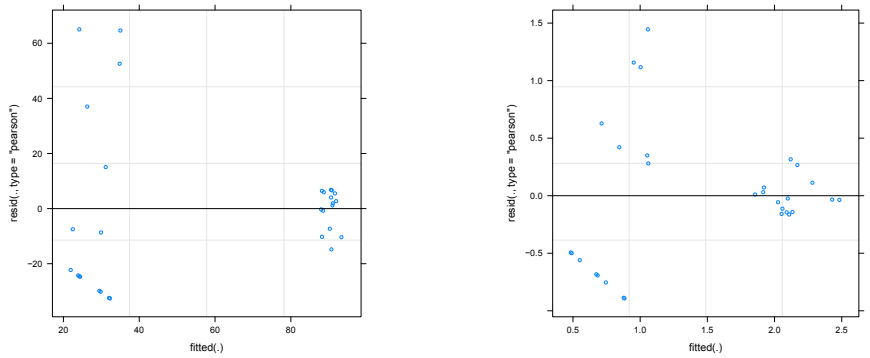
	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.04	0.04	1.00	12.00	0.09	0.7736
TASK	12.62	12.62	1.00	12.00	28.47	0.0002
SLICING	0.28	0.28	1.00	12.00	0.63	0.4431
GROUP	0.15	0.15	1.00	13.00	0.33	0.5739

5.12.3.2. Chequeo del análisis estadístico

Analizamos la homogeneidad de varianzas y la distribución normal de los residuos bajo las mismas consideraciones que en los anteriores experimentos. Remarcamos que los factores tienen solo dos niveles por lo que la linealidad siempre se cumple.

Homogeneidad de varianzas

En los gráficos de valores predichos vs. residuos estandarizados (5.172a) no se aprecia la existencia de ninguna figura en embudo que sugiera heterogeneidad entre varianzas.



(a) Calidad (b) Productividad

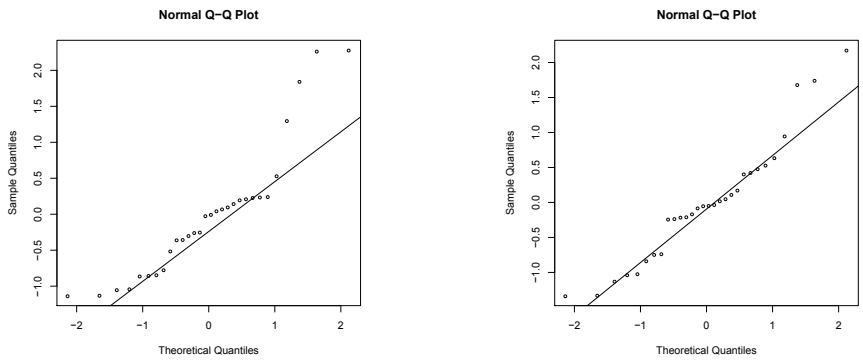
Figura 5.172: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

Normalidad de residuos

Los datos sin transformar resultan no normales al aplicar el test Shapiro-Wilks a los residuos de los efectos fijos, así como también a los residuos de los efectos aleatorios. Después de probar varias transformaciones, hemos conseguido normalizar la variable PROD usando una transformación Box-Cox $\lambda = -5$. Tal y como podemos observar en las figuras 5.173 y 5.174, los puntos se aproximan a la línea diagonal, aunque la coincidencia dista de ser perfecta. El test de Shapiro-Wilks confirma la impresión visual para los residuos fijos ($W = 0{,}94$, $p - value = 0{,}09$) y aleatorios ($W = 0{,}89$, $p - value = 0{,}06$).

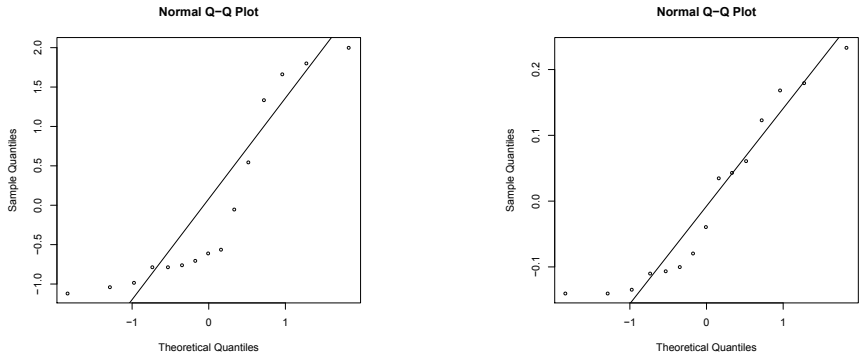
En el caso de la Calidad, ha sido imposible alcanzar algo parecido a la normalidad independientemente de la transformación aplicada. Nótese como los puntos se separan de la línea diagonal en las figuras 5.173 y 5.174. En términos del test Shapiro-Wilks, tanto los residuos ($W = 0{,}87$, $p-value = 0$) como los factores aleatorios ($W = 0{,}81$, $p-value = 0$) no resultan normales.

Tampoco podemos apoyarnos en los valores de asimetría y curtosis, los cuales son bastante elevados. En consecuencia, los resultados de los análisis de la Calidad deben tomarse con cautela.



(a) Calidad (b) Productividad

Figura 5.173: Gráficos QQ para los factores fijos



(a) Calidad (b) Productividad

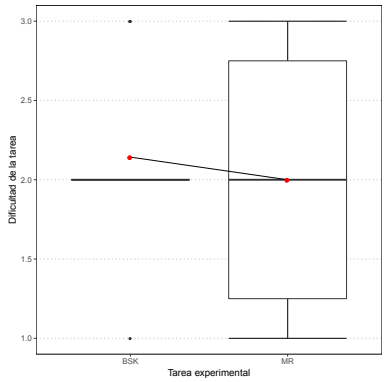
Figura 5.174: Gráficos QQ para los factores aleatorios

5.12.3.3. Influencia de factores instrumentales

En esta sección analizamos la dificultad de la tarea experimental percibida por los sujetos. Como hemos comentado en anteriores experimentos, las tareas fueron seleccionadas ad-hoc y por ende los resultados obtenidos podrían deberse al diseño experimental.

Tarea experimental

La figura 5.175a sugiere que la dificultad de las tareas experimentales MR y BSK es prácticamente similar. El test Kruskal-Wallis entre la tarea experimental y la dificultad percibida arroja un p-valor = 0,61, el cual no es significativo. En consecuencia, no parece que la dificultad de la tarea produzca las diferencias significativas entre BSK-MR para QLTY y PROD reportadas en la tabla 5.185.



(a) Dificultad percibida de la tarea

Figura 5.175: Efecto de la dificultad de la tarea percibida por los sujetos

5.12.4. Análisis de subgrupos

De acuerdo a los datos demográficos de los participantes mostrados en la sección 5.12.1.1, los aspectos que se pueden considerar en el análisis son la experiencia en Programación y la experiencia en Java. En cuanto a otros factores como el conocimiento del entorno Eclipse (que se incluye en este análisis), el uso de la técnica TDD y el uso de herramientas de pruebas, apenas un 20 % de sujetos presentan diferencias entre subgrupos. En el resto de aspectos preguntados, no existen diferencias entre sujetos que puedan ser usadas para el análisis.

5.12.4.1. Experiencia en programación

La experiencia en programación tiene una influencia negativa para la Calidad y es casi nula para la Productividad (B =-2.64 B =0). Los efectos no son significativos ($P-value$ =0.6 y $P-value$ =0.18). Existe una tendencia a mejorar notablemente tanto la Calidad como la Productividad cuando aplicaron TDD los desarrolladores menos experimentados (con menos de dos años de experiencia en programación). Los desarrolladores más experimentados prácticamente mantienen la misma Calidad y Productividad, como se aprecia en la figura 5.176.

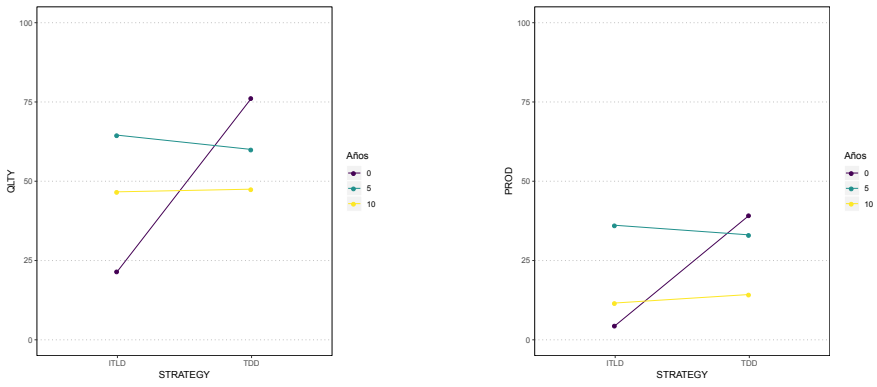
Tabla 5.186: Resultados del análisis estadístico para la Experiencia en Programación

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	196.28	196.28	1.00	11.00	0.22	0.6464
programmingExperience	251.32	251.32	1.00	12.00	0.28	0.6033
TASK	28784.16	28784.16	1.00	11.00	32.62	0.0001
SLICING	112.83	112.83	1.00	11.00	0.13	0.7274
GROUP	71.88	71.88	1.00	12.00	0.08	0.7802
STRATEGY:programmingExperience	101.79	101.79	1.00	11.00	0.12	0.7405

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.01	0.01	1.00	11.00	0.02	0.8958
programmingExperience	1.00	1.00	1.00	12.00	2.06	0.1768
TASK	12.25	12.25	1.00	11.00	25.33	0.0004
SLICING	0.21	0.21	1.00	11.00	0.43	0.5233
GROUP	0.09	0.09	1.00	12.00	0.18	0.6798
STRATEGY:programmingExperience	0.00	0.00	1.00	11.00	0.00	0.9888



(a) Calidad (b) Productividad

Figura 5.176: Efecto de la experiencia en programación. La experiencia se reporta redondeada a lustros.

5.12.4.2. Experiencia en programación Java

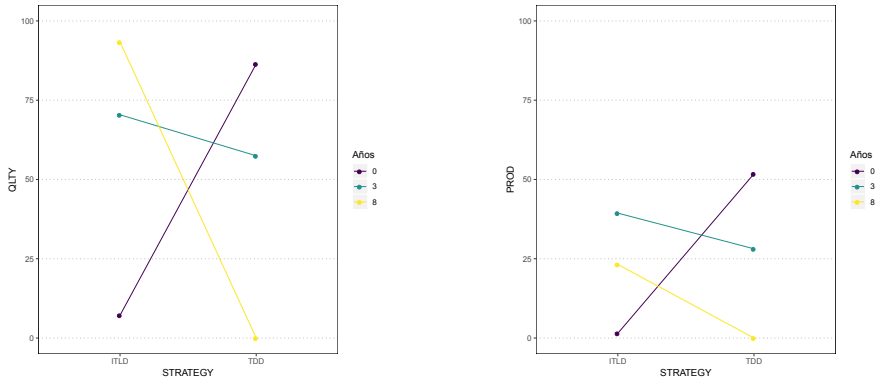
Tabla 5.187: Resultados del análisis estadístico para la Experiencia en Java

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	673.76	673.76	1.00	11.00	0.87	0.3701
javaExperience	255.95	255.95	1.00	12.00	0.33	0.5753
TASK	16602.94	16602.94	1.00	11.00	21.52	0.0007
SLICING	5.81	5.81	1.00	11.00	0.01	0.9324
GROUP	271.08	271.08	1.00	12.00	0.35	0.5644
STRATEGY:javaExperience	1320.28	1320.28	1.00	11.00	1.71	0.2175

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.39	0.39	1.00	11.00	0.92	0.3570
javaExperience	0.00	0.00	1.00	12.00	0.00	0.9804
TASK	7.01	7.01	1.00	11.00	16.65	0.0018
SLICING	0.38	0.38	1.00	11.00	0.91	0.3606
GROUP	0.09	0.09	1.00	12.00	0.22	0.6481
STRATEGY:javaExperience	0.69	0.69	1.00	11.00	1.63	0.2278



(a) Calidad (b) Productividad

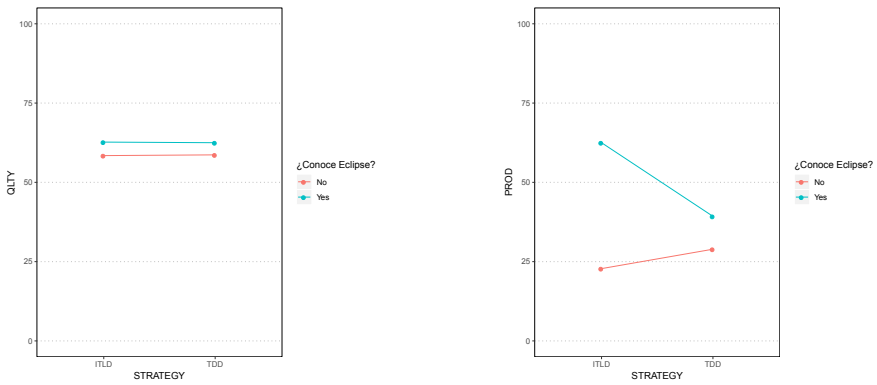
Figura 5.177: Efecto de la experiencia en programación Java. La experiencia se reporta redondeada en años.

El uso del lenguaje de programación Java presenta una influencia positiva para la Calidad y neutra para la Productividad (B =6.17 B =0). Los efectos tampoco son significativos en ambos casos ($P - value$ =0.58 y $P - value$ =0.98).

Los programadores con menor experiencia en Java, al igual que aquellos que tenían menor experiencia en programación, mejoraron notablemente al aplicar TDD. Al contrario, aquellos con mayor experiencia en Java son los

que obtuvieron menor Calidad y Productividad al aplicar TDD.

5.12.4.3. Conocimiento del entorno Eclipse



(a) Calidad (b) Productividad

Figura 5.178: Efecto del conocimiento del entorno Eclipse.

El conocimiento del entorno de desarrollo Eclipse presenta una influencia positiva para las variables Calidad y Productividad (B =14.94, B =0.01). Los efectos sin embargo, no son significativos ($P - value$ =0.81 y $P - value$ =0.8).

5.12.4.4. Resumen general

En las tablas 5.188 y 5.189 mostramos un resumen de los resultados de los análisis de subgrupos realizados. Ninguno de los factores analizados tiene un efecto que sea estadísticamente significativo.

Tabla 5.188: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	-2.64	0.6	2.19	0.74
Experiencia en Java	6.17	0.58	-8.36	0.22
Conocimiento del Entorno Eclipse	14.94	0.81	-22.97	0.41

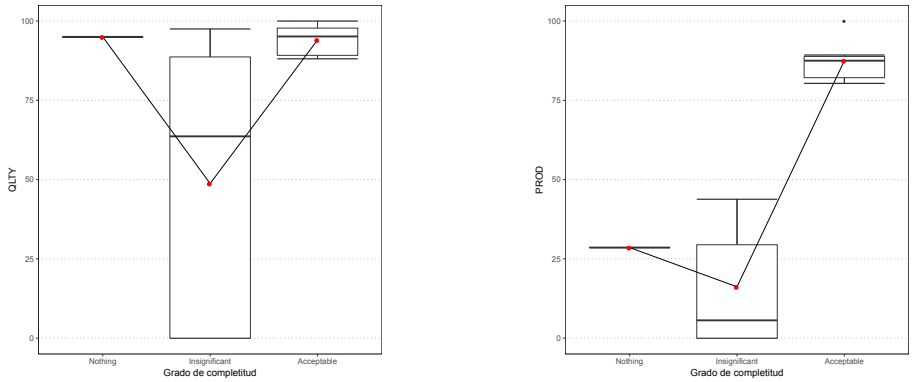
Tabla 5.189: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	0	0.18	0	0.99
Experiencia en Java	0	0.98	0	0.23
Conocimiento del Entorno Eclipse	0.01	0.8	-0.09	0.34

5.12.5. Grado de completitud

5.12.5.1. Aspectos generales

Como hemos venido realizando en otros experimentos, en esta sección analizamos manualmente el código entregado por los sujetos. La relación entre el grado de completitud frente a las variables QLTY y PROD se presenta en la figura 5.179. Adicionalmente, en la Tabla 5.190 se cuantifica el grado de completitud del trabajo realizado.



(a) Calidad (b) Productividad

Figura 5.179: Relación entre el grado de completitud y la calidad y productividad

Se entregaron un total de 30 tareas, de las cuales revisamos que 1 sujetos no hicieron nada. Por otro lado, 23 hicieron una mínima cantidad de trabajo y 6 sujetos realizaron el trabajo como era esperado.

Como ha ocurrido en experimentos anteriores, al incluir las tareas vacías los resultados difieren significativamente. Por ejemplo, podemos ver los siguientes datos: Promedio de QLTY para la estrategia TDD incluidos los sujetos que no realizaron un trabajo sustancial: 59.43 %. Promedio de QLTY para la estrategia TDD incluyendo únicamente a los sujetos que hicieron un

	Número de tareas entregadas
Nothing	1
Insignificant	23
Acceptable	6

Tabla 5.190: Grado de completitud

trabajo aceptable: 92.79 %.

5.12.5.2. Impacto de la estrategia de programación y la tarea experimental

Como se muestra en la tabla tabla 5.191, la estrategia de programación al parecer no tiene impacto en la distribución entre las tareas entregadas con los grados de finalización analizados. La diferencia es con una sola tarea que tiene calificación *Nothing* para TDD y que produce el desbalance al restarse de las tareas calificadas como *Insignificant* para esta estrategia.

	Nothing	Insignificant	Acceptable
ITLD	0	12	3
TDD	1	11	3

Tabla 5.191: Estrategia de programación

El test χ^2 que arroja los siguientes resultados: $\chi^2 = 1{,}04$, $df = 2$, $p - value = 0{,}59$, lo cual comprueba la inexistencia de una relación estadísticamente significativa entre la estrategia de programación y el grado de finalización de la tarea.

	Nothing	Insignificant	Acceptable
BSK	1	9	5
MR	0	14	1

Tabla 5.192: Grado de completitud por tarea experimental (BSK, MR)

Adicionalmente, la tabla 5.192 muestra que la mayor parte de tareas calificadas como *Acceptable* fueron para BSK. Sin embargo, las diferencias entre BSK y MR no significativas como muestra el test χ^2 ($\chi^2 = 4{,}75$, $df = 2$, $p - value = 0{,}09$).

5.12.5.3. Impacto de las variables demográficas

En las siguintes secciones analizaremos si el grado de completitud de las tareas, se encuentra reacionado con las variables demográficas obtenidas

de los sujetos experimentales. Vamos a analizar las mismas variables de la sección 5.12.4, esto es:

- Experiencia en Programación.
- Experiencia en Java.
- Conocimiento del entorno Eclipse.

Como podemos apreciar en la figura 5.180, la experiencia en programación al parecer no influye en el grado de completitud de las tareas. El test Kruskal-Wallis entre la experiencia en programación y el grado de completitud muestra un p-valor = 0,57 que no es significativo y nos confirma lo mostrado en la figura.

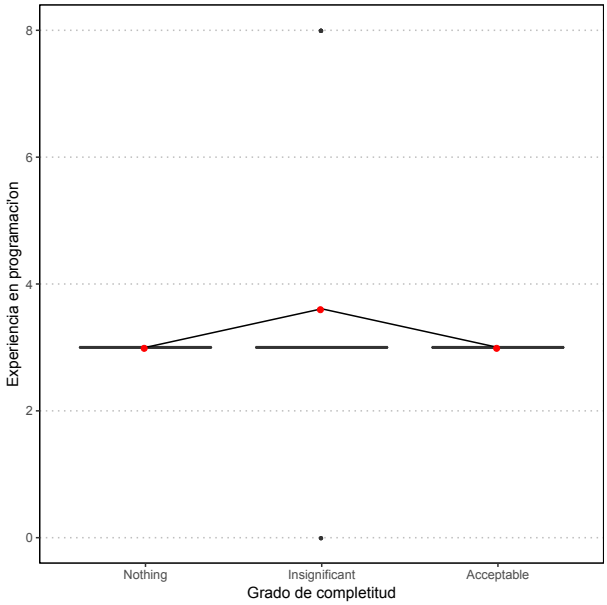


Figura 5.180: programmingExperience

Experiencia en programación Java

La experiencia en desarrollo con el lenguaje Java no parece influir en el grado de presentación de las tareas como podemos apreciar en la figura 5.181. Al realizar el test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud obtenemos un p-valor = 0,85 que no es significativo y nos confirma la impresión visual de la figura 5.181.

Conocimiento del entorno Eclipse

El conocimiento del entorno de desarrollo Eclipse si tiene influencia en el grado de completitud de las tareas experimentales. El test χ^2 nos indica

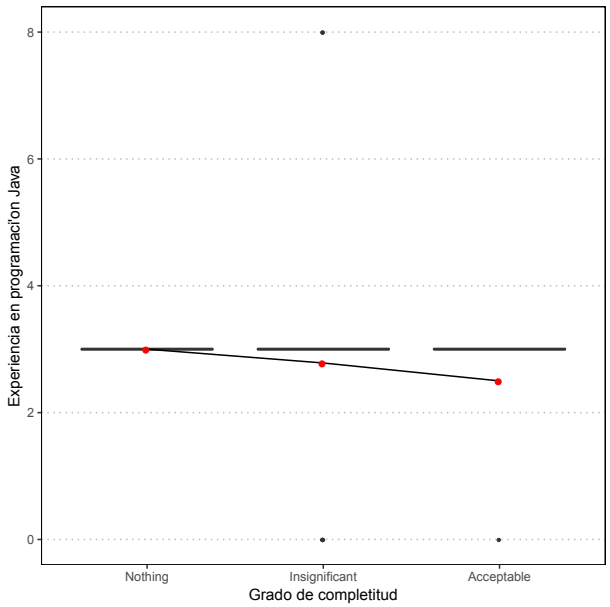


Figura 5.181: javaExperience

	Nothing	Insignificant	Acceptable
No	0	21	3
Yes	1	2	3

Tabla 5.193: Conocimiento del entorno eclipse

este hecho ($\chi^2 = 9{,}21$, $df = 2$, $p-value = 0{,}01$). En la tabla 5.193, podemos apreciar la distribución de los sujetos que indicaron tener o no conocimiento del entorno Eclipse, frente al grado de completitud de las tareas. Existe un elevado número de tareas donde el trabajo realizado fue catalogado como *Insignificant* y donde los sujetos indicaron que no tenían conocimiento previo del entorno Eclipse. Este hecho podría ser la causa de las diferencias significativas.

5.12.5.4. Resumen general impacto variables demográficas

En la tabla 5.194 se resume los resultados de los análisis de la influencia de las variables demográficas estudiadas. Existen diferencias significativas entre el *Conocimiento del entono Eclipse* y el grado de completitud de las tareas entregadas.

Tabla 5.194: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Experiencia en programación	0.57
Experiencia en Java	0.85
Conocimiento del entorno Eclipse	0.01

5.13. Experimento UNLP2015

5.13.1. Ejecución

El experimento UNLP2015 es el primero de los experimentos realizados con estudiantes de postgrado, y es una replicación del experimento ESPE2015 en el ámbito académico. Este experimento se realizó con un grupo de estudiantes de doctorado de la Universidad Nacional de La Plata de Argentina (UNLP) mientras realizaban sus actividades de investigación.

5.13.1.1. Muestra

En el experimento UNLP2015 participaron 9 estudiantes de post-grado de la Universidad Nacional de La Plata de Argentina. En general los estudiantes fueron jóvenes menores de 30 años sin mayor experiencia profesional, como se indica en la tabla 5.195. Las diferencias más significativas que encontramos entre grupos de sujetos son:

- La experiencia en programación se encuentra distribuida en niveles intermedio, novato y sin experiencia.
- Aproximadamente el 45 % no tiene experiencia en lenguaje Java e igual porcentaje indicó tener entre 2 y 5 años de experiencia. Un estudiante que representa el 10 % restante indicó tener entre 6 y 10 años de experiencia.
- Existe un 44 % de estudiantes que tienen entre 2 y 5 años de experiencia en JUnit. El resto indica no tener mayor experiencia.

5.13.1.2. Preparación

Este experimento se lo realizó en uno de los laboratorios de computación de la UNLP y, como en anteriores experimentos, se instaló la máquina virtual de Java, el framework Junit, el plugin para empaquetar y realizar mediciones y el IDE de desarrollo Eclipse.

Tabla 5.195: Características demográficas de los sujetos del experimento UNLP2015

EXPERIMENTO: UNLP2015		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	9
	30 >= Edad < 40 años	9
	40 >= Edad < 50 años	9
	Edad >= 50 años	9
Nivel de Educación	Other	0
	Undergraduate	0
	Bachelor	0
Experiencia profesional	Sin experiencia (<2 años)	9
	Novato (2 - 5 años)	9
	Intermedio (6 - 10 años)	9
	Experto (>10 años)	9
Experiencia en programación	Sin experiencia (<2 años)	2
	Novato (2 - 5 años)	3
	Intermedio (6 - 10 años)	4
	Experto (>10 años)	0
Uso de herramientas de pruebas	Yes	8
	No	1
Experiencia en lenguaje Java	Sin experiencia (<2 años)	4
	Novato (2 - 5 años)	4
	Intermedio (6 - 10 años)	1
	Experto (>10 años)	0
Experiencia en framework JUnit	Sin experiencia (<2 años)	6
	Novato (2 - 5 años)	3
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	1
	No	8
Experiencia en TDD	Sin experiencia (<2 años)	9
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	1
	No	8
Conocimiento del entorno Eclipse	Yes	0
	Yes	9
Función actual en la organización	Tester	0
	Developer	0
	Student	9

5.13.1.3. Realización

El experimento se realizó como una actividad colaborativa de investigación organizada por Claudia Pons con alumnos doctorandos del Laboratorio de Investigación y Formación en Informática Avanzada (Lifia). El experimento se realizó siguiendo el mismo protocolo utilizado en el experimento UADY2015, esto es, mediante un seminario taller de 5 días con un horario de cuatro horas por día y con la participación de los mismos investigadores de UADY2015. Por esta razón, hemos omitido la descripción en detalle del protocolo en este experimento.

5.13.1.4. Desviaciones

No existieron desviaciones que se consideren importantes durante la realización de experimento. En general se siguió el cronograma establecido y los estudiantes asistieron puntualmente a cada una de las sesiones, excepto uno de ellos que no asistió a la sesión final.

5.13.1.5. Reducción del conjunto de datos

A este experimento asistieron 9 sujetos, y de ellos un único sujeto no asistió a la segunda sesión experimental y por tanto no realizó la tarea aplicando la técnica de TDD. No pudimos conocer la causa de este abandono.

5.13.2. Estadísticos descriptivos

En la siguiente sección, describimos los datos estadísticos obtenidos para las variables respuestas de Calidad y Productividad.

5.13.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	83.57	17.10	-0.61	-0.94
TDD	83.86	16.99	-1.10	0.10

Tabla 5.196: Estadísticos descriptivos para QLTY

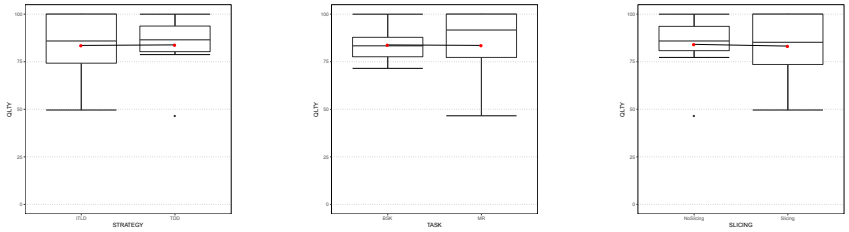
La calidad obtenida por los participantes es relativamente alta. Las medias obtenidas por los grupos ITLD y TDD son similares, tal y como se indica en la tabla 5.196. La asimetría y la curtosis para ITLD es negativa, en tanto que para TDD la asimetría es negativa y la curtosis positiva. Esto implica que la cola de distribución se alarga para valores inferiores a la media existiendo una menor concentración de datos entorno a la misma.

En lo gráficos de box-plot de la figura 5.182 podemos apreciar que las medianas para las dos estrategias (ITLD y TDD) son bastante coincidentes.

Además, ambas cajas se sitúan sobre el primer cuadrante por lo que no prevemos la existencia de significación estadística.

En cuanto a la tarea, al contrario de la mayoría de experimentos antes analizados, el promedio de MR es ligeramente superior al de BSK, como indica la figura 5.182b. No podemos apreciar claramente si los efectos de la tarea son significativos para la Calidad.

El efecto del nivel de descripción de la tarea (slicing) tiene una mediana similar y al parecer esta factor no ejerce efectos significativos sobre la calidad como puede observarse en la figura 5.182c.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.182: Box plots para la variable respuesta QLTY

5.13.2.2. Productividad

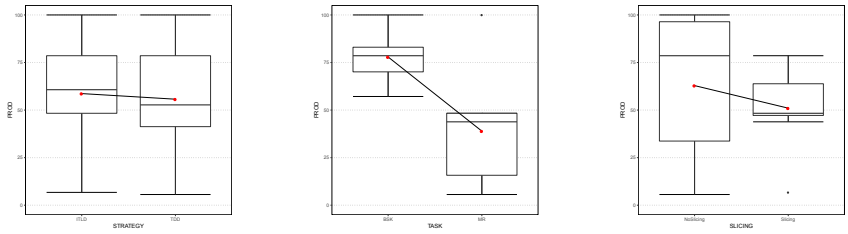
Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	58.67	32.60	-0.28	-1.44
TDD	55.72	29.76	-0.13	-1.25

Tabla 5.197: Estadísticos descriptivos para PROD

La Productividad tiene promedios menores que la Calidad, como se muestra en la tabla 5.197, con desviaciones estándar relativamente elevadas para las dos estrategias. Tanto la asimetría como la curtosis en este caso son negativas. En la figura 5.183a, se aprecia gráficamente que no existen diferencias sustanciales entre ITLD y TDD. La mediana de ITLD es ligeramente superior a la de TDD, y la mayoría de datos se agrupan en el tercer cuartil.

En la figura 5.183b se nota claramente el impacto de la tarea para la Productividad. Al contrario de la Calidad, en este caso BSK obtiene valores considerablemente superiores que MR, y el efecto parece significativo.

En cuanto al nivel de definición de la tarea (Slicing), al parecer no tiene influencia sobre la productividad, aunque la mediana se ubica por sobre el 75 % para el nivel No-Slicing y por debajo del 50 % para el nivel slicing.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.183: Box plots para la variable respuesta PROD

5.13.3. Prueba de hipótesis

5.13.3.1. Análisis estadístico

La estrategia de análisis estadístico en este caso, sigue las mismas pautas de los experimentos hechos con estudiantes. Los resultados de la ANOVA respecto a las variables respuesta Calidad y Productividad se muestran en la tabla 5.198. Los resultados no son significativos en casi todos los casos. La única excepción es la Tarea que obtiene resultados estadísticamente significativos para el caso de la Productividad (aunque BSK sigue siendo mayor que MR como ha sido habitual).

Tabla 5.198: Resultados del análisis estadístico

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	5499459605.56	5499459605.56	1.00	8059.80	0.12	0.7248
TASK	5710811418.35	5710811418.35	1.00	8059.80	0.13	0.7198
SLICING	2198160713.15	2198160713.15	1.00	4307.17	0.05	0.8239
GROUP	15172439169.26	15172439169.26	1.00	6.83	0.34	0.5775

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	32.27	32.27	1.00	4.72	0.55	0.4948
TASK	8404.24	8404.24	1.00	4.72	142.36	0.0001
SLICING	82.63	82.63	1.00	4.74	1.40	0.2927
GROUP	27.54	27.54	1.00	6.63	0.47	0.5177

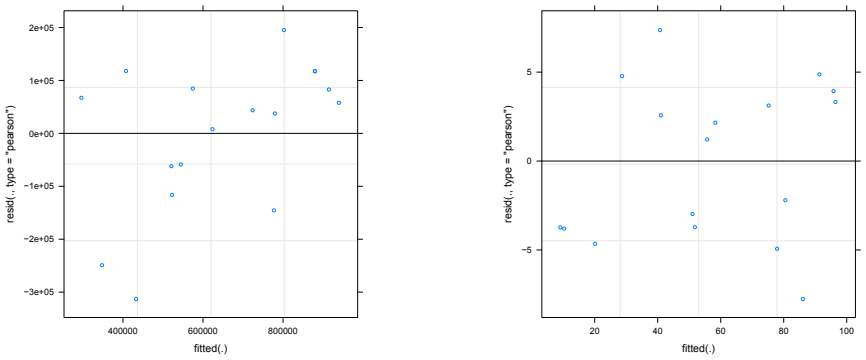
5.13.3.2. Chequeo del análisis estadístico

El chequeo del análisis estadístico sigue el mismo patrón que el experimento base ESPE2015. Como se ha mencionado, de las tres asunciones

dadas para los modelos mixtos (linealidad entre factores, homogeneidad de varianzas y distribución normal de residuos) comprobaremos la segunda y tercera condición ya que en este experimento al igual que en los restantes, los factores tienen solo dos niveles, por lo tanto, la linealidad siempre se cumple.

Homogeneidad de varianzas

Los gráficos de valores predichos vs. residuos estandarizados se muestran en la figura 5.184a. No se aprecia la existencia de ninguna figura de embudo que sugiera heterogeneidad entre varianzas.



(a) Calidad (b) Productividad

Figura 5.184: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

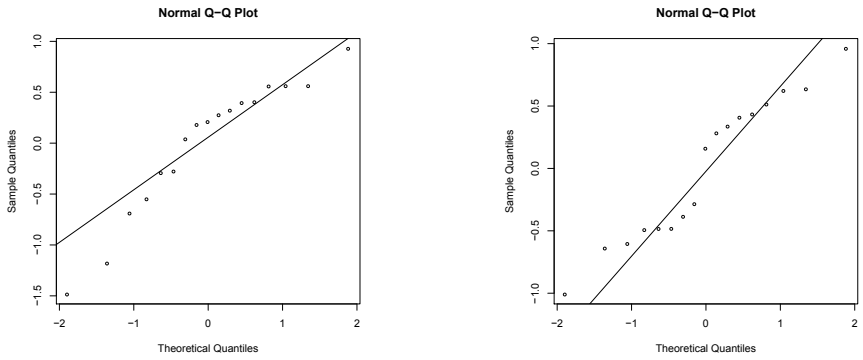
Normalidad de residuos

Estudiamos los residuos tanto de los factores fijos como de los factores aleatorios. En lo que respecta a los factores fijos, en la figura 5.185 podemos apreciar la distribución de los residuos para las variables respuesta QLTY y PROD. Notamos desviaciones a lo largo de la recta, por lo que la gráfica no nos permite apreciar la existencia de normalidad de residuos.

El test de Shapiro-Wilks indica que existe normalidad de los residuos para la Productividad ($W = 0{,}93$, $p - value = 0{,}23$). Sin embargo, en el caso de la Calidad, el test indica que los residuos no están distribuidos normalmente. Por ello, hemos aplicado una transformación Box-Cox con $\lambda = 3$, lo que produce que los residuos sean normales ($W = 0{,}91$, $p - value = 0{,}11$).

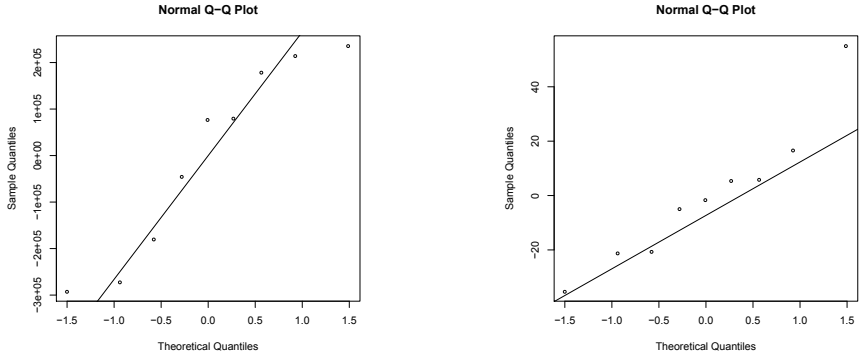
En lo que respecta a los factores aleatorios, la figura 5.186 muestra el gráfico QQ de los residuos para las variables QLTY y PROD. Excepto por los extremos, los datos se ajustan razonablemente a la distribución normal.

El test de Shapiro-Wilks nos permite confirmar la existencia de normalidad para la Calidad ($W = 0{,}9$, $p - value = 0{,}23$) y para la Productividad ($W = 0{,}93$, $p - value = 0{,}49$).



(a) Calidad (b) Productividad

Figura 5.185: Gráficos QQ para los factores fijos



(a) Calidad (b) Productividad

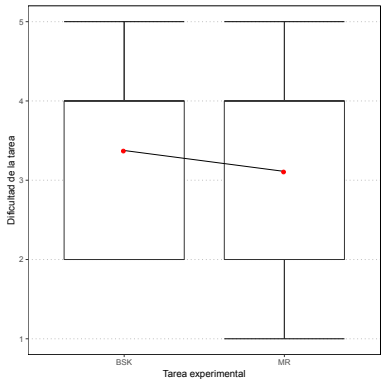
Figura 5.186: Gráficos QQ para los factores aleatorios

5.13.3.3. Influencia de factores instrumentales

En esta sección analizamos la dificultad de la tarea experimental percibida por los sujetos. Como en todas las instancias experimentales, las tareas fueron seleccionadas ad-hoc. Nuestro interés con este análisis es determinar si los resultados obtenidos podrían deberse al diseño experimental.

Tarea experimental

Como podemos observar en la figura 5.187a, la complejidad percibida por los sujetos es similar para MR y BSK. Esto lo podemos confirmar mediante el test Kruskal-Wallis entre la tarea experimental y la dificultad percibida, que arroja un p-valor = 0,68. Por lo tanto, aunque la tarea mostró diferencias significativas para la Productividad (ver sección **??**), los sujetos consideraron que el grado de dificultad no es un factor que probablemente incida en las variables estudiadas.



(a) Dificultad percibida de la tarea

Figura 5.187: Efecto de la dificultad de la tarea percibida por los sujetos

5.13.4. Análisis de subgrupos

Conforme las diferencias encontradas en los datos demográficos de la sección 5.13.1.1, en esta para el análisis de subgrupos consideramos los siguientes aspectos:

- Experiencia en programación.
- Experiencia en Java.
- Experiencia de el framework de pruebas JUnit.

5.13.4.1. Experiencia en programación

La experiencia en programación muestra una influencia negativa para las variable Calidad (B =-33.73) y Productividad (B =-1.67). El análisis también nos indica que los efectos no son significativos en ambos casos ($P-valor$ =0.82 y $P-valor$ =0.77). En el gráfico 5.188 se aprecia que los desarrolladores con un nivel de experiencia en programación intermedia (en el rango de 6 a 10 años de experiencia) mejoran su Calidad y Productividad al aplicar TDD. En contraste, aquellos programadores menos experimentados (considerados como novatos, entre 0 y menos de 2 años de experiencia),

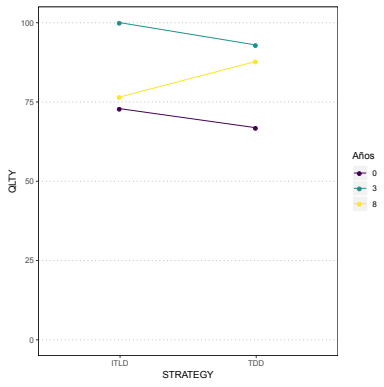
Tabla 5.199: Resultados del análisis estadístico para la experiencia en programación

(a) Calidad

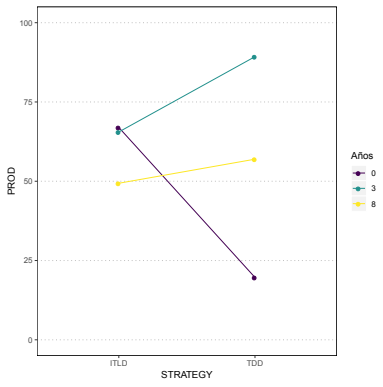
	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	95163037862.58	95163037862.58	1.00	250.02	3.31	0.0700
programmingExperience	1687455975.30	1687455975.30	1.00	6.02	0.06	0.8166
TASK	65430893996.31	65430893996.31	1.00	533.85	2.28	0.1320
SLICING	65965231464.74	65965231464.74	1.00	246.18	2.29	0.1311
GROUP	2518272071.17	2518272071.17	1.00	6.23	0.09	0.7769
STRATEGY:programmingExperience	130757753215.11	130757753215.11	1.00	332.57	4.55	0.0337

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	10.04	10.04	1.00	3.81	0.13	0.7384
programmingExperience	7.58	7.58	1.00	5.12	0.10	0.7671
TASK	4830.67	4830.67	1.00	3.66	62.16	0.0020
SLICING	0.26	0.26	1.00	3.81	0.00	0.9568
GROUP	33.03	33.03	1.00	5.22	0.43	0.5421
STRATEGY:programmingExperience	27.58	27.58	1.00	3.74	0.35	0.5855



(a) Calidad

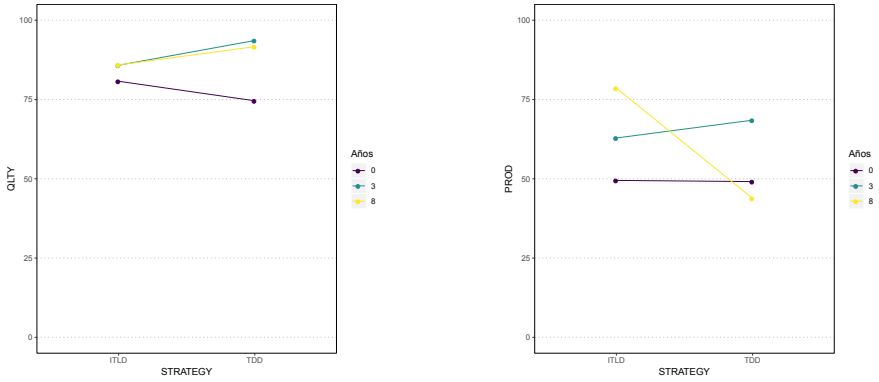


(b) Productividad

Figura 5.188: Efecto de la experiencia en programación. La experiencia se reporta redondeada a años.

disminuyeron su Calidad y Productividad. Por otra parte, los programadores dentro del rango de 2 a 5 años de experiencia, disminuyeron la Calidad al aplicar TDD y mejoraron su Productividad al usar esta misma técnica. La interacción *STRATEGY*programmingExperience*, en este caso resulta ser significativa para la Calidad.

5.13.4.2. Experiencia en programación Java



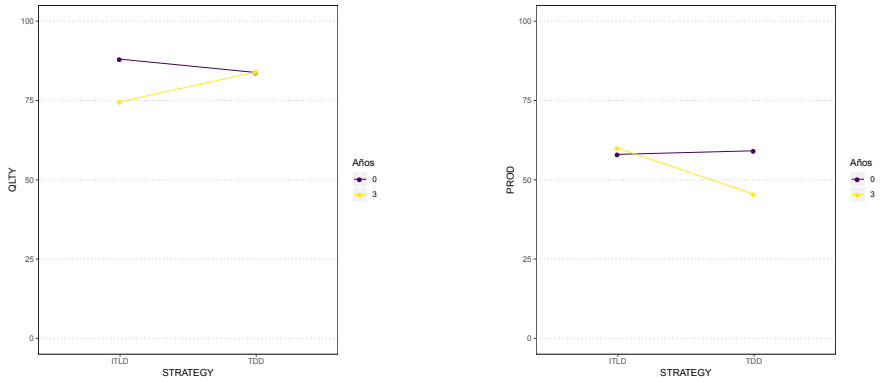
(a) Calidad (b) Productividad

Figura 5.189: Efecto de la experiencia en programación Java. La experiencia se reporta redondeada en años.

La experiencia que indicaron tener los sujetos en el lenguaje de programación Java tiene una influencia positiva para la Calidad y para la Productividad (B =28.59 B =2.95). Los efectos en este caso tampoco son significativos ($P-value$ =0.35 y $P-value$ =0.44) por lo que omitimos la tabla de análisis. En la figura 5.189 se observa el patrón de compartamiento al aplicar TDD e ITLD tanto para la Calidad como para la Productividad. Los programadores con mayor experiencia en Java mejoraron la Calidad al aplicar TDD, no así para la Productividad, donde la tendencia es decreciente al aplicar TDD. Los desarrolladores menos experimentados también muestran una ligera tendencia decreciente al aplicar TDD para la Calidad. Para la Productividad no existen deiferencias notables cuando este grupo de desarrolladores aplicaron las dos técnicas. El grupo de desarrolladores con una experiencia intermedia en Java, muestran una cierta mejora tanto de Calidad como de Productividad al aplicar TDD (existe cierto paralelismo entre lineas, como en casos anteriores).

5.13.4.3. Experiencia en el Framework JUnit

La experiencia de los sujetos en el framework JUnit presenta efectos negativos para la Calidad (B =-44.16) y positivos para la Productividad



(a) Calidad (b) Productividad

Figura 5.190: Efecto de la experiencia en el Framework JUnit. La experiencia se reporta redondeada en años

(B =2.2). Una vez más, los efetos no son significativos ($P-value$ =0.98 y $P-value$ =0.64). En este caso, la gráfica 5.190 muestra que los desarroladores con más experiencia en el Framework JUnit mejoran la Calidad, pero empeoran su Productividad al aplicar TDD. Los desarrolladores sin experiencia en el Framework JUnit prácticamente mantienen constante su Calidad y Productividad con ambas técnicas. La interacción *STRATEGY*jUnitExperience* también ha resultado significativa en este caso.

5.13.4.4. Resumen general

En las tablas 5.200 y 5.201 mostramos un resumen de los resultados de los análisis de subgrupos realizados. En este experimento, no existen efectos significativos para ninguna de las variables consideradas. Sin embargo, se observa que las interacciones *STRATEGY * Variable* para la experiencia en programación y para la experiencia en el framework Junit resultan significativas para la Calidad. La limitada cantidad de sujetos no nos permite hacer afirmaciones que sean relevantes.

Tabla 5.200: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	-33.73	0.82	45.15	**0.03**
Experiencia en Java	28.59	0.35	33.02	0.45
Experiencia en el Framework JUnit	-44.16	0.98	55.18	**0**

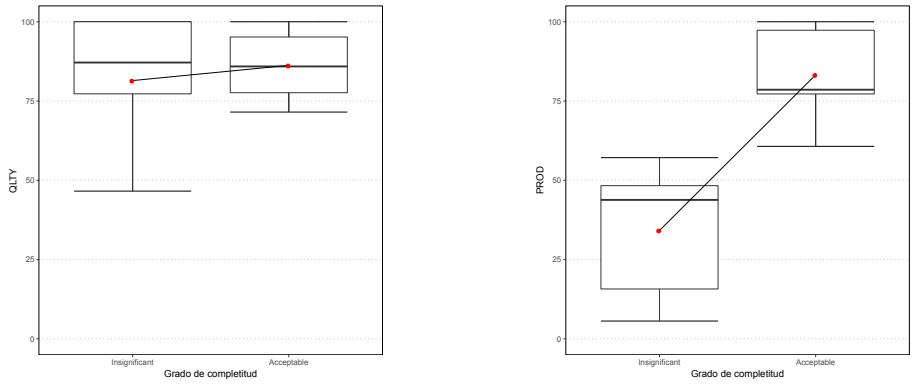
Tabla 5.201: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Experiencia en Programación	-1.67	0.77	1.42	0.59
Experiencia en Java	2.95	0.35	0.81	0.45
Experiencia en el Framework JUnit	2.2	0.64	3.48	0.32

5.13.5. Grado de completitud

5.13.5.1. Aspectos generales

En este experimento, observamos que aproximadamente la mitad de sujetos hicieron las tareas de acuerdo a como se esperaba, por lo que se les calificó como *Aceptable*. La otra mitad realizó unas pocas líneas de código y se calificó como *Insignificant*. Ninguno de los participantes entregó tareas vacías como puede verse en la tabla 5.202. Por otro lado, en la figura 5.191, se muestra la relación entre el grado de completitud frente a las variables QLTY y PROD.



(a) Calidad (b) Productividad

Figura 5.191: Relación entre el grado de completitud y la calidad y productividad

De un total de 17 tareas recogidas, 0 sujetos no hicieron nada. Por otro lado, 9 hicieron una mínima cantidad de trabajo y 8 sujetos realizaron el trabajo como era esperado.

Al no tener tareas vacías, en este caso el análisis no se ve afectado significativamente para la Productividad. Al comparar el promedio de la QLTY

	Número de tareas entregadas
Nothing	0
Insignificant	9
Acceptable	8

Tabla 5.202: Grado de completitud

de la segunda sesión, incluidos los sujetos que no realizaron un trabajo sustancial, es del 83.86 %; cuando se excluyen estos sujetos, la QLTY es del 88.22 %.

Al realizar el mismo análisis para la Productividad, por el contrario, obtenemos que el promedio de PROD de la segunda sesión, incluidos los particpantes que no realizaron un trabajo sustancial es del 55.72 %; sin considerar estos sujetos, la PROD es del 85.71 %.

5.13.5.2. Impacto de la estrategia de programación y la tarea experimental

La tabla 5.203, nos induce a pensar que la *estrategia de programación* no parece estar relacionada con el grado de finalización de las tareas. Existe el mismo número de sujetos que no hicieron nada tanto para TDD como para ITLD. Existe una mínima diferencia entre el número de sujetos que hicieron las tareas de manera aceptable y la estrategia de programación.

	Nothing	Insignificant	Acceptable
ITLD	0	4	5
TDD	0	5	3

Tabla 5.203: Estrategia de programación

Podemos indicar que no existe relación entre la estrategia de programación y el grado de finalización de las tareas. Esto se comprueba mediante la realización de un test χ^2 (hemos excluído de la tabla los valores nulos (*Insignificant*) para que se pueda realizar el test). El test arroja los siguientes resultados: $\chi^2 = 0{,}07$, $df = 1$, $p - value = 0{,}8$.

	Nothing	Insignificant	Acceptable
BSK	0	1	7
MR	0	8	1

Tabla 5.204: Grado de completitud por tarea experimental (BSK, MR)

La tabla 5.204 muestra cierta relación entre el grado de completitud y las tareas. En este caso, el doble de sujetos no hicieron nada en la tarea MR, aunque casi el mismo número de sujetos entregaron las dos tareas de

forma aceptable. Al excluir las tareas nulas (*Nothing*), las diferencias entre BSK y MR son significativas como muestra el test χ^2 ($\chi^2 = 7{,}09$, $df = 1$, $p - value = 0{,}01$). Esto indica que MR es probablemente más complejo que BSK.

5.13.5.3. Impacto de las variables demográficas

Siguiendo con la misma metodología de análisis utilizado en todos los experimentos, analizamos si el grado de completitud de las tareas se encuentra reacionado con las variables demográficas obtenidas de los sujetos experimentales. Analizaremos las mismas variables de la sección 5.13.4, esto es:

- Experiencia en programación.
- Experiencia en Java.
- Experiencia en el framework de pruebas JUnit.

Experiencia en programación

Como se aprecia en la figura 5.192, la experiencia en programación al parecer no tiene impacto sobre el grado de completitud de las tareas. Los máximos y mínimos que se observan en el gráfico tienen cierta diferencia en el nivel inferior. Algunos sujetos que entregaron la tarea de manera aceptable son programadores menos experimentados. El test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,74 el mismo que no es significativo.

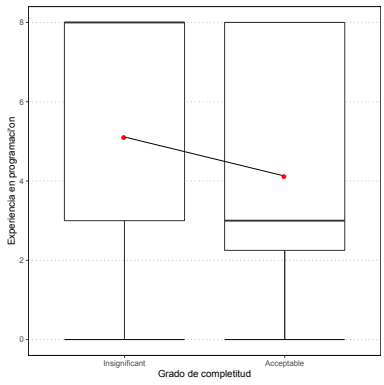


Figura 5.192: programmingExperience

Experiencia en programación Java

En cuanto a la experiencia en desarrollo con el lenguaje Java, al parecer tampoco influye en el grado de presentación de las tareas tal como se aprecia en la figura 5.193. El test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud no es significativo, con un p-valor = 0,96.

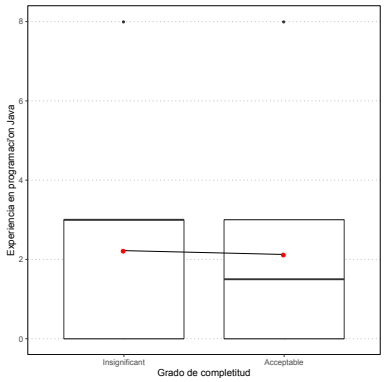


Figura 5.193: javaExperience

Experiencia en el framework de pruebas JUnit

En cuanto a la experiencia en el Framework de pruebas JUnit, en la figura 5.194. El test Kruskal-Wallis entre la experiencia en JUnit y el grado de completitud no es significativo, con un p-valor = 0,72.

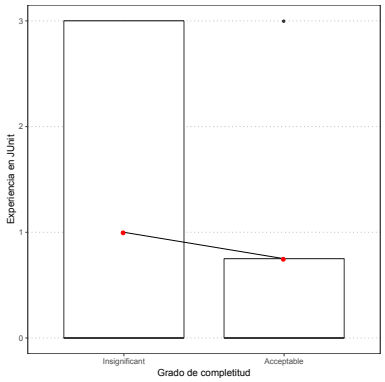


Figura 5.194: jUnitExperience

5.13.5.4. Resumen general impacto variables demográficas

La tabla 5.205 resume los resultados de los análisis de la influencia de las variables demográficas estudiadas frente al grado de completitud de las

tareas realizadas. No existen diferencias que sean estadísticamente significativas.

Tabla 5.205: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Experiencia en programación	0.74
Experiencia en Java	0.96
Experiencia en el framework JUNit	0.72

5.14. Experimento EcuadorUTN2017

5.14.1. Ejecución

El experimento UTN2017 es el último de la serie de experimentos realizados en el ámbito académico. Este experimento se llevó a cabo en la Universidad Técnica del Norte de Ecuador (UTN) con estudiantes de postgrado, todos ellos profesionales en el área de Ciencias de la Computación. La mayoría se encontraba desempeñando funciones relacionadas con el desarrollo de software. UTN2017 es una replicación del experimento ESPE2015.

5.14.1.1. Muestra

En el experimento UTN2017 participaron 27 estudiantes de la maestría en Ingeniería de Software perteneciente a la Universidad Técnica del Norte (UTN). En la tabla 5.206 podemos observar que ningún participante indicó tener experiencia previa en TDD. La mayoría reportó que no tenía experiencia en el framework JUnit (apenas un 7 % indicó tener entre 2 y 5 años de experiencia). El 85 % indicó que no había utilizado la técnica TDD y la mayoría de participantes (aproximadamente el 81 %) señaló tener conocimiento del entorno de desarrollo eclipse. Como puede notarse, las diferencias más notables se dan en los siguientes aspectos:

- La mayor parte de los participantes (70 %) tenían edades comprendidas entre 30 y 40 años. Un 11 % eran más jóvenes y un 19 % superaba los 40 años de edad.
- Existen participantes con diferentes niveles de experiencia en programación, siendo la mayor parte (un 40 %) programadores con nivel intermedio.
- La mayoría indicó no tener experiencia en Java (un 56 %). Los restantes programadores se caracterizaron como expertos (un 7 %), intermedios (un 11 %) y novatos (un 26 %).

- Aproximadamente un 37 % indicó haber utilizado herramientas de pruebas.
- Aproximadamente un 34 % indicó tener entrenamiento previo en pruebas unitarias.
- Los participantes desempeñaban diferentes funciones en la organización. La mayor parte eran desarrolladores (33 %), seguidos por analistas (19 %) y gestores (15 %). El 33 % restante, se encontraba desempeñando otras funciones.

5.14.1.2. Preparación

Este experimento fue realizado en las aulas de clase de la Universidad. Los participantes instalaron en sus computadores personales las herramientas necesarias. Se les solicitó instalar la máquina virtual de Java y el framework Junit. En este caso, los participantes entregaron el código realizado en un archivo empaquetado, y posteriormente se hizo las mediciones utilizando el plugin desarrollado para el efecto.

5.14.1.3. Realización

El experimento se realizó como parte del módulo de Gestión de proyectos de software, de la Maestría en Ingeniería de Software ofertada por la Universidad Técnica del Norte. No obstante, los estudiantes eran profesionales en activo, 18 de ellos se encontraban desempeñando funciones relacionadas con el desarrollo de software. El experimento se realizó en dos fines de semana consecutivos dentro del horario de clase del módulo que fue de 4 horas diarias. El experimento se realizó en cuatro días con el siguiente protocolo:

- **Día 1:** Los sujetos llenaran el cuestionario demográfico previo al inicio del entrenamiento. Durante el primer día se realizó una sesión de introducción al desarrollo ágil y formación en pruebas unitarias utilizando el framework Junit.
- **Día 2:** En el segundo día de la primera semana, se capacitó en las técnicas ITLD y slicing. Posteriormente se realizó la primera sesión experimental que consistió en la solución de las atreas BSK o MR con o sin slicing y aplicando la técnica ITLD. Posteriormente.
- **Día 3:** En el tercer día de la segunda semana, se realizó una sesión de entrenamiento en la técnica TDD.
- **Día 4:** Finalmente, en el cuarto día de la segunda semana se realizó la segunda tarea experimental, en este caso solucionar BSK o MR con o sin Slicing y aplicando la estrategia TDD.

Tabla 5.206: Características demográficas de los sujetos del experimento UTN2017

EXPERIMENTO: UTN2017		
Características	**Nivel**	**Número de sujetos**
Edad	Edad < 30 años	3
	30 >= Edad < 40 años	19
	40 >= Edad < 50 años	5
	Edad >= 50 años	0
Nivel de Educación	Other	0
	Undergraduate	0
	Bachelor	0
Experiencia profesional	Sin experiencia (<2 años)	1
	Novato (2 - 5 años)	15
	Intermedio (6 - 10 años)	8
	Experto (>10 años)	3
Experiencia en programación	Sin experiencia (<2 años)	1
	Novato (2 - 5 años)	8
	Intermedio (6 - 10 años)	11
	Experto (>10 años)	7
Uso de herramientas de pruebas	Yes	10
	No	17
Experiencia en lenguaje Java	Sin experiencia (<2 años)	15
	Novato (2 - 5 años)	7
	Intermedio (6 - 10 años)	3
	Experto (>10 años)	2
Experiencia en framework JUnit	Sin experiencia (<2 años)	25
	Novato (2 - 5 años)	2
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Uso de la técnica TDD	Yes	4
	No	23
Experiencia en TDD	Sin experiencia (<2 años)	27
	Novato (2 - 5 años)	0
	Intermedio (6 - 10 años)	0
	Experto (>10 años)	0
Entrenamiento previo en desarrollo de pruebas unitarias	Yes	9
	No	18
Conocimiento del entorno Eclipse	Yes	22
	No	5
Función actual en la organización	Manager	4
	Developer	9
	Analyst	5

Como en la mayoría de la serie de experimentos realizados, la capacitación estuvo a cargo de Geovanny Raura con la colaboración de Rodrigo Fonseca y se utilizaron los materiales de entrenamiento preparados por Oscar Dieste.

5.14.1.4. Desviaciones

El protocolo se cumplió de acuerdo a lo establecido. Las horas de inicio y culminación de las actividades experimentales y de la capacitación se cumplieron dentro de lo establecido sin retrasos.

5.14.1.5. Reducción del conjunto de datos

Prácticamente no existieron variaciones en el número de sujetos que asistieron al experimento, excepto por un único participante que por razones de fuerza mayor no asistió la primera semana de ejecución del experimento. Se presentaron 27 participantes y de ellos 26 realizaron la primera tarea experimental, es decir la aplicación de la estrategia ITLD. El total de participantes realizaron la segunda tarea utilizando la estrategia TDD.

5.14.2. Estadísticos descriptivos

Se describen por separado cada una de las variables respuestas de Calidad y Productividad obtenidas en esta instancia experimental.

5.14.2.1. Calidad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	62.58	35.02	-0.86	-0.79
TDD	70.77	24.70	-0.53	-1.20

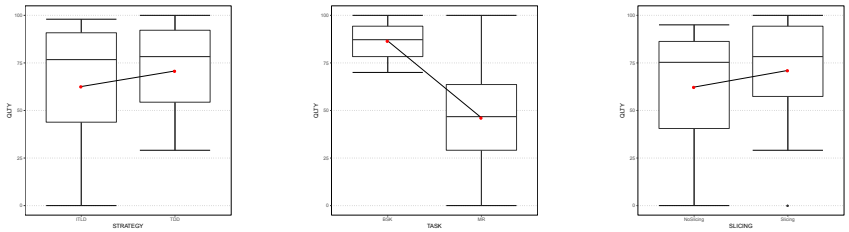
Tabla 5.207: Estadísticos descriptivos para QLTY

La tabla 5.207 muestra que la Calidad obtenida con la estrategia TDD es algo superior a lo obtenido para ITLD y superan el sesenta por ciento. Sin embargo, las desviaciones estándar son relativamente elevadas con asimetría y curtosis negativas en ambos casos y no tan altas.

De acuerdo al box-plot mostrado en la figura 5.195 las medianas son casi coincidentes para ambas estrategias como se muestra la tabla 5.207). No se visualiza la presencia de valores nulos para la estrategia TDD y la dispersión de puntos es menor que para ITLD.

En relación a la tarea, BSK sigue obteniendo mayores valores de Calidad que MR, tal y como indica la figura 5.195b. Al parecer los resultados son significativos para la tarea.

Finalmente, el efecto del factor Slicing presenta medianas similares con la mayor concentración de datos por encima del 50 %, como se aprecia en la figura: 5.195c.



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.195: Box plots para la variable respuesta QLTY

5.14.2.2. Productividad

Estrategia de Programación	Promedio	Desviación estandard	Asimetría	Curtosis
ITLD	26.82	30.19	1.31	0.53
TDD	37.40	31.34	0.58	-1.07

Tabla 5.208: Estadísticos descriptivos para PROD

La Productividad también es algo superior para la estrategia TDD sobre ITLD, aunque los promedios obtenidos son significativamente menores que para la Calidad. Como se observa en la tabla 5.208, los promedios están por debajo del 50 por ciento. La asimetría es positiva para las dos estrategias y la curtosis es opuesta con valores no excesivamente altos.

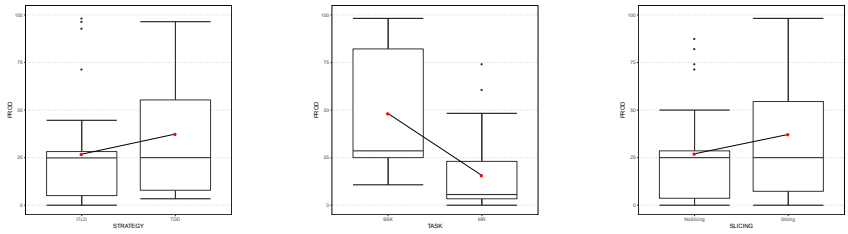
Esta tendencia podemos visualizarla de mejor manera en la figura 5.196. Se aprecia que los valores de las medianas son coincidentes para ITLD y TDD con similar cantidad de datos dispersos en los cuatro cuadrantes. Una vez más se aprecia que existe un impacto de la tarea para la Productividad: La tarea BSK obtiene una mediana superior en comparación con MR.

Finalmente, el nivel de definición de la tarea (slicing) presenta medianas similares con la mayor concentración de datos por debajo del 50 %.

5.14.3. Prueba de hipótesis

5.14.3.1. Análisis estadístico

Siguiendo la misma estrategia de análisis estadístico realizado con el experimento base ESPE2015, se realiza una ANOVA respecto a las variables



(a) Estrategia de Programación (b) Tarea (c) Slicing

Figura 5.196: Box plots para la variable respuesta PROD

respuesta Calidad y Productividad. Como se muestran en la tabla 5.209, los resultados son significativos en la mayoría de casos:

- La estrategia de programación ha resultado significativa para la variable respuesta Productividad y se acerca al nivel de significación para la Calidad. No es necesario realizar un análisis post-hoc; la tabla 5.208 indica claramente que la Productividad obtenida con TDD es mayor que la obtenida con ITLD.
- Por otra parte, tal y como anticipábamos en la sección: 5.14.2, la tarea arroja resultados significativos tanto para Calidad como Productividad. De nuevo, la tarea BSK obtiene mejores valores que MR.
- Finalmente, el factor slicing también ha resultado significativo tanto para la Calidad como para la Productividad, aunque esto no se pudo visualizar claramente en los gráficos box plot analizados en la sección 5.14.2.

5.14.3.2. Chequeo del análisis estadístico

El chequeo del análisis estadístico sigue el mismo patrón que todos los experimentos en el ámbito académico. De las tres asunciones dadas para los modelos mixtos (linealidad entre factores, homogeneidad de varianzas y distribución normal de residuos) comprobaremos la segunda y tercera condición ya que la linealidad siempre se cumple (los factores tienen sólo dos niveles).

Homogeneidad de varianzas

No se aprecia la existencia de ninguna figura en embudo que sugiera heterogeneidad entre varianzas para la Productividad, como puede comprobarse en el gráfico de valores predichos vs. residuos estandarizados de la figura 5.197b. En el caso de la Calidad, la figura 5.197a sugiere un patrón

Tabla 5.209: Resultados del análisis estadístico

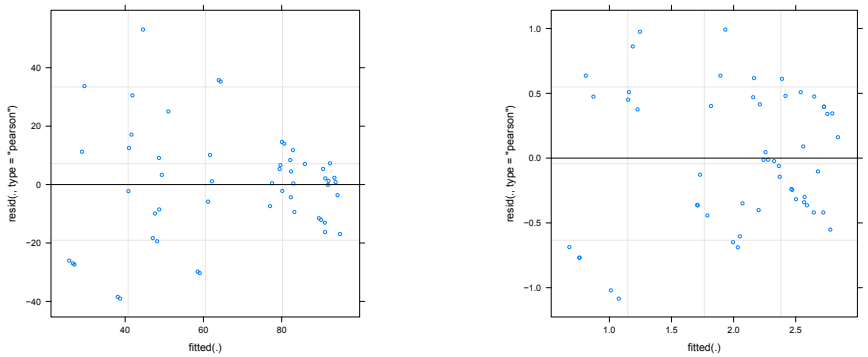
(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1718.01	1718.01	1.00	24.29	4.13	0.0532
TASK	23054.71	23054.71	1.00	24.29	55.43	0.0000
SLICING	1883.12	1883.12	1.00	24.29	4.53	0.0437
GROUP	998.90	998.90	1.00	25.29	2.40	0.1336

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	4.02	4.02	1.00	23.95	12.46	0.0017
TASK	15.51	15.51	1.00	23.95	48.03	0.0000
SLICING	1.40	1.40	1.00	23.95	4.34	0.0480
GROUP	2.00	2.00	1.00	24.98	6.20	0.0198

de embudo (la varianza decrece a medida que aumenta el tamaño de efecto). De todas formas, el patrón está lejos de ser concluyente.



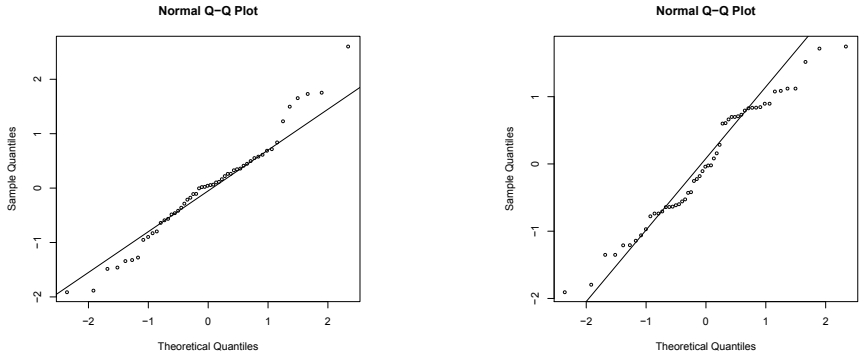
(a) Calidad (b) Productividad

Figura 5.197: Chequeo de la relación lineal entre el modelo y las variables respuesta Calidad y Productividad

Normalidad de residuos

Se estudian tanto de los factores fijos como de los factores aleatorios siguiendo la estrategia de análisis de todos los experimentos. En lo que respecta a los factores fijos, podemos apreciar en la figura 5.198, que la distribución de los residuos de la variable respuesta PROD se solapan razonablemente con la distribución normal (representada en la línea recta diagonal) con desviaciones en los extremos. Para el caso de la Productividad las desviaciones parecen más pronunciadas.

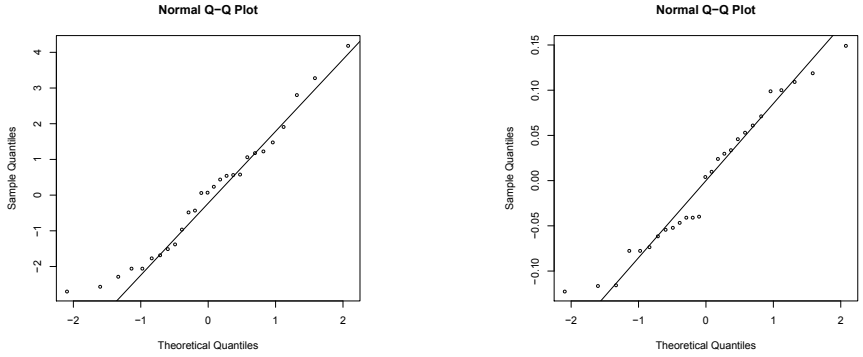
El test de Shapiro-Wilks indica que los residuos no se distribuyen normalmente para la Productividad, ($W = 0{,}97$, $p - value = 0{,}14$). Para el caso de la Calidad el test confirma la normalidad de residuos ($W = 0{,}98$, $p - value = 0{,}39$).



(a) Calidad (b) Productividad

Figura 5.198: Gráficos QQ para los factores fijos

En lo que respecta a los factores aleatorios, el test de Shapiro-Wilks una vez más confirma la normalidad de residuos para la Calidad ($W = 0{,}96$, $p - value = 0{,}37$) y no para la Productividad ($W = 0{,}95$, $p - value = 0{,}26$).



(a) Calidad (b) Productividad

Figura 5.199: Gráficos QQ para los factores aleatorios

```
> lmpt <- lmer(PROD^(1/4) ~  1 + STRATEGY +
+                              TASK +
+                              SLICING +
+                              GROUP +
```

```
+                                   (1 | subjectID),
+               data = all_expData)
> npt <- shapiro.test(resid(lmpt, scaled = TRUE))
> npat <- shapiro.test(ranef(lmpt)$subjectID$`(Intercept)`)
```

Debido a la falta de normalidad de residuos, aplicamos la transformación Box-Cox con $\lambda = \frac{1}{4}$, para la Productividad, con lo cual alcanzamos la normalidad para las dos variables respuesta (Calidad Efectos fijos: $W = 0{,}98$, $p-value = 0{,}39$, Efectos aleatorios: $W = 0{,}96$, $p-value = 0{,}37$; Productividad Efectos fijos: $W = 0{,}97$, $p-value = 0{,}14$, Efectos aleatorios: $W = 0{,}95$, $p - value = 0{,}26$). Los análisis realizados en la sección 5.14.3 ya incluyen esta transformación.

5.14.3.3. Influencia de factores instrumentales

En esta sección analizamos la dificultad de la tarea experimental percibida por los sujetos y su posible influencia debido al diseño experimental (como ha sido habitual en todos los experimentos).

Tarea experimental

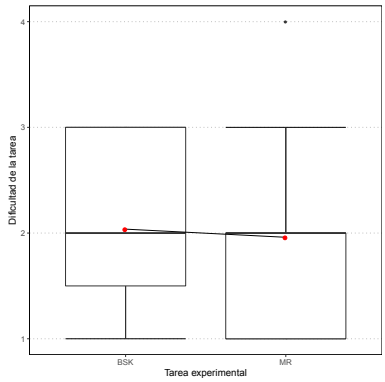
En la figura 5.200a, observamos que la complejidad percibida para MR es prácticamente similar a la complejidad de BSK (utilizando una escala de Likert de 1 a 5). El test Kruskal-Wallis entre la tarea experimental y la dificultad percibida arroja un p-valor = 0,62, lo cual confirma la impresión visual obtenida en la figura 5.200a.

Aunque el análisis estadítico arrojó resultados significativos para la tarea experimental (con mayores porecentajes de Calidad y Productividad para BSK sobre MR), sin embargo, de acuerdo a la percepción de los participantes la dificultad de la tarea no juega un papel relevante en la realización del experimento.

5.14.4. Análisis de subgrupos

Conforme a los datos demográficos de los participantes (véase sección 5.14.1.1) vamos a analizar los aspectos que contienen diferencias entre sujetos que merecen considerarse, estos son:

- Edad.
- Experiencia profesional
- Experiencia en programación.
- Experiencia en Java.
- Uso de herramientas de pruebas.



(a) Dificultad percibida de la tarea

Figura 5.200: Efecto de la dificultad de la tarea percibida por los sujetos

- Entrenamiento previo en desarrollo de pruebas unitarias.
- Función actual en la organización.

5.14.4.1. Edad

```
> lmq <- lmer(QLTY ~ 1 +
+                     STRATEGY * age +
+                     TASK +
+                     SLICING +
+                     GROUP +
+                     (1 | subjectID),
+             data = all_expData)
> lmp <- lmer(PROD^(1/4) ~  1 +
+                           STRATEGY * age +
+                           TASK  +
+                           SLICING +
+                           GROUP +
+                           (1 | subjectID),
+             data = all_expData)
```

El análisis realizado para determinar el impacto de la edad de los sujetos en la calidad del código y en la productividad de los programadores posee una influencia negativa (B = -2.4) para la Calidad y neutra ($B = 0$) para la Productividad. Los efectos no son significativos para la Calidad ($p-value =$ 0.12) pero si lo son para la Productividad $p-value = 0.02$). La introducción de la **edad** produce cambios en el nivel de significación de las estrategias de programación $ITLD/TDD$, tanto para la Calidad como la Productividad. Esto sugiere que los efectos observados para la estrategia de programación se deben en parte a la edad, y no únicamente a la estrategia de programación.

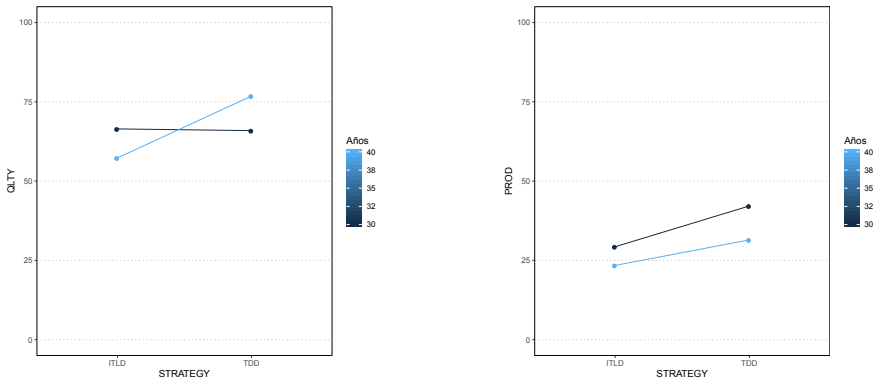
Tabla 5.210: Resultados del análisis estadístico para la Edad

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1074.92	1074.92	1.00	22.79	2.90	0.1024
age	965.46	965.46	1.00	23.95	2.60	0.1198
TASK	24091.19	24091.19	1.00	23.30	64.92	0.0000
SLICING	1733.03	1733.03	1.00	23.22	4.67	0.0412
GROUP	1172.10	1172.10	1.00	24.34	3.16	0.0880
STRATEGY:age	1442.48	1442.48	1.00	22.84	3.89	0.0609

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.24	0.24	1.00	22.55	0.78	0.3855
age	1.89	1.89	1.00	23.65	6.09	0.0212
TASK	16.07	16.07	1.00	23.07	51.78	0.0000
SLICING	1.34	1.34	1.00	22.99	4.30	0.0494
GROUP	2.73	2.73	1.00	24.04	8.79	0.0067
STRATEGY:age	0.54	0.54	1.00	22.61	1.75	0.1996

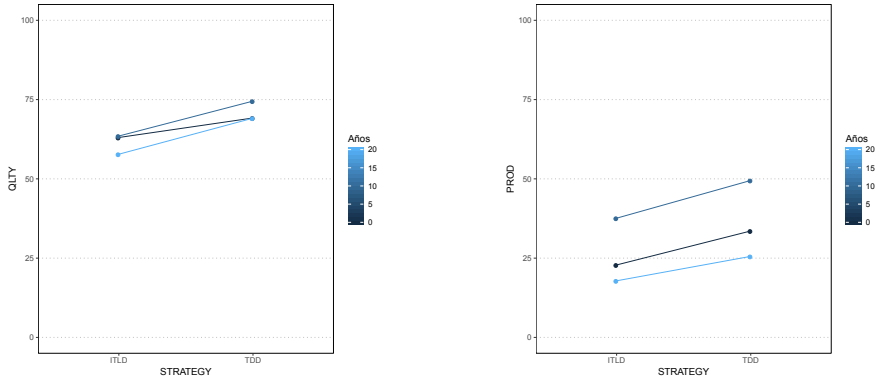


(a) Calidad (b) Productividad

Figura 5.201: Efecto de la Edad. La edad se reporta en décadas. Por ejemplo, el valor 20 representa el rango de 20-29 años de experiencia. El valor 30 representa el rango 30-39, etc.

La influencia de la edad en la estrategia de programación se muestra en la Fig. 5.201. Puede observarse que, a mayor edad, mayor es la Calidad y Productividad alcanzadas con TDD. Los participantes más jóvenes mantuvieron constante la Calidad tanto con $ITLD$ como con TDD, aunque mejoraron su Productividad al aplicar TDD.

5.14.4.2. Experiencia profesional



(a) Calidad (b) Productividad

Figura 5.202: Efecto de la experiencia profesional. La experiencia se reporta en décadas. Por ejemplo, el valor 0 representa 0-9 años de experiencia. El valor 10 representa 10-19 años de experiencia, etc.

En cuanto a la experiencia profesional, el análisis muestra una influencia negativa (B = -0.37) para la Calidad y neutra ($B = 0$) para la Productividad. Los efectos no son significativos ($p-value = 0.9$ y $p-value = 0.47$) en los dos casos, por esta razón se omiten las tablas de resultados. En este caso se observa en la figura 5.202, una marcada tendencia creciente a mejorar la Calidad y Productividad al aplicar TDD.

5.14.4.3. Experiencia en programación

La experiencia en programación muestra una influencia negativa para las variable calidad (B =-0.31) y neutra para la productividad (B =0). Los efectos no son significativos en ambos casos ($P-valor$ =0.54 y $P-valor$ =0.75). En lo que respecta a la aplicación de las estrategias de programación, en la figura 5.203 se observa una tendencia de los participantes con mayor experiencia en programación a mejorar la Calidad y Productividad cuando aplican TDD. Los participantes menos experimentados en programación disminuyeron ligeramente la Calidad con la estrategia TDD, aunque su Productividad se incrementó con esta estrategia.

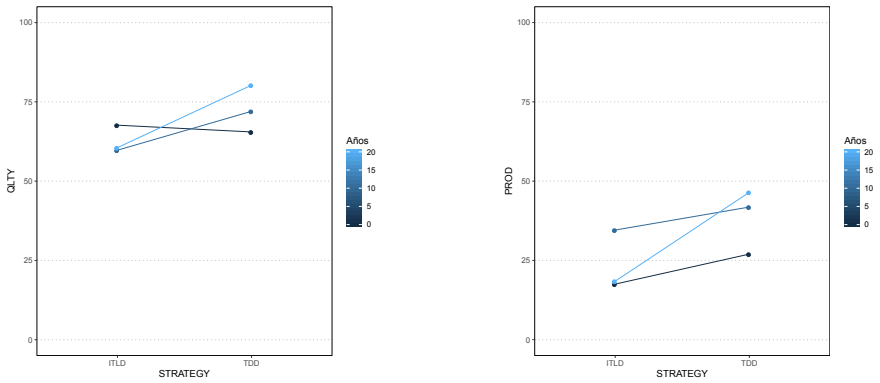
Tabla 5.211: Resultados del análisis estadístico para la experiencia en programación

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.61	0.61	1.00	22.88	0.00	0.9696
programmingExperience	155.32	155.32	1.00	23.79	0.38	0.5447
TASK	23253.66	23253.66	1.00	23.19	56.54	0.0000
SLICING	2018.68	2018.68	1.00	23.24	4.91	0.0368
GROUP	710.24	710.24	1.00	24.19	1.73	0.2011
STRATEGY:programmingExperience	523.47	523.47	1.00	22.72	1.27	0.2710

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.19	0.19	1.00	22.56	0.58	0.4556
programmingExperience	0.03	0.03	1.00	23.51	0.10	0.7546
TASK	15.62	15.62	1.00	22.87	48.42	0.0000
SLICING	1.50	1.50	1.00	22.92	4.64	0.0419
GROUP	1.67	1.67	1.00	23.91	5.18	0.0321
STRATEGY:programmingExperience	0.33	0.33	1.00	22.40	1.03	0.3207



(a) Calidad (b) Productividad

Figura 5.203: Efecto de la experiencia en programación. La experiencia se reporta en décadas. Por ejemplo, el valor 0 representa 0-9 años de experiencia. El valor 10 representa 10-19 años de experiencia, etc.

5.14.4.4. Experiencia en programación Java

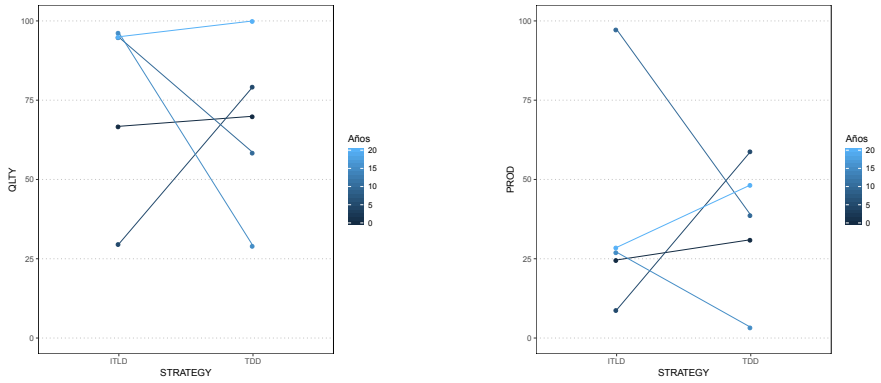
Tabla 5.212: Resultados del análisis estadístico para la Experiencia en Java

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	823.23	823.23	1.00	23.21	1.91	0.1802
javaExperience	252.04	252.04	1.00	23.79	0.58	0.4521
TASK	22586.73	22586.73	1.00	23.19	52.39	0.0000
SLICING	1904.96	1904.96	1.00	23.26	4.42	0.0466
GROUP	771.27	771.27	1.00	24.18	1.79	0.1935
STRATEGY:javaExperience	71.57	71.57	1.00	22.75	0.17	0.6875

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	2.09	2.09	1.00	22.90	6.27	0.0199
javaExperience	0.04	0.04	1.00	23.53	0.13	0.7238
TASK	15.34	15.34	1.00	22.88	46.03	0.0000
SLICING	1.42	1.42	1.00	22.95	4.26	0.0505
GROUP	1.78	1.78	1.00	23.92	5.34	0.0298
STRATEGY:javaExperience	0.10	0.10	1.00	22.45	0.29	0.5977



(a) Calidad (b) Productividad

Figura 5.204: Efecto de la experiencia en programación Java. La experiencia se reporta en lustros (5 años). Por ejemplo, el valor 0 representa 0-4 años de experiencia. El valor 5 representa 5-9 años de experiencia, etc.

La experiencia en el lenguaje de programación Java presenta una influencia positiva para la Calidad y nuetra para la Productividad (B =0.27 B =0). Los efectos no alcanzan el nivel de significación ($P - value$ =0.45 y $P - value$ =0.72).

En este caso la figura 5.204b muestra que algunos grupos de sujetos, principalmente los que más experiencia tienen en lenguaje Java, mejoraron

la Calidad y Productividad al aplicar TDD. Aunque también existen casos donde la Calidad y Productividad disminuyó drásticamente al aplicar TDD.

5.14.4.5. Uso de herramientas de pruebas

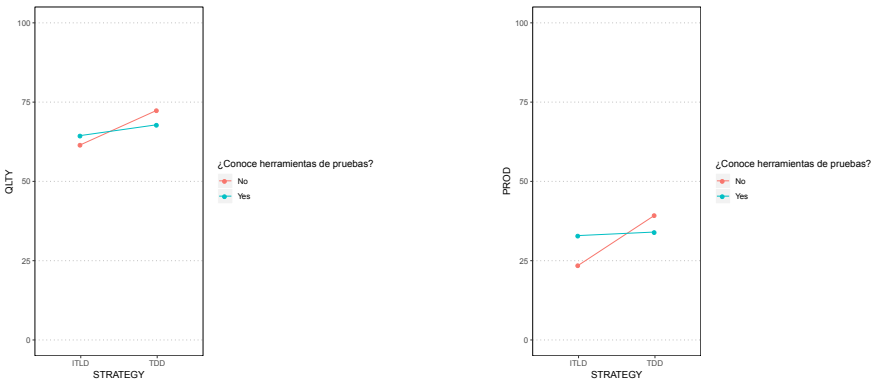
Tabla 5.213: Resultados del análisis estadístico para el uso de herramientas de pruebas

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1321.19	1321.19	1.00	23.61	3.09	0.0916
testingToolUsage	56.94	56.94	1.00	24.55	0.13	0.7182
TASK	23098.62	23098.62	1.00	23.24	54.07	0.0000
SLICING	1899.98	1899.98	1.00	23.24	4.45	0.0459
GROUP	966.90	966.90	1.00	24.24	2.26	0.1454
STRATEGY:testingToolUsage	156.45	156.45	1.00	23.46	0.37	0.5509

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3.44	3.44	1.00	23.35	10.24	0.0039
testingToolUsage	0.00	0.00	1.00	24.32	0.00	0.9604
TASK	15.51	15.51	1.00	22.98	46.20	0.0000
SLICING	1.40	1.40	1.00	22.98	4.18	0.0525
GROUP	1.98	1.98	1.00	24.01	5.90	0.0230
STRATEGY:testingToolUsage	0.06	0.06	1.00	23.20	0.17	0.6853



(a) Calidad (b) Productividad

Figura 5.205: Efecto del uso de herramientas de pruebas.

El uso de herramientas de pruebas presenta una influencia positiva (B =1.27) para la Calidad y neutra para la Productividad (B =0). Se puede notar también que los efectos una vez más no son significativos ($P-value$ =0.72 y $P-value$ =0.96).

Loa participantes que indicaron no conocer de herramientas de pruebas, mejoraron ligeramente la Calidad y Productividad al aplicar TDD, como se observa en la figura 5.205. Aquellos participantes que si conocían el uso de herramientas de prueba, prácticamente mantuvieron la misma Calidad y Productividad con las dos estrategias.

5.14.4.6. Función actual en la organización

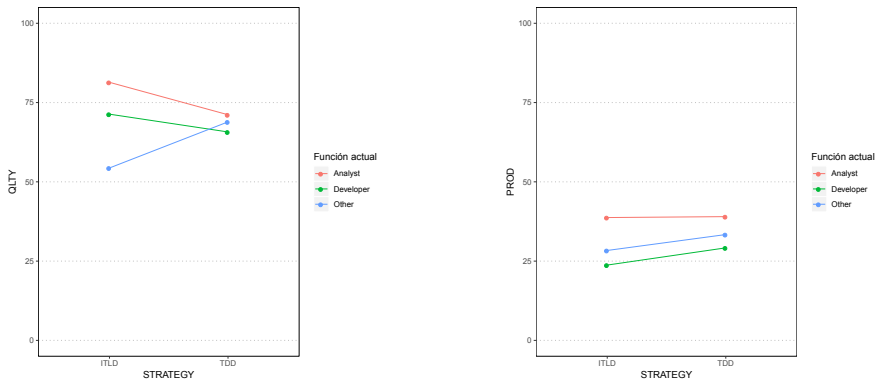
Tabla 5.214: Resultados del análisis estadístico para la Función Actual en la Organización

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	389.41	389.41	1.00	19.01	1.40	0.2506
currentFunction	479.20	239.60	2.00	19.70	0.86	0.4368
TASK	18453.48	18453.48	1.00	18.46	66.55	0.0000
SLICING	1939.21	1939.21	1.00	18.46	6.99	0.0162
GROUP	146.02	146.02	1.00	19.51	0.53	0.4766
STRATEGY:currentFunction	2345.72	1172.86	2.00	18.61	4.23	0.0306

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1.81	1.81	1.00	17.83	9.11	0.0074
currentFunction	0.38	0.19	2.00	18.78	0.96	0.4011
TASK	11.64	11.64	1.00	17.34	58.58	0.0000
SLICING	1.60	1.60	1.00	17.34	8.04	0.0113
GROUP	0.65	0.65	1.00	18.60	3.29	0.0860
STRATEGY:currentFunction	1.25	0.62	2.00	17.47	3.14	0.0682



(a) Calidad (b) Productividad

Figura 5.206: Efecto de la función actual en la organización

La función que se encontraban los sujetos desempeñando en la organi-

zación, presenta una influencia positiva para la Calidad y Productividad (B =8.64, B =0.01). Los efectos no son significativos en ambos casos ($p-value = 0.44$ y $p-value = 0.4$). La interacción *STRATEGY*currentFuntion*, resulta significativa para la Calidad y se aproxima al nivel de significación para la Productividad.

La figura 5.206 indica que los analistas y desarrolladores disminuyeron ligeramente la Calidad al aplicar TDD, aunque su productividad prácticamente es la misma con ambas estrategias. Los participantes con otras funciones, mejoraron ligeramente tanto en Calidad como en Productividad al aplicar TDD.

5.14.4.7. Entrenamiento previo en desarrollo de pruebas unitarias

Tabla 5.215: Resultados del análisis estadístico para el conocimiento de Pruebas Unitarias

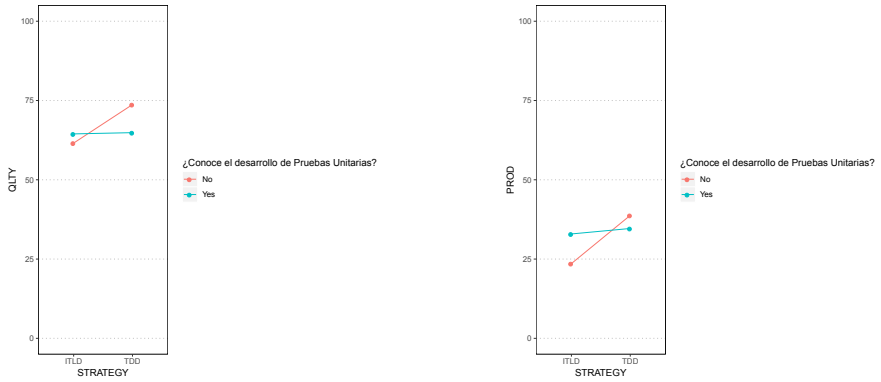
(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1215.54	1215.54	1.00	23.04	2.85	0.1050
UTTraining	68.75	68.75	1.00	23.97	0.16	0.6917
TASK	22814.60	22814.60	1.00	23.24	53.46	0.0000
GROUP	1019.50	1019.50	1.00	24.25	2.39	0.1351
SLICING	1823.79	1823.79	1.00	23.24	4.27	0.0500
STRATEGY:UTTraining	173.29	173.29	1.00	22.89	0.41	0.5303

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3.46	3.46	1.00	22.70	10.30	0.0039
UTTraining	0.01	0.01	1.00	23.67	0.03	0.8689
TASK	15.46	15.46	1.00	22.90	45.97	0.0000
SLICING	1.39	1.39	1.00	22.90	4.14	0.0536
GROUP	1.99	1.99	1.00	23.94	5.92	0.0229
STRATEGY:UTTraining	0.01	0.01	1.00	22.56	0.04	0.8354

El entrenamiento previo en desarrollo de pruebas unitarias presenta una influencia positiva para la Calidad(B =1.25) y neutra para la Productividad (B =0). Los efectos no son significativos en ambos casos (B =0.69 y B =0.87). Como se aprecia en la figura 5.207, los participantes que no tenían entrenamiento previo en desarrollo de pruebas unitarias, mejoraron ligeramente la Calidad y Productividad con la estrategia TDD. Aquellos participantes que indicaron tener entrenamiento previo en desarrollo de pruebas unitarias, no mejoraron en Calidad y Productividad al aplicar tanto $ITLD$ como TDD.



(a) Calidad (b) Productividad

Figura 5.207: Efecto del entrenamiento previo en desarrollo de Pruebas Unitarias.

5.14.4.8. Resumen general

Tabla 5.216: Resumen de la influencia de las variables demográficas estudiadas (Calidad)

	Calidad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Edad	-2.4	0.12	2.53	0.06
Experiencia profesional	-0.37	0.9	0.92	0.52
Experiencia en Programación	-0.31	0.54	1.5	0.27
Experiencia en Programación Java	0.27	0.45	0.54	0.69
Uso de herramientas de pruebas	1.27	0.72	-7.18	0.55
Conocimiento de Pruebas Unitarias	1.25	0.69	-7.65	0.53
Función actual en la organización	8.64	0.44	**-4.6**	**0.03**

Las tablas 5.216 y 5.217 muestran el resumen de los análisis de subgrupos realizados.

5.14.5. Grado de completitud

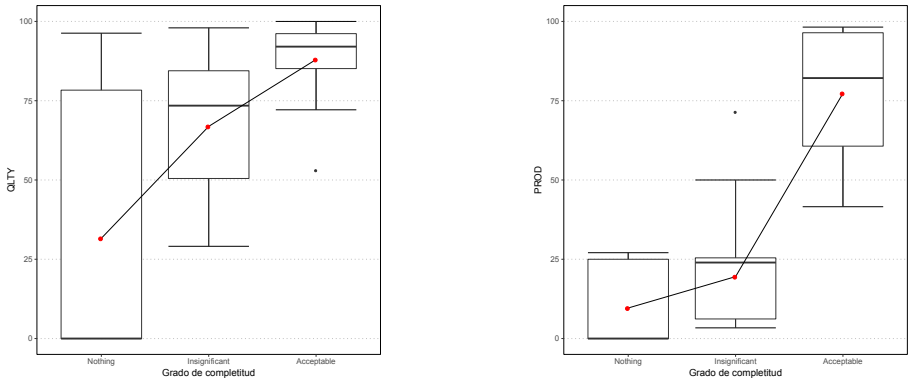
5.14.5.1. Aspectos generales

En la Tabla 5.218 se muestra el total de sujetos que no hicieron nada de la tarea experimental (*Nothing*), los que hicieron una mínima cantidad de trabajo (*Insignificant*) y los que realizaron el trabajo como se esperaba (*Acceptable*). Como en todos los experimentos, esta clasificación obedece a

Tabla 5.217: Resumen de la influencia de las variables demográficas estudiadas (Productividad)

	Productividad			
	Variable		STRATEGY * Variable	
	B	p-value	B	p-value
Edad	**0**	**0.02**	0	0.2
Experiencia profesional	0	0.47	0	0.44
Experiencia en Programación	0	0.75	0	0.32
Experiencia en Programación Java	0	0.72	0	0.6
Uso de herramientas de pruebas	0	0.96	0	0.69
Conocimiento de Pruebas Unitarias	0	0.87	0	0.84
Función actual en la organización	0.01	0.4	0.02	0.07

una revisión manual realizada al código entregado. Adicionalmente, en la figura 5.208 se presenta la relación entre el grado de completitud frente a las variables QLTY y PROD.



(a) Calidad (b) Productividad

Figura 5.208: Relación entre el grado de completitud y la calidad y productividad

De un total de 53 tareas que fueron entregadas, 8 sujetos no hicieron nada. Por otro lado, 32 participantes hicieron una mínima cantidad de trabajo y 13 sujetos realizaron el trabajo de la forma esperada.

En este caso un 15 % de sujetos entregaron las tareas vacías (*Nothing*), lo cual podría comprometer el análisis aunque en menor medida que en otros experimentos.

	Número de tareas entregadas
Nothing	8
Insignificant	32
Acceptable	13

Tabla 5.218: Grado de completitud

5.14.5.2. Impacto de la estrategia de programación y la tarea experimental

En la tabla 5.219, podemos ver que la totalidad de sujetos que no hicieron nada de las tareas fueron aquellos que utilizaron la técnica ITLD. Por otro lado, un 19 % del total de tareas entregadas fueron calificadas como *Acceptable* con la estrategia TDD, en contraste con el 6 % de tareas calificadas de igual forma al utilizar la estrategia ITLD. Al parecer la *Estrategia de programación* tendría influencia en el grado de completitud de las tareas.

	Nothing	Insignificant	Acceptable
ITLD	8	15	3
TDD	0	17	10

Tabla 5.219: Estrategia de programación

Para verificar la existencia de una posible relación entre la estrategia de programación y el grado de completitud de las tareas, realizamos un test χ^2 que arroja los siguientes resultados: $\chi^2 = 11{,}88$, $df = 2$, $p - value = 0$. Por lo tanto, se comprueba la existencia de una relación estadísticamente significativa entre estos factores.

	Nothing	Insignificant	Acceptable
BSK	3	16	8
MR	5	16	5

Tabla 5.220: Grado de completitud por tarea experimental (BSK, MR)

Por otra parte, la tabla 5.220 muestra el grado de completitud en función de la tarea. En este caso, la distribución es más o menos uniforme, por lo que no se esperaría la existencia de una relación estadíticamente significativa, como lo demuestra el test χ^2 ($\chi^2 = 1{,}17$, $df = 2$, $p - value = 0{,}56$).

5.14.5.3. Impacto de las variables demográficas

En las siguientes secciones analizaremos si el grado de completitud de las tarea se encuentra relacionado con las variables demográficas obtenidas

de los sujetos experimentales. Vamos a analizar las mismas variables de la sección 5.14.4, esto es:

- Edad.
- Experiencia profesional.
- Experiencia en programación.
- Experiencia en Java.
- Uso de herramientas de pruebas.
- Función actual en la organización.
- Entrenamiento previo en desarrollo de pruebas unitarias.

Edad

En la figura 5.209, podemos observar cierta tendencia a que los sujetos de mayor edad entreguen tareas incompletas. Sin embargo, el test Kruskal-Wallis entre la edad y el grado de completitud arroja un p-valor = 0,21, lo cual es insuficiente para confirmar esta tendencia.

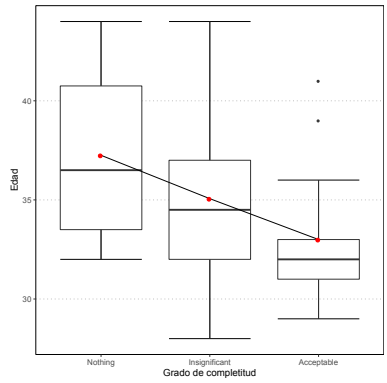


Figura 5.209: Relación entre la edad y el grado de completitud de las tareas

Experiencia profesional

Los años de experiencia al parecer no tienen influencia en el grado de completitud de las tareas, aunque la tendencia es que los sujetos que no hicieron nada fueron los que indicaron tener más años de experiencia (véase la figura 5.210). El test Kruskal-Wallis entre los años de experiencia y el grado de completitud arroja un p-valor = 0,65, el mismo que no es significativo y confirma la impresión visual de la figura 5.210.

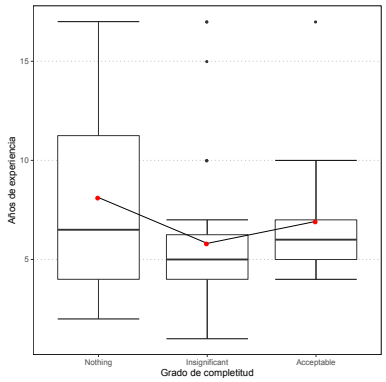


Figura 5.210: Relación entre los años de experiencia profesional y grado de completitud de las tareas

Experiencia en programación

La experiencia programación al parecer no tiene relación con el grado de completitud de las tareas, como se muestra en la Fig. 5.211. Como era de esperarse, los sujetos con más experiencia en programación son los que mejor hicieron las tareas, aunque el test Kruskal-Wallis entre la experiencia en programación y el grado de completitud arroja un p-valor = 0,47 no significativo.

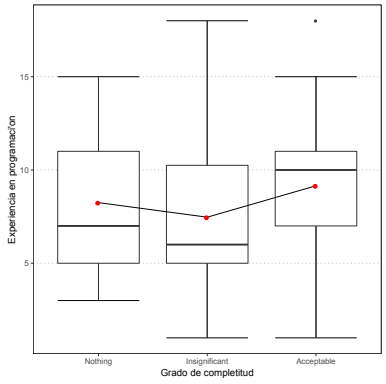


Figura 5.211: programmingExperience

Experiencia en programación Java

La experiencia en desarrollo con el lenguaje Java al igual que la experiencia en programación al parecer influye en el grado de presentación de las tareas (véase la figura 5.212). El test Kruskal-Wallis entre la experiencia en programación Java y el grado de completitud nos da un p-valor = 0,45 el cual no es significativo. Aunque los resultados del test no son significativos, sin embargo, podemos notar un mejor desempeño de los estudiantes

con mayor experiencia en Java, lo cual es un resultado previsible.

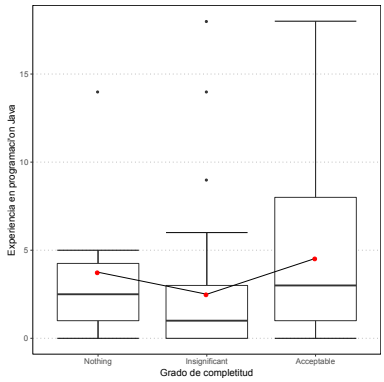


Figura 5.212: javaExperience

Uso de herramientas de prueba

	Nothing	Insignificant	Acceptable
No	6	20	8
Yes	2	12	5

Tabla 5.221: Conocimiento del entorno eclipse

El uso de herramientas de prueba no tiene influencia en el grado de completitud de las tareas experimentales. El test χ^2 nos confirma este hecho ($\chi^2 = 0{,}49$, $df = 2$, $p - value = 0{,}78$).

Función actual en la organización

	Nothing	Insignificant	Acceptable
Analyst	1	5	3
Developer	0	15	3
Manager	2	3	3
Other	5	9	4

Tabla 5.222: Funcion actual en la organización

La función que se encontraban desempeñando los sujetos en la organización tampoc tiene influencia en el grado de completitud de las tareas como podemos comprobar mediante el test χ^2 que arroja resultados no significativos ($\chi^2 = 9{,}2$, $df = 6$, $p - value = 0{,}16$).

Entrenamiento previo en desarrollo de pruebas unitarias

	Nothing	Insignificant	Acceptable
No	6	21	8
Yes	2	11	5

Tabla 5.223: Entrenamiento previo en pruebas unitarias

El entrenamiento previo en el desarrollo de pruebas unitarias tampoco presenta influencia en el grado de completitud de las tareas. El test χ^2 confirma este hecho ($\chi^2 = 0{,}41$, $df = 2$, $p - value = 0{,}82$). Por lo tanto no es significativo.

5.14.5.4. Resumen general impacto variables demográficas

La tabla 5.224 resume los resultados de los análisis de la influencia de las variables demográficas estudiadas frente al grado de completitud de las tareas realizadas. Como se observa, las variables no ejercen un impacto que sea estadísticamente significativo, aunque hemos observado tendencias como la influencia de la edad en el grado de terminación de las tareas: Los sujetos de mayor edad entregaron la mayor parte de tareas vacías o realizaron un trabajo insignificante.

Tabla 5.224: Resumen de la influencia de las variables demográficas estudiadas

	p-value
Edad	0.21
Experiencia profesional	0.65
Experiencia en programación	0.47
Experiencia en Java	0.45
Uso de herramientas de prueba	0.78
Función actual en la organización	0.16
Entrenamiento previo en desarrollo de pruebas unitarias	0.82

Capítulo 6

SÍNTESIS DE RESULTADOS

Una síntesis vale por diez análisis

Eugeni dÓrs

Resumen:

Los experimentos reportados en las secciones 5.8 - 5.14 han producido pocos resultados estadísticamente significativos, especialmente en lo tocante a las variables independientes que representan los aspectos personales. Esto era esperable, por tres razones:

- El número de participantes en cada experimento individual es reducido. En algún caso (ej: ESPE2015) hemos obtenido datos de casi cuarenta sujetos. Sin embargo, el número de sujetos es habitualmente más pequeño (ej: en BABEL2016 participaron únicamente diez programadores). En consecuencia, el poder estadístico de los experimentos es bastante reducido.
- Estamos explorando un buen numero de variables independientes (ej: STRATEGY, TASK, etc.) simultáneamente, lo cual reduce todavía más el poder estadístico.
- La motivación de los sujetos a la hora de participar en el experimento es mucho más influyente de lo que nos esperábamos. Algunos estudiantes que realizaron el experimento recibieron una calificación en función de su rendimiento (ej: UTN2017, ESPE2015, ESPE2016). Sin embargo, en otros casos tenían como única motivación la entrega de un certificado de participación en los talleres de formación (ej: UADY215, UNLP2015). A pesar de ello, prácticamente todos estos estudiantes entregaron las tareas con un grado de completitud entre aceptable e insignificante. Esto contrasta vivamente con los profesionales, los cuales también recibieron un certificado de participación, pero sin embargo entregaron muchas tareas sin llegar a realizar ningún código.

Es incluso posible que algunos resultados estadísticamente significativos sean espúreos. Debido al elevado número de test estadísticos realizados, existe la posibilidad de cometer un error tipo-I. El presente capítulo de síntesis persigue solventar los problemas anteriores o, si se prefiere, amenazas a la validez. Realizaremos una síntesis de tipo *Individual Patient Data* (IPD) con descomposición en subgrupos. Para la realización de este capítulo se ha utilizado [134], y los paquetes [143], [141], [138], [148], [149], [150], [140], [137], [151], [152], [153], [154], [155], [156], [157], [158].

6.1. Meta-análisis TDD vs. ITLD

La presente tesis tiene como objetivo identificar la influencia de los aspectos personales en la estrategia de programación TDD. Los aspectos personales considerados son:

- la edad,
- la experiencia profesional,
- la experiencia en programación,
- el nivel y grado de educación,
- el conocimiento previo en pruebas unitarias,
- el conocimiento del entorno Eclipse,
- la experiencia en java,
- la experiencia en el uso de herramientas de prueba, y finalmente,
- la experiencia en el framework Junit.

Para alcanzar este objetivo, hemos realizado siete experimentos comparando TDD con la estrategia de programación ITLD. La realización de dichos experimentos permite estudiar el efecto agregado de TDD vs. ITLD (además de los efectos agregados de los otros factores ensayados, esto es, TASK y SLICING) independientemente de la influencia de los aspectos personales.

Aunque comparar TDD vs. ITLD no es el objetivo de esta tesis, no deja de ser un resultado interesante para la misma. Los resultados de la comparación podrán contrastarse con los publicados en los estudios secundarios mencionados en la Introducción y Estado de la Cuestión, y de esta manera mejorar nuestro conocimientos acerca de la utilidad de TDD. La comparación TDD vs. ITLD es, asimismo, un buen punto de partida para la síntesis, ya que los meta-análisis de aspectos personales que se realizan en este trabajo se construyen sobre la base de la comparación TDD vs. ITLD.

El efecto agregado se calculará mediante meta-análisis IPD. Dentro de las principales ventajas de este tipo de meta-análisis está su mayor flexibilidad para agregar experimentos con múltiples variables independientes, que es precisamente lo que deseamos realizar (los factores personales son una variable independiente más). Asimismo, los procedimientos estadísticos son bien conocidos[145]. El análisis estadístico apropiado es el mostrado en el Listado 6.1.

```
lmer(QLTY ~ 1 +                # también PROD
             Provenance +
             STRATEGY +
             TASK +
             SLICING +
             GROUP +
             (1 | subjectID),
     data = ...)
```

Listing 6.1: Meta-análisis general

Tabla 6.1: Meta-análisis general (independiente de los aspectos personales)

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	25813.42	5162.68	5	148.05	6.63	0.000
STRATEGY	0.09	0.09	1	149.60	0.00	0.991
TASK	67904.00	67904.00	1	149.69	87.17	0.000
SLICING	401.32	401.32	1	185.67	0.52	0.474
GROUP	2020.30	2020.30	1	136.96	2.59	0.110

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	16009.27	3201.85	5	139.46	8.76	0.000
STRATEGY	4114.42	4114.42	1	128.18	11.26	0.001
TASK	72041.39	72041.39	1	128.49	197.18	0.000
SLICING	3247.19	3247.19	1	147.10	8.89	0.003
GROUP	0.26	0.26	1	129.05	0.00	0.979

El modelo es exactamente el mismo que hemos utilizado anteriormente en los experimentos, con la adición de un factor (Provenance) que representa al experimento del que provienen los datos y que se usará para determinar el grado de heterogeneidad del meta-análisis.

La Tabla 6.1 muestra el meta-análisis de seis (aclararemos inmediatamente por qué) de los siete experimentos realizados. En ambos análisis (Calidad y Productividad), la síntesis indica que la Tarea es significativa. Esto era esperable, ya que TASK ha resultado significativa en prácticamente todos los experimentos. A partir de este punto, los resultados difieren. Respecto a la Calidad, no se obtiene ningún resultado estadísticamente significativo.

Respecto a la Productividad, todos los factores excepto GROUP son significativos.

No obstante, este meta-análisis posee dos problemas importantes que impide, en nuestra opinión, considerar dichos resultados como aceptables:

1. El experimento QUITO2016 (ver sección 5.8) ha sido eliminado del meta-análisis, ya que en el mismo no pudo ensayarse el factor SLICING. Por este motivo, el meta-análisis mostrado en la Tabla 6.1 sólo contiene seis experimentos, en lugar de los siete realizados.
2. La variable Provenance es estadísticamente significativa. Esto implica que los experimentos arrojan resultados heterogéneos. La heterogeneidad de resultados aconseja una descomposición en subgrupos

Es conveniente tratar los problemas en secuencia. Primero debemos decidir si usamos sólo seis experimentos, o ignoramos el factor SLICING. Teniendo en cuenta que el número de sujetos experimentales es el mayor problema al que nos enfrentamos en IS empírica, prescindir de un experimento no parece una alternativa razonable.

En segundo lugar, y como veremos inmediatamente continuación, tendremos problemas de heterogeneidad a la hora de realizar los meta-análisis, lo que nos llevará a realizar una descomposición de los experimentos en los subgrupos profesionales/estudiantes. Esto nos permitirá estudiar el factor SLICING en los experimentos de estudiantes.

Finalmente, el meta-análisis, cuando incluya sujetos profesionales, no puede incluir el factor GROUP. Esto se debe a que el experimento QUITO2016 posee tres grupos que no son perfectamente compatibles con los restantes experimentos. Ahora bien, cuando analicemos estudiantes, el factor GROUP si podrá ser considerado, al igual que el factor SLICING.

En consecuencia, meta-analizaremos los experimentos sin los factores SLICING o GROUP, restringiendo el análisis de los mismos para el subgrupo de estudiantes. Dado que ambas variables no hacen referencia al objetivo principal de esta tesis, su exclusión parcial no tiene excesiva gravedad. Desarrollaremos esta idea en el capítulo de Amenazas a la Validez.

6.2. Meta-análisis TDD vs. ITLD sin los factores SLICING o GROUP

La Tabla 6.2 muestra el meta-análisis excluyendo los factores SLICING y GROUP. El modelo utilizado es una simplificacion del indicado en el Listado 6.1:

```
lmer(QLTY ~ 1 +              # también PROD
            Provenance +
            STRATEGY +
```

```
                TASK +
                (1 | subjectID),
        data = ...)
```

Listing 6.2: Meta-análisis, excluyendo los factores SLICING y GROUP

Tabla 6.2: Meta-análisis general (independiente de los aspectos personales) sin los factores SLICING o GROUP

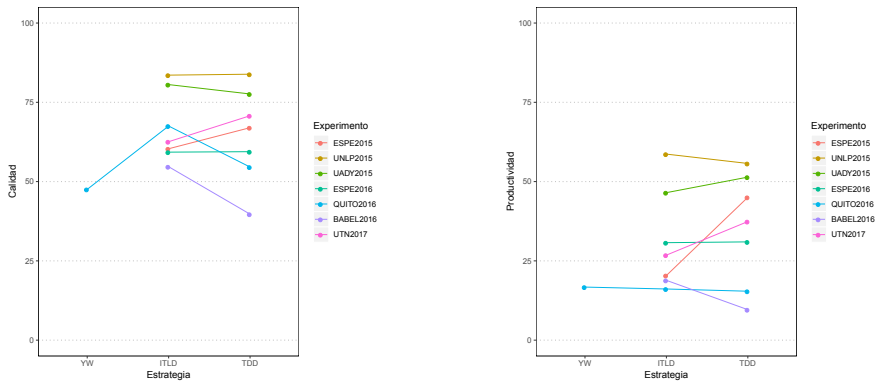
(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	25180.75	4196.79	6	185.00	5.51	0.000
STRATEGY	3136.92	1568.46	2	202.15	2.06	0.130
TASK	89148.85	44574.43	2	199.15	58.51	0.000

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	20780.80	3463.47	6	176.49	9.57	0.000
STRATEGY	3306.70	1653.35	2	180.77	4.57	0.012
TASK	74255.76	37127.88	2	177.48	102.55	0.000

Al excluir los factores SLICING y GROUP, es posible considerar el experimento QUITO2016, lo que a su vez implica que el meta-análisis incorpora los siete experimentos realizados y un total de 160 sujetos. Apenas existen diferencias entre este análisis y el mostrado en la Tabla 6.1: La Estrategia de Programación es significativa para la Productividad, y la Tarea tanto para la Calidad como la Productividad.



(a) Calidad (b) Productividad

Figura 6.1: Gráfico de perfil de todos los experimentos realizados.

El factor Provenance sigue siendo estadísticamente significativo. Ya hemos indicado que este resultado implica que los experimentos son hetero-

géneos. La figura 6.1 hace visible dicha heterogeneidad. El aspecto clave a observar es la posición de la línea que describe los resultados de cada experimento. Se puede apreciar fácilmente que las líneas no forman un haz, sino que están apiladas, con distancia entre ellas. El mayor o menor rendimiento medio (tanto en Calidad como en Productividad) es lo que produce la heterogeneidad de resultados.

La heterogeneidad de resultados es perjudicial para el análisis, ya que nos advierte de que estamos mercando experimentos con distintas características. No obstante, la heterogeneidad debe estudiarse y determinar su impacto en cada caso. En el presente meta-análisis tenemos interés en determinar el efecto de la estrategia de programación. La figura 6.1 muestra que existen experimentos donde ITLD se comporta mejor que TDD, y viceversa. Mezclar ambos experimentos producirá un efecto de nivelación que dará como resultado $ITLD \approx TDD$. En consecuencia, convendría identificar alguna variable entre-experimentos que, mediante descomposición de subgrupos, evitase este efecto pernicioso.

La variable entre-experimentos que con mayor rigurosidad puede dividir los experimentos en dos grupos es el tipo de sujeto participante en los experimentos. Esto es, nos planteamos dividir, en la siguiente sección, los experimentos en dos subgrupos: Estudiantes vs. Profesionales.

6.3. Descomposición en subgrupos del meta-análisis: Estudiantes vs. Profesionales

6.3.1. Profesionales

Tabla 6.3: Meta-análisis con sujetos profesionales (sin los factores SLICING o GROUP)

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	4484.78	2242.39	2	71.76	3.60	0.032
STRATEGY	2919.27	1459.63	2	66.62	2.35	0.104
TASK	50179.90	25089.95	2	66.44	40.33	0.000

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	3489.27	1744.63	2	74.71	5.28	0.007
STRATEGY	613.10	306.55	2	69.38	0.93	0.400
TASK	19137.69	9568.85	2	69.19	28.98	0.000

El meta-análisis para los experimentos con profesionales se muestra en la Tabla 6.3. El modelo de análisis es (como no puede ser de otra manera)

el indicado en el listado 6.2.

La Tabla 6.3 sólo arroja resultados significativos para el factor TASK, como ya hemos observado con anterioridad. El factor STRATEGY no posee ninguna influencia ni en la Calidad ni en la Productividad. Provenance sigue siendo estadísticamente significativo, lo que indica que tampoco hemos resuelto el problema de la heterogeneidad.

6.3.2. Estudiantes

Tabla 6.4: Meta-análisis con sujetos estudiantes

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	17181.83	5727.28	3	182.00	6.98	0.000
STRATEGY	122.89	122.89	1	182.00	0.15	0.699
TASK	39533.55	39533.55	1	182.00	48.19	0.000
SLICING	66.56	66.56	1	182.00	0.08	0.776
GROUP	348.96	348.96	1	182.00	0.43	0.515

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	9660.47	3220.16	3	105.16	9.12	0.000
STRATEGY	2730.44	2730.44	1	92.92	7.73	0.007
TASK	55748.28	55748.28	1	93.11	157.84	0.000
SLICING	1522.44	1522.44	1	110.42	4.31	0.040
GROUP	110.18	110.18	1	94.20	0.31	0.578

El meta-análisis para los experimentos con estudiantes se muestra en la tabla 6.4. El modelo de análisis es el indicado en el listado 6.1. Nótese que el meta-análisis de los sujetos estudiantes incluye los factores SLICING y GROUP, ya que los experimentos con estudiantes son replicaciones exactas.

Tal y como ha ocurrido a lo largo de la investigación, la Tarea resulta estadísticamente significativa tanto para la Calidad como la Productividad. Aparte de este resultado recurrente, sólo es relevante el análisis de la Productividad, donde los factores STRATEGY y SLICING resultan estadísticamente significativos.

Tal y como se indica en la tabla 6.5a, los estudiantes son más productivos cuando usan TDD. La diferencia $TDD - ITLD$ es de (7.82 puntos), lo que corresponde con un tamaño de efecto de 0.24 (pequeño).

Del mismo modo, la productividad cuando se usan especificaciones con Slicing es mayor que las especificaciones sin Slicing (6.12 puntos), lo que corresponde con un tamaño de efecto de 0.19 (muy pequeño).

En lo que respecta a SLICING, las especificaciones divididas en slices producen una mayor productividad que las especificaciones monolíticas, tal y

Tabla 6.5: Post-hoc análisis de los factores STRATEGY y SLICING para variable respuesta PROD (estudiantes)

(a) STRATEGY

	Estimate	Std. Error	z value	Pr(>|z|)
TDD - ITLD	7.82	2.81	2.78	0.005

(b) SLICING

	Estimate	Std. Error	z value	Pr(>|z|)
Slicing - NoSlicing	6.12	2.95	2.08	0.038

como muestra el análisis post-hoc de la Tabla 6.5b. La diferencia es pequeña (6.12 puntos), pero estadísticamente significativa. En términos de tamaño de efecto, la diferencia es de 0.19 (muy pequeño).

Finalmente, podemos observar en la tabla 6.4 que Provenance sigue arrojando un resultado estadísticamente significativo. En consecuencia, la heterogeneidad sigue presente a pesar de la descomposición en subgrupos realizada.

6.4. Descomposición en subgrupos utilizando la variable entre-experimentos *recruitment*

Profundizando un poco más en los gráficos de la figura 6.1, pueden realizarse observaciones bastante interesantes:

- Los experimentos que han obtenido mejores resultados son UNLP2015 y UADY2015. En ambos casos, se trata de experimentos académicos, pero no realizados en el marco de una asignatura, sino como seminarios de libre participación.
- Los experimentos en academia (ESPE2015, ESPE2016 y UTN2017) exhiben resultados intermedios. Aquí cabe realizar dos consideraciones adicionales:
  - En este grupo de experimentos, TDD se comporta mejor que ITLD. En todos los demás casos, TDD se comporta peor que ITLD.
  - UTN2017 es un experimento realizado con profesionales en un ámbito académico. Sin embargo, el comportamiento de los sujetos se asemeja más a un experimento académico que industrial.
- Los experimentos industriales *stricto sensu* (QUITO2016, BABEL2016) producen los peores resultados. En este caso, los experimentos fueron

realizados en el marco de cursos de formación que los asistentes podían abandonar en cualquier momento.

Distinguir entre experimentos realizados con estudiantes "voluntarios" y estudiantes "obligados" es muy discutible, pero los datos indican que esta agrupación (u otra que tenga distinto nombre pero englobe los mismos experimentos) puede tener sentido. Por ello, realizaremos una descomposición de los experimentos en tres subgrupos: estudiantes voluntarios (Volunteer), estudiantes en academia (Conscripted), y profesionales (Training course), y razonaremos si esta agrupación es más conveniente que Profesional vs. Estudiante.

Tabla 6.6: Descomposición de subgrupos en función del tipo de reclutamiento (Calidad)

(a) Voluntarios

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	137.82	137.82	1	37.59	0.54	0.466
STRATEGY	168.31	168.31	1	35.53	0.66	0.421
TASK	99.55	99.55	1	35.53	0.39	0.536

(b) Estudiantes

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	1070.41	535.21	2	99.05	0.79	0.457
STRATEGY	28.04	28.04	1	98.35	0.04	0.839
TASK	86216.30	86216.30	1	98.42	127.06	0.000

(c) Profesionales

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	2661.98	2661.98	1	37.64	4.20	0.048
STRATEGY	5027.17	2513.59	2	32.35	3.96	0.029
TASK	27480.11	13740.06	2	32.49	21.67	0.000

Las tablas 6.6 y 6.7 muestran los resultados de la descomposición de subgrupos usando la variable entre-experimentos *recruitment*. Hemos optado por usar el análisis descrito en el listado 6.2 para proporcionar una visión coherente de los resultados.

La primera cosa que salta a la vista es la homogeneidad de los resultados. Con la única excepción de la tabla 6.7c, Provenance arroja resultados no significativos. Asimismo, algunos resultados son bastante coherentes respecto a la sección anterior. Por ejemplo, los estudiantes siguen siendo más productivos cuando usan TDD (véase tabla 6.4b).

La diferencia más notable aparece en los profesionales. Dado que el experimento UTN2017 se parecía más a un experimento académico que a un experimento industrial, hemos eliminado UTN2017 del subgrupo de experi-

Tabla 6.7: Descomposición de subgrupos en función del tipo de reclutamiento (Productividad)

(a) Voluntarios

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	499.57	499.57	1	41.37	1.34	0.253
STRATEGY	244.10	244.10	1	39.21	0.66	0.423
TASK	15862.86	15862.86	1	39.21	42.65	0.000

(b) Estudiantes

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	46.09	23.05	2	90.53	0.06	0.943
STRATEGY	4709.84	4709.84	1	83.32	12.05	0.001
TASK	54776.87	54776.87	1	83.54	140.17	0.000

(c) Profesionales

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	104.42	104.42	1	38.93	1.95	0.170
STRATEGY	270.70	135.35	2	31.80	2.53	0.096
TASK	5291.19	2645.60	2	31.99	49.43	0.000

mentos profesionales y lo hemos añadido al subgrupo de experimentos con estudiantes. Esto no ha perjudicado al subgrupo de experimentos con estudiantes (es homogéneo) y a su vez ha mejorado la homogeneidad del subgrupo de experimentos con profesionales.

Esta nueva descomposición en subgrupos obtiene un resultado significativo para la Calidad. Sin embargo, este resultado no es nuevo; ya había sido reportado en la sección 5.8.3.2. Esto nos hace pensar que esta descomposición es demasiado granular, y que debemos reconsiderar algunas de las decisiones tomadas.

6.5. Modelo final de meta-análisis

Los resultados obtenidos son bastante convincentes, pero no *completamente convincentes*. El mayor problema es que la descomposición en subgrupos desaprovecha el tamaño muestral. Cada grupo contiene únicamente dos o tres experimentos, lo que a su vez disminuye el poder estadístico. Podría incluso afirmarse que la ausencia de heterogeneidad no es genuina, sino artifactural, esto es, derivada del bajo poder estadístico de los análisis. No creemos que esto sea cierto, pero es un aspecto que no puede simplemente ignorarse.

En nuestra opinión, creemos que la síntesis debería realizarse de la forma siguiente:

- Para evitar la disminución del número de sujetos experimentales por grupo (disminuyendo en proporción el poder estadístico), incorporaremos la nueva variable *recruitment* en el modelo mixto. Eliminaremos la variable Provenance, ya que asumimos la posible heterogeneidad de resultados gracias a toda la exploración previa realizada.

- Los experimentos académicos y profesionales exhiben distintos resultados respecto a la Estrategia de Programación. En los primeros, $TDD > ITLD$, mientras que en los segundos $ITLD > TDD$. En consecuencia, parece que el tipo de reclutamiento interacciona con la Estrategia de Programación. En consecuencia, introduciremos la interacción *STRATEGY * recruitment* en el modelo.

- El experimento UNLP2015 tiene un comportamiento respecto a STRATEGY similar al de los experimentos con profesionales. Del mismo modo, UADY2015 se parece mucho a los experimentos con estudiantes. Sin embargo, reasignar dichos experimentos parece altamente especulativo. Mantendremos dichos experimentos como realizados por voluntarios,

El modelo final será el indicado en el listado 6.3.

```
lmer(QLTY^(3/2) ~ 1 +
                  STRATEGY * recruitment +
                  TASK +
                  (1 | subjectID),
     data = ...)

lmer(PROD^(1/2) ~ 1 +
                  STRATEGY * recruitment +
                  TASK +
                  (1 | subjectID),
     data = ...)
```

Listing 6.3: Modelo de meta-análisis final

Nótese que al no considerar los factores SLICING ni GROUP, tal y como hemos hecho en las tablas 6.6 y 6.7, el análisis conjunto de todos los experimentos es posible.

Antes de proseguir con el análisis estadístico, es conveniente indicar el motivo de las transformaciones realizadas en la variable respuesta. Tal y como se ha hecho a lo largo del Capítulo 5, hemos chequeado la linealidad, normalidad y homogeneidad de varianzas del modelo de análisis. En el caso de los modelos no transformados del Listado 6.3 se producía un serio problema de normalidad. La transformación soluciona en gran medida el problema: El test Kolmogorov-Smirnov (tenemos más de 100 puntos de datos) confirma la normalidad de los factores fijos y, en el caso de los factores aleatorios,

donde existen desviaciones los valores de asimetría y curtosis son muy reducidos. Por otra parte, los modelos con y sin transformación no parecen poseer problemas de no-normalidad o heterocedasticidad. En consecuencia, usaremos dichas transformaciones a lo largo de este capítulo de síntesis.

Tabla 6.8: Modelo de meta-análisis final

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	110900.66	110900.66	1	173.58	1.46	0.229
recruitment	1892359.54	946179.77	2	173.77	12.45	0.000
TASK	7269197.68	3634598.84	2	200.88	47.81	0.000
STRATEGY:recruitment	144280.69	72140.34	2	166.54	0.95	0.389

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1.27	1.27	1	161.19	0.34	0.559
recruitment	206.85	103.43	2	165.41	27.98	0.000
TASK	834.22	417.11	2	185.52	112.85	0.000
STRATEGY:recruitment	37.47	18.74	2	153.65	5.07	0.007

Tabla 6.9: Análisis post-hoc del modelo de meta-análisis final (mostrado en la tabla 6.8). Se analiza la interacción *STRATEGY:recruitment* para la variable PROD.

	contrast	estimate	SE	df	t.ratio	p.value
recruitment = Training course						
1	ITLD - TDD	0.8651	0.6454	172.87	1.340	0.1819
recruitment = Conscripted						
2	ITLD - TDD	-1.2152	0.3110	139.44	-3.907	0.0001
recruitment = Volunteer						
3	ITLD - TDD	-0.1705	0.4195	133.20	-0.406	0.6851

Results are averaged over the levels of: TASK

$PROD^{\frac{1}{2}}$ scale

El resultado de este modelo final de meta-análisis se muestra en la tabla 6.8. El único efecto estadísticamente significativo (además de TASK y *recruitment*, como era esperable) ocurre en la interacción *STRATEGY:recruitment*.

La tabla 6.9 muestra unos resultados muy parecidos[1] a los ya mostrados en las tablas 6.3 y 6.4: Los estudiantes (Conscripted) aumentan su productividad con el uso de TDD. El efecto es de -0.75 puntos, lo que en términos de effect size corresponde a $d = -0{,}2$ (efecto pequeño).

Como dice el refrán, para este viaje (llegar a los mismos resultados) no hacían falta estas alforjas (hacer una agrupación de experimentos que se

[1] Esta tabla difiere de las usadas anteriormente debido a la existencia de una interacción que nos obliga a utilizar un nuevo comando `lsmeans` no usado con anterioridad.

parece tanto a la dicotomía Profesional vs. Estudiante). Sin embargo, nos sentimos cómodos con las decisiones tomadas, y creemos que algo hemos aprendido acerca de las motivaciones de los sujetos experimentales por el camino.

Procedemos en este momento al análisis del impacto de las características personales usando el modelo con la variable *recruitment*. No obstante, podemos ya anticipar que en la sección **??** compararemos este modelo de meta-análisis con el propuesto en la sección 6.3. De esta manera, podremos comprobar que las decisiones que hemos tomado no influyen, al menos excesivamente, en los resultados obtenidos.

6.6. Meta-análisis de la influencia de los factores personales

En esta sección vamos a determinar la influencia de los factores personales en las variables respuesta Calidad y Productividad al aplicar TDD o ITLD en base a los datos obtenidos de los experimentos realizados.

Para realizar el análisis, es necesario decidir los efectos a estudiar. Está claro que el punto de partida es el modelo mostrado en el listado 6.3. Los factores personales pueden introducirse en este modelo de tres formas:

1. Mediante una interacción con *STRATEGY*, pero no con *recruitment*
2. Mediante interacciones con *STRATEGY* y *recruitment* por separado.
3. Mediante una interacción triple (*STRATEGY * recruitment * factor personal*).

En nuestro caso, usaremos la opción 2). La opción 1) es un tanto simple, y la opción 3) se diferencia muy poco de la opción 2), ya que la interacción *STRATEGY * recruitment* ya está presente. Sin embargo, la opción 3) contiene una interacción triple de difícil interpretación. Por todo ello, el modelo de análisis será de la forma siguiente:

```
lmer(QLTY^(3/2) ~ 1 +
                  STRATEGY * recruitment +
                  STRATEGY * factor personal
                     +
                  recruitment * factor
                     personal +
                  TASK +
                  (1 | subjectID),
     data = ...)

lmer(PROD^(1/2) ~ 1 +
                  STRATEGY * recruitment +
```

```
                    STRATEGY * factor personal
                       +
                    recruitment * factor
                       personal +
                    TASK +
                    (1 | subjectID),
          data = ...)
```

Listing 6.4: Meta-análisis incluyendo los factores personales

6.6.1. Meta-análisis incluyendo el entrenamiento previo en pruebas unitarias (UTTTraining)

Tabla 6.10: Meta-análisis incluyendo el entrenamiento previo en pruebas unitarias (UTTTraining)

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	95151.27	95151.27	1	155.66	1.25	0.265
recruitment	1128887.79	564443.89	2	156.66	7.43	0.001
UTTraining	87775.75	87775.75	1	159.10	1.16	0.284
TASK	7196514.26	3598257.13	2	196.99	47.36	0.000
STRATEGY:recruitment	147596.97	73798.48	2	165.98	0.97	0.381
STRATEGY:UTTraining	5774.76	5774.76	1	150.33	0.08	0.783
recruitment:UTTraining	185371.78	92685.89	2	156.62	1.22	0.298

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.16	0.16	1	143.24	0.04	0.835
recruitment	91.85	45.92	2	147.34	12.39	0.000
UTTraining	2.55	2.55	1	149.11	0.69	0.408
TASK	827.92	413.96	2	181.57	111.67	0.000
STRATEGY:recruitment	37.23	18.62	2	153.27	5.02	0.008
STRATEGY:UTTraining	0.42	0.42	1	137.65	0.11	0.738
recruitment:UTTraining	4.65	2.32	2	147.26	0.63	0.536

Los análisis mostrados en la tabla 6.10 no señalan ningún resultado estadísticamente significativo que no conozcamos ya. Por un lado, están los efectos ya conocidos para las varianles *TASK* y *recruitment*. Por otro lado, existe una interacción con $p - valor = 0{,}008$ para la Productividad, pero se trata del mismo efecto observado anteriormente en las tablas 6.4 y 6.8b.

La variable *UTTraining* no muestra ningún efecto estadísticamente significativo.

Tabla 6.11: Meta-análisis incluyendo el conocimiento del entorno de desarrollo Eclipse (knowEclipse)

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	108124.44	108124.44	1	167.61	1.39	0.240
recruitment	1977115.21	988557.60	2	165.39	12.70	0.000
knowEclipse	60062.90	60062.90	1	163.95	0.77	0.381
TASK	7138465.17	3569232.59	2	196.13	45.84	0.000
STRATEGY:recruitment	150433.81	75216.91	2	160.80	0.97	0.383
STRATEGY:knowEclipse	11.67	11.67	1	152.93	0.00	0.990
recruitment:knowEclipse	147510.31	73755.15	2	155.87	0.95	0.390

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1.75	1.75	1	158.91	0.48	0.489
recruitment	199.51	99.76	2	161.48	27.32	0.000
knowEclipse	0.41	0.41	1	157.80	0.11	0.737
TASK	845.38	422.69	2	183.52	115.77	0.000
STRATEGY:recruitment	35.99	18.00	2	151.69	4.93	0.008
STRATEGY:knowEclipse	11.44	11.44	1	143.34	3.13	0.079
recruitment:knowEclipse	3.17	1.58	2	152.00	0.43	0.649

Tabla 6.12: Análisis post-hoc de la interacción *STRATEGY*knowEclipse* para la variable PROD.

contrast	estimate	SE	df	t.ratio	p.value
knowEclipse = No					
1 ITLD - TDD	-0.6178	0.3731	145.80	-1.656	0.0999
knowEclipse = Yes					
2 ITLD - TDD	0.2126	0.3514	151.77	0.605	0.5461

Results are averaged over the levels of: recruitment, TASK

$PROD^{\frac{1}{2}}$ scale

6.6.2. Meta-análisis incluyendo el conocimiento del entorno de desarrollo Eclipse (knowEclipse)

Los análisis mostrados en la tabla 6.11 reproducen los resultados estadísticamente significativo habituales, pero también sugieren que la variable *knowEclipse* interacciona con la Estrategia de Programación para la Productividad. Aunque el p-valor es mayor que 0.05, merece la pena echar un vistazo. La tabla 6.12 muestra que los sujetos sin conocimiento de Eclipse son más productivos usando TDD. El efecto es de -0.38 puntos, lo que en términos de effect size corresponde a $d = 0{,}27$ (efecto pequeño).

6.6.3. Meta-análisis incluyendo la experiencia en programación (programmingExperience)

Tabla 6.13: Meta-análisis incluyendo la experiencia en programación (programmingExperience)

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	72486.22	72486.22	1	170.20	0.95	0.332
recruitment	645049.99	322525.00	2	177.81	4.21	0.016
programmingExperience	48019.38	48019.38	1	189.56	0.63	0.430
TASK	7291767.59	3645883.79	2	198.18	47.58	0.000
STRATEGY:recruitment	152966.47	76483.24	2	164.62	1.00	0.371
STRATEGY:programmingExperience	3196.11	3196.11	1	162.45	0.04	0.838
recruitment:programmingExperience	908.80	454.40	2	184.11	0.01	0.994

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.02	0.02	1	156.82	0.01	0.937
recruitment	82.20	41.10	2	167.63	11.04	0.000
programmingExperience	1.42	1.42	1	171.42	0.38	0.538
TASK	835.63	417.82	2	183.23	112.22	0.000
STRATEGY:recruitment	37.17	18.59	2	151.71	4.99	0.008
STRATEGY:programmingExperience	0.38	0.38	1	148.22	0.10	0.750
recruitment:programmingExperience	1.78	0.89	2	170.89	0.24	0.788

La tabla 6.13 muestra que la variable *programmingExperience* no posee ningún efecto estadísticamente significativo. Los efectos de la experiencia en programación no son diferentes a los antes mencionados.

6.6.4. Meta-análisis incluyendo la edad de los sujetos participantes (age)

La edad (al igual que la experiencia profesional, que aparece inmediatamente más abajo) no puede estudiarse de la forma en que hemos hecho

hasta ahora. La razón es que no hemos registrado la edad de los sujetos estudiantes, sino sólo la de los profesionales. Los sujetos estudiantes pertenecen a los mismos cursos, y por lo tanto tienen edades muy parecidas. El análisis de dichas edades no resultaría útil para detectar efectos en la Calidad o Productividad.

En el Apéndice A se muestra el meta-análisis de los factores personales descomponiendo los datos en profesionales vs. estudiantes. Como se puede observar en la Sección A.1.4, la edad ejerce un efecto negativo y estadísticamente significativo en la Calidad y en la Productividad.

Rogamos se consulte dicha sección. Para evitar redundancias, no repetiremos el texto aquí.

6.6.5. Meta-análisis incluyendo los años de experiencia profesional (experienceYears)

Sólo hemos registrado la experiencia profesional de los sujetos profesionales. En este caso, probablemente hemos pecado de ser excesivamente conservadores, ya que los sujetos estudiantes pueden poseer experiencias muy variadas. No obstante, una vez tomada la decisión, no nos ha sido posible deshacerla ya que no podemos contactar de nuevo con los sujetos estudiantes.

En consecuencia, al igual que en el caso de la edad, la experiencia profesional sólo puede estudiarse para sujetos profesionales. El análisis aparece en la Sección A.1.5 del Apéndice A. La experiencia profesional no resulta estadísticamente significativa ni para la Calidad ni para la Productividad.

Rogamos se consulte dicha sección. Para evitar redundancias, no repetiremos el texto aquí.

6.6.6. Meta-análisis incluyendo el grado de estudios de los participantes (Seniority)

Los datos de la variable *Seniority* provienen de un único experimento (UADY2015) donde fue posible comparar grupos de sujetos de distintos grados de estudios. Por esta razón no se incluye esta variable en el meta-análisis, ya que no aporta nada nuevo a lo antes observado.

6.6.7. Nivel de educación (educationLevel)

La tabla 6.14 muestra que la variable *educationLevel* tampoco posee efectos estadísticamente significativos para las variables Calidad y Productividad.

Tabla 6.14: Meta-análisis incluyendo el nivel de educación (educationLevel)

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	148465.16	148465.16	1	201.53	1.94	0.165
recruitment	1182957.13	591478.57	2	154.26	7.72	0.001
educationLevel	151460.73	50486.91	3	166.50	0.66	0.578
TASK	7235119.90	3617559.95	2	198.06	47.22	0.000
STRATEGY:recruitment	11928.57	5964.28	2	155.08	0.08	0.925
STRATEGY:educationLevel	189786.43	63262.14	3	169.76	0.83	0.481
recruitment:educationLevel	613.18	613.18	1	147.70	0.01	0.929

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.48	0.48	1	188.11	0.13	0.721
recruitment	116.47	58.23	2	145.45	15.43	0.000
educationLevel	5.76	1.92	3	158.07	0.51	0.677
TASK	826.39	413.19	2	186.10	109.46	0.000
STRATEGY:recruitment	16.00	8.00	2	143.72	2.12	0.124
STRATEGY:educationLevel	1.68	0.56	3	156.87	0.15	0.930
recruitment:educationLevel	0.02	0.02	1	140.21	0.00	0.945

6.6.8. Uso de herramientas de prueba (testingToolUsage)

Tabla 6.15: Meta-análisis incluyendo el uso de herramientas de prueba (testingToolUsage)

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	143439.58	143439.58	1	174.52	1.89	0.171
recruitment	1551322.96	775661.48	2	184.33	10.22	0.000
testingToolUsage	367265.65	367265.65	1	211.74	4.84	0.029
TASK	7365448.16	3682724.08	2	199.40	48.54	0.000
STRATEGY:recruitment	204013.32	102006.66	2	166.04	1.34	0.264
STRATEGY:testingToolUsage	5224.14	5224.14	1	158.06	0.07	0.793
recruitment:testingToolUsage	509245.81	254622.91	2	179.35	3.36	0.037

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.01	0.01	1	158.74	0.00	0.967
recruitment	137.70	68.85	2	174.56	18.59	0.000
testingToolUsage	15.22	15.22	1	197.13	4.11	0.044
TASK	842.63	421.31	2	182.04	113.75	0.000
STRATEGY:recruitment	44.16	22.08	2	150.66	5.96	0.003
STRATEGY:testingToolUsage	0.87	0.87	1	142.70	0.24	0.628
recruitment:testingToolUsage	18.11	9.06	2	169.10	2.45	0.090

El uso de herramientas de prueba *testingToolUsage* resulta significativo

tanto para la Calidad como para la Productividad, como se observa en la tabla 6.15 con un tamaño de efecto de 50.8 y 9.03 respectivamente. Además de la interacción *recruitment*testingToolUsage* resulta también significativa para la Calidad y se aproxima al nivel de significación para la Productividad.

Tabla 6.16: Análisis post-hoc de *testingToolUsage* para la variable QLTY

	contrast	estimate	SE	df	t.ratio	p.value
1	No - Yes	-132.7222	60.6943	200.10	-2.187	0.0299

Results are averaged over the levels of: STRATEGY, recruitment, TASK
$QLTY^{\frac{3}{2}}$ scale

Tabla 6.17: Análisis post-hoc de *testingToolUsage* para la variable PROD.

	contrast	estimate	SE	df	t.ratio	p.value
1	No - Yes	-1.0262	0.5093	192.91	-2.015	0.0453

Results are averaged over the levels of: STRATEGY, recruitment, TASK
$PROD^{\frac{1}{2}}$ scale

El análisis post-hoc de *testingToolUsage* muestra que independientemente de la técnica utilizada, la variable ejerce influencia por ella misma tanto para la Calidad como para la Productividad,como se aprecia en las tablas 6.16 y 6.17

Tabla 6.18: Análisis post-hoc de la interacción *recruitment*testingToolUsage* para la variable QLTY

	contrast	estimate	SE	df	t.ratio	p.value
testingToolUsage = No						
1	Training course - Conscripted	-113.3133	61.2189	171.11	-1.851	0.1564
2	Training course - Volunteer	-248.2204	64.0860	170.82	-3.873	0.0004
3	Conscripted - Volunteer	-134.9071	49.4066	131.63	-2.731	0.0195
testingToolUsage = Yes						
4	Training course - Conscripted	296.8404	163.4887	228.68	1.816	0.1667
5	Training course - Volunteer	57.7670	170.2253	221.28	0.339	0.9385
6	Conscripted - Volunteer	-239.0735	66.7031	136.66	-3.584	0.0014

Results are averaged over the levels of: STRATEGY, TASK
$QLTY^{\frac{3}{2}}$ scale
P value adjustment: tukey method for comparing a family of 3 estimates

Por otra parte, el análisis post-hoc de la interacción *recruitment*testingToolUsage* que se muestra en la tabla 6.18 indica que cuando no se han utilizado herramientas de prueba, los *volunter* (profesionales) tienen un mejor rendimiento que los *conscripted* (estudiantes), así como los *volunter* con *training course*. Cuando se han usado herramientas de prueba, los voluntarios sólo superan a los *conscripted*, en cuanto a la Calidad.

Tabla 6.19: Análisis post-hoc de la interacción *recruitment*testingToolUsage* para la variable PROD.

	contrast	estimate	SE	df	t.ratio	p.value
testingToolUsage = No						
1	Training course - Conscripted	-1.5086	0.5228	168.47	-2.886	0.0122
2	Training course - Volunteer	-3.4293	0.5479	166.54	-6.259	<.0001
3	Conscripted - Volunteer	-1.9207	0.4337	134.89	-4.429	0.0001
testingToolUsage = Yes						
4	Training course - Conscripted	1.5483	1.3469	221.06	1.150	0.4847
5	Training course - Volunteer	-0.9159	1.4097	212.94	-0.650	0.7927
6	Conscripted - Volunteer	-2.4642	0.5834	138.63	-4.224	0.0001

Results are averaged over the levels of: STRATEGY, TASK

$PROD^{\frac{1}{2}}$ scale

P value adjustment: tukey method for comparing a family of 3 estimates

En lo que respecta a la Productividad, los participantes *volunter* que han utilizado herramientas de prueba tienen mejor rendimiento que los *conscripted*. Cuando no han utilizado herramientas de prueba, además de que los *conscripted* superan a los *volunter*, también mejoraron el rendimiento los *volunter* y *conscripted* con *training course* (Ver tabla 6.19).

6.6.9. Experiencia en el uso del Framework JUnit (jUnitExperience)

La tabla 6.20 muestra que la variable *jUnitExperience* no posee efectos estadísticamente significativos diferentes a los conocidos, aunque la interacción *STRATEGY*recruitment* ha resultado estadísticamente significativa para la productividad.

Tabla 6.20: Meta-análisis incluyendo la experiencia en JUnit (jUnitExperience)

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	148431.23	148431.23	1	167.55	1.95	0.164
recruitment	1474739.95	737369.97	2	166.34	9.69	0.000
jUnitExperience	19834.79	19834.79	1	266.55	0.26	0.610
TASK	6766743.51	3383371.75	2	194.42	44.44	0.000
STRATEGY:recruitment	129761.16	64880.58	2	163.67	0.85	0.428
STRATEGY:jUnitExperience	76458.69	76458.69	1	157.01	1.00	0.318
recruitment:jUnitExperience	225672.87	112836.44	2	196.69	1.48	0.230

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.68	0.68	1	156.98	0.18	0.669
recruitment	179.46	89.73	2	158.65	24.40	0.000
jUnitExperience	8.48	8.48	1	255.69	2.31	0.130
TASK	793.75	396.87	2	175.89	107.93	0.000
STRATEGY:recruitment	36.52	18.26	2	152.65	4.97	0.008
STRATEGY:jUnitExperience	1.46	1.46	1	145.07	0.40	0.529
recruitment:jUnitExperience	5.80	2.90	2	188.91	0.79	0.456

6.6.10. Experiencia en programación Java (javaExperience)

Tabla 6.21: Meta-análisis incluyendo la experiencia en Java (javaExperience)

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	77518.00	77518.00	1	164.84	1.03	0.313
recruitment	911528.15	455764.07	2	169.84	6.03	0.003
javaExperience	379369.76	379369.76	1	183.35	5.02	0.026
TASK	7208744.76	3604372.38	2	198.52	47.69	0.000
STRATEGY:recruitment	163592.89	81796.44	2	167.38	1.08	0.341
STRATEGY:javaExperience	666.80	666.80	1	152.82	0.01	0.925
recruitment:javaExperience	350709.08	175354.54	2	177.71	2.32	0.101

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	0.23	0.23	1	150.19	0.06	0.801
recruitment	67.66	33.83	2	158.05	9.16	0.000
javaExperience	3.87	3.87	1	168.22	1.05	0.307
TASK	836.79	418.40	2	181.14	113.30	0.000
STRATEGY:recruitment	35.53	17.77	2	152.68	4.81	0.009
STRATEGY:javaExperience	3.30	3.30	1	137.98	0.89	0.346
recruitment:javaExperience	2.50	1.25	2	164.43	0.34	0.714

La tabla 6.21 muestra que la variable *javaExperience* es significativa

para la Calidad. Además, la interacción *STRATEGY*recruitment* resulta también significativa para la Productividad.

El tamaño de efecto del modelo es de 16.34, que representa cuánto aumenta la Calidad por cada año que se aumenta la experiencia en java.

6.6.11. Función en la organización

Tabla 6.22: Meta-análisis incluyendo la función actual en la organización (currentFunction)

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	44328.33	44328.33	1	143.09	0.58	0.446
recruitment	1526292.32	763146.16	2	162.31	10.06	0.000
currentFunction	100936.47	20187.29	5	155.07	0.27	0.931
TASK	6751644.99	3375822.50	2	186.27	44.52	0.000
STRATEGY:recruitment	3244.46	1622.23	2	152.66	0.02	0.979
STRATEGY:currentFunction	252366.40	50473.28	5	151.02	0.67	0.650
recruitment:currentFunction	72247.08	72247.08	1	185.03	0.95	0.330

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1.31	1.31	1	129.91	0.36	0.550
recruitment	141.34	70.67	2	153.08	19.34	0.000
currentFunction	5.36	1.07	5	145.01	0.29	0.916
TASK	772.04	386.02	2	170.30	105.64	0.000
STRATEGY:recruitment	13.21	6.61	2	140.06	1.81	0.168
STRATEGY:currentFunction	7.77	1.55	5	137.64	0.43	0.830
recruitment:currentFunction	2.32	2.32	1	167.23	0.63	0.427

La función que se encontraban desempeñando los participantes en la organización, no produce resultados significativos tanto para la Calidad como para la Productividad, como se indica en la tabla 6.22.

6.6.12. Resumen de los resultados obtenidos

En la tabla 6.23, se muestra un resumen de los factores que han resultado estadísticamente significativos, o que se aproximan al nivel de significación.

Tabla 6.23: Resumen Meta-análisis de la influencia de factores personales que han resultado estadísticamente significativos

	p-valores	
FACTOR	QLTY	PROD
STRATEGY:knowEclipse	0.99	**0.079**
testingToolUsage	**0.029**	**0.044**
javaExperience	**0.026**	0.307

6.7. Influencia de los factores instrumentales

Tabla 6.24: Meta-análisis de la dificultad percibida en la tarea experimental (taskDifficulty)

(a) Calidad

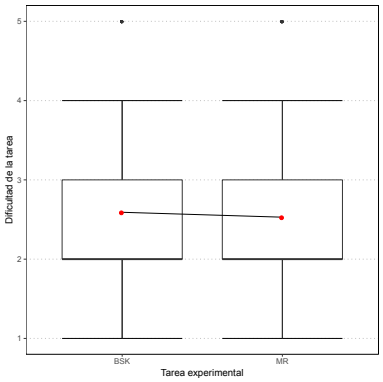
	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	100204.85	100204.85	1	210.00	1.30	0.255
recruitment	538933.96	269466.98	2	210.00	3.51	0.032
taskDifficulty	2093.41	2093.41	1	210.00	0.03	0.869
TASK	5491011.93	5491011.93	1	210.00	71.47	0.000
STRATEGY:recruitment	85896.15	42948.07	2	210.00	0.56	0.573
STRATEGY:taskDifficulty	97486.17	97486.17	1	210.00	1.27	0.261
recruitment:taskDifficulty	117687.85	58843.93	2	210.00	0.77	0.466

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	4.31	4.31	1	191.50	1.30	0.256
recruitment	26.93	13.47	2	207.48	4.05	0.019
taskDifficulty	5.62	5.62	1	209.98	1.69	0.195
TASK	507.26	507.26	1	121.96	152.67	0.000
STRATEGY:recruitment	3.89	1.95	2	141.87	0.59	0.558
STRATEGY:taskDifficulty	2.04	2.04	1	191.10	0.61	0.434
recruitment:taskDifficulty	1.21	0.61	2	203.17	0.18	0.833

En cuanto a la influencia de los factores instrumentales, la dificultad percibida de la tarea experimental no muestra resultados que sean estadísticamente significativos como se puede ver en la tabla 6.24. La figura 6.2a nos corrobora estos resultados. De acuerdo a los participantes, tanto MR como BSK tienen similar grado de dificultad. Hemos excluido la tarea SS ya que esta se aplicó en un único experimento (Quito2016), por lo tanto no aporta nada nuevo en nuestro meta-análisis. El hecho de que los participantes indiquen que la dificultad de la tarea experimental es similar, no necesariamente está en la misma dirección de los resultados obtenidos. Hemos visto a lo largo de todas las replicaciones, que por lo general los participantes obtienen

mejor Calidad y Productividad con BSK en comparación con MR, siendo esta diferencia estadísticamente significativa.



(a) Dificultad percibida de la tarea

Figura 6.2: Efecto de la dificultad de la tarea percibida por los sujetos

6.8. Meta-análisis del Grado de completitud

En esta sección analizaremos si existe alguna influencia entre el grado de completitud de la tarea realizada por los participantes y los factores personales que hemos considerado en este estudio.

Hemos definido el grado de completitud como la cantidad de código original entregado por los sujetos en las diferentes tareas experimentales. Los niveles que hemos definido son los siguientes:

- *Nothing*: Cuando el código entregado era prácticamente el mismo código base que fue proporcionado a los participantes.
- *Insignificant*: Se usó cuando los cambios realizados al código de base se limitaban sólo a unas pocas líneas.
- *Acceptable*: Cuando el participante realizó de manera significativa el código de la tarea.

Con el propósito de que el meta-análisis sea más sencillo de interpretar, utilizamos diferentes aproximaciones como la dicotomización de la variable grado de completitud, el uso del meta-análisis de datos agregados, y su representación mediante gráficos tipo *forest plot*. Para el caso de las variables continuas, también utilizamos gráficos de dispersión. Finalmente, descartamos en el meta-análisis la inclusión de la variable `SLICING`, ya que ha sido poco relevante durante este capítulo de síntesis.

6.8.1. ITLD vs. TDD

En primer lugar, analizaremos la relación entre la estrategia de programación (ITLD o TDD) y el grado de completitud. En este análisis, no tendremos en consideración ni `SLICING` ni `TASK`, ya que el presente meta-análisis es de carácter exploratorio, no confirmatorio. Esto nos permite explorar de un modo más sencillo (y visible) las relaciones entre variables, aunque su carácter causal quedará en entredicho.

El procedimiento que vamos a utilizar para la síntesis es el meta-análisis de datos agregados. Dado que la variable independiente `degreeOfCompletion` es ordinal, procedemos a dicotomizarla como se indica en [?]. Existen otras opciones de meta-análisis tales como los *proportional odds models* y las tablas de contingencia. Evitamos los primeros por su difícil interpretación, y los segundos porque no nos permiten considerar los experimentos por separado ni determinar la homogeneidad de los mismos.

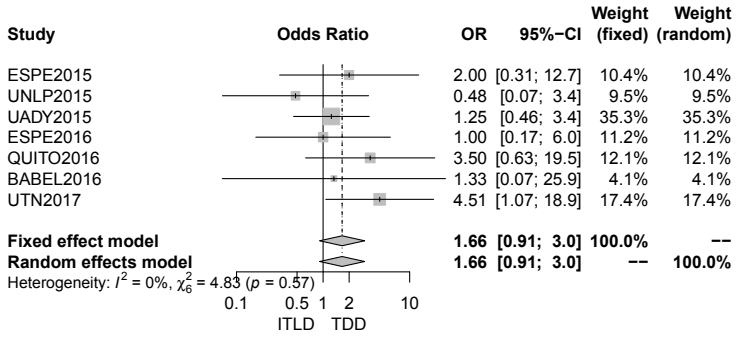


Figura 6.3: Meta-análisis ITLD vs. TDD vs. respecto al grado de completitud

El *forest plot* ITLD vs. TDD respecto al grado de completitud se muestra en la Fig. 6.3. Los pasos para obtener este diagrama han sido los siguientes:

1. Dicotomizar la variable `degreeOfCompletion` en dos categorías: *NothingOrInsignificant* y *Acceptable*. La primera categoría significa que el trabajo realizado por el sujeto experimental no ha sido satisfactorio, y la segunda lo contrario.
2. Calcular odds ratios.

3. Aplicar el método genérico de ponderación por inversa de varianza para el cálculo del odds ratio conjunto.

4. Aplicar el procedimiento de DerSimonian and Laird para la determinación del grado de heterogeneidad.

La Fig. 6.3 proporciona tanto el meta-análisis de efectos fijos como aleatorios. En este caso, ambos coinciden porque los estudios son homogéneos ($\chi^2 = 4.83 < 6$, p-value $= 0.57$).

La interpretación del *odds ratio* puede llegar a ser complicada, por lo que emplearemos explicaciones estándar. En la Fig. 6.3, lo que pretendemos estudiar es qué estrategia (TDD o ITLD) produce grados de completitud aceptables. Los valores mayores que 1 indican que los sujetos experimentales que TDD produce más valores *Acceptable* que ITLD (o, lo que es lo mismo, que ITLD produce más valores *NothingOrInsignificant* que TDD). El resultado es estadisticamente significativo cuando el intervalo de confianza (el diamante sombreado inferior) no toca el 1.

En este caso, se observa claramente que los sujetos experimentales trabajan más cuando aplican la estrategia TDD. El resultado no es estadísticamente significativo, pero la distribución de estudios en el gráfico sugiere que el origen de la no-significación reside en el bajo poder estadístico de los estudios individuales. El tamaño de efecto es OR = 1.66, lo que corresponde con un efecto pequeño[?].

6.8.2. MR vs. BSK

Estudiamos la influencia de las tareas Mars Rover (MR) y Bowling Score Keeper (BSK) con respecto al grado de completitud. No podemos hacer lo mismo con `SLICING`, ya que lo hemos ensayado poco.

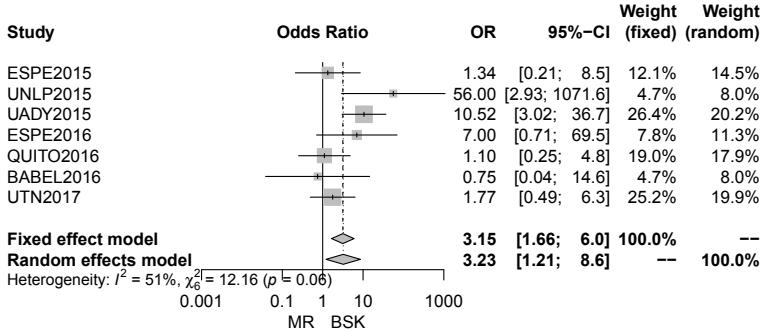


Figura 6.4: Meta-análisis influencia de la tarea

El meta-análisis de datos agregados de la influencia de las tareas respecto al grado de completitud, se presenta en la figura 6.4. En este caso, el meta-análisis indica que los grupos son homogéneos ($\chi^2 = 12.16 < 6$, p-value = 0.06), con un valor en el OR = 3.15 en el modelo de efectos fijos, lo que corresponde con un tamaño de efecto medio.

Como era previsible, la tarea experimental influye en el comportamiento de los sujetos cuando realizan los experimentos. Los resultados obtenidos en los experimentos individuales han mostrado que la tarea, aunque sea un factor instrumental, tiene un efecto significativo en las variables respuesta. En la gráfica 6.4, se observa que los sujetos realizan más código cuando desarrollan la tarea BSK en la mayor parte de experimentos, con un efecto global que es estadísticamente significativo.

6.8.3. Entrenamiento en unit testing

La figura 6.5 muestra los resultados del meta-análisis de efectos fijos y aleatorios del grado de completitud de la tarea, en función del entrenamiento previo que tenían los sujetos en el desarrollo de pruebas unitarias.

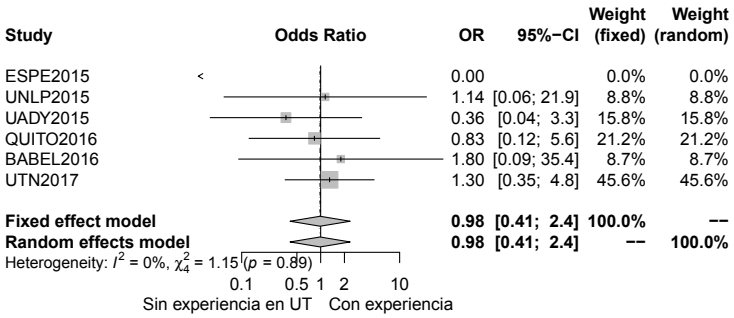


Figura 6.5: Meta-análisis del entrenamiento en pruebas unitarias, en función del grado de completitud.

El meta-análisis de efectos fijos y aleatorios son homogéneos ($\chi^2 = 1.15 < 4$, p-value = 0.89). El entrenamiento en previo en pruebas unitarias no tiene influencia en el grado de completitud de la tarea, como puede observarse en la figura 6.5. El tamaño de efecto es OR = 0.98, lo que corresponde con un efecto muy pequeño.

Cabe indicar que los experiementos donde no existan datos referentes a los factores analizados, no aparecerán en el `forest-plot`. En el experimento ESPE2016, por ejemplo, ninguno de los participantes indicó tener entrenamiento previo en el desarrollo de pruebas unitarias, por lo que este experimento no se muestra en la figura 6.5.

6.8.4. Conocimiento del entorno de desarrollo Eclipse

El meta-análisis del conocimiento previo de los participantes en desarrollo con el IDE Eclipse, frente al grado de completitud de las tareas, se muestra en el gráfico 6.6.

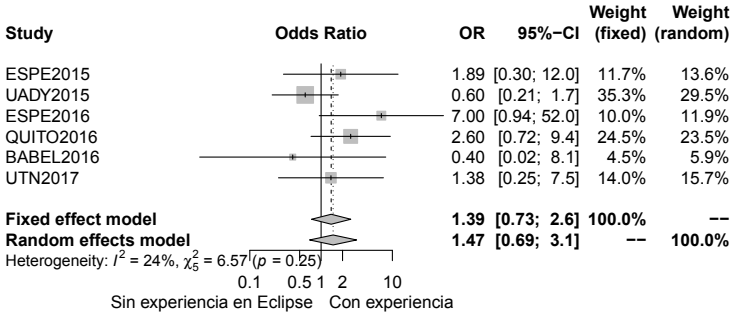


Figura 6.6: Meta-análisis del conocimiento del entorno Eclipse, en función del grado de completitud.

El meta-análisis indica que los grupos son homogéneos ($\chi^2 = 6.57 < 5$, p-value $= 0.25$) con un tamaño de efecto OR $= 1.39$, el cual puede interpretarse como muy pequeño. En este caso, el conocimiento previo en la herramienta Eclipse no afecta significativamente en el grado de completitud de las tareas.

6.8.5. Edad

La edad es una variable de tipo ratio. Realizar un meta-análisis de una variable independiente ratio, y una variable dependiente ordinal (o dicotómica) es inusual. No somos conscientes de ningún procedimiento habitual que permita hacer dicho meta-análisis.

Para evitar utilizar procedimientos complejos, vamos a invertir el papel de las variables. Usaremos `Age` como variable dependiente, y `DegreeOf-Completion` como variable independiente. De este modo, podemos usar un meta-análisis de tamaños de efectos ordinarios para calcular un efecto global. Dado que los meta-análisis realizados en esta sección son de carácter exploratorio, la inversión del papel de las variables no debería suponer amenaza a la validez alguna.

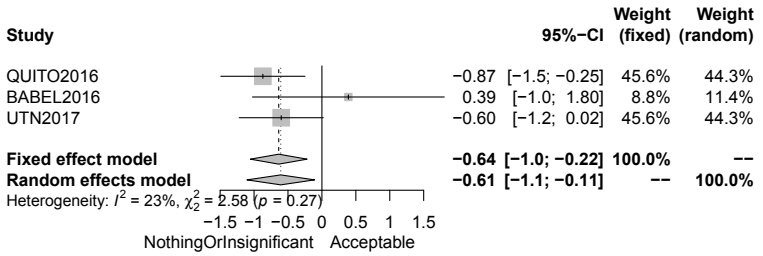


Figura 6.7: Meta-análisis de la edad, en función del grado de completitud.

El meta-análisis de la edad, en función del grado de completitud, se muestra en la Fig. 6.7. El meta-análisis de efectos fijos y aleatorios son homogéneos ($\chi^2 = 2.58 < 2$, p-value = 0.27).

La interpretación del *forest plot* es el siguiente:

1. Si el effect size es negativo, los sujetos que han entregado código con un grado de completitud `NothingOrInsignaificant` son mayores (esto es, tienen más edad) que los sujetos que han entregado código con un grado de completitud `Acceptable`.

2. Si el effect size es positivo, la interpretación es exactamente la inversa.

Como podemos comprobar en la Fig. 6.7, el tamaño de efecto es negativo, con valor d =-0.64, lo que corresponde con un tamaño de efecto medio [?]. El efecto en este caso es estadísticamente significativo ($p - value <= 0.0029$). En otras palabras: los datos muestran que los sujetos que entregan tareas poco o nada trabajadas son de mayor edad que aquellos que realizan las tareas de forma aceptable.

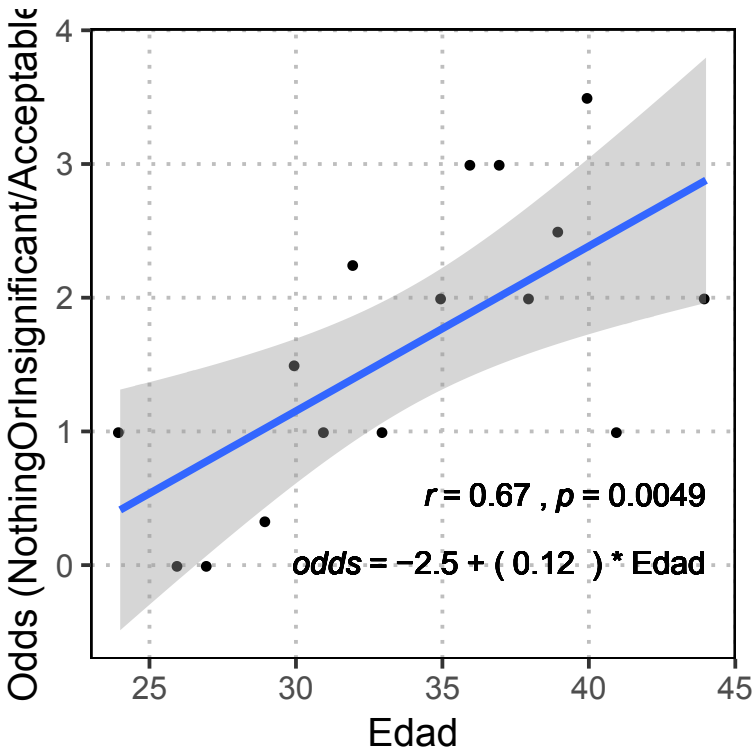


Figura 6.8: Scatter de la influencia de la edad en el grado de completitud. Se ve claramente que el odds ratio (NothingOrInsignificant/Acceptable) aumenta con la edad.

Aunque la Fig. 6.7 indica claramente la influencia de la edad en el grado de completitud, el cambio en el papel de las variables (independientes por dependientes, y viceversa) no deja de provocar cierta confusión. Por lo tanto, habiendo ya establecido el carácter estadísticamente significativo de la relación, mostramos un gráfico más entendible en la Fig. 6.8. Se trata de un gráfico de dispersión donde en el eje X se representa la edad, y en el eje Y los *odds* que corresponden a dicha edad.

La Fig. 6.7 muestra claramente que los *odds*, esto es, la probabilidad de que la tarea haya sido realizada de forma *NothingOrInsignificant* aumenta con la edad. De hecho, la correlación es casi estadísticamente significativa, lo que es notable con tan pocos puntos de datos.

6.8.6. Experiencia en Programación

Para analizar la experiencia en programación, en función al grado de completitud, aplicamos el mismo procedimiento que con la edad, esto es, usaremos `programmingExperience` como variable dependiente, y `DegreeOfCompletion` como variable independiente.

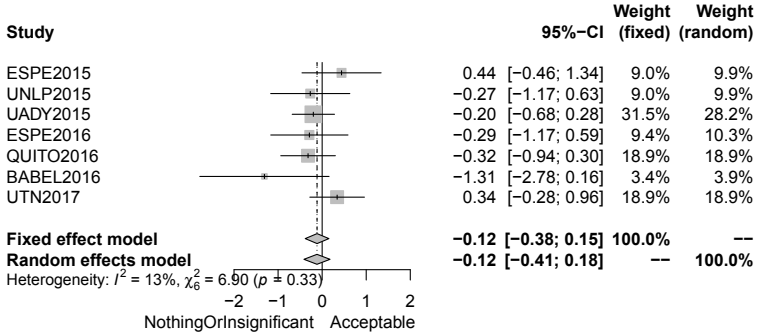


Figura 6.9: Meta-análisis de la experiencia en programación, en función del grado de completitud.

En la Fig. 6.9 se muestra el meta-análisis de la experiencia en programación en función del grado de completitud. El meta-análisis de efectos fijos y aleatorios son homogéneos ($\chi^2 = 6.9 < 6$, p-value = 0.33).

Como podemos observar, el efecto es positivo, con valor d =-0.12, lo que corresponde con un tamaño de efecto muy pequeño. En este caso, el efecto no resulta estadísticamente significativo ($p - value <= 0.401$).

6.8.7. Nivel de educación

El nivel de educación de los participantes, es una variable ordinal (between-subjects) pero no hay diversidad dentro de los experimentos. La forma en que hemos realizado el análisis es mediante una tabla de contingencia, con un análisis post-hoc posterior.

Dimension	NothingOrInsignificant	Acceptable
Undergraduate	0.15	0.15
Bachelor	1.00	1.00
Master	0.56	0.56
Other	1.00	1.00

Tabla 6.25: Tabla de p-valores correspondientes al test post-hoc de la tabla de contingencia.

NothingOrInsignificant	Acceptable
137	34
31	15
54	26
3	0

Tabla 6.26: Tabla total de tareas por nivel de educación y grado de completitud

La prueba chi-cuadrado de Pearson, nos permite contrastar si existe correlación entre el nivel de educación y el grado de completitud de la tarea. En este caso la prueba arroja un $p - valor = 0.0655$, que se aproxima al nivel de significación. Decidimos descartar la hipótesis nula de que no existe correlación entre las variables estudiadas, aunque los resultados del análisis sean sólo relativamente fiables.

El análisis post-hoc de la tabla de contingencia y los p-valores correspondientes se muestran en la tabla 6.25. Como se observa, las diferencias entre grupos no es estadísticamente significativa en ninguno de los casos. Sin embargo, considerando la proporción entre la cantidad de tareas calificados como *NothingOrInsignificant*, frente al nivel de educación (ver tabla 6.26), se obtiene que el grupo de participantes con nivel de educación de *Master* y *Bachellor* han realizado la mayor cantidad de tareas calificadas como *Acceptable* que equivalen aproximadamente al 32.5 %, frente al 19.9 % del grupo *Undergraduate*. El grupo *Other* no ha realizado ninguna tarea *Acceptable*.

La explicación de este resultado, podría tener varias aristas, por lo que no nos resulta evidente emitir un juicio bien fundado. Lo que no deja lugar a dudas es que el grupo de participantes con mayor nivel de conocimientos tienden a realizar mayor cantidad de código en las tareas experimentales.

Finalmente, debemos indicar que la relación entre un mayor nivel de conocimiento y el grado de completitud no significa que exista correlación con otros factores analizados. Por ejemplo, al analizar la edad en función del grado de completitud de las tareas, encontramos que los participantes mayores son los que más realizaron tareas calificadas como *NothingInsignificant*.

6.8.8. Experiencia profesional

En el gráfico 6.10, se presenta el meta-análisis de la experiencia profesional de los participantes en relación con el grado de completitud de la tarea.

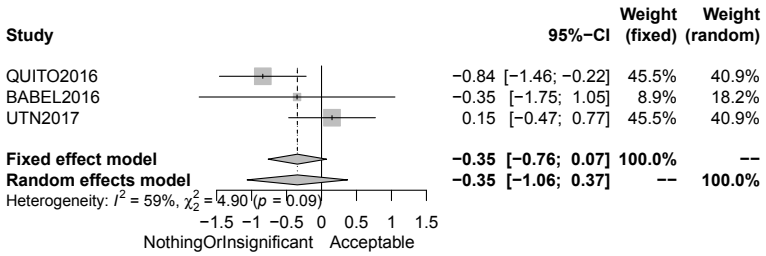


Figura 6.10: Meta-análisis de la experiencia profesional, en función del grado de completitud.

El meta-análisis de efectos fijos y aleatorios son homogéneos ($\chi^2 = 4.9 < 2$, p-value = 0.09).

El meta-análisis de la experiencia profesional de los participantes en función del grado de completitud produce un efecto positivo, con valor d =-0.35, lo que corresponde con un tamaño de efecto pequeño. Sin embargo no es estadísticamente significativo ($p - value <= 0.1055$).

6.8.9. Seniority

El factor *Seniority* indica el curso o grado de estudios de los estudiantes de pre-grado que han participado en los experimentos. El problema que tenemos con esta variable es que, en todos los casos, los estudiantes que han participado en un experimento tenían la misma seniority. La única excepción es UADY2015, en el cual participaron estudiantes de una misma carrera con diferentes niveles de estudios.

Por lo tanto, este análisis fue presentado como parte de los resultados de 5.11.5.3.

6.8.10. Uso de herramientas de prueba

En la figura 6.11 se presenta el meta-análisis referente al uso de herramientas de prueba en función del grado de completitud de la tarea.

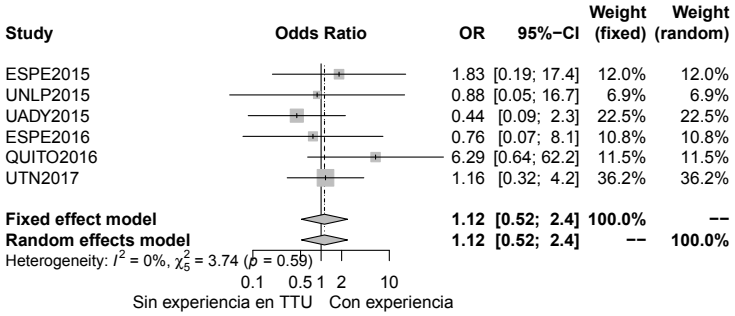


Figura 6.11: Meta-análisis del uso de herramientas de prueba, en función del grado de completitud.

El meta-análisis indica que los grupos son homogéneos para el modelo de efectos fijos y aleatorios ($\chi^2 = 3.74 < 5$, p-value $= 0.59$), con un tamaño de efecto OR $= 0.59$, lo que corresponde a un efecto muy pequeño. No obstante, en la figura 6.11 se observa que el efecto del uso de herramientas de prueba no es estadísticamente significativo.

6.8.11. Experiencia en el Framework JUnit

En cuanto a la experiencia previa que los participantes indicaron tener en el Framework JUnit y su relación con el grado de completitud de las tareas, el meta-análisis se presenta en la figura 6.12.

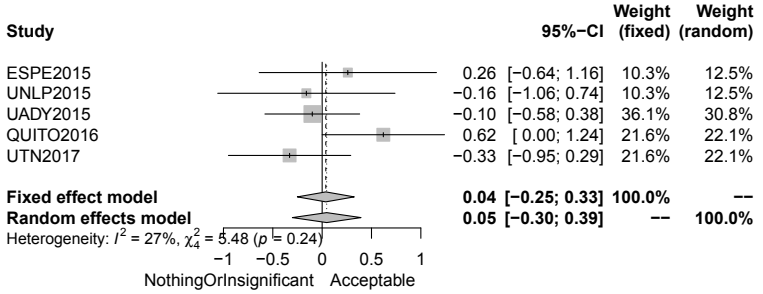


Figura 6.12: Meta-análisis de la experiencia en el Framework JUnit, en función del grado de completitud.

Los grupos son homogéneos para el modelo de efectos fijos y aleatorios ($\chi^2 = 5.48 < 4$, p-value = 0.24). El meta-análisis produce un tamaño de efecto positivo, con valor $d = 0.04$ en el modelo de efectos fijos, lo que corresponde con un tamaño de efecto muy pequeño. No obstante, este resultado no es estadísticamente significativo ($p - value <= 0.8014$).

6.8.12. Experiencia en el lenguaje Java

El meta-análisis de la experiencia en el lenguaje Java y el grado de completitud de la tarea, se presenta en la figura 6.13.

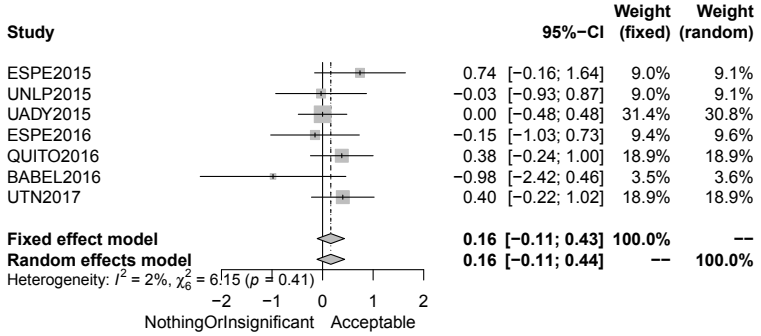


Figura 6.13: Meta-análisis de la experiencia en el lenguaje Java, en función del grado de completitud.

El meta-análisis de efectos fijos y aleatorios son homogéneos ($\chi^2 = 6.15 < 6$, p-value $= 0.41$). El meta-análisis indica que los grupos son homogéneos con un efecto positivo $d = 0.16$ en el modelo de efectos fijos y aleatorios, lo que corresponde con un tamaño de efecto muy pequeño, que no es estadísticamente significativo ($p - value <= 0.2368$).

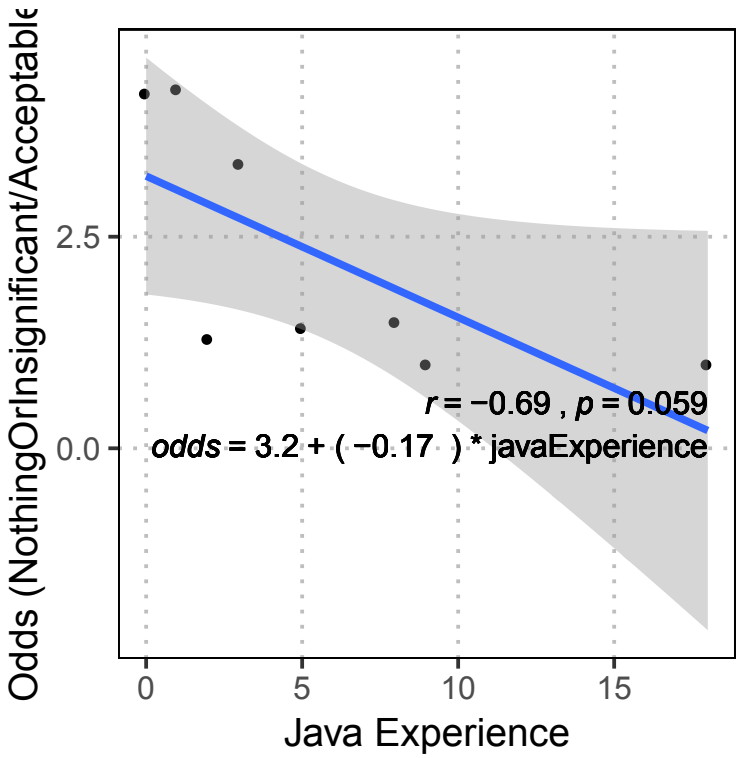


Figura 6.14: Probabilidad de realización de tareas en función de la experiencia en Java

Dado que el p-valor obtenido en el meta-análisis es bastante reducido, hemos optado por realizar un gráfico de scatter. Este gráfico muestra la probabilidad de que los sujetos realicen las tareas de forma *NothingOrInsignificant* en función de la experiencia en el lenguaje Java, y se presenta en la figura 6.14. En este caso, vemos una clara tendencia de que los sujetos con menor experiencia en Java tienen mayor probabilidad de realizar las tareas *NothingOrInsignificant*.

El coeficiente de correlación en este caso es $r = -0{,}689$, lo que corresponde con un efecto grande, y que es estadísticamente significativo ($p-value <= 0{,}0588$), aunque como hemos manifestado, los resultados producidos por los gráficos de scatter deben tomarse con cautela.

6.8.13. Resumen del Meta-análisis del Grado de completitud

En la table 6.27 se presenta un cuadro resumen del meta-análisis del Grado de completitud de la tarea. Como se observa, la edad de los sujetos influye significativamente en la cantidad de código que realizan los sujetos en las tareas experimentales. El nivel de educación se aproxima al nivel de significación, aunque debido a la poca cantidad de datos, no podemos extraer

conclusiones fiables.

Tabla 6.27: Resumen del Meta-análisis del Grado de completitud

FACTOR	Odds Ratio	p-value
UTTTraining	0.98	0.9709
knowEclipse	1.39	0.3138
ProgrammingExp	-0.12	0.401
Age	-0.64	0.0029
ExperienceYears	-0.35	0.1055
EducationLevel	-	0.0655
TestingToolUsage	1.12	0.7725
jUnitExperience	0.04	0.8014
javaExperience	0.16	0.2368

El meta-análisis nos ha permitido determinar que la edad de los participantes es un factor personal que ha influido significativamente en la realización de nuestros experimentos, por lo menos en aquellos realizados con profesionales. Los profesionales de mayor edad son quienes menos se mostraron motivados a entregar las tareas de la forma esperada, independientemente de la técnica de programación utilizada. Este hecho lo analizaremos con mayor detalle en la sección de discusión 8.

Capítulo 7

AMENAZAS A LA VALIDEZ

...

...

Resumen:

Es común que en los estudios experimentales algunos aspectos como el diseño experimental, la heterogeneidad de los sujetos, la instrumentación utilizada, la fatiga de los participantes, etc., puedan afectar la validez de los resultados. De acuerdo a[159], estos aspectos negativos, denominados habitualmente *amenazas a la validez*, se pueden clasificar en cuatro categorías: *Amenazas a la validez de conclusión estadística, amenazas a la validez interna, amenazas a la validez externa, y amenazas a la validez del constructo.* En este capítulo presentamos las amenazas a la validez que consideramos relevantes para nuestro estudio.

7.1. Amenazas a la validez de conclusión estadística

Las amenazas a la validez de conclusión son un conjunto de aspectos de carácter esencialmente estadístico, tales como el poder estadístico o las características de los sujetos experimentales, que pueden dar lugar a que el experimento obtenga resultados erróneos.

Nótese que sin algún tipo de referencia externa (por ejemplo, un conjunto de replicaciones experimentales), es imposible distinguir los resultados experimentales correctos de los erróneos. En consecuencia, los investigadores debemos minimizar las amenazas a la validez de la conclusión o, si ello no es posible, identificar el posible impacto de las amenazas no controladas.

El listado de amenazas a la validez de conclusión es bastante largo (véase[159]) o, para una descripción más pormenorizada[160]. En lo que sigue, analizaremos únicamente las amenazas que operan en esta investigación: Tasa de error tipo I, poder estadístico (o tasa de error tipo II), fiabilidad de la medición, fiabilidad de la administración de los tratamientos y heterogeneidad de los sujetos.

7.1.1. Tasa de error tipo I

Nuestro estudio tiene un carácter exploratorio. Para la realización de este estudio exploratorio, hemos utilizado *multiple testing* sin corrección, lo que da lugar a tasas de error tipo I considerablemente infladas y, sin duda, algunos falsos positivos. Sin embargo, esto entraña únicamente un riesgo relativo, ya que buscamos precisamente **posibles relaciones** entre aspectos personales y la efectividad de TDD. Dichas relaciones deben ser confirmadas o refutadas en estudios confirmatorios posteriores.

7.1.2. Poder estadístico

Tal y como se ha indicado en el Capítulo 4, es necesario disponer de 90 sujetos por grupo (180 sujetos totales) para identificar efectos medios de las covariables alcanzando un poder estadístico del 80 %. Para los factores, esta cifra se reduce a 18 sujetos por grupo (36 sujetos totales).

Hemos realizado siete experimentos. El experimento con más sujetos fue ESPE2015, que tenía 43 sujetos totales. Este experimento debería ser capaz por sí solo de identificar los efectos de los factores, pero está lejos de ser suficientemente sensible para identificar los efectos de las covariables. Por ello, hemos decidido realizar un meta-análisis *Individual Patient Data*[133] aunando, si así puede decirse, el poder estadístico de los experimentos individuales. Los siete experimentos engloban en total a 160 sujetos experimentales, o lo que es lo mismo, 80 por grupo. No todos los datos son usables (se han producido abandonos), pero aun así hemos alcanzado un tamaño muestral muy similar al requerido (90 sujetos por grupo). En consecuencia, esperamos que nuestros resultados no se vean seriamente afectados por errores de tipo II.

7.1.3. Fiabilidad de la medición

Para evitar sesgos en la medición, las variables respuesta QLTY y PROD fueron obtenidas de manera automática mediante el desarrollo de un software de medición específico. La variable respuesta *grado de completitud* fue medida por el autor de forma subjetiva. Sin embargo, creemos que al haber medido el *grado de completitud* en una escala de Likert de 3 valores (ninguno, insignificante y aceptable), la posibilidad de error de medición es baja. No obstante, hemos fusionado los niveles *ninguno* e *insignificante* de

la variable respuesta *grado de completitud* antes de proceder a su análisis. De este modo, los posibles efectos perniciosos de la falta de fiabilidad en la medición se reducen considerablemente.

7.1.4. Fiabilidad de la administración de los tratamientos

La falta de conformidad puede adoptar dos formas distintas. Por un lado, está la no utilización de TDD (o su uso en grados variables) por parte de los sujetos cuando deberían hacerlo (esto es, en la correspondiente sesión experimental). Sin embargo, es difícil averiguar en la práctica como esto ocurre. Hemos decidido no descartar los datos de ningún sujeto en función de la aplicación correcta o incorrecta de TDD. Al ser un estudio fundamentalmente exploratorio, no hemos querido prescindir de cualquier información que nos permitiera llegar a algún tipo de conclusión, aun sabiendo por anticipado que ello podía reducir considerablemente el poder estadístico de los experimentos. Otra alternativa es que los sujetos experimentales no se esfuercen en la resolución de las tareas experimentales. Para afrontar esta amenaza, hemos definido la variable respuesta *grado de completitud*, lo que nos ha permitido estudiar por qué algunos sujetos no trabajaban durante la realización de las tareas experimentales.

7.1.5. Heterogeneidad de los sujetos

Hemos incluido esta amenaza en al capítulo, precisamente para dejar claro que no aplica, o bien que tiene muy poca importancia. Dado que nuestro objetivo es identificar las características personales influyentes, hemos afrontado directamente esta amenaza. En otras palabras: hemos analizado explícitamente si los aspectos personales de cada sujeto experimental puede explicar los resultados que dicho sujeto obtiene al aplicar ITLD/TDD.

7.2. Amenazas a la validez interna

La validez interna hace relación al grado de certidumbre de que la variable independiente sea realmente responsable de los cambios producidos en la variable dependiente (esto es, la variable respuesta). Un experimento de mediadas repetidas, como es nuestro caso, podría estar expuesto a las siguientes amenazas a la validez (Shadishet al. 2001): fatiga, maduración, arrastre, pérdida de participación, selección, e instrumentación.

7.2.1. Fatiga de los sujetos experimentales

Dependiendo del contexto, sea este académico o industrial, los experimentos fueron realizados durante el horario de trabajo o de estudios regular. Durante este período, los desarrolladores no realizaron sus actividades

laborales normales; su única responsabilidad era la participación en el experimento y el entrenamiento previo en las técnicas de TDD relacionadas. Por lo tanto, el experimento no indujo ningún esfuerzo adicional a los sujetos. De hecho, se procuró en la mayoría de casos que el horario de realización de los experimentos fuera más relajado que un día de trabajo regular.

En los experimentos con profesionales no pudimos evitar, sin embargo, que en ciertos casos tuvieran que ocuparse de atender a algún incidente no previsto, lo que les obligó a abandonar las sesiones experimentales (este hecho se reportó en cada uno de los experimentos). Finalmente, las sesiones experimentales fueron realizadas en días distintos a las sesiones de entrenamiento en la mayoría de los casos, justamente para permitir que los participantes descansen y así evitar que la fatiga tuviera un efecto pernicioso.

7.2.2. Maduración

En experimentos de medidas repetidas, existe la posibilidad de que las tareas realizadas en primer lugar influyan de algún modo en las tareas realizadas a continuación. En esta investigación, todos los sujetos aplicaron de manera secuencial ITLD y TDD, en este orden. Esto podría suponer una amenaza a la validez, ya que los valores de las variables respuesta para TDD aumentarán o disminuirán (lo más probable es que aumenten debido al efecto de maduración) en comparación a un experimento *between-subjects*.

No obstante, en la presente investigación, la maduración produce un efecto necesario y beneficioso. ITLD es una estrategia de programación que se parece mucho a la programación *test-last* tradicional y que, al mismo tiempo, sirve de referencia para aprender TDD. Después de dominar ITLD, los desarrolladores pueden aprender TDD más rápidamente, lo cual es importante debido al limitado tiempo disponible para la formación. La maduración, en consecuencia, apoya la adecuada administración de TDD mejorando la efectividad de los sujetos experimentales, contribuyendo al éxito del experimento.

7.2.3. Arrastre

En inglés, esta amenaza se conoce como *carry-over*, y afecta a experimentos de tipo *cross-over*, como es el caso de nuestra investigación. Tiene similitudes con la maduración, pero en este investigación su efecto no está relacionado con las estrategias de programación sino con la secuencia de realización de las tareas experimentales MR y BSK que se modeliza mediante la variable independiente GROUP de manera consistente, lo que sugiere que no existe el *carry-over*.

En cualquier caso, tal y como indica [161], el *carry-over* es difícil de detectar estadísticamente, por lo que deberíamos pensar si existe o no desde un punto de vista teórico. En nuestros experimentos, la secuencia de realización

de tratamientos es siempre ITLD ->TDD, por lo que no puede existir un efecto de arrastre. La diferencia entre las secuencias se reduce a efectuar la tarea MR antes que BSK, o viceversa.

Dado que no existe una razón que sugiera la existencia de un efecto de arrastre entre tareas, lo más razonable es considerar que el *carry-over* no está afectando a esta investigación.

7.2.4. Perdida de participación

En los experimentos con profesionales, varios sujetos abandonaron el experimento antes de su finalización. Hemos indicado las posibles causas en los experimentos correspondientes. En general, creemos que los motivos son: 1) la carga de trabajo del sujeto o 2) que la formación no cumplió con sus expectativas. El abandono no puede corregirse de ningún modo, por lo que tenemos que reconocer que esta amenaza puede estar operando en la presente investigación. En los estudios con estudiantes el efecto de abandono fue mínimo en razón de que la motivación de estos sujetos es distinta.

7.2.5. Selección

La amenaza de selección puede ocurrir debido al enfoque de muestreo utilizado para seleccionar los sujetos del experimento. Debido al entorno en el que se llevaron a cabo los experimentos, no tuvimos la oportunidad de muestrear al azar a los sujetos de una población determinada. En su lugar, tuvimos que confiar en una muestra de conveniencia. Nótese que esta situación es común en muchos estudios experimentales. A este respecto, solo podemos esperar que futuros estudios de otros investigadores complementen nuestros resultados y pongan de manifiesto lo adecuado/erróneo de la muestra seleccionada.

7.2.6. Instrumentación

La amenaza de instrumentación puede ocurrir debido al formato de las especificaciones de las tareas y, o, a la dificultad de las mismas. Para intentar contrarrestar este efecto: 1) las tareas experimentales fueron incluidas en el experimento como variables independientes, 2) ensayamos en seis de los siete experimentos el nivel de especificación de la tarea (con slicing o sin slicing).

7.3. Amenazas a la validez de constructo

La validez de constructo hace referencia, esencialmente, al grado en que las variables respuesta y las mediciones realizadas representan realmente lo que deseamos medir. Por ejemplo, en esta investigación usamos la variable respuesta QLTY para valorar la *calidad* del código. ¿En qué medida QLTY

representa la *calidad* real? Esta pregunta es la que se desea contestar analizando las amenazas a la validez de constructo.

7.3.1. Sesgo de mono-operación

Nuestro experimento no sufrió (al menos, de forma excesiva) un sesgo de mono operación, ya que usamos dos variables respuesta con métricas diferentes. La incorporación de una tercera variable respuesta (el *grado de completitud*) aumentó el repertorio de operacionalizaciones del constructo subyacente (cuanto de "bien.^{es}taba desarrollado el código).

7.3.2. Sesgo de mono-método

Usamos Java como lenguaje de programación tanto durante el entrenamiento como durante los tratamientos, y usamos el mismo tipo de instrumento (suites de prueba de aceptación) para medir las variables respuesta para las tres tareas (SS, MR y BSK). El sesgo de mono-método no puede descartarse. No obstante, debe notarse que QLTY y PROD se han medido de forma similar a trabajos previos (Erdogmus et al., 2005; George y Williams, 2004 ; Fucci y Turhan, 2013), lo que permite, a costa de la presente amenaza de validez, que nuestros resultados puedan compararse más fácilmente con experimentos anteriores y estudios secundarios.

7.3.3. Expectativas del experimentador

Los experimentadores no teníamos ninguna expectativa. En prevención de que los sujetos pudieran realizar alguna suposición, nos cuidamos de transmitir el modo en que evaluaríamos el desempeño de los sujetos.

7.4. Amenazas a la validez externa

La validez externa se ocupa de la generalización de los resultados experimentales en términos de los objetos (tareas), sujetos y tecnologías utilizadas:

- Las tareas experimentales que elegimos fueron pequeñas en términos de líneas de código, y sus complejidades también fueron menores que las aplicaciones industriales típicas de la vida real. Nuestros resultados no se pueden generalizar a grandes sistemas de software.

- Las tareas experimentales implicaban el desarrollo de un software desde cero. En consecuencia, nuestros resultados no deberían generalizarse a actividades de mantenimiento de software.

- Los sujetos experimentales participantes fueron profesionales y estudiantes, desde novatos hasta desarrolladores senior. Tal y como se puede observar en las tablas demográficas presentadas para cada uno de los estudios, la diversidad de sujetos experimentales ha sido amplia. Sin embargo, prácticamente todos los sujetos no tenían experiencia en TDD. En un único experimento (ESPE2015), tres de los 43 participantes indicaron tener entre 2 a 5 años de experiencia previa en TDD. Esto representa apenas el 1,8 % (aproximadamente) del total de sujetos participantes en los 7 estudios realizados. Por lo tanto, creemos que nuestros resultados se pueden generalizar a profesionales o estudiantes que aunque conozcan la técnica TDD no tienen experiencia práctica previa en su aplicación.

- La mayoría de los experimentos reportados en la literatura utilizan Java como lenguaje de programación, Eclipse como entorno de desarrollo y jUnit como herramienta de prueba unitaria (por ejemplo, Erdogmuset al. 2005; George y Williams2004; Fucci y Turhan2013). Elegimos la misma configuración tecnológica y la mayoría de los sujetos se sintieron cómodos con ella. Técnicamente, existe el riesgo de que nuestros hallazgos dependan de las tecnologías utilizadas en este estudio, pero este riesgo es bastante pequeño, ya que: 1) los enfoques de desarrollo aplicados son independientes del lenguaje de programación y la herramienta de prueba unitaria utilizada, 2)jUnit es el estándar de referencia para la realización de pruebas unitarias, y 3) la mayoría de lenguajes e IDE tienen funcionalidades y herramientas análogas a las utilizadas en este experimento.

Capítulo 8

DISCUSIÓN

Creo que en la discusión de los problemas naturales, deberíamos comenzar no con las escrituras, sino con experimentos y demostraciones

Galileo Galilei

Resumen:

En este capítulo presentamos las principales conclusiones obtenidas en la presente tesis doctoral, organizadas por cada pregunta de investigación.

8.1. Discusión

Para contestar las preguntas de investigación planteadas, hemos adoptado ciertas posturas que amplían la perspectiva dominante en la literatura:

- En la mayoría de los estudios secundarios, el efecto de la experiencia se ha determinado comparando estudios realizados con estudiantes y estudios realizados con profesionales. Nosotros también hemos adoptado esta postura, tal y como se puede comprobar en el capítulo de Síntesis y en el Apéndice A.

  No obstante lo anterior, la síntesis se ha realizado teniendo en cuenta no sólo la dicotomía profesional-estudiante, sino el tipo de reclutamiento de los sujetos experimentales. Hay tres ventajas en descomponer los experimentos en subgrupos usando el tipo de reclutamiento:

  - Desaparece la heterogeneidad entre-experimentos. Las diferencias pueden adscribirse a la nueva variable *recruitment*.

  - Añadimos un nivel adicional de explicación de los datos obtenidos. Algunos resultados pueden explicarse en función del tipo de reclutamiento de los sujetos.
  - Mejora la estabilidad de los resultados. Cuando se utiliza una descomposición profesional vs. estudiante, el factor STRATEGY pasa de ser significativo a no significativo, y viceversa, en función del factor personal considerado. Esto no ocurre con la variable *recruitment*, la cual hace que los distintos análisis proporcionen resultados coherentes en prácticamente todos los casos.

- La edad de los sujetos participantes de los experimentos no ha sido un factor explícitamente estudiado en otros trabajos encontrados en la literatura. Creemos que este factor es un aspecto que podría tener cierta relación con los resultados experimentales y, de hecho, los datos respaldan nuestra posición.

- Finalmente, el grado de completitud no fue un aspecto que en un principio consideramos analizar en la presente investigación. Decidimos incluirlo dada la cantidad de participantes que entregaron las tareas experimentales prácticamente sin realizar nada de código. El grado de completitud enriquece la interpretación de los resultados experimentales, ya que no sólo podemos saber cuánto (Productividad) o cuánto de bien (Calidad) han trabajado los sujetos experimentales, sino también algo del modo en que lo han hecho (Completitud).

Contestaremos a las preguntas de investigación en las siguientes secciones:

8.1.1. ¿Cuál es la influencia de los factores humanos sobre la calidad externa cuando se utiliza TDD? (RQ1)

8.1.1.1. Independientemente de los aspectos personales

Nuestros resultados indican que TDD no mejora la calidad externa. Si analizamos todos los estudios conjuntamente, el factor STRATEGY no resulta estadísticamente significativo (ver Tabla 6.1a).

8.1.1.2. Realizando una descomposición en subgrupos

Al comparar TDD con la estrategia ITLD, el resultado de nuestro meta-análsis arrojó resultados heterogéneos tanto para la variable calidad externa como para la productividad. Por tal razón, realizamos dos descomposiciones en subgrupos: La primera considerando el tipo de reclutamiento, y la segunda separando a estudiantes y profesionales. A este respecto:

- Si tenemos en consideración la variable de descomposición en subgrupos *recruitment*, el factor STRATEGY sigue siendo no significativo (ver Tabla 6.8a).
- El tipo de sujeto (profesional o estudiante) tampoco influye en la calidad del código obtenido (ver Tablas 6.3a y 6.4a).

Varios estudios secundarios han reportado que la calidad externa mejora cuando TDD es utilizado por profesionales[16],[15],[19],[20],[21]. En nuestro caso, la calidad disminuye en los experimentos BABEL2016 y QUITO2016, aunque no se alcance la significación estadística. En el caso del experimento UTN2017, la calidad aumenta de forma estadísticamente significativa (ver Sección 5.14). La diferencia fundamental entre los experimentos BABEL2016/QUITO2016 y UTN2017 es el tipo de reclutamiento: Los experimentos en BABEL2016/QUITO2016 tienen el formato de un curso de formación, mientras que en UTN2017 los sujetos están obligados a participar al tratarse de un experimento realizado en academia.

Los estudios[16] y[21] reportan que la calidad mejora en los experimentos con estudiantes. Eso es ciertamente lo que ocurre en los experimentos ESPE2015 y ESPE2016. Sin embargo, la calidad decrece claramente en los experimentos UADY2016 y UNLP2015. La diferencia fundamental entre ESPE2015/ESPE2016 y UADY2016/UNLP2015 residen en que, en el primer, caso los sujetos están obligados a participar en el experimento; en el segundo caso, los estudiantes participan voluntariamente.

8.1.1.3. Considerando factores personales concretos

Podemos distinguir dos tipos de resultados:

- Factores influyentes **independientemente del tipo de sujeto**: Se trata de las variables *testingToolUsage* (experiencia en el uso de herramientas de prueba) y *javaExperience* (experiencia en programación Java). Tal y como dicta el sentido común, cuando los sujetos poseen experiencia en cualquiera de dichos factores, la calidad del código mejora independientemente de si se usa ITLD o TDD (ver Tablas 6.18 y 6.21a). Los efectos son de 50.8 para *testingToolUSage* y 16.34 para *javaExperience*, siendo estadísticamente significativos.

  En lo que respecta a la experiencia en el uso de herramientas de prueba, los resultados son coherentes con lo indicado por *Geras et al.*[57]. Los autores indican que la efectividad de TDD mejora cuando aumentan las habilidades en pruebas de los sujetos.

  Los resultados anteriores se obtienen cuando se utiliza la variable de descomposición en subgrupos *recruitment*. En el Apéndice A proporcionamos el análisis de los factores personales utilizando una descom-

posición de los experimentos en los subgrupos profesional vs. estudiante. En este caso, la variable *javaExperience* sigue siendo significativa para la calidad cuando los sujetos son profesionales. Este hecho aumenta nuestra confianza en, al menos, el efecto positivo de la experiencia en Java.

Es conveniente indicar que *Fucci et al.*[111] indican que TDD no tiene un impacto significativo en la calidad externa, aunque este trabajo está limitado a estudiantes de maestría. En nuestro caso, los únicos estudiantes de maestría pertenecen al experimento UNLP2015 y, coincidimos en que la calidad obtenida para ITLD es muy similar a la obtenida para TDD; los resultados son obviamente no significativos (ver sección 5.13).

Hubiésemos querido hacer una comparación más específica entre nuestros resultados y aquellos reportados en estudios primarios, sin embargo, como podrá advertir el lector en la tabla `https://github.com/georaura/tddexperiments/tree/master/appendices`, conseguimos identificar las métricas utilizadas en los estudios primarios, sin embargo, la diversidad de las mismas, hace que sea inútil realizar una comparación medianamente convincente. A modo de ejemplo podemos citar algunas métricas utilizadas y que no necesariamente miden, al menos de forma directa, la calidad externa o productividad: *Process conformance, Developers knowledge of Java Programming, Unit Testing, Lines of code, Number of test, Test covergage, Changed Lines of Code, Editing Speed, Duration of Implementation, Quality of Tests, Number of Failed Tests, Size of Developed Programs, Individual satisfaction, Performance accuracy of code, Progress Tracking, Discipline, etc.*

- Factores no influyentes **independientemente del tipo de sujeto**: El entrenamiento en pruebas unitarias, la edad, el conocimiento del entorno de desarrollo eclipse, la experiencia profesional, la experiencia en programación, el nivel de educación, la *seniority*, la función actual en la organización y la experiencia en el framework JUnit no produjeron diferencias significativas en lo que respecta a la calidad externa.

  El entrenamiento en TDD debería aumentar la calidad externa según *Lui and Chan*[104]. No hemos podido comprobar este extremo debido a que la inmensa mayoría de nuestros sujetos no habían recibido entrenamiento en TDD y por ello no hemos podido estudiar este factor personal.

8.1.2. ¿Cuál es la influencia de los factores humanos sobre la productividad cuando se utiliza TDD? (RQ2)

8.1.2.1. Independientemente de los aspectos personales

El análisis conjunto de los datos indica que el factor STRATEGY es estadísticamente significativo (ver Tabla 6.1b). Los desarrolladores que usan TDD son más productivos que los que usan ITLD (el análisis post-hoc no aparece en el texto pero se obtiene fácilmente del código del capítulo de Síntesis). Este resultado es coherente con lo indicado en algunos estudios secundarios de TDD. Por ejemplo *Bissi al.*[21] y *Rafique and Misic*[15] concluyen que TDD tiene un efecto positivo sobre la productividad en estudios académicos.

No obstante, la realidad es un tanto más compleja, como indicaremos inmediatamente a continuación.

8.1.2.2. Realizando una descomposición en subgrupos

Si tenemos en consideración la variable de descomposición en subgrupos *recruitment*, STRATEGY pasa a ser no significativo (ver Tabla 6.8b). En su lugar, la interacción STRATEGY**recruitment* pasa a ser estadísticamente significativa. El análisis post-hoc (ver Tabla 6.9) muestra que TDD mejora la productividad sólo para los *conscripted*, pero no para los restantes tipos de reclutamiento.

La variable de descomposición en subgrupos *recruitment* esta fuertemente relacionada con el tipo de sujeto (profesional o estudiante). Si atendemos a lo indicado en las Tablas 6.3b y 6.4b, podemos comprobar que el factor STRATEGY es estadísticamente significativo para los estudiantes. Este resultado es coherente con *Bissi al.*[21].

Los profesionales no mejoran su productividad. Esto último es muy relevante, debido a una particularidad de la familia de experimentos realizada en esta investigación. Los sujetos que han participado en el experimento UTN2017 (ver Capítulo 5.14) son profesionales, pero el experimento se realizó en academia. Los sujetos de UTL2017 fueron más productivos usando TDD. Podría decirse, en cierta medida, que el experimento UTN2017 se parece mucho a los experimentos con estudiantes.

En los otros dos experimentos con profesionales – QUITO2016 (ver Capítulo 5.8) y BABEL2016 (ver Capítulo 5.9) – los sujetos fueron menos productivos usando TDD, aunque las diferencias no son estadísticamente significativas. Esto coincide con estudios secundarios analizados en la literatura, donde se indica que TDD disminuye la productividad con profesionales[15,16,21].

Makinen et al.[19] indica que los profesionales son más rápidos escribiendo código cuando usan TDD. Podría inferirse de esto que los profesionales

son más productivos. No obstante, la velocidad no es equivalente a nuestra productividad, por lo que los resultados de *Makinen et al.*[19] podrían ser compatibles con los nuestros.

Con la debida precaución, creemos que el aspecto esencial que influye en la productividad no es el carácter profesional o estudiante del sujeto, sino el modo en que ha sido reclutado y, más probablemente, el efecto que el tipo de reclutamiento ejerce en el sujeto. Es posible que los sujetos *conscripted* perciban que TDD es una materia que deben "aprobar", y por ello se esfuerzan en su aplicación.

Por el contrario, los sujetos que realizaron el experimento como un *training course* pierden productividad a medida que pasa el tiempo (recuérdese que TDD se aplica el último lugar). ¿Están desmotivados o cansados?

Cabría preguntarse por qué estos sujetos no abandonaron el curso. En realidad, eso es justamente lo que ocurre. El número de abandonos en QUITO2016 y BABEL2016 es enorme. Los que es quedan es porque *no pueden marcharse*, ya que la empresa les ha concedido licencia para participar en el entrenamiento en TDD. La información de que disponemos sugiere que la participación forzada causaría desinterés, lo que motivaría la caída de productividad.

8.1.2.3. Considerando factores personales concretos

Podemos distinguir tres tipos de resultados:

- Factores influyentes **independientemente del tipo de sujeto**: *testingToolUsage* (experiencia en el uso de herramientas de prueba) tiene un efecto de 9.03 estadísticamente significativo (ver Tabla 6.17).

  Asimismo, *knowEclipse* (conocimiento del IDE Eclipse) posee un pequeño efecto positivo, estadísticamente significativo, cuando se utiliza TDD (ver Tabla 6.11b). Esto es, el conocimiento del IDE Eclipse no tiene ningún impacto para ITLD.

  En el Apéndice A proporcionamos el análisis de los factores personales utilizando una descomposición de los experimentos en los subgrupos profesional vs. estudiante. En este caso, la variable *knowEclipse* presenta un efecto significativo para la Calidad cuando los sujetos son estudiantes. La variable *knowEclipse* no resulta significativa para la productividad. No es posible conjugar todos estos resultados, pero la tendencia de fondo sugiere un cierto efecto positivo (no sabemos si en la Calidad o Productividad, o para ciertos tipos de sujetos) de la variable *knowEclipse*.

- Factores influyentes **en poblaciones concretas**: La variable *age* sólo ha podido ser estudiada en los experimentos con profesionales. Tal y como observarse en la Tabla A.8b, la edad ejerce un efecto negativo

de -0.06 puntos y (a efectos prácticos) estadísticamente significativo en la productividad (p-value = 0.055). Esto significa que los sujetos son menos productivos a medida que aumenta su edad, y ello independientemente de si aplican ITLD o TDD.

El descenso de la productividad con la edad puede observarse en la Figura 8.1. La reducción ocurre en todos los experimentos con profesionales, incluyendo UTN2017, lo que sugiere que el tipo de reclutamiento no interacciona con la edad.

La Figura 8.1 posee otra característica interesante. La recta de regresión (línea continua) es claramente descendente, pero el predictor de la media (*loess*) no es estrictamente lineal, sino que sube y baja ligeramente. En los experimentos QUITO2016 y UTN2017, el máximo se alcanza a los 30 años, baja hacia los 35, sube de nuevo en torno a los 40, y después baja ya de forma ininterrumpida. Quizás la figura sea más sencilla de interpretar si nos atenemos a los extremos del intervalo de confianza, donde destaca el máximo en los 30 años y la bajada de productividad a edades mayores, particularmente a partir de los 40 (ver el gráfico del conjunto de experimentos con profesionales).

No disponemos de datos de edad para estudiantes, pero es posible que se observe el patrón mostrado en BABEL2016 y, mucho más claramente, en el conjunto de experimentos con profesionales, esto es, una subida progresiva de en la productividad de los 20 a 30 años.

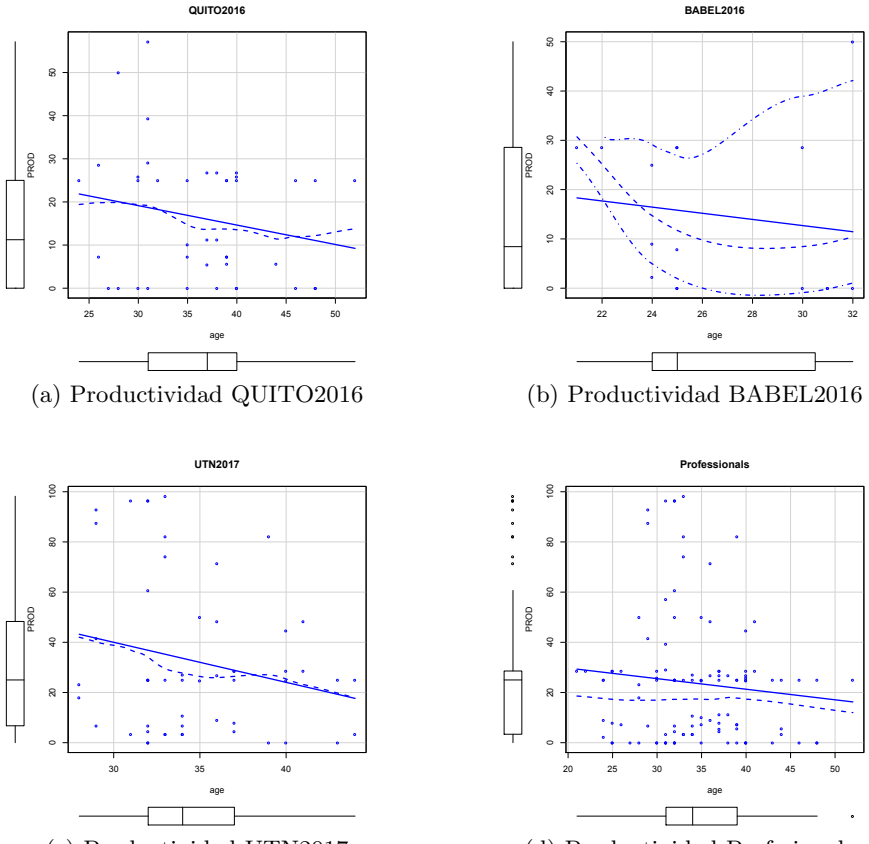
El efecto de la edad en la Productividad podría explicar la inconsistencia entre nuestros resultados y estudios anteriores ([15],[16],[21]). La media de edad de nuestros sujetos es 34 años, ya en el periodo de declive. Si los sujetos de otros experimentos fueran más jóvenes, las productividades podrían ser más altas.

Bien es cierto que los estudios de[15],[16],[21] indican que la Productividad disminuye con TDD con profesionales, y que el efecto de la edad que hemos identificado es independiente de ITLD/TDD. Pero también es cierto que la Productividad aumenta en UTN2017 donde los sujetos fueron profesionales en activo. La conclusión más razonable es que tanto el tipo de reclutamiento como la edad tienen efectos separados en la productividad. En cuanto a la variable *Seniority* o el grado en el cual se encontraban los estudiantes al momento de realizar el experimento, únicamente en el experimento UADY2015 encontramos diferencias entre grupos de sujetos que podían compararse, resultando (a efectos prácticos) significativos para la Productividad (ver tabla Tabla 5.172b con un tamaño de efecto negativo de -5.83 puntos. Como se había comentado, el efecto es inverso a lo esperado: los estudiantes de mayor grado de estudios, resultaron ser menos productivos. Probablemente la explicación radica en que los grupos no estuvieron balanceados, un

único sujeto estaba en el nivel de maestría y la mayor parte en el quinto nivel de su carrera.

Al igual que para el caso de la Calidad, no hemos sido capaces de comparar la Productividad obtenida en nuestros estudios, con los resultados de estudios primarios identificados en nuestra revisión de literatura. La falta de homogeneidad en las métricas utilizadas, hace que esta tarea sea inútil y que no podamos obtener conclusiones razonables, con excepción de lo antes comentado.

- Factores no influyentes **independientemente del tipo de sujeto**: El entrenamiento en pruebas unitarias, la edad, la experiencia profesional, la experiencia en programación, el nivel de educación, la *seniority*, la función actual en la organización, la experiencia en el framework JUnit, la experiencia en programación Java, no produjeron diferencias significativas en lo que respecta a la Productividad.



(a) Productividad QUITO2016

(b) Productividad BABEL2016

(c) Productividad UTN2017

(d) Productividad Profesionales

Figura 8.1: Análisis de regresión de los experimentos con profesionales

8.1.3. ¿Cuál es la influencia de los factores humanos sobre el grado de completitud de las tareas experimentales cuando se utiliza TDD? (RQ3)

Uno de los principales hallazgos encontrados en el experimento realizado en la industria (el experimento Quito2016) fue que un elevado número de sujetos abandonaba las sesiones experimentales o no realizaba ningún trabajo en las sesiones experimentales. Este comportamiento no ocurría generalmente con los experimentos realizados en la academia. En consecuencia, definimos el grado de completitud como la cantidad de código nuevo o modificado (respecto a la plantilla proporcionada al inicio del experimento) entregado por los sujetos participantes al final de las sesiones experimentales. Nuestro meta-análisis produjo los siguientes resultados:

- La edad ha resultado estadísticamente significativa para sujetos profesionales (ver Figura 6.7). Hemos comprobado que conforme los sujetos tienen mayor edad, las tareas se entregaron menos completas. Muchas de ellas fueron entregadas sin ningún tipo de modificación.

  Este resultado es compatible y, de hecho, probablemente explica la baja productividad de los sujetos profesionales. Los sujetos de mayor edad, que son la mayoría, tienden a no realizar las tareas experimentales, lo que a su vez produce un descenso de la productividad.

  La experiencia profesional, que está fuertemente correlada con la edad ($r = 0{,}66$, $p - value < 0{,}001$), exhibe un comportamiento semejante a la edad, aunque no estadísticamente significativo (ver Figura 6.10).

  Esto se puede explicar de forma razonable por el tipo de motivación de los participantes y también por el trabajo asignado en la empresa durante la realización del experimento. En el caso de QUITO2016, la asistencia era voluntaria sin mayor motivación que la entrega de un certificado de participación. En el caso de BABEL2016, los sujetos tenían actividades pendientes que resolver y algunos decidieron asistir a una o dos sesiones de entrenamiento pero no así a las sesiones experimentales.

  La excepción es UTN2017, cuya realización tiene más en común con los experimentos en academia que con los experimentos QUITO2016 y BABEL2016.

- Un factor que se acercó al nivel de significación estadística del 5 % fue el nivel de educación para el caso de los experimentos realizados con estudiantes. El grupo de participantes de Bachelor y Master fueron quienes entregaron las tareas experimentales más completas. Tiene sentido ya que los sujetos formados en la universidad está más acostumbrados, y probablemente aprecia más, los cursos de formación del estilo impartido durante la realización del experimento.

- Hemos dejado para el final un resultado que nos intriga. Hemos achacado anteriormente la falta de productividad y la incompletitud de las tareas a problemas de motivación. No obstante, no estamos seguros de que éste sea realmente el caso. Los sujetos experimentales entregan tareas más completas cuando aplican TDD que cuando aplican ITLD (ver Figura 6.3). En particular, los sujetos profesionales hacen tantas o más modificaciones en el código que los estudiantes cuando usan TDD, independientemente del tipo de reclutamiento (véanse los datos de QUITO2016 y UTN2017). ¿Por qué razón hacen los profesionales tantos o más cambios que los estudiantes si están desmotivados?

  El cansancio/aburrimiento no parece una explicación razonable, ya que TDD es la segunda tarea experimental, esto es, el cansancio debería tener como efecto un mayor grado de completitud para ITLD, y no al revés.

  Otra explicación que parecería razonable es que los sujetos profesionales no llegaron a entender la necesidad de una tarea experimental aplicando ITLD, y por ese motivo entregaron las tareas ITLD incompletas. Sin embargo, no se entendería en este caso que la productividad disminuya, aunque de forma no significativa, para TDD.

  La única explicación plausible que encontramos para los efectos de la edad y estrategia de programación en la productividad y grado de completitud es la efectividad del curso de formación. Los sujetos profesionales están realmente interesados en TDD y ello se refleja en la cantidad de trabajo realizado, pero ese trabajo no es efectivo. Parecería que la formación no tiene efecto, ya sea porque no se ha impartido correctamente (cosa que dudamos después de haber realizado muchos cursos), ha sido demasiado corta (lo que podría ser cierto) o no ha tenido impacto en los sujetos.

  Lo anterior aplica sólo para sujetos profesionales. Los sujetos estudiantes parecen igualmente interesados en ITLD y TDD, ya que el grado de completitud es parecido en ambos casos. La productividad de los estudiantes se incrementa con TDD. Y la efectividad de los cursos de formación en estudiantes se supone, incluyendo los sujetos de UTN2017 que realizan el experimento en un ámbito académico.

  Es como si los sujetos profesionales ya no supieran aprender. El hecho de que las tareas incompletas aumenten con la edad apoya esta hipótesis.

Capítulo 9

CONCLUSIONES Y FUTURAS LÍNEAS DE INVESTIGACIÓN

Tu tarea no es predecir el futuro, sino hacerlo posible.

Antoine de Saint-Exupéry, *Ciudadela*

9.1. Conclusiones

9.1.1. Influencia de TDD en la Calidad Externa y Productividad

El desarrollo de software es un fenómeno complejo, en el que influyen aspectos personales como la experiencia, aspectos psicológicos como la motivación, el conocimiento previo de determinadas técnicas o tecnologías, etc. Todos estos aspectos están presentes de forma natural en los programadores, por lo que afectan a la calidad del código desarrollado, a la productividad, y seguramente a otros aspectos importantes del proceso de desarrollo de software. Los aspectos personales han sido discutidos con cierta amplitud en la literatura, como hemos podido corroborar en el estado de la cuestión. Sin embargo, también hemos podido comprobar la escasa evidencia empírica existente, lo que limita la capacidad de la comunidad de IS para determinar de forma rigurosa la influencia de los factores personales en el proceso de desarrollo.

Actualmente, la experimentación en Ingeniería de Software se centra, ya no sólo en la realización de experimentos aislados, sino en una serie de experimentos relacionados (o familias de experimentos según Victor Basili [31]). En este sentido, durante la realización de este trabajo de investigación, he-

mos podido evidenciar la importancia de la replicación de experimentos. La familia de experimentos que hemos realizado nos ha permitido obtener resultados estadísticamente significativos que, usando experimentos aislados, no hubieran sido detectados.

En esta investigación, hemos realizado una síntesis de resultados de 7 experimentos. Independientemente de los factores humanos, la aplicación del desarrollo basado en pruebas (TDD) en comparación con el desarrollo incremental con pruebas al final (ITLD), no mejoró la calidad externa del código, pero sí la productividad de los programadores.

9.1.2. Influencia de TDD en la Calidad Externa y Productividad, considerando distintos tipos de sujetos

No obstante, cuando realizamos una descomposición en subgrupos separando estudiantes de profesionales, así como considerando el tipo de reclutamiento (dividiendo entre participantes voluntarios *volunteer*, no voluntarios *conscripted* y aquellos que participaron de los experimentos como un curso de entrenamiento *Training course*), la síntesis produce resultados más matizados.

Los resultados indican que la Calidad externa no produjo diferencias significativas al aplicar TDD o ITLD, aunque en ciertos casos si hubo mejora en experimentos con estudiantes (que participaron como *conscripted*), pero en otros casos la Calidad externa decreció cuando fueron estudiantes que participaron como voluntarios.

Para el caso de la Productividad, la variable de descomposición en subgrupos *recruitment* está fuertemente relacionada con el tipo de sujeto (profesional o estudiante). En algunos de los experimentos realizados en la academia, los participantes estuvieron, de cierta manera, condicionados a realizar las tareas experimentales de una manera adecuada, independientemente si fueron profesionales o estudiantes (fueron motivados por obtener una calificación). En otros experimentos, los participantes no tenían la .ºbligación"de realizar las tareas experimentales, ya que su participación fue voluntaria a modo de un curso de entrenamiento. El tipo de reclutamiento, sería la clave de las diferencias encontradas. . Nuestra síntesis produjo resultados significativos en la Productividad para la estrategia de programación con estudiantes y con aquellos participantes no voluntarios (*conscripted*). Sin embargo, no arrojó resultados significativos en el caso de los profesionales participantes como un *training curse* o *volunteer*.

De lo anterior se concluye, con la debida precaución, que la manera en que fueron reclutados los participantes de nuestros experimentos incidió en la Calidad externa y la Productividad, más allá de la técnica aplicada. La motivación del sujeto pudo influir en el grado de interés por realizar las tareas experimentales y, en consecuencia, en la Calidad y Productividad

alcanzadas.

9.1.3. Influencia de TDD en la Calidad Externa y Productividad, considerando aspectos personales

La diversidad de sujetos experimentales que participaron en nuestra investigación ha sido amplia. En función de su experiencia, hemos experimentado con desarrolladores desde novatos hasta seniors. La mayor parte de los sujetos no tenían experiencia en el uso de TDD. Otras aspectos personales pudieron influir en nuestros resultados, tales como el conocimiento de herramientas de desarrollo, la experiencia en el lenguaje de programación, y las tareas experimentales utilizadas. De hecho, la tarea experimental siempre arrojó resultados que mostraron una influencia estadísticamente significativa en las variables respuesta.

Encontramos en la literatura que los estudios de tipo experimental sobre TDD se enfrentan a estas mismas circunstancias y cuyos efectos han sido ampliamente discutidos como parte de las amenazas a la validez. En el meta-análisis de Rafique and Misic [15], por ejemplo, se advierte sobre la influencia de la complejidad de la tarea y su posible impacto negativo en la productividad. Altos niveles de complejidad podrían hacer más difícil la aplicación de TDD ya que implicaría un mayor esfuerzo. Así mismo los autores indican, en relación al tamaño de la tarea, que la evidencia empírica muestra que los efectos de TDD pueden observarse con el tiempo, en consecuencia los efectos son menores en los estudios en la academia que en general son de corta duración, en comparación con estudios en la industria.

Al considerar factores humanos concretos, la Calidad externa mejora independientemente de la técnica utilizada (TDD o ITLD) cuando los participantes indicaron tener conocimiento previo de herramientas de prueba y experiencia en programación Java. Al analizar la Productividad, obtuvimos resultados significativos para el aspecto personal Edad: los participantes mayores tienden a ser menos productivos que los jóvenes en experimentos con profesionales.

Para el caso de los estudiantes, el aspecto personal Edad no fue considerado debido a que la mayoría eran jóvenes universitarios con edades parecidas. Posteriormente, nos resultó imposible obtener la edad de los estudiantes para poder compararlos con los profesionales.

Nuestros resultados nos inducen a concluir también que tanto el tipo de reclutamiento como la edad, tienen efectos separados en la productividad. Algunos participantes profesionales no voluntarios (*conscripted*) obtuvieron mejor Productividad que otros participantes también profesionales que realizaron el experimento como un curso de entrenamiento (*Training course*).

Como un caso particular, ya que sólo fue posible realizar el análisis en un único experimento (UADY2015), el grado de estudios (*Seniority*) de los

estudiantes produjo resultados significativos para la Productividad. Tal y como se esperaba, los estudiantes de menor nivel fueron más productivos de los que se encontraban en un nivel superior. No obstante, esto puede explicarse en razón de que los grupos no estaban balanceados (la mayoría de estudiantes en este experimento fueron de 5 nivel representando aproximadamente el 50 % de alumnos, mientras que el 50 % restante estaban distribuidos en 4to, 6to y 7mo niveles, y un único estudiante estaba cursando una maestría).

En el análisis del estado de la cuestión, observamos que no existe literatura sustancial acerca del efecto de los factores personales en TDD. Pudimos identificar únicamente 39 de 114 estudios primarios, donde se consideraron factores personales clasificados como individuales, interpersonales y organizacionales (ver tabla 2.7). El aspecto mayormente analizado ha sido la experiencia ya sea profesional, en el lenguaje de programación o en la realización de tareas específicas. En general la literatura muestra resultados no concluyentes, por un lado existen estudios que muestran un impacto positivo de TDD al considerar algunos factores personales y otros donde no se observan diferencias significativas al compararse con técnicas tradicionales de desarrollo (ver tabla 2.6). No obstante si hemos podido matizar ciertas coincidencias como la efectividad del uso de TDD cuando aumentan las habilidades de pruebas de los sujetos, lo que se alinea al trabajo de Geras et al.[57]. Además pudimos comprobar que la estrategia de programación es estadísticamente significativa en estudios con estudiantes al igual que lo obtenido por Bissi et al.[21] . Por otra parte, nuestros estudios coinciden con[15],[16],[21] en que la productividad disminuye al aplicar TDD con profesionales.

Esperábamos realizar un análisis comparativo mucho más convincente entre nuestros resultados y los estudios previos, sin embargo, nos encontramos con la dificultad de la diversidad de métricas utilizadas para medir los efectos de TDD. La incompatibilidad en la forma de operacionalizar las variables respuesta entre los diferentes estudios, hizo infructuoso cualquier intento de comparación que nos pudimos plantear. Este problema ya ha sido reportado por algunos autores como[15],[19],[?],[16]. Por lo tanto, la heterogeneidad de estudios hace que los resultados no sean totalmente comparables y que la agregación de estudios sea engorrosa y produzca una amenaza a la validez[?].

Vale mencionar que la edad, y la motivación (en nuestros estudios considerada por el tipo de reclutamiento de los participantes), han sido factores humanos que nos arrojaron resultados que han merecido especial atención dada la significación estadística obtenida. No obstante, en lo que respecta a la edad, no logramos determinar ningún estudio que analice formalmente su relación con la Calidad y Productividad al aplicar TDD, lo cuál nos sorprende significativamente. Si bien es cierto, la motivación ha sido un tema abordado como un aspecto psicológico en estudios experimentales en Ingenie-

ría de Software, sin embargo tampoco fuimos capaces de encontrar estudios directamente relacionados con este aspecto y su influencia sobre la calidad y productividad con TDD, a excepción de estudios como el de [16], [63], [54] que apuntan a la motivación como un posible factor que podría influir en sus resultados.

9.1.4. Influencia de TDD en el Grado de Completitud, considerando aspectos personales

Creemos que uno de los principales aportes de nuestro estudio es haber comprobado cómo el interés de los participantes al realizar las tareas experimentales (a lo que denominamos *grado de completitud*) influye en su productividad.

Observamos que conforme los participantes tienen mayor Edad, el *grado de completitud* de sus tareas experimentales fue disminuyendo. Quizás este efecto podría explicarse por la falta de motivación. Los sujetos en este caso fueron profesionales que participaron como un curso de entrenamiento, y en ciertos casos no podían abandonar el experimento dado que tenían que realizar el entrenamiento dentro de su horario de trabajo.

Resulta intrigante el hecho de que, si bien la motivación pudo ser un factor relacionado con la entrega de tareas incompletas o sin realizar, no obstante cuando los sujetos aplicaron TDD, hicieron más intentos por entregar mejor la tarea. La explicación más razonable a este fenómeno es que la efectividad del curso de formación pudo haber incidido más fuertemente de lo que pensamos. Quizá la expectativa de aprender TDD por los profesionales incidió en que intentasen hacer algo más con TDD que con ITLD, aunque a efectos prácticos, el resultado en la productividad fue pobre. Como este comportamiento se reflejó sólo en el caso de los profesionales, sólo podemos concluir que a medida que los desarrolladores van avanzando en años, su interés por aprender se va diluyendo, y cuando algo no comprenden, el esfuerzo por conseguir el objetivo se desvanece. Respecto a los estudiantes, no disponemos de datos que nos permitan comparar su comportamiento con el de los profesionales.

Como corolario podemos indicar que el presente estudio es de carácter exploratorio, dado el escaso número de publicaciones encontradas en nuestra línea de investigación. Por lo tanto, no hemos pretendido establecer una relación causa-efecto sino más bien, despertar el interés por la realización de un mayor número de estudios empíricos sobre los factores humanos que podrían incidir en los resultados. Desde nuestra perspectiva, hemos llegado a obtener ciertos hallazgos interesantes, sin embargo, vale recalcar que el limitado número de sujetos experimentales que participaron en los 7 experimentos, hace que nuestros resultados deban tomarse con la debida cautela, considerando también las amenazas a la valides discutidas en el capítulo anterior.

9.1.5. Aspectos finales

Conviene indicar que nuestro estudio se restringe al ámbito exploratorio. Aunque hemos identificado ciertas relaciones entre variables, la evidencia empírica respecto a los factores personales y su influencia sobre TDD es todavía muy escasa. Consideramos que hace falta realizar más estudios que permitan confirmar, refutar y, sobre todo, ampliar, nuestros resultados.

9.2. Futuras líneas

La familia de de experimentos que hemos realizado nos han permitido introducir los factores humanos como una variable moderadora que influye en los resultados de las estrategias de programación TDD, y también ITLD. Como ejemplo, podemos citar la edad de los sujetos y la experiencia en el lenguaje de programación, en donde hemos obtenido resultados que han influído significativamente sobre la Calidad Externa y Productividad. A nuestro criterio, los aspectos humanos que han sido considerados aún se encuentran en una fase primaria de estudio, por lo que se plantean las siguientes futuras líneas de investigación:

- Replicar los experimentos considerando los factores humanos que hemos identificado en la investigación. En este caso, la selección de sujetos y asignación a grupos podría realizarse de forma segmentada (por valores/niveles de los factores humanos). Como consecuencia de este cambio, los factores humanos adquirirían la condición de factores, no covariables, y su análisis implicaría relaciones causales. Dicho de otra manera: la investigación pasaría de exploratoria a confirmatoria.

- Otro aspecto que debería estudiarse con mayor detalle es si la motivación de los desarrolladores tiene una correlación con la edad y, por tanto, con su interés en realizar las tareas experimentales adecuadamente. Como pudimos apreciar, los sujetos de mayor edad fueron los que menos se esforzaron por cumplir las tareas adecuadamente y esto influyó en los resultados obtenidos. Esta investigación no tendría una relación directa con TDD, pero si con la investigación experimental en Ingeniería de Software.

- Finalmente, otro aspecto que hemos encontrado en nuestra revisión de literatura, y que creemos puede guiar futuras líneas de investigación sobre el efecto de TDD en la calidad y productividad, es lo concerniente al mantenimiento del software. El mantenimiento apenas se menciona en las revisiones de literatura. De hecho, encontramos un único estudio secundario que analiza este aspecto ([19]). Dentro de sus resultados, los autores indican que TDD tiene un efecto positivo en la etapa de man-

tenimiento, pero su conclusión se basa en un único estudio primario encontrado ([162]).

Parte I

Apéndices

Apéndice A

SÍNTESIS DE RESULTADOS AGRUPANDO ESTUDIANTES Y PROFESIONALES

TBD

TBD

Resumen:

En este apéndice se incluye el meta-análisis de la influencia de los factores personales entre estudiantes y profesionales que complementa el análisis del Capítulo 6. Debido al elevado número de test estadísticos realizados, existe la posibilidad de cometer un error tipo-I. Realizaremos una síntesis de tipo *Individual Patient Data* (IPD) con descomposición en subgrupos. Para ello se ha utilizado[163], y los paquetes *ggplot2*[164], *multcomp*[165], *xtable*[166], *pspearman*[167], *corrplot*[168], *lmerTest*[?], *effectsize*[169] y *FSA*[?].

A.1. Meta-análisis de la influencia de los factores personales, considerando estudiantes y profesionales

En esta sección vamos a determinar la influencia de los factores personales en las variables respuesta Calidad y Productividad al aplicar TDD o

ITLD en base a los datos obtenidos de los experimentos realizados. En este caso, el análisis se realizará sobre los mismos aspectos determinados en el Capítulo 6, considerando estudiantes y profesionales por separado, esto es:

- la edad,
- la experiencia profesional,
- la experiencia en programación,
- el nivel y grado de educación,
- el conocimiento previo en pruebas unitarias,
- el conocimiento del entorno Eclipse,
- la experiencia en java,
- la experiencia en el uso de herramientas de prueba, y finalmente,
- la experiencia en el framework Junit.

Vale mencionar que algunos aspectos no han sido considerados en todos los experimentos. Por ejemplo la edad, es un factor que fue considerado para el caso de los experimentos con profesionales, ya que para el caso de los experimentos con estudiantes, no existían mayores diferencias entre grupos de participantes.

Para una mejor comprensión del lector, realizaremos un análisis completo para la variable *Entrenamiento previo en pruebas unitarias* (identificada como UTTraining), ya que reúne las características idóneas para ejemplificar adecuadamente el análisis realizado para el resto de variables. Posteriormente comentaremos únicamente aquellas variables que arrojen resultados significativos o de interés para los objetivos de la investigación. El modelo de análisis será de la forma siguiente:

```
lmer(QLTY ~ 1 +
            Provenance +
            STRATEGY*factor personal +
            TASK +
            (1 | subjectID),
     data = ...)

lmer(PROD ~ 1 +
            Provenance +
            STRATEGY*factor personal +
            TASK +
            (1 | subjectID),
     data = ...)
```

Listing A.1: Meta-análisis (profesionales y estudiantes) incluyendo los factores personales

A.1.1. Entrenamiento previo en pruebas unitarias (UTTTraining)

Comenzamos el meta-análisis verificando si los experimentos son homogéneos una vez que la variable demográfica `UTTraining` ha sido introducida en el modelo lineal general. El modelo de análisis puede adoptar varias formas. En nuestro caso hemos decidido que:

- El objetivo principal de esta tesis es verificar si ciertos factores personales influyen en la aplicación de la estrategia de programacion TDD. Por tanto, nos interesa especialmente analizar la interacción `STRATEGY : UTTraning`.
- No podemos descartar que el entrenamiento previo en pruebas unitarias tenga un impacto en la forma en que los individuos programan, independientemente de la estrategia (ITLD o TDD) utilizada. En consecuencia, creemos necesario introducir el factor principal `UTTraning` en el análisis.

El análisis estadístico utilizado es el mostrado en el Listado A.2, especificando el factor personal a analizar, en este caso `UTTraining + STRATEGY:UTTraining` (lo que en `R` se escribe como `STRATEGY*UTTraining`).

```
lmer(QLTY ~ 1 +              # también PROD
            Provenance +
            STRATEGY*UTTraining +
            TASK +
            (1 | subjectID),
     data = ...)
```

Listing A.2: Meta-análisis incluyendo los factores personales

La tabla A.1 indica que el factor `Provenance` es significativo tanto para `QLTY` como `PROD`, lo que implica que los experimentos son heterogéneos. La variable independiente que puede ser tomada como un factor entre-experimentos es la correspondiente al carácter profesional/estudiante de los sujetos, por lo que realizaremos análisis separados para profesionales y estudiantes.

La tabla A.2 muestra que el análisis de los profesionales sigue siendo heterogéneo. En consecuencia, transformamos el análisis de efectos fijos de la Tabla A.2 en un modelo de factores aleatorios, el cual se muestra en la Tabla A.3.

El modelo de factores aleatorios de la Tabla A.3 muestra que, para los profesionales, ninguno de los factores (con excepción de la tarea, como ha ocurrido con frecuencia a lo largo de la síntesis) arroja resultados estadísticamente significativos. En particular, no existe ninguna relación entre la estrategia de programación (TDD o ITLD) y el entrenamiento previo que

Tabla A.1: Meta-análisis de los experimentos, incluyendo a todos los sujetos. Los factores SLICING y GROUP no se han considerado. Se incluye el término `STRATEGY * UTTraining`

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	24726.18	4121.03	6	184.20	5.37	0.000
STRATEGY	1754.45	877.23	2	204.85	1.14	0.321
UTTraining	552.17	552.17	1	255.16	0.72	0.397
TASK	88824.02	44412.01	2	199.58	57.90	0.000
STRATEGY:UTTraining	10.55	5.27	2	209.93	0.01	0.993

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	20396.74	3399.46	6	175.60	9.34	0.000
STRATEGY	856.12	428.06	2	186.06	1.18	0.311
UTTraining	128.81	128.81	1	252.22	0.35	0.552
TASK	73346.07	36673.03	2	179.30	100.81	0.000
STRATEGY:UTTraining	309.59	154.79	2	192.34	0.43	0.654

Tabla A.2: Meta-análisis de efectos fijos, incluyendo sólo a los sujetos profesionales (siguiendo el modelo del Listado **??**. Los factores SLICING y GROUP no se han considerado. Se incluye el término `STRATEGY * UTTraining`

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	4155.26	2077.63	2	69.82	3.28	0.043
STRATEGY	1654.87	827.44	2	66.93	1.31	0.277
UTTraining	818.16	818.16	1	81.62	1.29	0.259
TASK	49773.27	24886.63	2	65.78	39.31	0.000
STRATEGY:UTTraining	12.93	6.47	2	68.98	0.01	0.990

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	3304.31	1652.16	2	72.18	5.05	0.009
STRATEGY	286.24	143.12	2	68.92	0.44	0.647
UTTraining	10.05	10.05	1	83.48	0.03	0.861
TASK	19482.83	9741.41	2	67.69	29.78	0.000
STRATEGY:UTTraining	618.33	309.16	2	70.90	0.95	0.393

Tabla A.3: Meta-análisis de efectos aleatorios, incluyendo sólo a los sujetos profesionales (siguiendo el modelo del Listado ??. Los factores SLICING y GROUP no se han considerado. Se incluye el término `STRATEGY * UTTraining`

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1412.20	706.10	2	67.64	1.13	0.328
UTTraining	909.97	909.97	1	81.83	1.46	0.230
TASK	49038.33	24519.16	2	67.22	39.41	0.000
STRATEGY:UTTraining	10.47	5.23	2	69.83	0.01	0.992

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	323.03	161.52	2	69.66	0.50	0.612
UTTraining	8.04	8.04	1	84.39	0.02	0.876
TASK	19554.63	9777.32	2	68.98	29.98	0.000
STRATEGY:UTTraining	626.21	313.10	2	71.15	0.96	0.388

indicaron tener los sujetos en el desarrollo de pruebas unitarias. Nótese que la variable `UTTrainig` analizada de forma individual tampoco influye significativamente en la calidad y productividad.

De manera análoga que para el caso de los profesionales, realizamos el meta-análisis de todos los experimentos para el grupo de estudiantes. La tabla A.4 muestra que el meta-análisis de efectos fijos es heterogéneo, por lo que procedemos a utilizar un modelo de factores aleatorios que se presenta en la tabla A.5.

La interacción entre la estrategia de programación y el entrenamiento previo en desarrollo de pruebas unitarias tampoco es significativa para el caso de los estudiantes. Han resultado nuevamente significativos los factores (STRATEGY y SLICING) de forma independiente. Esto último era esperable, ya que la introducción de nuevas variables en el modelo lineal no acostumbra a cambiar la significación de factores introducidos previamente.

En resumen, en esta sección hemos presentado un meta-análisis de la influencia del entrenamiento previo en pruebas unitarias, considerando en primer lugar tanto a estudiantes como a profesionales. Como el factor `Provenance` resulta significativo, nos hemos decantado por aplicar el meta-análisis dividiendo a profesionales y estudiantes. Como el factor `Provenance` ha resultado de nuevo significativo, aplicamos un modelo efectos aleatorios. En ninguno de los casos `UTTraining` ni `STRATEGY : UTTraining` alcanzan la significación estadística.

Aplicaremos este mismo procedimiento al resto de factores personales de interés. Sin embargo, mostraremos únicamente las tablas finales y no todas las tablas intermedias, lo que redundará en una mejora de la concisión y

Tabla A.4: Meta-análisis de efectos fijos, incluyendo sólo a los sujetos estudiantes (siguiendo el modelo del Listado **??**. Se incluye el término `STRATEGY * UTTraining`

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	16913.70	5637.90	3	180.00	6.81	0.000
STRATEGY	312.49	312.49	1	180.00	0.38	0.540
UTTraining	30.98	30.98	1	180.00	0.04	0.847
TASK	37439.00	37439.00	1	180.00	45.21	0.000
GROUP	382.92	382.92	1	180.00	0.46	0.497
SLICING	64.10	64.10	1	180.00	0.08	0.781
STRATEGY:UTTraining	202.92	202.92	1	180.00	0.25	0.621

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	9977.05	3325.68	3	103.86	9.39	0.000
STRATEGY	1406.38	1406.38	1	86.48	3.97	0.049
UTTraining	280.61	280.61	1	88.01	0.79	0.376
TASK	55217.70	55217.70	1	92.33	155.85	0.000
GROUP	62.63	62.63	1	93.24	0.18	0.675
SLICING	1458.11	1458.11	1	108.95	4.12	0.045
STRATEGY:UTTraining	230.51	230.51	1	86.85	0.65	0.422

Tabla A.5: Meta-análisis de efectos aleatorios, incluyendo sólo a los sujetos estudiantes (siguiendo el modelo del Listado **??**. Se incluye el término `STRATEGY * UTTraining`

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	314.96	314.96	1	179.85	0.38	0.538
UTTraining	4.32	4.32	1	181.38	0.01	0.943
TASK	37179.96	37179.96	1	180.01	44.87	0.000
GROUP	399.48	399.48	1	180.18	0.48	0.488
SLICING	67.77	67.77	1	179.87	0.08	0.775
STRATEGY:UTTraining	206.44	206.44	1	179.87	0.25	0.618

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1401.98	1401.98	1	84.36	3.94	0.050
UTTraining	233.62	233.62	1	89.34	0.66	0.420
TASK	54995.97	54995.97	1	90.16	154.73	0.000
GROUP	27.68	27.68	1	95.63	0.08	0.781
SLICING	1411.41	1411.41	1	107.17	3.97	0.049
STRATEGY:UTTraining	227.85	227.85	1	84.72	0.64	0.426

simplicidad de esta sección.

A.1.2. Conocimiento del entorno de desarrollo Eclipse (knowEclipse)

El modelo de efectos aleatorios indica que el conocimiento del entorno de desarrollo Eclipse no resulta influyente en el caso de los profesionales, tanto para QLTY como para PROD. Sin embargo, en el caso de los estudiantes si se obtiene un resultado significativo. La Tabla A.6a muestra una influencia estadísticamente significativa del conocimiento acerca del IDE Eclipse en la Calidad (los restantes factores significativos ya han sido discutidos anteriormente).

Tabla A.6: Meta-análisis de efectos aleatorios, incluyendo sólo a los sujetos estudiantes (siguiendo el modelo del Listado **??**). Se incluye el término `STRATEGY * knowEclipse`

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	170.73	170.73	1	179.78	0.21	0.644
knowEclipse	4798.02	4798.02	1	181.32	6.01	0.015
TASK	39796.99	39796.99	1	179.81	49.88	0.000
SLICING	169.53	169.53	1	179.72	0.21	0.645
GROUP	90.79	90.79	1	180.17	0.11	0.736
STRATEGY:knowEclipse	208.61	208.61	1	179.71	0.26	0.610

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	2169.74	2169.74	1	90.17	6.24	0.014
knowEclipse	0.67	0.67	1	98.27	0.00	0.965
TASK	56605.92	56605.92	1	90.43	162.86	0.000
SLICING	1311.64	1311.64	1	106.94	3.77	0.055
GROUP	61.04	61.04	1	96.14	0.18	0.676
STRATEGY:knowEclipse	1101.45	1101.45	1	90.50	3.17	0.078

Tabla A.7: Post-hoc análisis para el factor `knowEclipse` (estudiantes)

(a) Calidad

	Estimate	Std. Error	z value	Pr(>|z|)
Yes - No	-8.87	5.93	-1.50	0.135

(b) Productividad

	Estimate	Std. Error	z value	Pr(>|z|)
Yes - No	4.81	5.26	0.91	0.361

El análisis post-hoc para el factor demográfico `knowEclipse` se muestra en la Tabla A.7. La diferencia es pequeña (8.87 puntos), pero **no es** estadísticamente significativa. Aunque parezca una contradicción, es perfectamente

posible que el modelo lineal y los tests post-hoc no coincidan. En este caso, hay que atender a lo que indique el test post-hoc. La diferencia en calidad entre aquellos que conocen Eclipse y aquellos que no es de 8.87 puntos (curiosamente, aquellos que no conocen eclipse obtienen resultados de mayor calidad. En términos de tamaño de efecto, la diferencia es $Hedges'g = 0{,}27$) (pequeño).

A.1.3. Experiencia en programación (programmingExperience)

Los análisis realizados no arrojan ningún resultado estadísticamente significativo, más allá de los ya reportados en secciones anteriores.

A.1.4. Edad de los sujetos participantes (age)

Tabla A.8: Meta-análisis de los experimentos, incluyendo sólo a los sujetos profesionales. El factor slicing no se ha considerado. Se analiza la interacción STRATEGY * age

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3828.19	1914.10	2	80.57	3.13	0.049
age	1606.28	1606.28	1	58.90	2.63	0.111
TASK	47173.14	23586.57	2	63.94	38.55	0.000
GROUP	7643.69	1910.92	4	2.00	3.12	0.257
STRATEGY:age	2552.55	1276.28	2	80.05	2.09	0.131

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	521.07	260.54	2	82.03	0.78	0.461
age	1271.61	1271.61	1	63.88	3.82	0.055
TASK	18699.90	9349.95	2	66.91	28.05	0.000
GROUP	778.98	194.75	4	2.00	0.58	0.710
STRATEGY:age	615.17	307.59	2	81.52	0.92	0.401

En el caso de los profesionales, la edad ejerce una influencia que se aproxima (aunque no alcanza) al nivel de significación del 5 % para la Productividad. La tabla A.8 muestra que, en el caso de la Productividad, el efecto es -0.06, lo que corresponde a un tamaño muy pequeño.

Para el caso de los estudiantes, la mayoría de los sujetos participantes fueron jóvenes de edades similares. La edad, por lo tanto, es un factor que no ha sido considerado en el meta-análisis.

A.1.5. Años de experiencia profesional (experienceYears)

Tabla A.9: Meta-análisis de los experimentos, incluyendo sólo a los sujetos profesionales. El factor slicing no se ha considerado. Se analiza la interacción STRATEGY * experienceYears

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1615.19	807.59	2	66.80	1.21	0.305
experienceYears	1580.22	1580.22	1	46.01	2.36	0.131
TASK	47312.06	23656.03	2	61.61	35.40	0.000
GROUP	8662.91	2165.73	4	2.00	3.24	0.249
STRATEGY:experienceYears	23.95	11.97	2	67.76	0.02	0.982

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	24.08	12.04	2	72.60	0.04	0.965
experienceYears	314.40	314.40	1	57.79	0.93	0.339
TASK	18596.89	9298.45	2	67.62	27.52	0.000
GROUP	708.34	177.08	4	2.00	0.52	0.738
STRATEGY:experienceYears	247.09	123.55	2	73.09	0.37	0.695

Los años de experiencia de los profesionales no influye significativamente ni en la Calidad ni en la Productividad.

El análisis de la influencia de la experiencia profesional en el caso de estudiantes es siempre controvertido. Los estudiantes no acostumbran a tener una experiencia profesional sustancial. Sin embargo, cada día es más frecuente que los estudiantes trabajen en paralelo con sus estudios, por lo que no es extraño encontrar alumnos con una experiencia significativa en los cursos superiores.

En este caso, hemos optado por realizar el análisis. Su resultado ha sido estadísticamente no significativos, esto es, la experiencia de los estudiantes no influye ni en la Calidad ni en la Productividad.

A.1.6. Grado académico (Seniority)

La variable *Seniority* hace referencia al nivel de académico de los estudiantes, por lo que no se considera a los profesionales. Para los estudiantes, el meta-análisis basado en el modelo de efectos aleatorios se presenta en la tabla A.10. Se aprecia que el factor *Seniority* se aproxima al nivel de significación del 5 % para la Productividad, lo que corresponde a un tamaño de efecto de -0.17 (muy pequeño). En los demás casos, los resultados no fueron estadísticamente significativos.

Tabla A.10: Meta-análisis de los experimentos, incluyendo sólo a los sujetos estudiantes. El factor slicing no se ha considerado. Se analiza la interacción STRATEGY * Seniority

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	70.93	70.93	1	180.03	0.09	0.768
Seniority	917.67	917.67	1	29.09	1.13	0.297
TASK	38953.79	38953.79	1	180.09	47.87	0.000
GROUP	100.93	100.93	1	183.36	0.12	0.725
STRATEGY:Seniority	123.18	123.18	1	180.03	0.15	0.698

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1310.28	1310.28	1	91.59	3.49	0.065
Seniority	1487.36	1487.36	1	58.29	3.96	0.051
TASK	55996.39	55996.39	1	91.40	148.98	0.000
GROUP	327.16	327.16	1	97.01	0.87	0.353
STRATEGY:Seniority	608.30	608.30	1	90.54	1.62	0.207

A.1.7. Nivel de educación (educationLevel)

Tabla A.11: Meta-análisis de los experimentos, incluyendo sólo a los sujetos profesionales. El factor slicing no se ha considerado. Se analiza la interacción STRATEGY * educationLevel

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	2057.37	1028.69	2	47.04	1.69	0.195
STRATEGY	3724.41	1862.21	2	55.88	3.07	0.054
educationLevel	456.78	228.39	2	42.54	0.38	0.689
TASK	42621.64	21310.82	2	59.60	35.10	0.000
GROUP	6800.08	2266.69	3	43.21	3.73	0.018
STRATEGY:educationLevel	3581.93	1193.98	3	58.73	1.97	0.129

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
Provenance	444.81	222.41	2	57.61	0.68	0.509
STRATEGY	14.21	7.10	2	61.36	0.02	0.978
educationLevel	96.54	48.27	2	54.70	0.15	0.862
TASK	17535.17	8767.59	2	65.22	26.95	0.000
GROUP	593.70	197.90	3	55.34	0.61	0.612
STRATEGY:educationLevel	1226.89	408.96	3	63.87	1.26	0.297

El nivel de educación hace referencia a los estudios de los sujetos, esto es, a si han obtenido una titulación de grado, maestría o doctorado/tercer nivel.

Nótese que, en nuestro caso, hemos realizado experimentos con estudiantes de los tres tipos, por lo que el nivel de educación puede estudiarse tanto a nivel de profesionales como estudiantes.

La tabla A.11 muestra el análisis de efectos fijos para los profesionales (no ha sido necesario realizar un análisis de efectos aleatorios ya que los experimentos resultan homogéneos, tal y como indica el factor `Provenance`). A nivel de la PROD, los resultados son non significativos en todos los casos. Sin embargo, en lo que respecta a la QLTY, la situación cambia.

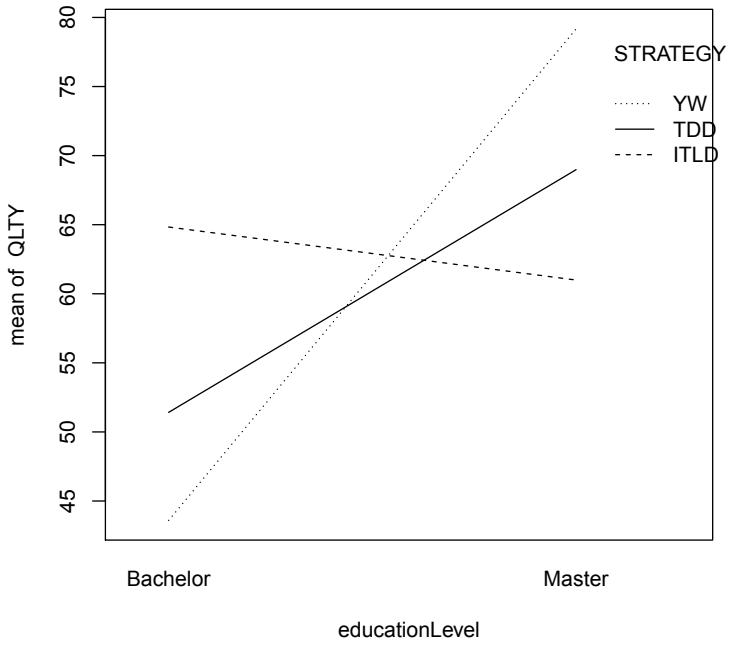


Figura A.1: Interacción entre el nivel de educación y la estrategia de programación (sólo profesionales)

El factor personal `educationLevel` produce una variación sustancial entre-grupos, lo que se refleja en un p-valor casi significativo. Es probable que, de contar con más sujetos, el resultado hubiera sido estadísticamente significativo. El gráfico de perfil mostrado en la Figura A.1 indica que a mayor nivel de educación, mayor efectividad de las estrategias YW y TDD, mientras que ITLD disminuye.

Pero lo relevante es que `STRATEGY` casi alcanza la significación de 5 %, esto es, la estrategia de programación, corregida por `educationLevel`, es a efectos prácticos estadísticamente significativa. Desafortunadamente, el efecto no hace referencia a TDD. Tal y como podemos observar en la Tabla

	Estimate	Std. Error	z value	Pr(>|z|)
ITLD - YW	76.17	38.12	2.00	0.105
TDD - YW	29.17	21.41	1.36	0.346
TDD - ITLD	-47.00	32.23	-1.46	0.297

Tabla A.12: Post-hoc análisis para el factor `STRATEGY` (sólo profesionales y variable respuesta QLTY)

A.12, las diferencias sustanciales se producen entre YW e ITLD, exhibiendo un tamaño de efecto grande (h = 2.17).

Para el caso de los estudiantes, el nivel de educación no produce efectos significativos ni en la Calidad ni en la Productividad. Por brevedad, no hemos incluido el análisis estadístico correspondiente.

A.1.8. Uso de herramientas de prueba (testingToolUsage)

Para el caso de los profesionales, el efecto del uso de herramientas de prueba sobre QLTY y PROD no produce efectos significativos.

Tabla A.13: Meta-análisis de los experimentos, incluyendo sólo a los sujetos estudiantes. El factor slicing no se ha considerado. Se analiza la interacción STRATEGY * testingToolUsage

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	40.59	40.59	1	181.06	0.05	0.824
testingToolUsage	72.48	72.48	1	161.35	0.09	0.767
TASK	38264.11	38264.11	1	181.09	46.63	0.000
GROUP	379.91	379.91	1	181.24	0.46	0.497
STRATEGY:testingToolUsage	733.53	733.53	1	180.94	0.89	0.346

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	2259.78	2259.78	1	92.65	5.99	0.016
testingToolUsage	112.99	112.99	1	96.10	0.30	0.585
TASK	55299.62	55299.62	1	92.05	146.57	0.000
GROUP	51.41	51.41	1	96.08	0.14	0.713
STRATEGY:testingToolUsage	47.32	47.32	1	92.42	0.13	0.724

Para el caso de los estudiantes, el uso de herramientas de prueba produce resultados significativos en el factor `STRATEGY` en la variable respuesta `PROD`, tal y como se observa en la tabla A.13.

El análisis post-hoc se muestra en la tabla A.14. Se observa que los estudiantes que usan TDD son más productivos que los que usan ITLD. El tamaño de efecto es medio (h = 0.55).

	Estimate	Std. Error	z value	Pr(>|z|)
TDD - ITLD	8.18	3.80	2.15	0.031

Tabla A.14: Post-hoc análisis para el factor `STRATEGY` (sólo estudiantes y variable respuesta PROD)

A.1.9. Experiencia en el uso del Framework JUnit (jUnitExperience)

Tabla A.15: Meta-análisis de los experimentos, incluyendo sólo a los sujetos profesionales. El factor slicing no se ha considerado. Se analiza la interacción STRATEGY * jUnitExperience

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	3975.36	1987.68	2	64.53	3.10	0.052
jUnitExperience	32.31	32.31	1	77.87	0.05	0.823
TASK	40740.31	20370.16	2	65.80	31.74	0.000
GROUP	7573.63	1893.41	4	2.00	2.95	0.269
STRATEGY:jUnitExperience	758.67	379.34	2	67.99	0.59	0.557

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	422.83	211.41	2	67.91	0.62	0.539
jUnitExperience	1.90	1.90	1	82.01	0.01	0.940
TASK	16521.75	8260.88	2	68.69	24.36	0.000
GROUP	811.54	202.88	4	2.00	0.60	0.703
STRATEGY:jUnitExperience	117.34	58.67	2	70.22	0.17	0.841

La tabla A.15 muestra el Meta-análisis con un modelo de efectos aleatorios sobre los efectos de la experiencia de los profesionales en el uso del framework JUnit. El factor *Strategy* muestra resultados significativos para la variable QLTY, sin embargo la interacción *Strategy:jUnitExperience*, no es significativa tanto para QLTY como para PROD.

La tabla A.16 muestra el Meta-análisis con un modelo de efectos aleatorios sobre la experiencia de los estudiantes en el uso del framework JUnit. Al igual que para el caso de los profesionales, el factor *Strategy* muestra resultados significativos para la variable QLTY, sin embargo la interacción *Strategy:jUnitExperience* no es significativa tanto para QLTY como para PROD.

Tabla A.16: Meta-análisis de los experimentos, incluyendo sólo a los sujetos estudiantes. El factor slicing no se ha considerado. Se analiza la interacción STRATEGY * jUnitExperience

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	222.06	222.06	1	180.98	0.27	0.604
jUnitExperience	5.40	5.40	1	183.96	0.01	0.936
TASK	36614.22	36614.22	1	181.00	44.45	0.000
GROUP	390.11	390.11	1	181.38	0.47	0.492
STRATEGY:jUnitExperience	316.11	316.11	1	180.92	0.38	0.536

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	1845.84	1845.84	1	91.26	4.96	0.028
jUnitExperience	510.10	510.10	1	99.38	1.37	0.244
TASK	52033.63	52033.63	1	92.14	139.88	0.000
GROUP	5.70	5.70	1	96.57	0.02	0.902
STRATEGY:jUnitExperience	224.95	224.95	1	94.20	0.60	0.439

A.1.10. Experiencia en programación Java (javaExperience)

Tabla A.17: Meta-análisis de los experimentos, incluyendo sólo a los sujetos profesionales. El factor slicing no se ha considerado. Se analiza la interacción STRATEGY * javaExperience

(a) Calidad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	4265.21	2132.60	2	67.07	3.28	0.044
javaExperience	2527.96	2527.96	1	86.60	3.89	0.052
TASK	48108.80	24054.40	2	61.85	37.00	0.000
GROUP	8726.56	2181.64	4	2.00	3.36	0.243
STRATEGY:javaExperience	1151.50	575.75	2	64.01	0.89	0.417

(b) Productividad

	Sum Sq	Mean Sq	NumDF	DenDF	F value	Pr(>F)
STRATEGY	327.52	163.76	2	71.92	0.49	0.615
javaExperience	713.78	713.78	1	89.94	2.14	0.147
TASK	18561.69	9280.85	2	66.91	27.77	0.000
GROUP	722.10	180.52	4	2.00	0.54	0.730
STRATEGY:javaExperience	316.69	158.34	2	69.85	0.47	0.625

El Meta-análisis de efectos aleatorios muestra que la experiencia de los profesionales en el lenguaje de programación Java, produce efectos significativos sobre la Calidad (que corresponde a un tamaño de efecto pequeño).

Para la Productividad los efectos no son significativos, como se muestra en la tabla: A.17. El factor *Strategy* también es significativo para el caso de la Calidad.

En el caso de los estudiantes, la experiencia en lenguaje de programación Java, no produce efectos significativos sobre la Calidad y Productividad al realizar un Meta-análisis aplicando un modelo de efectos aleatorios como se muestra en la tabla: A.17.

A.1.11. Resumen de los resultados obtenidos

En la tabla A.18, se muestra un resumen de los p-valores obtenidos al analizar los efectos de los factores personales sobre las variables respuesta Calidad y Productividad que se aproximan o que son estadísticamente significativos.

Tabla A.18: Factores personales que han resultado significativos o se aproximan al nivel de significación. Se muestran los p-valores

	PROFESIONALES		**ESTUDIANTES**	
FACTOR	**QLTY**	**PROD**	**QLTY**	**PROD**
UTTTraining				
Strategy				0.05
knowEclipse			**0.015**	
Strategy				0.014
Str:KnowEclipse				0.078
Age		**0.055**		
Strategy	0.049			
Seniority				**0.051**
EducationLevel				
Strategy	0.054			
TestingToolUsage				
Strategy				0.016
jUnitExperience				
Strategy	0.052			0.028
javaExperience	**0.052**			
Strategy	0.044			

Para el caso de los profesionales, la experiencia en el lenguaje Java es un factor personal que influye significativamente sobre la calidad del código desarrollado, con un tamaño de efecto de **0.48**. Para el caso de la Productividad, se observa que la edad se aproxima al nivel de significación y tiene un tamaño de efecto de **-0.06**. La variable *Strategy*, también es significativa en el caso de la Calidad para las variables *Age, JavaExperience* y se aproximan al nivel de significación en el caso de *EducationLevel,jUnitExperience*.

En lo que respecta a los estudiantes, el conocimiento del entorno de desarrollo Eclipse resulta significativo para la Calidad, con un tamaño de efecto de **-0.27**. La variable *Seniority* también se aproxima al nivel de significación, aunque es el resultado de un único experimento. El factor *Strategy*, es significativo para *UTTTraining, KnowEclipse, JunitExpeerience* y se aproxima al nivel de significación para la interacción *Str:KnowEclipse* en cuanto a la Productividad.

Bibliografía

Biblio.

Miguel de Cervantes Saavedra.

[1] L. Pizadeh, "Human factors in software development: A systematic literature review," Master's thesis, Department of Computer Science and Engineering, Chalmers University of Technology, Sep. 2010.

[2] B. Cartaxo, A. AraÃºjo, A. S. Barreto, and S. Soares, "The impact of scrum on customer satisfaction: An empirical study," in *2013 27th Brazilian Symposium on Software Engineering*, 2013, pp. 129–136.

[3] K. Beck, *Test Driven Development. By Example (Addison-Wesley Signature)*. Addison-Wesley Longman, Amsterdam, 2002.

[4] CollabNet VersionOne. (2020) 14^{th} annual state of agile^{TM} report. [Online]. Available: https://explore.digital.ai/state-of-agile/14th-annual-state-of-agile-report

[5] K. Beck and C. Andres, *Extreme Programming Explained: Embrace Change*, ser. XP Series. Pearson Education, 2004.

[6] K. Beck, "Aim, fire," *IEEE Software*, vol. 18, pp. 87–89, 09 2001. [Online]. Available: doi.ieeecomputersociety.org/10.1109/52.951502

[7] B. George and L. Williams, "A structured experiment of Test-Driven Development," *Information and Software Technology*, vol. 46, no. 5, pp. 337 – 342, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584903002040

[8] A. Gupta and P. Jalote, "An experimental evaluation of the effectiveness and efficiency of the Test Driven Development," in *First International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM 2007, Sept 2007, pp. 285–294.

[9] M. M. Muller and O. Hagner, "Experiment about Test-First programming," *IEE Proceedings - Software*, vol. 149, no. 5, pp. 131–136, 2002.

[10] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. A. Visaggio, "Evaluating advantages of Test Driven Development: A controlled experiment with professionals," in *Proceedings of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering*, ser. ISESE '06. ACM, 2006, pp. 364–371. [Online]. Available: http://doi.acm.org/10.1145/1159733.1159788

[11] A. Höfer and M. Philipp, "An empirical study on the tdd conformance of novice and expert pair programmers," in *Agile Processes in Software Engineering and Extreme Programming*, P. Abrahamsson, M. Marchesi, and F. Maurer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 33–42.

[12] L. Huang and M. Holcombe, "Empirical investigation towards the effectiveness of test first programming," *Information and Software Technology*, vol. 51, no. 1, pp. 182 – 194, 2009, special Section - Most Cited Articles in 2002 and Regular Research Papers.

[13] L. Madeyski, "The impact of Pair Programming and Test-Driven Development on package dependencies in Object-Oriented Design — an experiment," in *Product-Focused Software Process Improvement: 7th International Conference, PROFES 2006, Amsterdam, The Netherlands, June 12-14, 2006. Proceedings.* Springer Berlin Heidelberg, 2006, pp. 278–289. [Online]. Available: http://dx.doi.org/10.1007/11767718_24

[14] D. Fucci, B. Turhan, N. Juristo, O. Dieste, A. Tosun-Misirli, and M. Oivo, "Towards an operationalization of test-driven development skills: An industrial empirical study," *Information and Software Technology*, vol. 68, pp. 82–97, 2015.

[15] Y. Rafique and V. B. Mišić, "The effects of test-driven development on external quality and productivity: A meta-analysis," *IEEE Transactions on Software Engineering*, vol. 39, no. 6, pp. 835–856, 2012.

[16] B. Turhan, L. Layman, M. Diep, F. Shull, and H. Erdogmus, *How Effective is Test Driven Development.* O Reilly Media, 10 2010, pp. 45–55.

[17] S. Kollanus, "Test-driven development - still a promising approach?" in *2010 Seventh International Conference on the Quality of Information and Communications Technology*, Sep. 2010, pp. 403–408.

[18] A. Causevic, D. Sundmark, and S. Punnekkat, "Factors limiting industrial adoption of test driven development: A systematic review," in *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*, March 2011, pp. 337–346.

[19] S. Mäkinen and J. Münch, "Effects of test-driven development: A comparative analysis of empirical studies," in *Software Quality. Model-Based Approaches for Advanced Software and Systems Engineering*, D. Winkler, S. Biffl, and J. Bergsmann, Eds. Cham: Springer International Publishing, 2014, pp. 155–169.

[20] H. Munir, M. Moayyed, and K. Petersen, "Considering rigor and relevance when evaluating test driven development: A systematic review," *Information and Software Technology*, vol. 56, no. 4, pp. 375 – 394, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584914000135

[21] W. Bissi, A. G. S. S. Neto, and M. C. F. P. Emer, "The effects of test driven development on internal quality, external quality and productivity: A systematic review," *Information and Software Technology*, vol. 74, pp. 45 – 54, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584916300222

[22] H. Munir, M. Moayyed, and K. Petersen, "Considering rigor and relevance when evaluating test driven development: A systematic review," *Information and Software Technology*, vol. 56, no. 4, p. 375, 2014.

[23] I. Karac and B. Turhan, "What do we (really) know about test-driven development?" *IEEE Software*, vol. 35, no. 4, pp. 81–85, July 2018.

[24] L. McLeod and S. MacDonell, "Factors that affect software systems development project outcomes: A survey of research," *ACM Comput. Surv.*, vol. 43, p. 24, 10 2011.

[25] F. Anwar, R. Razali, and K. Ahmad, "Achieving effective communication during requirements elicitation-a conceptual framework," in *International Conference on Software Engineering and Computer Systems*. Springer, 2011, pp. 600–610.

[26] S. Chakraborty, S. Sarker, and J. Valacich, "Understanding analyst effectiveness in requirements elicitation: A gestalt fit perspective." 01 2007, pp. 771–782.

[27] S. Acuna, M. Pérez, J. Hannay, N. Juristo, and D. Pfahl, "Are team personality and climate related to satisfaction and software quality? aggregating results from a twice replicated experiment," *Information and Software Technology*, vol. 57, p. 141, 01 2015.

[28] C. Amrit, M. Daneva, and D. Damian, "Human factors in software development: On its underlying theories and the value of learning from related disciplines; a guest editorial introduction to the special issue," *Information and Software Technology*, 09 2014.

[29] G. Matturro, "Soft skills in software engineering: A study of its demand by software companies in uruguay," 05 2013, pp. 133–136.

[30] N. Juristo and A. M. Moreno, *Basics of Software Engineering Experimentation*, 1st ed. Springer Publishing Company, Incorporated, 2010.

[31] V. Basili, F. Shull, and F. Lanubile, "Building knowledge through families of experiments," *Software Engineering, IEEE Transactions on*, vol. 25, pp. 456 – 473, 08 1999.

[32] N. Juristo, "Experiences conducting experiments in industry: The eseil fidipro project," in *2016 IEEE/ACM 4th International Workshop on Conducting Empirical Studies in Industry (CESI)*, 2016, pp. 1–3.

[33] G. V. Glass, "Primary, secondary, and meta-analysis of research," *Educational Researcher*, vol. 5, no. 10, pp. 3–8, 1976. [Online]. Available: http://www.jstor.org/stable/1174772

[34] B. A. Kitchenham, D. Budgen, and P. Brereton, *Evidence-Based Software Engineering and Systematic Reviews*, R. LeBlanc, Ed. Taylor & Francis Group, 2015.

[35] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Information and Software Technology*, vol. 64, pp. 1 – 18, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584915000646

[36] O. Dieste and A. G. Padua, "Developing search strategies for detecting relevant experiments for systematic reviews," in *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)*, Sep. 2007, pp. 215–224.

[37] C. E. Anchundia and E. R. Fonseca C., "Resources for reproducibility of experiments in empirical software engineering: Topics derived from a secondary study," *IEEE Access*, vol. 8, pp. 8992–9004, 2020.

[38] M. Ghafari, T. Gross, D. Fucci, and M. Felderer, "Why research on test-driven development is inconclusive?" in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, ser. ESEM 20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3382494.3410687

[39] B. M. Napoleao, K. R. Felizardo, E. F. de Souza, F. Petrillo, N. L. Vijaykumar, E. Y. Nakagawa, and S. Halle, "Establishing a search string to detect secondary studies in software engineering," 2019.

[40] T. Bhat and N. Nagappan, "Evaluating the efficacy of test-driven development: industrial case studies," in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical Software Engineering*. ACM, 2006, pp. 356–363.

[41] L.-O. Damm and L. Lundberg, "Results from introducing component-level test automation and Test-Driven Development," *Journal of Systems and Software*, vol. 79, no. 7, pp. 1001 – 1014, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121205001573

[42] ——, "Quality impact of introducing component-level test automation and Test-Driven Development," in *Proceedings in Software Process Improvement: 14th European Conference, EuroSPI 2007, Potsdam, Germany, September 26-28*. Springer, 2007, pp. 187–199. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-75381-0_17

[43] C. Desai, D. S. Janzen, and J. Clements, "Implications of integrating Test-driven Development into CS1/CS2 curricula," in *Proceedings of the 40th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '09. ACM, 2009, pp. 148–152. [Online]. Available: http://doi.acm.org/10.1145/1508865.1508921

[44] S. H. Edwards, "Using Test-Driven Development in the classroom: Providing students with automatic, concrete feedback on performance," in *Proceedings of the International Conference on Education and Information Systems: Technologies and Applications EISTA*, vol. 3, 2003.

[45] H. Erdogmus, M. Morisio, and M. Torchiano, "On the effectiveness of the Test-First approach to programming," *IEEE Transactions on Software Engineering*, vol. 31, no. 3, pp. 226–237, 2005.

[46] D. Janzen and H. Saiedian, "Does Test-Driven Development really improve software design quality?" *IEEE Software*, vol. 25, no. 2, pp. 77–84, 2008.

[47] E. M. Maximilien and L. Williams, "Assessing test-driven development at ibm," in *Proceedings. 25th International Conference on Software Engineering, 2003*, May 2003, pp. 564–569.

[48] G. Melnik and F. Maurer, "A cross-program investigation of students'perceptions of agile methods," in *Proceedings 27th International Conference on Software Engineering, 2005. ICSE 2005.*, May 2005, pp. 481–488.

[49] N. Nagappan, E. M. Maximilien, T. Bhat, and L. Williams, "Realizing quality improvement through Test Driven Development:

results and experiences of four industrial teams," *Empirical Software Engineering*, vol. 13, no. 3, pp. 289–302, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10664-008-9062-z

[50] M. Pancur, M. Ciglaric, M. Trampus, and T. Vidmar, "Towards empirical evaluation of Test-Driven Development in a university environment," in *The IEEE Region 8 EUROCON 2003. Computer as a Tool*, vol. 2, Sept 2003, pp. 83–86 vol.2.

[51] J. C. Sanchez, L. Williams, and E. M. Maximilien, "On the sustained use of a Test-Driven Development practice at IBM," in *Agile Conference (AGILE), 2007*, Aug 2007, pp. 5–14.

[52] L. Williams, E. M. Maximilien, and M. Vouk, "Test-Driven Development as a defect-reduction practice," in *14th International Symposium on Software Reliability Engineering, ISSRE 2003.*, Nov 2003, pp. 34–45.

[53] R. A. Ynchausti, "Integrating unit testing into a software development team's process," in *XP 2001: Proceedings of the 2nd International Conference on Extreme Programming and Flexible Processes in Software Engineering, Sardinia, Italy*, 2001, pp. 84–87.

[54] T. Flohr and T. Schneider, "Lessons learned from an xp experiment with students: Test-first needs more teachings," in *Product-Focused Software Process Improvement*, J. Munch and M. Vierimaa, Eds. Berlin, HeidelbergV: Springer Berlin Heidelberg, 2006, pp. 305–318.

[55] B. George, *Analysis and Quantification of Test-Driven Development Approach.* Master's thesis. North Carolina State University, 2002.

[56] A. Geras, M. Smith, and J. Miller, "A prototype empirical evaluation of Test Driven Development," in *Proceedings. 10th International Symposium on Software Metrics, 2004.*, Sept 2004, pp. 405–416.

[57] A. M. Geras, "The effectiveness of test driven development," Master's thesis, Department of Electrical and Computer Engineering, University of Calgary, April 2004.

[58] D. S. Janzen, "An empirical evaluation of the impact of test-driven development on software quality," Ph.D. dissertation, University of Kansas, 1993.

[59] L. Madeyski, "Preliminary analysis of the effects of Pair Programming and Test-Driven Development on the external code quality," in *Proc. 2005 Conf. Softw. Eng.: Evol. Emerg. Technol.*, 2005, p. 113.

[60] L. Madeyski and Ł. Szała, "The impact of Test-Driven Development on software development productivity — an empirical study," in *Software Process Improvement: 14th European Conference, EuroSPI 2007, Potsdam, Germany, September 26-28, 2007. Proceedings.* Springer Berlin Heidelberg, 2007, pp. 200–211. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-75381-0_18

[61] M. Siniaalto and P. Abrahamsson, "Does Test-Driven Development improve the program code? alarming results from a comparative case study," in *Balancing Agility and Formalism in Software Engineering.* Springer Berlin Heidelberg, 2008, pp. 143–156. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-85279-7_12

[62] O. P. N. Slyngstad, J. Li, R. Conradi, H. Ronneberg, E. Landre, and H. Wesenberg, "The impact of Test Driven Development on the evolution of a reusable framework of components – An industrial case study," in *The Third International Conference on Software Engineering Advances, 2008. ICSEA '08.*, Oct 2008, pp. 214–223.

[63] J. H. Vu, N. Frojd, C. Shenkel-Therolf, and D. S. Janzen, "Evaluating test-driven development in an industry-sponsored capstone project," in *2009 Sixth International Conference on Information Technology: New Generations*, April 2009, pp. 229–234.

[64] S. Yenduri and L. A. Perkins, "Impact of using Test-Driven Development: A case study." in *Software Engineering Research and Practice*, 2006, pp. 126–129.

[65] L. Zhang, S. Akifuji, K. Kawai, and T. Morioka, "Comparison between Test Driven Development and Waterfall Development in a Small-Scale project," in *Extreme Programming and Agile Processes in Software Engineering: 7th International Conference, XP 2006, Oulu, Finland, June 17-22, 2006. Proceedings.* Springer Berlin Heidelberg, 2006, pp. 211–212. [Online]. Available: http://dx.doi.org/10.1007/11774129_29

[66] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. A. Visaggio, "Productivity of Test Driven Development: A controlled experiment with professionals," in *In Proceedings of 7th International Conference: Product-Focused Software Process Improvement, PROFES 2006, Amsterdam, The Netherlands, June 12-14, 2006.* Springer, 2006, pp. 383–388. [Online]. Available: http://dx.doi.org/10.1007/11767718_32

[67] S. H. Edwards, "Using software testing to move students from trial-and-error to reflection-in-action," *SIGCSE Bull.*, vol. 36, no. 1, pp. 26–30, 2004. [Online]. Available: http://doi.acm.org/10.1145/1028174.971312

[68] B. George and L. Williams, "An initial investigation of Test Driven Development in industry," in *Proceedings of the 2003 ACM Symposium on Applied Computing*, ser. SAC '03. ACM, 2003, pp. 1135–1139. [Online]. Available: http://doi.acm.org/10.1145/952532.952753

[69] D. S. Janzen and H. Saiedian, "On the influence of Test-Driven Development on software design," in *19th Conference on Software Engineering Education Training (CSEET'06)*, April 2006, pp. 141–148.

[70] D. S. Janzen, C. S. Turner, and H. Saiedian, "Empirical software engineering in industry short courses," in *20th Conference on Software Engineering Education Training (CSEET'07)*, July 2007, pp. 89–96.

[71] L. Madeyski, "The impact of test-first programming on branch coverage and mutation score indicator of unit tests: An experiment," *Inf. Softw. Technol.*, vol. 52, no. 2, pp. 169–184, Feb. 2010. [Online]. Available: https://doi.org/10.1016/j.infsof.2009.08.007

[72] A. Rendell, "Effective and pragmatic Test Driven Development," in *Agile Conference, 2008. AGILE '08*, Aug 2008, pp. 298–303.

[73] M. Siniaalto and P. Abrahamsson, "A comparative case study on the impact of Test-Driven Development on program design and test coverage," in *First International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM 2007, Sept 2007, pp. 275–284.

[74] S. Xu and T. Li, "Evaluation of Test-Driven Development: An academic case study," in *Software Engineering Research, Management and Applications 2009*. Berlin, Heidelberg: Springer, 2009, pp. 229–238. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-05441-9_20

[75] P. Abrahamsson, A. Hanhineva, and J. Jäälinoja, "Improving business agility through technical solutions: A case study on test-driven development in mobile software development," in *Business Agility and Information Technology Diffusion: IFIP TC8 WG 8.6 International Working Conference May 8–11, 2005, Atlanta, Georgia, U.S.A.* Springer US, 2005, pp. 227–243. [Online]. Available: http://dx.doi.org/10.1007/0-387-25590-7_14

[76] M. A. Domino, R. W. Collins, A. R. Hevner, and C. F. Cohen, "Conflict in collaborative software development," in *Proceedings of the 2003 SIGMIS Conference on Computer Personnel Research: Freedom in Philadelphia-Leveraging Differences and Diversity in the IT Workforce*, ser. SIGMIS CPR'03. New York, NY, USA: Association for Computing Machinery, 2003, pp. 44–51. [Online]. Available: https://doi.org/10.1145/761849.761856

[77] L. Huang, C. Thomson, and M. Holcombe, "How good are your testers? an assessment of testing ability," in *Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION (TAICPART-MUTATION 2007)*, Sep. 2007, pp. 82–88.

[78] S. Kollanus and V. Isomöttönen, "Understanding tdd in academic environment: Experiences from two experiments," in *Proceedings of the 8th International Conference on Computing Education Research*, ser. Koli'08. New York, NY, USA: Association for Computing Machinery, 2008, pp. 25–31. [Online]. Available: https://doi.org/10.1145/1595356.1595362

[79] N. F. LeJeune, "Teaching Software Engineering practices with Extreme Programming," *J. Comput. Sci. Coll.*, vol. 21, no. 3, pp. 107–117, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1089182.1089196

[80] L. Madeyski, "On the effects of Pair Programming on thoroughness and fault-finding effectiveness of unit tests," in *Product-Focused Software Process Improvement: 8th International Conference, PROFES 2007, Riga, Latvia, July 2-4, 2007. Proceedings*. Springer Berlin Heidelberg, 2007, pp. 207–221. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-73460-4_20

[81] ——, "Impact of Pair Programming on thoroughness and fault detection effectiveness of unit test suites," *Software Process: Improvement and Practice*, vol. 13, no. 3, pp. 281–295, 2008. [Online]. Available: http://dx.doi.org/10.1002/spip.382

[82] A. Marchenko, P. Abrahamsson, and T. Ihme, "Long-Term effects of Test-Driven Development a case study," in *Agile Processes in Software Engineering and Extreme Programming: 10th International Conference, XP 2009, Pula, Sardinia, Italy, May 25-29, 2009. Proceedings*. Springer Berlin Heidelberg, 2009, pp. 13–22. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-01853-4_4

[83] V. B. Mišić, "Perceptions of Extreme Programming: An exploratory study," *SIGSOFT Softw. Eng. Notes*, vol. 31, no. 2, pp. 1–8, 2006. [Online]. Available: http://doi.acm.org/10.1145/1118537.1118542

[84] M. M. Muller and A. Hofer, "The effect of experience on the test-driven development process," *Empirical Software Engineering*, vol. 12, no. 6, pp. 593–615, Dec 2007. [Online]. Available: https://doi.org/10.1007/s10664-007-9048-2

[85] P. Sfetsos, L. Angelis, and I. Stamelos, "Investigating the extreme programming system–an empirical study," *Empirical Software Engineering*, vol. 11, no. 2, pp. 269–301, Jun 2006. [Online]. Available: https://doi.org/10.1007/s10664-006-6404-6

[86] L. B. Sherrell and J. J. Robertson, "Pair programming and agile software development: Experiences in a college setting," *J. Comput. Sci. Coll.*, vol. 22, no. 2, pp. 145–153, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1181901.1181927

[87] H. Wasmus and H.-G. Gross, "Evaluation of Test-Driven Development: An industrial case study," in *Second International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE*, 2007.

[88] T. Dogša and D. Batič, "The effectiveness of Test-Driven Development: an industrial case study," *Software Quality Journal*, vol. 19, no. 4, pp. 643–661, 2011. [Online]. Available: http://dx.doi.org/10.1007/s11219-011-9130-2

[89] M. Pancur and M. Ciglaric, "Impact of Test-Driven Development on productivity, code and tests: A controlled experiment," *Information and Software Technology*, vol. 53, no. 6, pp. 557 – 573, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950584911000346

[90] J. W. Wilkerson, J. F. Nunamaker, and R. Mercer, "Comparing the defect reduction benefits of code inspection and Test-Driven Development," *IEEE Transactions on Software Engineering*, vol. 38, no. 3, pp. 547–560, 2012.

[91] M. F. Aniche and M. A. Gerosa, "Most common mistakes in test-driven development practice: Results from an online survey with developers," in *2010 Third International Conference on Software Testing, Verification, and Validation Workshops*, April 2010, pp. 469–478.

[92] L. Crispin, "Driving software quality: How Test-driven Development impacts software quality," *IEEE Software*, vol. 23, no. 6, pp. 70–71, 2006.

[93] C. Desai, D. Janzen, and K. Savage, "A survey of evidence for Test-Driven Development in academia," *SIGCSE Bull.*, vol. 40, no. 2, pp. 97–101, 2008. [Online]. Available: http://doi.acm.org/10.1145/1383602.1383644

[94] M. Ficco, R. Pietrantuono, and S. Russo, "Bug localization in test-driven development," *Adv. Soft. Eng.*, vol. 2011, pp. 2:1–2:18, 2011. [Online]. Available: http://dx.doi.org/10.1155/2011/492757

[95] D. S. Janzen and H. Saiedian, "A leveled examination of Test-Driven Development acceptance," in *29th International Conference on Software Engineering (ICSE'07)*, May 2007, pp. 719–722.

[96] D. Janzen and H. Saiedian, "Test-driven learning in early programming courses," in *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, ser. SIGCSE '08. ACM, 2008, pp. 532–536. [Online]. Available: http://doi.acm.org/10.1145/1352135.1352315

[97] S. Kollanus and V. Isomöttönen, "Test-driven development in education: Experiences with critical viewpoints," in *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '08. ACM, 2008, pp. 124–127. [Online]. Available: http://doi.acm.org/10.1145/1384271.1384306

[98] N. Laranjeiro and M. Vieira, "Extending Test-Driven Development for robust web services," in *Second International Conference on Dependability, 2009. DEPEND '09.*, June 2009, pp. 122–127.

[99] M. F. Aniche and M. A. Gerosa, "How the practice of tdd influences class design in object-oriented systems: Patterns of unit tests feedback," in *2012 26th Brazilian Symposium on Software Engineering*, Sep. 2012, pp. 1–10.

[100] I. Turnu, M. Melis, A. Cau, A. Setzu, G. Concas, and K. Mannaro, "Modeling and simulation of open source development using an agile practice," *J. Syst. Archit.*, vol. 52, no. 11, pp. 610–618, Nov. 2006. [Online]. Available: https://doi.org/10.1016/j.sysarc.2006.06.005

[101] E. Tacconelli, "Crds guidance for undertaking reviews in health care," *Lancet Infectious Diseases - LANCET INFECT DIS*, vol. 10, pp. 226–226, 04 2010.

[102] B. Kitchenham, P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering-a systematic literature review," *Information and Software Technology*, vol. 51, pp. 7–15, 01 2009.

[103] M. Q. Patton, "Enhancing the quality and credibility of qualitative analysis," *Health services research*, vol. 41, p. 1189, 12 1999.

[104] K. M. Lui and K. C. C. Chan, "Test driven development and software process improvement in china," in *Extreme Programming and Agile Processes in Software Engineering*, J. Eckstein and H. Baumeister, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 219–222.

[105] O. Kobayashi, M. Kawabata, M. Sakai, and E. Parkinson, "Analysis of the interaction between practices for introducing xp effectively," in *Proceedings of the 28th International Conference on Software Engineering*, ser. ICSE'06. New York, NY, USA: Association for Computing Machinery, 2006, pp. 544–550. [Online]. Available: https://doi.org/10.1145/1134285.1134361

[106] D. S. Janzen and H. Saiedian, "Test-driven learning: Intrinsic integration of testing into the cs/se curriculum," *SIGCSE Bull.*, vol. 38, no. 1, pp. 254–258, Mar. 2006. [Online]. Available: https://doi.org/10.1145/1124706.1121419

[107] M. A. Domino, R. W. Collins, and A. R. Hevner, "Controlled experimentation on adaptations of pair programming," *Information Technology and Management*, vol. 8, no. 4, pp. 297–312, Dec 2007. [Online]. Available: https://doi.org/10.1007/s10799-007-0016-8

[108] D. S. Janzen and H. Saiedian, "A leveled examination of test-driven development acceptance," in *29th International Conference on Software Engineering (ICSE'07)*, May 2007, pp. 719–722.

[109] A. Marchenko, P. Abrahamsson, and T. Ihme, "Long-term effects of test-driven development a case study," in *Agile Processes in Software Engineering and Extreme Programming*, P. Abrahamsson, M. Marchesi, and F. Maurer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 13–22.

[110] R. Latorre, "Effects of developer experience on learning and applying unit test-driven development," *IEEE Transactions on Software Engineering*, vol. 40, no. 4, pp. 381–395, April 2014.

[111] D. Fucci, B. Turhan, and M. Oivo, "On the effects of programming and testing skills on external quality and productivity in a test-driven development context," in *Proceedings of the 19th International Conference on Evaluation and Assessment in Software Engineering*, ser. EASE'15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2745802.2745826

[112] L. Madeyski, "Is external code quality correlated with programming experience or feelgood factor?" vol. 4044, 06 2006, pp. 65–74.

[113] M. M. Muller and O. Hagner, "Experiment about test-first programming," *IEE Proceedings - Software*, vol. 149, no. 5, pp. 131–136, Oct 2002.

[114] L. Machuca-Villegas and G. P. Gasca-Hurtado, "Towards a social and human factor classification related to productivity in software development teams," in *International Conference on Software Process Improvement*. Springer, 2019, pp. 36–50.

[115] S. C. de Barros Sampaio, E. A. Barros, G. S. de Aquino Junior, M. J. C. e Silva, and S. R. de Lemos Meira, "A review of productivity factors and strategies on software development," in *2010 fifth international conference on software engineering advances*. IEEE, 2010, pp. 196–204.

[116] E. Oliveira, T. Conte, M. Cristo, and N. Valentim, "Influence factors in software productivity a tertiary literature review," *International Journal of Software Engineering and Knowledge Engineering*, vol. 28, no. 11–12, pp. 1795–1810, 2018.

[117] L. McLeod and S. G. MacDonell, "Factors that affect software systems development project outcomes: A survey of research," *ACM Computing Surveys (CSUR)*, vol. 43, no. 4, pp. 1–56, 2011.

[118] P. Lenberg, R. Feldt, and L. G. Wallgren, "Human factors related challenges in software engineering–an industrial perspective," in *2015 IEEE/ACM 8th International Workshop on Cooperative and Human Aspects of Software Engineering*. IEEE, 2015, pp. 43–49.

[119] F. Anwar, R. Razali, and K. Ahmad, "Achieving effective communication during requirements elicitation-a conceptual framework," in *International Conference on Software Engineering and Computer Systems*. Springer, 2011, pp. 600–610.

[120] C. França, F. Q. B. da Silva, and H. Sharp, "Motivation and satisfaction of software engineers," *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 118–140, Feb. 2020.

[121] E. Mellblom, I. Arason, L. Gren, and R. Torkar, "The connection between burnout and personality types in software developers," *IEEE Software*, vol. 36, no. 5, pp. 57–64, Sep. 2019.

[122] A. Trendowicz and J. Münch, "Factors influencing software development productivity - state-of-the-art and industrial experiences," ser. Advances in Computers. Elsevier, 2009, vol. 77, pp. 185–241.

[123] I. Turnu, M. Melis, A. Cau, A. Setzu, G. Concas, and K. Mannaro, "Modeling and simulation of open source development using an agile practice," *J. Syst. Archit.*, vol. 52, no. 11, pp. 610–618, Nov. 2006. [Online]. Available: http://dx.doi.org/10.1016/j.sysarc.2006.06.005

[124] A. Jedlitschka and D. Pfahl, "Reporting guidelines for controlled experiments in software engineering," 12 2005, pp. 10 pp.–.

[125] J. C. Carver and T. Al, "Towards reporting guidelines for experimental replications: A proposal," 2010.

[126] N. Juristo and O. S. Gómez, *Replication of Software Engineering Experiments*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 60–88. [Online]. Available: https://doi.org/10.1007/978-3-642-25231-0_2

[127] A. Santos, S. Vegas, F. Dieste Tubío, Uyaguari, A. Tosun, D. Fucci, B. Turhan, G. Scanniello, S. Romano, I. Karac, M. Kuhrmann, V. Mandic, R. Ramac, D. Pfahl, C. Engblom, J. Kyykka, K. Rungi, C. Palomeque, J. Spisak, and N. Juristo, "A family of experiments on test-driven development," *Empirical Software Engineering*, vol. 26, 05 2021.

[128] "Principios del manifiesto agil," http://agilemanifesto.org/iso/es/principles.html, accessed: 2010-09-30.

[129] I. Ratner and J. Harvey, "Vertical slicing: Smaller is better," 08 2011, pp. 240–245.

[130] "Github thesis files," https://github.com/georaura/tddexperiments, accessed: 2021-05-06.

[131] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates, 1988.

[132] I. Ramos, J. Tuya, and J. Dolado-Cosin, *Técnicas cuantitativas para la gestión en la IngenierÃa del Software*, 01 2007.

[133] A. Santos, S. Vegas, M. Oivo, and N. Juristo, "A procedure and guidelines for analyzing groups of software engineering replications," *IEEE Transactions on Software Engineering*, vol. Early access, 2019.

[134] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2021. [Online]. Available: https://www.R-project.org/

[135] D. Robinson, "broom: An r package for converting statistical analysis objects into tidy data frames," *arXiv preprint arXiv:1412.3565*, 2014.

[136] J. Fox and S. Weisberg, *An R Companion to Applied Regression*, 3rd ed. Thousand Oaks CA: Sage, 2019. [Online]. Available: https://socialsciences.mcmaster.ca/jfox/Books/Companion/

[137] H. Wickham, *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. [Online]. Available: https://ggplot2.tidyverse.org

[138] T. Hothorn, F. Bretz, and P. Westfall, "Simultaneous inference in general parametric models," *Biometrical Journal*, vol. 50, no. 3, pp. 346–363, 2008.

[139] W. Revelle, *psych: Procedures for Psychological, Psychometric, and Personality Research*, Northwestern University, Evanston, Illinois, 2021, r package version 2.1.9. [Online]. Available: https://CRAN.R-project.org/package=psych

[140] A. A. Dragulescu and C. Arendt, *xlsx: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files*, 2018, r package version 0.6.1. [Online]. Available: https://CRAN.R-project.org/package=xlsx

[141] D. B. Dahl, D. Scott, C. Roosen, A. Magnusson, and J. Swinton, *xtable: Export Tables to LaTeX or HTML*, 2019, r package version 1.8-4. [Online]. Available: https://CRAN.R-project.org/package=xtable

[142] H. Wickham, *stringr: Simple, Consistent Wrappers for Common String Operations*, 2019, r package version 1.4.0. [Online]. Available: https://CRAN.R-project.org/package=stringr

[143] A. Kuznetsova, P. B. Brockhoff, and R. H. B. Christensen, "lmerTest package: Tests in linear mixed effects models," *Journal of Statistical Software*, vol. 82, no. 13, pp. 1–26, 2017.

[144] S. Mangiafico, *rcompanion: Functions to Support Extension Education Program Evaluation*, 2019, r package version 2.2.1. [Online]. Available: https://CRAN.R-project.org/package=rcompanion

[145] S. Vegas, C. Apa, and N. Juristo, "Crossover designs in software engineering experiments: Benefits and perils," *IEEE Transactions on Software Engineering*, vol. 42, no. 2, pp. 120–135, 2016.

[146] A. Field, *Discovering Statistics Using SPSS*, ser. ISM (London, England). SAGE Publications, 2009. [Online]. Available: https://books.google.es/books?id=a6FLF1YOqtsC

[147] S. S. C. Cooperative. (2016, mar) Mixed models: Diagnostics and inference. [Online]. Available: https://www.ssc.wisc.edu/sscc/pubs/MM/MM_DiagInfer.html

[148] P. Savicky, *pspearman: Spearman's rank correlation test*, 2014, r package version 0.3-0. [Online]. Available: https://CRAN.R-project.org/package=pspearman

[149] T. Wei and V. Simko, *R package çorrplot": Visualization of a Correlation Matrix*, 2017, (Version 0.84). [Online]. Available: https://github.com/taiyun/corrplot

[150] M. S. Ben-Shachar, D. Makowski, and D. LÃ¼decke, "Compute and interpret indices of effect size," *CRAN*, 2020, r package. [Online]. Available: https://github.com/easystats/effectsize

[151] D. H. Ogle, P. Wheeler, and A. Dinno, *FSA: Fisheries Stock Analysis*, 2020, r package version 0.8.30. [Online]. Available: https://github.com/droglenc/FSA

[152] S. Balduzzi, G. RÃ¼cker, and G. Schwarzer, "How to perform a meta-analysis with R: a practical tutorial," *Evidence-Based Mental Health*, no. 22, pp. 153–160, 2019.

[153] A. C. D. Re, *compute.es: Compute Effect Sizes*, 2013. [Online]. Available: https://cran.r-project.org/package=compute.es

[154] H. Wickham, "Reshaping data with the reshape package," *Journal of Statistical Software*, vol. 21, no. 12, pp. 1–20, 2007. [Online]. Available: http://www.jstatsoft.org/v21/i12/

[155] D. Ebbert, *chisq.posthoc.test: A Post Hoc Analysis for Pearson's Chi-Squared Test for Count Data*, 2019, r package version 0.1.2. [Online]. Available: https://CRAN.R-project.org/package=chisq.posthoc.test

[156] M. Torchiano, *effsize: Efficient Effect Size Computation*, 2020, r package version 0.8.1. [Online]. Available: https://CRAN.R-project.org/package=effsize

[157] H. Zhu, *kableExtra: Construct Complex Table with 'kableánd Pipe Syntax*, 2020, r package version 1.3.1. [Online]. Available: https://CRAN.R-project.org/package=kableExtra

[158] R. V. Lenth, "Least-squares means: The R package lsmeans," *Journal of Statistical Software*, vol. 69, no. 1, pp. 1–33, 2016.

[159] C. Wohlin, P. Runeson, M. Host, M. C. Ohlsson, and B. Regnell, *Experimentation in Software Engineering.* Springer, 2012.

[160] W. SHADISH, W. Shadish, E. Chelimsky, T. Cook, D. Campbell, and C. Learning, *Experimental and Quasi-experimental Designs for Generalized Causal Inference*, ser. Experimental and Quasi-experimental Designs for Generalized Causal Inference. Houghton Mifflin, 2002, no. v. 1. [Online]. Available: https://books.google.es/books?id=o7jaAAAAMAAJ

[161] S. Senn, *Cross-over Trials in Clinical Research*, ser. Statistics in Practice. Wiley, 2003. [Online]. Available: https://books.google.es/books?id=bOks34_wUjcC

[162] T. Dogsa and D. Batic, "The effectiveness of test-driven development: An industrial case study," *Software Quality Journal*, vol. 19, pp. 643–661, 12 2011.

[163] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2016. [Online]. Available: https://www.R-project.org/

[164] H. Wickham, *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2009. [Online]. Available: http://ggplot2.org

[165] T. Hothorn, F. Bretz, and P. Westfall, "Simultaneous inference in general parametric models," *Biometrical Journal*, vol. 50, no. 3, pp. 346–363, 2008.

[166] D. B. Dahl, *xtable: Export Tables to LaTeX or HTML*, 2016, r package version 1.8-2. [Online]. Available: https://CRAN.R-project.org/package=xtable

[167] P. Savicky, *pspearman: Spearman's rank correlation test*, 2014, r package version 0.3-0. [Online]. Available: https://CRAN.R-project.org/package=pspearman

[168] T. Wei and V. Simko, *R package çorrplot": Visualization of a Correlation Matrix*, 2017, (Version 0.84). [Online]. Available: https://github.com/taiyun/corrplot

[169] M. S. Ben-Shachar, D. Makowski, and D. LÃ¼decke, "Compute and interpret indices of effect size," *CRAN*, 2020, r package. [Online]. Available: https://github.com/easystats/effectsize

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes