# ¿Qué factores personales afectan a la calidad y productividad de TDD? Un experimento con profesionales

Geovanny Raura[1,2] [0000−0002−2045−6350], Claudia Pons[2] [0000−0003−1149−0976], Efraín R. Fonseca C.[1] [0000−0001−7129−9335], and Oscar Dieste[3] [0000−0002−3060−7853]

[1] Universidad de las Fuerzas Armadas ESPE, Sangolquí, Ecuador
{erfonseca, jgraura}@espe.edu.ec
[2] Universidad Nacional de La Plata, Argentina
Universidad Abierta Interamericana, UAI, Argentina
cpons@info.unlp.edu.ar
[3] Universidad Politécnica de Madrid, España
odieste@fi.upm.es

**Resumen** —***Contexto:*** Test-Driven Developement (TDD) es una técnica de desarrollo de software ágil que es ampliamente utilizada en la industria, aunque su efectividad ha generado incertidumbre si se compara con técnicas de desarrollo tradicional. ***Objetivo:*** Estudiar la efectividad de TDD considerando el grado de influencia de distintos factores humanos. ***Metodología:*** Experimento aleatorizado (*crossover 2x2*) realizado con sujetos profesionales en un ámbito académico. ***Resultados:*** La calidad y productividad al aplicar TDD es algo superior a lo obtenido con el desarrollo iterativo incremental (ITLD). La edad de los participantes, la función que desempeñaban en su trabajo y el conocimiento previo de la técnica de TDD ejercen influencia sobre las variables respuesta.

**Keywords:** Desarrollo dirigido por pruebas · calidad · productividad · factores humanos · experimentación.

## 1. Introducción

Las metodologías ágiles se basan en distintas prácticas, tales como el desarrollo dirigido por pruebas ($TDD$, por sus siglas en inglés) [1]. Se han realizado múltiples estudios empíricos acerca de $TDD$ para verificar si mejora la calidad del software y la productividad de los programadores (dentro de ellos podemos citar estudios secundarios como: [13], [15], [2]). No obstante, los resultados no son concluyentes y, en varios casos, han resultado incluso contradictorios. Por ejemplo, [11] concluye que $TDD$ mejora la productividad de los programadores, mientras que [5] y [6] afirman lo contrario.

En un trabajo de síntesis, Rafique y Mišić [13] indican que, a pesar de las diferencias considerables entre los experimentos analizados, $TDD$ da como resultado

una pequeña mejora en la calidad. No obstante, esta postura no es necesariamente la correcta. Turhan et al. [15] indican que la calidad externa parece disminuir para los sujetos graduados y aumenta para los no graduados. Efectivamente, las diferencias entre [13] y [15] podrían explicarse por las características personales (estudiantes vs. profesionales) de los sujetos experimentales.

Las características personales de los desarrolladores, tales como la experiencia laboral, conocimiento previo en TDD, habilidad para realizar casos de pruebas, conocimiento del dominio, motivación, etc., no han sido frecuentemente estudiadas en $TDD$. Además, Bissi et al. [2] indican que hay mucha demanda para investigar $TDD$ en la práctica. Esto nos motivó a realizar un experimento con desarrolladores profesionales, con un propósito doble: (1) Aportar al cuerpo de conocimiento de $TDD$ un estudio realizado con profesionales, y (2) estudiar el impacto de distintas habilidades personales en la calidad externa del software y la productividad de los programadores cuando utilizan $TDD$.

Este artículo está organizado de acuerdo a las guías de reporte de Jedlitschka y Pfahl [9], con adaptaciones debido al espacio disponible. La información acerca del experimento se incluye en la Sección 2. La ejecución del experimento en la sección 3. El análisis y resultados obtenidos se muestran en la Sección 4. En las Sección 5 se describen las amenazas a la validez. Finalmente, en la Sección 6, se presentan las conclusiones de la investigación.

## 2. Planeo experimental

El experimento aquí reportado se basa en el experimento [10] realizado por N. Juristo y su equipo de investigación de la Universidad Politécnica de Madrid. El objetivo de este experimento fue estudiar la efectividad de TDD en comparación con Iterative Test-Last ($ITLD$). $ITLD$ es la aproximación más usada para el desarrollo de software apoyado por test automatizados. $ITLD$ consiste en el desarrollo de pequeñas porciones del código de producción, seguido inmediatamente por la realización de pruebas de unidad [8].

### 2.1. Factores y variables respuesta

Utilizamos los mismos factores y variables respuesta que el experimento original. Se ensayó como factor principal la *aproximación de desarrollo*, con los niveles $ITLD$ y $TDD$. Las variables respuesta han sido la *calidad externa* (QLTY) y la *productividad* (PROD). QLTY representa el grado de corrección del código desarrollado por los sujetos, y se define como:

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

donde $QLTY_i$ es la calidad de la historia de usuario i-ésima implementada por el sujeto. $QLTY_i$ se define como:

$$QLTY_i = \frac{\#Assert_i(Pass)}{\#Assert_i(All)} \tag{2}$$

mientras que #TUS es:

$$\#TUS = \sum_{i=1}^{\#us} \#Assert_i(Pass) \geq 0 \mapsto True \quad (3)$$

En ambos casos, $\#Assert_i(Pass)$ representa el número de aserciones de jUnit (ya que el experimento se realizó utilizando el lenguaje de programación Java) correctamente ejecutados en la historia de usuario i-ésima. PROD representa la cantidad de trabajo realizada por los sujetos, y se define como:

$$PROD = \frac{\#Assert(Pass)}{\#Assert(All)} \quad (4)$$

Las variables calidad y productividad se han utilizado en varios trabajos y la naturaleza de las mismas han sido particularmente descritas en [14].

### 2.2. Hipótesis

Se proponen dos hipótesis. La primera hace referencia a que la calidad externa del software no se ve alterada por el uso de $ITLD$ o $TDD$:
$H_{10} : QLTY_{ITLD} = QLTY_{TDD}$
$H_{11} : QLTY_{ITLD} <> QLTY_{TDD}$ *(2-colas)*

La segunda hipótesis indica lo mismo respecto a la productividad:
$H_{20} : PROD_{ITLD} = PROD_{TDD}$
$H_{21} : PROD_{ITLD} <> PROD_{TDD}$ *(2-colas)*

### 2.3. Diseño

El diseño experimental fue de tipo *cross over*. El diseño *cross-over* se utiliza con frecuencia para aumentar el poder estadístico cuando el número de sujetos experimentales es reducido. Sin embargo, el uso de este diseño exige introducir, además del factor principal, factores secundarios de tipo *between* y *within-subjects*. Estos factores son los siguientes:

- **Tarea experimental:** Se utilizaron dos tareas muy conocidas en el ámbito de $TDD$: *MarsRover* (MR) y *BowlingScoreKeeper* (BSK). MR es un ejercicio de programación que tiene por objetivo el desarrollo de una serie de métodos públicos o API (Aplication Program Interface), que permita simular el movimiento de un vehículo a diferentes puntos con diferentes orientaciones (Norte, Sur, Este, Oeste), dentro de un planeta representado por un plano de coordenadas. BSK tiene por objetivo calcular el marcador de un único juego de bolos. Tanto MR como BSK se limitan a la implementación de un algoritmo de programación, sin que sea necesario el desarrollo de una interface de usuario [4].

[4] La descripción de las tareas se puede encontrar en `https://github.com/georaura/tddexperiments/tree/master/ExperimentTemplates`

Tabla 1: Covariables

| Covariable | Descripción | Métrica |
|---|---|---|
| Edad | Indica la edad del sujeto participante | Medida mediante un valor numérico: Tipo entero positivo |
| Experiencia en programación<br>Experiencia profesional<br>Experiencia en lenguaje Java<br>Experiencia en framework JUnit | Indica la experiencia del sujeto en distintos aspectos | Medida mediante escala likert:<br>1 Sin experiencia (<2 años)<br>2 Novato (2 - 5 años)<br>3 Intermedio (6 - 10 años)<br>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:<br>1 Si<br>2 No |
| 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:<br>1 Si<br>2 No |
| Conocimiento del entorno Eclipse | Determina si el sujeto tiene conocimiento previo del IDE de desarrollo Eclipse | Medida mediante escala nominal:<br>1 Si<br>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:<br>1 Manager<br>2 Developer<br>3 Analyst<br>4 Other |

- **Nivel de especificación de la tarea:** La especificación de las tareas fueron descompuestas en una serie de pasos o 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. Una versión No-slicing, al contrario es una especificación que contiene simplemente una descripción textual de todas las funciones a codificar.
- **Grupo:** Este factor secundario hace referencia al orden en que los sujetos experimentales realizaron las tareas MR y BSK.

## 2.4. Covariables

A diferencia del experimento base, nuestro principal interés es determinar el grado de influencia de distintos aspectos personales en la efectividad de la programación con $TDD$. Los aspectos personales escogidos, que se usarán como covariables durante el análisis post-hoc, han sido los indicados en la Tabla 1. Aunque no se describirán explícitamente, para cada una de las covariables existe una hipótesis post-hoc asociada. Por ejemplo, respecto al aspecto personal *Edad*:

$H_{a0}$: No hay ninguna relación entre edad del sujeto participante y la efectividad en la aplicación de $ITLD/TDD$

$H_{a1}$: Existe una relación entre edad del sujeto participante y la efectividad en la aplicación de $ITLD/TDD$ *(2-colas)*

# 3. Ejecución

## 3.1. Preparación

El experimento fue realizado en la Universidad Técnica del Norte de Ecuador con un grupo de 27 estudiantes de la Maestría en Ingeniería de Software. Todos los estudiantes son profesionales en activo. Antes de realizar el experimento, las características personales se obtuvieron mediante un cuestionario demográfico.[5].

Por cuestión de espacio no hemos incluido la tabulación de datos, sin embargo podemos señalar que ningún participante indicó tener experiencia previa en $TDD$ [6]. La mayoría reportó que no tenía experiencia en el framework JUnit (apenas un 7 % indicó tener entre 2 y 5 años de experiencia), y el 85 % indicó que no había utilizado la técnica $TDD$. En las restantes covariables, aunque los valores están fuertemente desbalanceados, existe un número mínimo de sujetos en cada categoría, ej: aproximadamente un 20 % de los sujetos manifestó no tener conocimiento del entorno Eclipse, frente a un 80 % que si lo tenía.

El experimento se realizó en dos fines de semana consecutivos dentro del horario de clase de los participantes de acuerdo al siguiente protocolo:

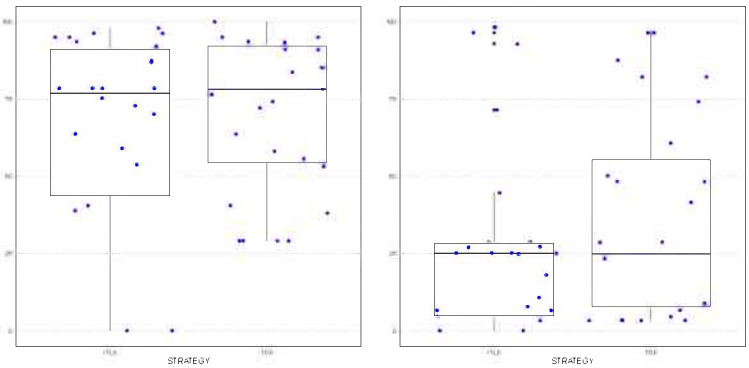
- **Día 1:** Los sujetos llenaran el cuestionario demográfico previo al inicio del entrenamiento. Durante el primer día se realizó una sesión de introducción al desarrollo ágil y formación en pruebas unitarias utilizando el framework Junit.
- **Día 2:** En el segundo día de la primera semana, se capacitó en las técnicas $ITLD$ con slicing. Posteriormente, 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 $ITLD$.
- **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/sin Slicing y aplicando la estrategia $TDD$. Para concluir la sesión experimental, se realizó una retroalimentación de los conocimientos adquiridos.

## 3.2. Desviaciones

Prácticamente no existieron variaciones en el número de sujetos que asistieron al experimento, excepto por un único participante que no asistió la primera semana de ejecución del experimento. Se presentaron 27 sujetos. De ellos, 26 realizaron la primera tarea experimental, es decir la aplicación de la estrategia $ITLD$, y 27 sujetos realizaron la segunda tarea aplicando la estrategia $TDD$.

[5] Los datos están disponibles en `https://github.com/georaura/tddexperiments/blob/master/TddExpData/DemographicsUTN2017.xlsx`.

[6] Por ello, esta variable no podrá ser analizada en las secciones que siguen.




(a) Calidad (b) Productividad

Figura 1: Box-plots para las variable respuesta Productividad y Calidad

## 4. Análisis

### 4.1. Estadísticos descriptivos

Tabla 2: Estadísticos descriptivos para la calidad

| Estrategia de Programación | Promedio | Desviación estándar | Asimetría | Curtosis |
|---|---|---|---|---|
| ITLD | 62.58 | 35.02 | -0.86 | -0.79 |
| TDD | 70.77 | 24.70 | -0.53 | -1.20 |

Tabla 3: Estadísticos descriptivos para la productividad

| Estrategia de Programación | Promedio | Desviación estándar | Asimetría | Curtosis |
|---|---|---|---|---|
| ITLD | 26.82 | 30.19 | 1.31 | 0.53 |
| TDD | 37.40 | 31.34 | 0.58 | -1.07 |

La tabla 2 muestra que la calidad [7] obtenida aplicando la estrategia $TDD$ es algo superior a lo obtenido para $ITLD$. Sin embargo, las desviaciones estándar son elevadas, lo que implica que existe una gran variación entre individuos. La desviación estándar de $TDD$ es sustancialmente menor a $ITLD$, lo que indica que los subjetos experimentales producen código de calidad más uniforme,

[7] Los datos están disponibles en `https://github.com/georaura/tddexperiments/blob/master/TddExpData/DataUTN2017.xlsx`. La medición fue realizada por G. Raura.

dentro de la variabilidad. El box-plot mostrado en la figura 1a corrobora las observaciones anteriores. La asimtetría y curtosis rondan el valor 1 o inferiores, lo que probablemente implica que los datos se distribuyen normalmente.

La variable respuesta productividad se comporta de forma parecida a la calidad, tal y como puede comprobarse en la tabla 3, excepto en lo que se refiere a la variabilidad. La desviación estándar de $TDD$ es mayor que la de $ITLD$, lo que implica que los resultados de la aplicación de $ITLD$ son más uniformes. Sin embargo, el box-plot de la figura 1a muestra claramente que dicha uniformidad ocurre con valores de productividad bajos. La mayor variabilidad en $TDD$ lleva aparejada un mejor rendimiento de varios de los sujetos que han usado esta estrategia de programación.

### 4.2. Prueba de hipótesis

El análisis se ha realizado siguiendo las indicaciones de Vegas et al. [16]. Con el propósito de facilitar el re-análisis, mostramos a continuación el código $R$ utilizado. Los análisis cumplen con las condiciones de normalidad de residuos (nótese el uso de la transformación Box-Cox con $\lambda = \frac{1}{4}$) para obtener normalidad de residuos), heterocedasticidad y normalidad de efectos aleatorios[8].

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

lmp <- lmer(PROD^(1/4) ~ STRATEGY + TASK + SLICING + GROUP +
                         (1 | subjectID),
            data = all_expData)
```

Los resultados se muestran en la tabla 4. Teniendo en cuenta el número de sujetos experimentales y el carácter exploratorio del experimento, no seremos estrictos con el criterio habitual $\alpha = 0{,}05$ de significación estadística. Ignoraremos las variables $TASK$ (que ya sabemos por otros estudios [14] que acostumbra ser estadísticamente significativa), $GROUP$ y $SLICING$, ya que no son relevantes para este artículo.

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, ya que las tablas 2 y 3 indican que $TDD$ es superior a $ITLD$. Los tamaños de efecto ($g$ de Hedges [7]) son 0.38 y 0.45 para la calidad y productividad, lo que equivale en términos cuantitativos [3] a efectos *pequeño* y *pequeño*, respectivamente.

[8] El lector interesado podrá encontrar un análisis más detallado en `https://bitbucket.org/tesistdd/tesisgeovanny/`.

Tabla 4: 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 |

### 4.3. Análisis de las características personales

Para cuantificar el impacto de las características personales, hemos estudiado la interacción entre cada característica y la estrategia de programación. A modo de ejemplo, **para la edad**, el código $R$ es el siguiente [9]:

```
lmq <- lmer(QLTY ~ STRATEGY * floor(age/10) +
                   TASK + SLICING + GROUP +
                   (1 | subjectID),
            data = all_expData)

lmp <- lmer(PROD^(1/4) ~ STRATEGY * floor(age/10) +
                         TASK + SLICING + GROUP +
                         (1 | subjectID),
            data = all_expData)
```

Lo que produce los resultados que se muestran en la tabla 5. 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.

La influencia de la edad en la estrategia de programación [10] se muestra en la Fig. 2. Puede observarse que, a mayor edad, menor la calidad y productividad alcanzadas tanto en $ITLD$ como $TDD$. Este patrón general presenta una ex-

[9] En el caso particular de la edad, hemos transformado los años a décadas para que los gráficos de perfil fueran más fácilmente interpretables.

[10] Los tamaños de efecto, en unidades naturales (porcentajes) pueden calcularse directamente usando los ejes de ordenadas.

Tabla 5: Resultados del análisis de la influencia de la Edad en $TDD$

(a) Calidad

| | Sum Sq | Mean Sq | NumDF | DenDF | F value | Pr(>F) |
|---|---|---|---|---|---|---|
| STRATEGY | 812.73 | 812.73 | 1.00 | 22.77 | 2.16 | 0.1553 |
| floor(age/10) | 1073.91 | 1073.91 | 1.00 | 23.77 | 2.85 | 0.1042 |
| TASK | 24059.42 | 24059.42 | 1.00 | 23.23 | 63.94 | 0.0000 |
| SLICING | 1434.20 | 1434.20 | 1.00 | 23.27 | 3.81 | 0.0630 |
| GROUP | 1231.51 | 1231.51 | 1.00 | 24.26 | 3.27 | 0.0828 |
| STRATEGY:floor(age/10) | 1308.27 | 1308.27 | 1.00 | 22.73 | 3.48 | 0.0752 |

(b) Productividad

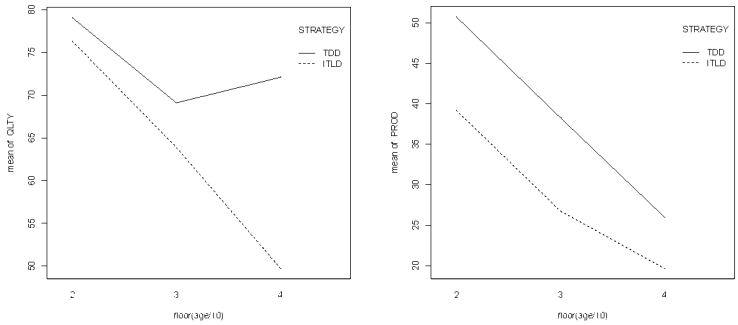
| | Sum Sq | Mean Sq | NumDF | DenDF | F value | Pr(>F) |
|---|---|---|---|---|---|---|
| STRATEGY | 0.18 | 0.18 | 1.00 | 22.46 | 0.58 | 0.4541 |
| floor(age/10) | 1.90 | 1.90 | 1.00 | 23.41 | 6.10 | 0.0212 |
| TASK | 16.02 | 16.02 | 1.00 | 22.92 | 51.55 | 0.0000 |
| SLICING | 1.12 | 1.12 | 1.00 | 22.96 | 3.61 | 0.0702 |
| GROUP | 2.84 | 2.84 | 1.00 | 23.90 | 9.13 | 0.0059 |
| STRATEGY:floor(age/10) | 0.61 | 0.61 | 1.00 | 22.42 | 1.96 | 0.1753 |

cepción para los desarrolladores mayores de 40 años, los cuales producen código de mayor calidad que los desarrolladores más jóvenes.

De la misma forma que para la edad, hemos analizado la influencia de las restantes caracteristicas personales. Hemos incluído en el Apéndice A aquellos análisis que han resultado estadísticamente significativos (con las matizaciones anteriormente indicadas). A este respecto, nótese que la realización de múltiples análisis aumenta el riesgo de cometer un error de tipo I. No obstante, asumismos dicho riesgo ya que este experimento emplea pocos sujetos experimentales, por lo que el riesgo de cometer un error tipo II es mucho mayor.

Dos características personales han arrojado resultados significativos: El **uso de TDD** y la **función actual en la organización**. Los análisis se muestan en las Tablas 6 y 7 del Apéndice A. A diferencia de la Edad, estas variables no alteran la significación de las estrategias de programación $ITLD/TDD$. El uso de TDD y la función actual en la organización son, probablemente, variables moderadoras y como tales su introducción en el análisis hace los efectos de las estrategias de programación $ITLD/TDD$ más evidentes.

La Fig. 3 muestra que aquellas personas que usan TDD consiguen alcanzan calidades y productividades similares al resto de sujetos cuando usan $ITLD$. Sin embargo, su rendimiento aumenta espectacularmente cuando usan $TDD$. En nuetra opinión, no hay nada sorprendente en este hecho; se trata de un resultado "de sentido común". No obstante, nos alegramos de haber obtenido un resultado tan predecible, ya que de alguna manera confirma que los sujetos se han comportado de forma natural durante la realización del experimento.




(a) Calidad (b) Productividad

Figura 2: Gráficos de perfil para STRATEGY * age, y las variables respuesta Productividad y Calidad. Nótese que los ejes de ordenadas de los gráficos **no son iguales**.

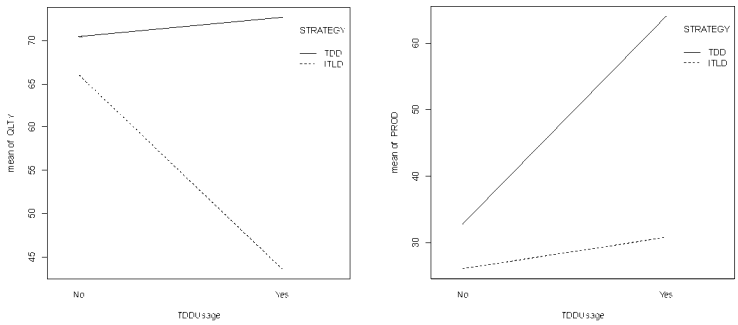
La Fig. 4 muestra los resultados alcanzados por los sujetos en función de su puesto actual en la organización. Se observa que $TDD$ produce una mejor calidad y productividad que $ITLD$, en prácticamente todos los casos. Esto es plenamente coherente con el análisis global mostrado en la Tabla 4.

En lo que respecta a los puestos concretos, el patrón no es claro. Para $ITLD$ se observa que la calidad calidad y productividad de los gestores es pobre, lo que es lógico ya que los gestores no realizan habitualmente tareas técnicas. Sin embargo, también referido a $ITLD$, no deja de ser curioso que los analistas obtengan mejor calidad y productividad que los desarrolladores. Bien es cierto que los conceptos de *analista* y *desarrollador* no siempre corresponden con perfiles precisos, por lo que estos resultados podrían probablemente matizarse de contar con más información. Lo que resulta realmente sorprendente es el pico que los gestores nuestran para la calidad y productividad cuando usan $TDD$. Los resultados de los gestores superan a todos los otros perfiles.

## 5. Amenazas a la validez

Al aplicar diseños de medidas repetidas, generalmente se encuentran las siguientes amenazas a la validez [4]: práctica, fatiga, carry over y orden/periodo. En razón a que las sesiones de entrenamiento y los experimentos ha sido realizados de manera intensiva, es altamente probable que en el presente experimento opere la amenaza de fatiga. Sin embargo, a nuestro criterio, las restantes amenazas no aplican por las siguientes razones:

- **Práctica:** Como se deriva de los datos demográficos, TDD es una técnica nueva para la mayoría de los sujetos experimentales. Esto sugiere que en




(a) Calidad (b) Productividad

Figura 3: Gráficos de perfil para STRATEGY * TDDUsage

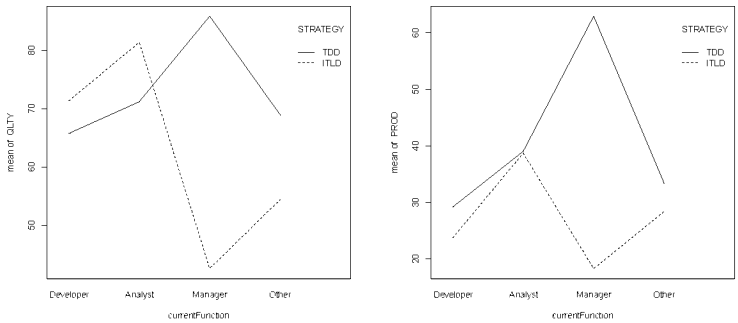
cuanto más práctica exista de esta técnica, se podrían identificar mejor los potenciales efectos beneficiosos. Los sujetos deben lograr cierto grado de pericia en las técnicas que fueron abordadas en las sesiones de entrenamiento (TDD e ITLD). Por lo tanto, la práctica no representa una amenaza a la validez sino una condición necesaria para alcanzar los objetivos experimentales.

- **Acarreo:** El efecto de acarreo ocurre cuando los efectos de un tratamiento no han desaparecido antes de introducir el siguiente tratamiento. La técnica ITLD utiliza estrategias parecidas a TDD, por lo que el carry-over, resulta favorable para el experimento al igual que la práctica.
- **Orden/periodo:** Este factor se relaciona con el paso del tiempo. En nuestro caso, el entrenamiento y las sesiones experimentales se realizan dentro del mismo fin de semana, y el experimento se completa en dos fines de semana consecutivos. No creemos que en tan breve espacio de tiempo puedan producirse eventos que afecten a la pericia o motivación de los sujetos experimentales.

## 6. Discusión y conclusiones

Nuestro estudio indica que $TDD$ produce un efecto positivo tanto para la calidad como para la productividad. En el caso de la productividad, los resultados obtenidos son estadísticamente significativos, mientras que la calidad se acerca al nivel de significación. Los tamaños de efecto son pequeños en ambos casos. La literatura coincide en que la calidad externa del código escrito por profesionales aumenta cuando aplican $TDD$ [12]. En cuanto a la productividad, los resultados reportados en la literatura son mixtos.

El número de sujetos experimentales utilizado era bastante pequeño, lo que conlleva un reducido poder estadístico. Esto perjudica la identificación de los




(a) Calidad (b) Productividad

Figura 4: Gráficos de perfil para STRATEGY * currentFunction

efectos de las características personales, ya que es difícil alcanzar la significación estadística. Aun así, hemos obtenido algunos hallazgos, que lógicamente deben tomarse con cautela.

La edad de los sujetos participantes de los experimentos ha arrojado resultados positivos: a mayor edad, menor la calidad y productividad alcanzadas tanto en $ITLD$ como $TDD$. Esto nos sugiere que otros factores humanos como la experiencia podrían ejercer efectos similares, aunque en nuestros análisis han resultado negativos a este respecto.

Los sujetos que conocían del uso de $TDD$ obtuvieron una mejor efectividad al aplicar esta técnica. Esto más que un hallazgo es una especie de control. Si el uso de $TDD$ no mejorase la eficacia de $TDD$, sería un sinsentido. Asimismo, nos sorprende que los profesionales que trabajan como gestores hayan obtenido un mejor desempeño al aplicar $TDD$ que los desarrolladores o analistas. La explicación puede deberse al hecho de que estos participantes eran programadores experimentados (en su mayoría con más de 10 años de experiencia) que se encontraban dirigiendo equipos de desarrollo y cumplían amabas funciones (developer y manager).

Este experimento pertenece a una línea de trabajo activa [11]. Como trabajo futuro inmediato, pretendemos realizar un meta-análisis de los experimentos realizados hasta la fecha, para poder identificar con mayor precisión el impacto de las características personales en $TDD$ e $ITLD$.

## Referencias

1. Beck, K.: Test Driven Development By Example. Addison-Wesley Longman, Amsterdam, 1 edn. (2002)

[11] `https://bitbucket.org/tesistdd/tesisgeovanny/`.

2. Bissi, W., Serra Seca Neto, A.G., Emer, M.C.F.P.: The effects of test driven development on internal quality, external quality and productivity: A systematic review. Information and Software Technology **74**, 45–54 (2016). https://doi.org/https://doi.org/10.1016/j.infsof.2016.02.004, https://www.sciencedirect.com/science/article/pii/S0950584916300222
3. Cohen, J.: Statistical power analysis for the behavioral sciences. Routledge, New York, 3 edn. (1988)
4. Cook, T., Campbell, D.: Quasi-Experimentation: Design and Analysis Issues for Field Settings. Houghton Mifflin (1979)
5. Desai, C., Janzen, D., Savage, K.: A survey of evidence for test-driven development in academia. SIGCSE Bull. **40**(2), 97–101 (Jun 2008). https://doi.org/10.1145/1383602.1383644, https://doi.org/10.1145/1383602.1383644
6. Dogła, T., Batic, D.: The effectiveness of test-driven development: An industrial case study. Software Quality Journal **19**, 643–661 (12 2011). https://doi.org/10.1007/s11219-011-9130-2
7. Hedges, L.V., Olkin, I.: Statistical Methods for Meta-Analyis. Academic Press (1985)
8. Janzen, D.S., Saiedian, H.: Test-driven learning: Intrinsic integration of testing into the cs/se curriculum. SIGCSE Bull. **38**(1), 254–258 (Mar 2006). https://doi.org/10.1145/1124706.1121419, https://doi.org/10.1145/1124706.1121419
9. Jedlitschka, A., Pfahl, D.: Reporting guidelines for controlled experiments in software engineering. pp. 10 pp.– (12 2005). https://doi.org/10.1109/ISESE.2005.1541818
10. Juristo, N.: Experiences conducting experiments in industry: The eseil fidipro project. In: 2016 IEEE/ACM 4th International Workshop on Conducting Empirical Studies in Industry (CESI). pp. 1–3 (2016). https://doi.org/10.1109/CESI.2016.009
11. Madeyski, L., Szała, Ł.: The impact of test-driven development on software development productivity — an empirical study. In: Abrahamsson, P., Baddoo, N., Margaria, T., Messnarz, R. (eds.) Software Process Improvement. pp. 200–211. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
12. Munir, H., Moayyed, M., Petersen, K.: Considering rigor and relevance when evaluating test driven development: A systematic review. Information and Software Technology **56**(4), 375 – 394 (2014). https://doi.org/https://doi.org/10.1016/j.infsof.2014.01.002, http://www.sciencedirect.com/science/article/pii/S0950584914000135
13. Rafique, Y., Mišić, V.B.: The effects of test-driven development on external quality and productivity: A meta-analysis. IEEE Transactions on Software Engineering **39**(6), 835–856 (2012)
14. Tosun, A., Dieste, O., Fucci, D., Vegas, S., Turhan, B., Erdogmus, H., Juristo, N.: An industry experiment on the effects of test-driven development on external quality and productivity. In: 2017 Empirical Software Engineering. vol. 1, p. 2763–2805 (2017). https://doi.org/doi.org/10.1007/s10664-016-9490-0
15. Turhan, B., Layman, L., Diep, M., Shull, F., Erdogmus, H.: How Effective is Test Driven Development, pp. 45–55. O Reilly Media (10 2010)
16. Vegas, S., Apa, C., Juristo, N.: Crossover designs in software engineering experiments: Benefits and perils. IEEE Transactions on Software Engineering **42**(2), 120–135 (2016). https://doi.org/10.1109/TSE.2015.2467378

# Apéndice A Análisis de las características personales que señalan relaciones estadísticamente significativas

Tabla 6: Resultados del análisis de la influencia del uso de *TDD* en *TDD*

(a) Calidad

| | Sum Sq | Mean Sq | NumDF | DenDF | F value | Pr(>F) |
|---|---|---|---|---|---|---|
| STRATEGY | 2351.45 | 2351.45 | 1.00 | 22.81 | 5.85 | 0.0239 |
| TDDUsage | 487.30 | 487.30 | 1.00 | 23.86 | 1.21 | 0.2817 |
| TASK | 22747.18 | 22747.18 | 1.00 | 23.27 | 56.62 | 0.0000 |
| SLICING | 1801.08 | 1801.08 | 1.00 | 23.27 | 4.48 | 0.0451 |
| GROUP | 924.13 | 924.13 | 1.00 | 24.28 | 2.30 | 0.1423 |
| STRATEGY:TDDUsage | 735.55 | 735.55 | 1.00 | 22.81 | 1.83 | 0.1893 |

(b) Productividad

| | Sum Sq | Mean Sq | NumDF | DenDF | F value | Pr(>F) |
|---|---|---|---|---|---|---|
| STRATEGY | 4.84 | 4.84 | 1.00 | 22.51 | 16.92 | 0.0004 |
| TDDUsage | 0.01 | 0.01 | 1.00 | 23.66 | 0.02 | 0.8933 |
| TASK | 15.28 | 15.28 | 1.00 | 22.96 | 53.40 | 0.0000 |
| SLICING | 1.34 | 1.34 | 1.00 | 22.96 | 4.69 | 0.0410 |
| GROUP | 1.69 | 1.69 | 1.00 | 24.08 | 5.91 | 0.0228 |
| STRATEGY:TDDUsage | 1.18 | 1.18 | 1.00 | 22.51 | 4.12 | 0.0542 |

Tabla 7: Resultados del análisis de la influencia de la Función Actual en la Organización en *TDD*

(a) Calidad

| | Sum Sq | Mean Sq | NumDF | DenDF | F value | Pr(>F) |
|---|---|---|---|---|---|---|
| STRATEGY | 2790.21 | 2790.21 | 1.00 | 21.46 | 11.48 | 0.0027 |
| currentFunction | 382.24 | 127.41 | 3.00 | 22.57 | 0.52 | 0.6700 |
| TASK | 23269.90 | 23269.90 | 1.00 | 21.32 | 95.75 | 0.0000 |
| SLICING | 2171.24 | 2171.24 | 1.00 | 21.32 | 8.93 | 0.0069 |
| GROUP | 567.82 | 567.82 | 1.00 | 22.55 | 2.34 | 0.1403 |
| STRATEGY:currentFunction | 4818.10 | 1606.03 | 3.00 | 21.29 | 6.61 | 0.0025 |

(b) Productividad

| | Sum Sq | Mean Sq | NumDF | DenDF | F value | Pr(>F) |
|---|---|---|---|---|---|---|
| STRATEGY | 5.70 | 5.70 | 1.00 | 20.45 | 29.58 | 0.0000 |
| currentFunction | 0.41 | 0.14 | 3.00 | 21.60 | 0.70 | 0.5608 |
| TASK | 15.98 | 15.98 | 1.00 | 20.32 | 82.99 | 0.0000 |
| SLICING | 1.71 | 1.71 | 1.00 | 20.32 | 8.87 | 0.0073 |
| GROUP | 1.23 | 1.23 | 1.00 | 21.58 | 6.39 | 0.0194 |
| STRATEGY:currentFunction | 3.57 | 1.19 | 3.00 | 20.28 | 6.18 | 0.0037 |