POLITÉCNICA

UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingenieros Informáticos

Impacto de la instrumentación en los experimentos de Ingeniería del Software: Aplicación a experimentos de Test-Driven Development

TESIS DOCTORAL

Fernando Manuel Uyaguari Uyaguari
Ingeniero de Sistemas

Dirigida por
Dr. Óscar Dieste y Dra. Sira Vegas

2017

Tribunal de Tesis

Dra. Natalia Juristo Juzgado

Universidad Politécnica de Madrid

(Presidenta)

Dr. Xavier Ferré Grau

Universidad Politécnica de Madrid

(Secretario)

Dra. Beatriz Bernárdez Jiménez

Universidad de Sevilla

(Vocal)

Dra. Marcela Fabiana Genero Bocco

Universidad de Castilla-La Mancha

(Vocal)

Dr. Jose Ignacio Panach Navarrete

Universidad de Valencia

(Vocal)

*A mis padres,
hermanos, sobrinos
y a la memoria de mi abuelita*

Agradecimientos

Esta Tesis es fruto del trabajo desarrollado durante casi cuatro años dentro del Grupo de Investigación en Ingeniería del Software Experimental (GrISE) de la UPM, liderado por la profesora Natalia Juristo, quién me permitió participar en el proyecto ESEIL, le expreso mi agradecimiento por compartir sus conocimientos y por su permanente apoyo durante el desarrollo de la Tesis. A mis tutores Óscar Dieste y Sira Vegas gracias por su apoyo constante y por los conocimientos compartidos. A los compañeros que colaboraron para que se pueda realizar esta investigación: Davide Fucci e Itir Karac de la U. de Oulu, Elizabeth Jaimes de la UPM.

Agradezco a Dios por las bendiciones recibidas, por permitirme vivir esta experiencia que la anhelaba desde hace varios años.

Agradezco a mis padres Manuel y Carmela, a mis hermanos Juan Pablo y Alvaro, a mis sobrinos Juan Diego, Pablito, Juanfer y Anita Belén por todo el cariño y el apoyo constante durante estos años lejos de casa, a pesar de la distancia parecía que estaban todo el tiempo cerca de mí.

A todos mis amigos y familiares que estuvieron presentes para compartir momentos durante estos años en España y en mis visitas a Ecuador. A Geovanny, Pablito, Marco Antonio, de manera especial a Cristina.

A mis compañeros del Doctorado gracias por el apoyo y el tiempo compartido en estos años de estudios: Edison, Marco, Rodrigo, Liliana, Jay, Richard, Patty, Roberto, Danilo, Jose.

A todas y cada una de las personas que de diferente manera contribuyeron a alcanzar este objetivo.

A todos, muchas gracias!

Un especial reconocimiento a la Secretaría Nacional de Educación Superior, Ciencia, Tecnología e Innovación (SENESCYT), organismo ecuatoriano que financió esta investigación.

Resumen

La experimentación se ha venido aplicando desde hace varias décadas como método de investigación en Ingeniería del Software. El incremento en la realización de experimentos ha sido notable en los últimos años. Sin embargo, hay ciertos aspectos relativos al proceso experimental que han sido poco estudiados y posiblemente estén afectando la validez de los resultados experimentales. En particular la operacionalización, que pertenece a la fase de diseño experimental, apenas ha sido tratada.

La operacionalización es el proceso por el cual los constructos que se desean estudiar, tanto en lo relativo a las variables dependientes como independientes, son asignados a métricas y factores medibles. Con frecuencia, la medición no es directa, sino que requiere de métodos de medición complejos, donde intervienen muchos elementos, tales como instrumentos, medidores y manipulaciones o intervenciones sobre las muestras a medir. Existen evidencias de que ciertos elementos de los métodos de medición (por ejemplo, los casos de prueba usados en la medición de muchos experimentos en TDD) producen resultados experimentales que no son consistentes entre sí.

El objetivo de esta Tesis es determinar qué componentes de los métodos de medición pueden influir en los resultados experimentales, y cuál es el origen de dichas variaciones.

Para realizar este estudio, se trabajó con una familia de experimentos en TDD realizados tanto en industria como en academia, dentro del proyecto FiDiPro ESEIL. La investigación se ha enfocado en tres elementos de los métodos de medida: Casos de prueba, medidores y tipos de intervención. Para realizar la investigación, se realizaron múltiples mediciones utilizando distintos casos de prueba, medidores y tipos de intervención. Las mediciones fueron posteriormente comparadas utilizando métodos provenientes de la ingeniería, como ISO 5725, las ciencias biomédicas, como es el caso de los diagramas Bland-Altman, y el Coeficiente de Correlación Intraclass, proveniente de la psicología.

Nuestros resultados muestran que la utilización de una u otra suite de casos de prueba modifica de forma sustancial los resultados de los análisis estadísticos de los experimentos. En contrapartida, los medidores y los tipos de intervención no influyen fuertemente en los resultados experimentales.

El aporte de esta Tesis es proporcionar visibilidad en la fase de operacionalización, poniendo a disposición de los experimentadores recomendaciones sobre ciertos aspectos de los métodos de medida que permitirán obtener resultados más precisos en los experimentos de Ingeniería del Software.

Abstract

Experimentation has been applied for several decades as a research method in Software Engineering. The increased number of conducted experiments has been remarkable in recent years. However, there are aspects related to the experimental process that have been seldom studied and may be affecting the validity of experimental results. In particular, operationalization, which belongs to the design phase, has scarcely been addressed.

Operationalization is the process where the constructs to be studied, both in terms of dependent and independent variables, are assigned to metrics and measurable factors. Measurement is often not direct but requires complex measurement methods, involving many elements, such as instruments, measurers and manipulations or interventions on the samples to be measured. There is evidence that certain elements of measurement methods (e.g. test cases used in the measurement of many TDD experiments) yield experimental results that are not consistent with each other.

The objective of this Thesis is to determine which components of measurement methods may influence experimental results and what the origin of these variations may be.

For this study we worked with a family of TDD experiments conducted both in industry and academia, within the FiDiPro ESEIL project. The research has focused on three elements of measurement methods: test cases, measurers and types of intervention. Measurements were subsequently compared using engineering methods as ISO 5275, methods applied in biomedical sciences such as Bland-Altman diagrams, and the Intraclass Correlation Coefficient from psychology.

Our results show that the use of one or another test cases substantially modifies the results of the experiment's statistical analysis. In contrast, measurers and types of intervention do not strongly influence experimental results.

The contribution of this thesis is to provide visibility about the operational phase, making available recommendations on certain aspects of measurement methods that experimenters may take into account to obtain more accurate experimental results in Software Engineering.

Índice abreviado

1. Introducción	1
2. Estado de la cuestión	15
3. Planteamiento del problema y proceso de resolución	39
4. Familia ESEIL de experimentos sobre TDD	47
5. Medición de los experimentos de la familia estudiada: Operacionalización	71
6. Medición de los experimentos de la familia estudiada: Tipos de intervención	97
7. Estrategias de comparación de resultados	113
8. Estudio de las diferencias entre test suites	121
9. Estudio de las diferencias entre medidores	137
10. Estudio de las diferencias entre tipos de intervención	149
11. Discusión	161
12. Amenazas a la validez	169
13. Conclusiones y futuras líneas	173
Bibliografía	177
A. Estudios empíricos en TDD	191
B. Especificación de las tareas	199
C. Protocolo de medición	213
D. Código fuente obtenido de los sujetos experimentales	227
E. Casos de prueba	231

F. Código fuente para la medición

253

Índice general

1. Introducción	1
1.1. Área de investigación	1
1.2. Dominio del problema	2
1.2.1. Experimento y proceso experimental	2
1.2.2. Operacionalización	3
1.2.3. Medición	5
1.2.4. Fiabilidad	6
1.3. Problema abordado	7
1.4. Objetivos	8
1.5. Aproximación a la solución	9
1.6. Publicaciones	10
1.7. Estructura de este documento	12
2. Estado de la cuestión	15
2.1. Estudios secundarios sobre TDD	15
2.1.1. Revisión sistemática de Kollanus [104]	18
2.1.2. Revisión sistemática de Causevic et. al [31]	19
2.1.3. Meta-análisis de Rafique & Misic [145]	21
2.1.4. Revisión sistemática de Munir et. al [134]	22
2.1.5. Análisis comparativo de estudios empíricos de los efectos de TDD de Makinen & Munch [121]	23
2.1.6. Revisión sistemática de Bissi et. al [18]	24
2.2. Experimentos controlados sobre TDD	25
2.3. Diferencias instrumentales entre los experimentos de TDD	31
2.3.1. Operacionalización e instrumentación de variables respuesta en ex- perimentos	31
2.3.2. Casos de prueba	34
2.3.3. Tarea experimental	36
3. Planteamiento del problema y proceso de resolución	39
4. Familia ESEIL de experimentos sobre TDD	47
4.1. Descripción del experimento	49
4.1.1. Factores y tratamientos	49
4.1.2. Variable respuesta	50
4.1.3. Tareas experimentales	52

4.1.4.	Casos de prueba de aceptación para obtención de métricas	53
4.2.	Operacionalización	56
4.3.	Replicaciones ejecutadas	58
4.3.1.	FSecure	58
4.3.2.	Elektrobit	61
4.3.3.	UPV	62
4.3.4.	UPM Seminario 2014	63
4.3.5.	Playtech	63
4.3.6.	ESPEL	64
4.3.7.	Universidad de Basilicata	65
4.3.8.	Ericsson	65
4.3.9.	University of Southern Denmark	66
4.3.10.	Mapfre	67
4.3.11.	PAF	68
4.4.	Diferencias de diseños	68
5.	Medición de los experimentos de la familia estudiada: Operacionalización	71
5.1.	Selección de la métrica	72
5.1.1.	Elaboración de la variable respuesta	72
5.1.2.	Definición de la métrica	77
5.2.	Preparación del instrumento de medición	79
5.2.1.	Definición del instrumento de medición	79
5.2.1.1.	Casos de prueba Ad hoc	79
5.2.1.2.	Casos de prueba con Particiones de Clases de Equivalencia	81
5.2.2.	Implementación del instrumento de medición	83
5.2.3.	Casos de prueba con PCdE - MR	84
5.2.4.	Verificación del instrumento de medición	85
5.3.	Medición	85
5.3.1.	Recepción del resultado de la operación experimental	85
5.3.2.	Aplicación del instrumento de medición	86
5.3.2.1.	Ambiente de desarrollo 1: Java, Eclipse - Manual	88
5.3.2.2.	Ambiente de desarrollo 2: Java-Eclipse Automático	89
5.3.2.3.	Ambiente de desarrollo 3: C++, Eclipse, Google Test	91
5.3.2.4.	Ambiente de desarrollo 4: C++, Eclipse, Boost	92
5.3.2.5.	Ambiente de desarrollo 5: C#, Visual Studio (VS), VS Testing Tool	93
5.3.3.	Obtención de los valores de la variable respuesta	94
6.	Medición de los experimentos de la familia estudiada: Tipos de intervención	97
6.1.	Medición sin intervención	97
6.2.	Medición con Intervención Sintáctica	100
6.3.	Medición con Intervención Semántica	104
6.3.1.	Revisión de las clases del código fuente	104
6.3.2.	Revisión de errores con la ayuda del depurador	105
6.3.3.	Revisión de los casos de prueba que fallan	106
6.4.	Experimentos tratados	108

7. Estrategias de comparación de resultados	113
7.1. Aproximación a la comparación	113
7.2. Exactitud	115
7.3. Incertidumbre expandida	117
7.4. Representación gráfica	117
7.5. Medición en las ciencias del comportamiento	118
8. Estudio de las diferencias entre test suites	121
8.1. Características generales de los datos	121
8.1.1. Experimentos	121
8.1.2. Mediciones	121
8.2. Análisis exploratorio	121
8.2.1. Distribución de los datos	122
8.2.2. Correlaciones	122
8.2.3. Análisis estadístico de experimentos	124
8.3. Análisis de reproducibilidad	125
8.4. Gráficos de Bland-Altman	126
8.4.1. Gráfico conjunto de los experimentos PT y EC	126
8.4.2. Gráficos diferenciados por experimento	126
8.4.3. Estudio gráfico de posibles variables influyentes	128
8.4.3.1. Factor <i>Tarea</i>	128
8.4.3.2. Factor <i>Grupo</i>	129
8.4.3.3. Posibles razones de las variaciones	129
8.5. ICC	130
8.6. Conclusiones	131
8.6.1. Respecto a la influencia de las test suites	131
8.6.2. Motivos de la diferencia entre mediciones generadas por Ad hoc y PCdE	131
8.6.3. Respecto a los tipos de análisis	134
9. Estudio de las diferencias entre medidores	137
9.1. Características generales de los datos	137
9.1.1. Experimentos	137
9.1.2. Mediciones	137
9.2. Análisis exploratorio	137
9.2.1. Distribución de los datos	137
9.2.2. Correlaciones	138
9.2.3. Análisis estadístico de experimentos	139
9.3. Análisis de reproducibilidad	140
9.4. Gráficos de Bland-Altman	141
9.4.1. Gráfico conjunto de los tres medidores	141
9.4.2. Gráficos diferenciados por experimento	142
9.4.3. Estudio gráfico de posibles variables influyentes	143
9.5. ICC	144
9.6. Conclusiones	145
9.6.1. Respecto a la influencia del medidor	146
9.6.2. Respecto a los tipos de análisis	146

9.6.3. Respecto a los tipos de intervención	147
10. Estudio de las diferencias entre tipos de intervención	149
10.1. Características generales de los datos	149
10.1.1. Experimentos	149
10.1.2. Mediciones	149
10.2. Análisis exploratorio	149
10.2.1. Distribución de los datos	149
10.2.2. Correlaciones	150
10.2.3. Análisis estadístico de experimentos	153
10.3. Análisis de reproducibilidad	154
10.4. Gráficos de Bland-Altman	155
10.4.1. Gráfico conjunto de los tres tipos de intervención	155
10.4.2. Gráficos diferenciados por experimento	157
10.4.3. Estudio gráfico de posibles variables influyentes	158
10.5. ICC	158
10.6. Conclusiones	159
10.6.1. Respecto a la influencia del tipo de intervención	159
10.6.2. Respecto a los tipos de análisis	159
11. Discusión	161
11.1. Resultados principales	161
11.2. Métodos de análisis de diferencias entre mediciones	162
11.3. Generalización	163
12. Amenazas a la validez	169
12.1. Validez de la conclusión	169
12.2. Validez interna	169
12.3. Validez de constructo	170
12.4. Validez externa	170
13. Conclusiones y futuras líneas	173
13.1. Visión general	173
13.2. Actividades	173
13.3. Resultados	175
13.4. Recomendaciones generales	175
13.5. Líneas futuras de investigación	176
Bibliografía	177
A. Estudios empíricos en TDD	191
B. Especificación de las tareas	199
B.1. MR especificación de grano grueso	199
B.2. MR especificación descompuesta en nanotareas	201
B.3. BSK especificación de grano grueso	202
B.4. BSK especificación descompuesta en nanotareas	204
B.5. Spreadsheet	207

B.6. Music Phone	208
B.7. Sudoku	211
C. Protocolo de medición	213
C.1. Medición de experimentos	214
C.2. Operacionalización de experimentos de la literatura	219
D. Código fuente obtenido de los sujetos experimentales	227
E. Casos de prueba	231
E.1. Casos de prueba Ad hoc	231
E.1.1. Casos de prueba Ad hoc - MR	231
E.1.2. Casos de prueba Ad hoc- BSK	237
E.2. Casos de prueba con PCdE	240
E.2.1. Clases de equivalencia - MR	240
E.2.2. Casos de prueba - MR	243
E.2.3. Clases de equivalencia - BSK	245
E.2.4. Casos de prueba - BSK	250
F. Código fuente para la medición	253
F.1. Métodos para medir con PCdE	253
F.1.1. Método gotoStart()	253
F.1.2. Método getSize()	253
F.1.3. Método public boolean findBonus()	254
F.2. Ejemplo de intervención en la elaboración de la variable que retorna los valores de un método	255
F.3. Ejemplo de intervención en la modificación del formato de salida de un método	257
F.4. Ejemplos de implementación del Bonus en la tarea BSK	262
F.5. Ejemplos de control de entradas inválidas en las tareas	265

Índice de figuras

1.1. Operacionalización de un experimento	4
1.2. Fiabilidad y validez [36]	6
2.1. Estudios en TDD	26
3.1. Esquema del proceso	41
5.1. Procedimiento de medición de los experimentos en TDD	72
5.2. Aplicación del instrumento de medición al código fuente	86
5.3. Visión parcial ejemplo del entorno de desarrollo 1: Java, Eclipse - Manual	88
5.4. Resultado de las aserciones en la hoja de cálculo para UPV	89
5.5. Código fuente del fichero: <i>AllTests.java</i>	90
5.6. Visión parcial ejemplo del entorno de desarrollo 2: Java-Eclipse Automático	91
5.7. Visión parcial ejemplo del entorno de desarrollo 3: C++, Google Test	92
5.8. Visión parcial ejemplo del entorno de desarrollo 4: C++, Boost	93
5.9. Visión parcial ejemplo del entorno de desarrollo 5: Visual Studio	94
5.10. Visión ejemplo del resultado de una medición	95
5.11. Visión ejemplo de una hoja de información de una medición	96
5.12. Visión de la interfase correspondiente al repositorio	96
6.1. Verificación del nombre de las librerías JRE y JUnit	98
6.2. Verificación del nombre del paquete importado	98
6.3. Vista parcial de de casos de prueba que dan Error o Falla en el IDE Eclipse	107
7.1. Componentes de exactitud según VIM	115
7.2. Gráfico de Bland-Altman	118
8.1. Gráficos de densidad para PT y EC	122
8.2. Gráficos de dispersión para PT y EC (recta de regresión en azul, concordancia perfecta en rojo).	123
8.3. Gráficos de Bland-Altman para los experimentos PT y EC conjuntamente	127
8.4. Gráficos de Bland-Altman para los experimentos PT y EC, por separado	127
8.5. Gráficos de Bland-Altman para los experimentos PT, separados por tarea	128
8.6. Gráficos de Bland-Altman para los experimentos EC, separados por tarea	129
8.7. Gráficos de Bland-Altman para los experimentos PT, separados por grupo	130
8.8. Gráficos de Bland-Altman para los experimentos EC, separados por grupo	130
8.9. Calidades generadas por las test suites utilizando el código de referencia MR	133

8.10. Calidades generadas por las test suites utilizando el código de referencia BSK	133
8.11. Calidades generadas por las test suites PCdE utilizando el código de referencia BSK con control de errores (izquierda) y sin casos de prueba de condiciones excepcionales (derecha)	134
9.1. QLTY density plot for FS, PT and DI	138
9.2. Gráficos de dispersión para FS, PT y DI (recta de regresión en azul, concordancia perfecta en rojo).	139
9.3. Gráfico de Bland-Altman (modificado) para los tres medidores conjuntamente	142
9.4. Gráficos de Bland-Altman para los experimentos FS, PT y DI	143
9.5. Gráficos de Bland-Altman para el experimento FS, diferenciando las tareas MR y BSK	144
9.6. Gráficos de Bland-Altman para el experimento PT, diferenciando las tareas MR y BSK	145
9.7. Gráficos de Bland-Altman por tipo de intervención	145
10.1. Diagrama de densidad para FS y PT	150
10.2. Correlaciones entre tipos de intervención	152
10.3. Correlaciones entre tipos de intervención, excluyendo ceros	152
10.4. Correlaciones entre tipos de intervención en FS	153
10.5. Correlaciones entre tipos de intervención en PT	153
10.6. Gráfico de Bland-Altman (modificado) para los tres tipos de intervención conjuntamente	156
10.7. Gráficos de Bland-Altman para los tipos de intervención, a pares	156
10.8. Gráficos de Bland-Altman para los tipos de intervención, a pares, para el experimento FS	157
10.9. Gráficos de Bland-Altman para los tipos de intervención, a pares, para el experimento PT	158
10.10. Gráficos de Bland-Altman para los tipos de intervención, a pares, para el experimento PT y grupo $MR \rightarrow BSK$	158
10.11. Gráficos de Bland-Altman para los tipos de intervención, a pares, para el experimento PT y grupo $BSK \rightarrow MR$	159
D.1. Ejemplo de código fuente de un sujeto	230
F.1. Método <i>gotoStart()</i>	253
F.2. Método <i>getSize()</i>	253
F.3. Método <i>findBonus()</i>	254
F.4. Método <i>executeCommad()</i> del sujeto FSK04	256
F.5. Intervención en el <i>return</i> del método <i>executeCommad()</i> del sujeto FSK04 .	256
F.6. Sentencia <i>return</i> del método <i>executeCommand</i> del sujeto PTE011	257
F.7. Intervención en la sentencia <i>return</i> del método <i>executeCommand</i> del sujeto PTE011	257
F.8. Formato incorrecto para la orientación del Rover, variable <i>dir</i>	259
F.9. Formato corregido para la orientación del Rover, variable <i>dir</i>	259
F.10. Uso de formato de tipo entero para la orientación del Rover	260
F.11. Uso de formato de tipo caracter para la orientación del Rover	260

F.12.MR: Variables de salida X e Y intercambiadas	261
F.13.MR: Variables de salida X e Y corregidas	261
F.14.Implementación de Bonus en la tarea BSK	263
F.15.Alternativa de implementación de Bonus en la tarea BSK	264
F.16.Control de entradas inválidas mediante alternativas válidas	266
F.17.Control de entradas inválidas mediante la clase <i>Exception</i>	266
F.18.Control de entradas inválidas mediante clase alternativa <i>customException</i> .	267

Índice de tablas

2.1. Estudios secundarios sobre TDD	16
2.2. Experimentos controlados estudiados en la revisión sistemática de Kollanus [104]	18
2.3. Experimentos controlados en la industria en Causevic et. al [31]	19
2.4. Experimentos controlados con profesionales y estudiantes en Causevic et. al [31]	19
2.5. Experimentos controlados en la academia en Causevic et. al [31]	20
2.6. Experimentos controlados en la academia según Rafique & Misic [145] . . .	21
2.7. Experimentos controlados en la industria según Rafique & Misic [145] . . .	22
2.8. Experimentos de la categoría B2 según Munir et. al [134]	23
2.9. Experimentos de la categoría C según Munir et. al [134]	23
2.10. Experimentos controlados tomados en cuenta en Makinen & Munch [121] .	24
2.11. Experimentos controlados en la academia según Bissi et. al [18]	25
2.12. Experimentos controlados en la industria según Bissi et. al [18]	25
2.13. Estudios empíricos de TDD no incluidos en las revisiones sistemáticas . . .	27
2.14. Experimentos en TDD que estudian calidad	29
2.15. Protocolo de Medición del Experimento TDD en Java (Playtech) - Operacionalización del constructo de la variable respuesta calidad	32
2.16. Operacionalización del constructo de la variable respuesta calidad del Experimento Erdogmus et. al [50]	33
2.17. Casos de prueba en los experimentos en TDD	35
2.18. Tarea Bowling Score Keeper en los experimentos en TDD	37
3.1. Mediciones de los experimentos con distintos tipos de intervenciones	42
3.2. Mediciones realizadas por otros medidores	43
3.3. Repetición de mediciones de los experimentos con el mismo medidor	43
3.4. Mediciones de experimentos	44
4.1. Familia ESEIL de experimentos en TDD	48
4.2. Factores y tratamientos por experimento	50
4.3. Composición de la funcionalidad según ISO/IEC 25010:2011 [84]	50
4.4. Variables respuesta por experimento	51
4.5. Tareas experimentales por experimento	53
4.6. Total de casos de prueba y aserciones por cada suite de casos de prueba . .	54
4.7. Número de casos de prueba y aserciones Ad hoc	54
4.8. Número casos de prueba y aserciones con PCdE	54

4.9. Número casos de prueba y aserciones de la tarea MP	55
4.10. Número casos de prueba y aserciones de la tarea Spreadsheet	55
4.11. Número casos de prueba y aserciones de la tarea Sudoku	56
4.12. Suite de casos de prueba por experimento	56
4.13. Cronograma del experimento FSecure (Planeado)	57
4.14. Roles por experimento	58
4.15. Configuraciones de los experimentos - Parte 1	59
4.16. Configuraciones de los experimentos - Parte 2	60
4.17. Diseño del experimento FSecure	61
4.18. Diseño del experimento Elektrobit	62
4.19. Diseño del experimento UPV	62
4.20. Diseño del experimento Seminario UPM	63
4.21. Diseño del experimento Playtech	64
4.22. Diseño del experimento ESPEL	64
4.23. Diseño del experimento U. de Basilicata	65
4.24. Diseño del experimento Ericsson	66
4.25. Diseño del experimento USD	66
4.26. Diseño del experimento Mapfre	67
4.27. Diseño del experimento PAF	68
4.28. Diseños usados en la familia de experimentos en TDD	70
5.1. Protocolo de medición del experimento TDD en Java (FSecure) - Operacio- nalización del constructo de la variable respuesta calidad	73
5.2. Operacionalización del constructo de la variable respuesta calidad del expe- rimento George, con estudiantes [65]	75
5.3. Operacionalización del constructo de la variable respuesta calidad del expe- rimento George, con profesionales [65]	76
5.4. Definición del objetivo de calidad	77
5.5. Casos de prueba Ad hoc para probar movimientos básicos de Mars Rover	80
5.6. Definición de clases de equivalencia de los movimientos básicos (f, b, l, r) de la tarea MR	82
5.7. Casos de prueba con PCdE de los movimientos básicos de la tarea MR	84
5.8. Ambientes de pruebas para las mediciones	87
6.1. Mediciones de los experimentos	109
8.1. Características de los experimentos que usaremos en el estudio de las dife- rencias entre test suites	121
8.2. Correlaciones entre test suites para los experimentos PT y EC	123
8.3. Análisis de la variable respuesta QLTY para el experimento PT	124
8.4. Análisis de la variable respuesta QLTY para el experimento EC	125
8.5. Análisis conjunto de reproducibilidad	125
8.6. Análisis de reproducibilidad para los experimentos PT y EC	126
9.1. Características de los experimentos que usaremos en el estudio de las dife- rencias entre medidores	137
9.2. Correlaciones entre medidores para los experimentos FS, PT y DI	138

9.3. Correlaciones entre los medidores 1, 2 y 3	138
9.4. Análisis de la variable respuesta QLTY para el experimento FS	139
9.5. Análisis de la variable respuesta QLTY para el experimento PT	140
9.6. Análisis de la variable respuesta QLTY para el experimento DI	140
9.7. Análisis conjunto de reproducibilidad	141
9.8. Análisis de reproducibilidad para los experientos FS, PT y DI	141
10.1. Características de los experimentos que usaremos en el estudio de las dife- rencias entre tipos de intervención	149
10.2. Correlaciones entre tipos de intervención	150
10.3. Correlaciones entre tipos de intervención para FS	151
10.4. Correlaciones entre tipos de intervención para PT	151
10.5. Análisis de la variable respuesta QLTY para el experimento FS	154
10.6. Análisis de la variable respuesta QLTY para el experimento PT	154
10.7. Análisis conjunto de reproducibilidad	155
10.8. Análisis de reproducibilidad para los experientos FS y PT	155
11.1. Impacto de los elementos estudiados	166
A.1. Estudios empíricos en TDD	192
B.1. Composición de la característica Funcionalidad	209
C.1. Protocolo de medición del experimento TDD en C++ (Elektrobit) - Opera- cionalización del constructo de la variable respuesta calidad	215
C.2. Protocolo de medición del experimento TDD en Java (UPM) - Operaciona- lización del constructo de la variable respuesta calidad	216
C.3. Protocolo de medición del experimento TDD en Java (UPV) - Operaciona- lización del constructo de la variable respuesta calidad	217
C.4. Protocolo de medición del experimento TDD en Java (ESPEL) - Operacio- nalización del constructo de la variable respuesta calidad	218
C.5. Operacionalización del constructo de la variable respuesta calidad del expe- rimento de Canfora [25]	220
C.6. Operacionalización del constructo de la variable respuesta calidad del expe- rimento de Geras et.al [69]	221
C.7. Operacionalización del constructo de la variable respuesta calidad del expe- rimento de Edwards [48]	223
C.8. Operacionalización del constructo de la variable respuesta calidad del expe- rimento de Desai et. al [42]	225
E.1. Casos de prueba Ad hoc tarea: Mars Rover	232
E.2. Casos de prueba Ad hoc tarea: BSK, Clase Frame	237
E.3. Casos de prueba Ad hoc tarea: BSK, clase Bowling Game	238
E.4. Definición de clases de equivalencia de la tarea Mars Rover	240
E.5. Casos de prueba con PCdE de la tarea: MR	243
E.6. Definición de clases de equivalencia de la tarea BSK: Clase Frame	245
E.7. Definición de clases de equivalencia de la tarea BSK: Clase BowlingGame	245
E.8. Casos de prueba con PCdE de la tarea: BSK, clase Frame	250

E.9. Casos de prueba con PCdE de la tarea: BSK, clase BowlingGame	251
---	-----

Capítulo 1

Introducción

1.1. Área de investigación

Los ingenieros del software aplican métodos aprendidos durante su formación académica o en base a su experiencia para desarrollar sus productos. La industria del software toma decisiones relativas a los métodos, técnicas, metodologías, herramientas utilizadas, en base a percepciones, prejuicios y opiniones subjetivas [70]. En una gama de opciones para producir software no se tiene evidencia de que un enfoque o técnica sea mejor que otra [98].

Históricamente las ingenierías han surgido de prácticas ad hoc, en donde un conjunto de problemas debían ser resueltos y se resolvían de cualquier manera, hasta llegar a un estado maduro en donde los diseños y construcciones se basan en conocimiento científico [158]. La necesidad de trabajar con hechos antes que con suposiciones es una propiedad en cualquier disciplina de la ingeniería [98].

La experimentación contrasta las creencias con la realidad convirtiéndolas en hechos o desechándolas [158,159]. En las ciencias maduras, la construcción de teorías permite adquirir y acumular conocimientos. La experimentación logra [40]: Identificar y comprender las variables que juegan un papel en el fenómeno bajo estudio y las relaciones entre tales variables; por tanto aprender relaciones causa-efecto, estableciendo leyes y teorías que explican los comportamientos observados.

La mayoría de la investigación científica se basa en la experimentación, en la observación de un fenómeno manipulado para propósitos de investigación y en la medición de variables involucradas en el fenómeno [98]. En los experimentos el investigador controla la situación, directa, precisa y sistemáticamente el comportamiento del fenómeno a estudiar [173].

La principal razón para usar la experimentación en Ingeniería del Software (IS) es con el propósito de permitir el entendimiento e identificar las relaciones entre las distintas variables [173]. La experimentación se realiza para evaluar, predecir, entender, controlar y mejorar el producto y el proceso de desarrollo del software [9]. La experimentación descubre las relaciones entre las variables involucradas en un fenómeno [98]. Por ejemplo, mediante la experimentación podríamos conocer cómo el uso de un determinado método afecta la calidad del software generado usando dicho método. En IS los laboratorios consisten en estudiantes desarrollando software de juguete mientras que la realidad que simula el laboratorio son los profesionales desarrollando proyectos reales.

Cuando experimentamos en IS, los sujetos son asignados a diferentes tratamientos de forma aleatoria. El propósito es manipular una o más variables y controlar las otras varia-

bles en valores fijos. El efecto de la manipulación es medido y analizado estadísticamente [173]. Guías para el diseño experimental en IS se pueden encontrar en [9] [144] [98] [173].

La Ingeniería del Software Experimental traslada a la IS el paradigma experimental. Los primeros experimentos en ingeniería del software se llevaron a cabo en los años 60 por Grant y Sackmann [71], en donde se comparó la performance de programadores bajo ciertas condiciones controladas. A mediados de los 80, Basili [9] presenta un framework para experimentación determinando la necesidad de que la experimentación sea sistemática. Desde entonces se han realizado muchos experimentos. En estos últimos 15 años la cantidad de experimentos en IS se ha incrementado dramáticamente. Sin embargo aún queda mucho camino por recorrer para contar con una metodología experimental madura para IS. Esta tesis pretende ser una contribución metodológica a la maduración del paradigma experimental aplicado a IS.

1.2. Dominio del problema

1.2.1. Experimento y proceso experimental

Hemos considerado las siguientes definiciones de **experimento**:

- Un experimento es una alteración deliberada del valor de una o más propiedades para ver qué otras propiedades son afectadas y de ser posible en qué grado [24].
- Un experimento es un estudio en el que se introduce deliberadamente una intervención para observar sus efectos [157].

Un experimento posee dos tipos de variables destacadas: factores y variable respuesta.

Se denominan **factores** a cada característica del desarrollo de software a ser estudiado que afecte la variable respuesta. Los factores son intencionalmente variados durante el experimento y eso afecta el resultado del experimento [98].

La variable respuesta es la salida de un experimento. La variable respuesta de un experimento es la característica del proyecto, fase, producto o recurso que es medida para estudiar el efecto de los factores.

Un proceso ofrece los pasos para apoyar una actividad. Para ejecutar un experimento se debe tomar en cuenta varios pasos que se deben ejecutar en un determinado orden. Un **proceso experimental** se elabora para asegurar que se toman acciones apropiadas para realizar un experimento exitoso [173].

El proceso experimental implica una interacción de un proceso de hipótesis y prueba. Las hipótesis sobre los modelos, métodos, técnicas de ingeniería del software son comprobadas. El primer proceso experimental para IS lo define Basili [9] y consiste de cuatro fases: Definición, Planificación, Operación e Interpretación de un experimento.

La propuesta más reciente de Wohlin [173] y que nosotros seguimos en esta tesis. Según Wohlin las fases del proceso experimental son las siguientes: Alcance del experimento, planificación, operación, análisis e interpretación y empaquetado.

En la fase de **alcance del experimento** se define el objetivo del experimento y se fija claramente la hipótesis. También se define el alcance. El alcance puede hacerse mediante el framework GQM [8].

En la fase de **planificación** se determina en detalle el contexto del experimento. Se identifican los sujetos, si son profesionales o estudiantes, además el ámbito en donde se

ejecutará el experimento. Se definen los factores, las variables respuesta y la escala de medida.

Durante esta fase se determina el tipo de diseño a usar en el experimento. Se seleccionan y preparan los instrumentos.

Se evalúa la validez de los resultados, respecto a las distintas clases: validez interna, externa, de constructo y de conclusión.

La fase de **operacionalización** corresponde con la preparación de los sujetos y materiales para el experimento. Se ejecuta el experimento y se realiza la recogida de datos. Para la recogida de datos se usa los instrumentos definidos en la fase de planificación.

Este trabajo de tesis doctoral se enmarca en esta fase, específicamente la operacionalización de la variable respuesta.

Durante la fase de **análisis e interpretación**, se prueba la hipótesis. Dependiendo de las escalas de medida, valores de entrada y tipo de resultado que estamos buscando seleccionamos una prueba estadística.

Por último, en la fase de **presentación y empaquetado** los investigadores elaboran un documento con los resultados, podría ser algo como un documento de investigación para publicación. Esta fase implica también la elaboración de un paquete de laboratorio con propósitos de replicación del experimento.

1.2.2. Operacionalización

Para que una hipótesis pueda ser comprobada, es necesario que sea operacionalizada [24]. Operacionalización es el proceso de enlazar una definición conceptual a un conjunto específico de técnicas o procedimientos de medición [137]. La figura 1.1 (adaptada de [166,173]) muestra la tarea de la operacionalización.

Los constructos son los medios para conectar las operaciones usadas en los experimentos a la teoría. Los constructos científicos (conceptos, hipótesis, teorías) se refieren a hechos imperceptibles, como por ejemplo procesos mentales [24]. Cuando tenemos una hipótesis sobre una relación causa-efecto lo que tenemos es la creencia de una relación entre un constructo causa y un constructo efecto [173].

En esta investigación la operacionalización de la variable respuesta es nuestro principal interés. Más concretamente el método de medición en la operacionalización de la variable respuesta.

Según Bunge [24], la operacionalización de constructos está sujeta a distintos tipos de error, y se debería dar énfasis en:

- Operacionalizar cada constructo de varias formas dentro y entre los estudios.
- Examinar el patrón entre las características de la medida y del constructo.
- Poner atención en la calidad de la unión del constructo con las operaciones.

Para ilustrar la operacionalización vamos a utilizar el ejemplo de la operacionalización del constructo efecto: longitud de la mesa. La operacionalización consiste en el procedimiento para obtener el valor de la longitud de la mesa en una unidad de medida (centímetros). El instrumento de medida es la cinta métrica. La aplicación del instrumento de medida sobre el objeto a medir consiste en tender la cinta de medida sobre el largo de la mesa, haciendo coincidir el inicio de la cinta con el inicio de la mesa, y cubrir la cinta hasta el

final de la mesa. Por último, tomar la lectura en centímetros de la cinta de medida. De esta manera se ha obtenido valor de la longitud de la mesa.

Figura 1.1: Operacionalización de un experimento

El **método de medición** proporciona medios para ejecutar el experimento y monitorearlo, sin afectar el control del experimento. El resultado del experimento debería ser el mismo independientemente de cómo es medido. Si el método afecta los resultados del experimento, los resultados no son válidos [173]. Nótese que esta relación entre método de medición y resultado es desconocida hasta que se comprueban los resultados con diversos métodos.

Durante la fase de planificación del experimento se definen los métodos de medición. Los métodos contienen distintos tipos de elementos: objetos, guías e instrumentos de medición [173]. Por ejemplo, el código fuente generado por los sujetos experimentales durante un experimento de TDD son los objetos del método de medida, como guías tenemos los procedimientos para adaptar el código de los sujetos al contexto del método de medición (lo que denominamos “tipos de intervención”). Un ejemplo de instrumento de medición que será especialmente importante para nosotros son los casos de prueba que, en muchos casos, se utilizan para medir la calidad del código. Finalmente, no podemos olvidar al operador, esto es, a la persona que mide. En la mayoría de las ocasiones, en la medición de las variables respuesta implican algún tipo de decisión o intervención humana. Concretamente, los investigadores juegan el papel de medidores. Por ejemplo, calcular las líneas de código de un programa exige determinar si un comentario cuenta como línea de código o no.

En el ejemplo de la operacionalización del constructo efecto longitud de la mesa descrito en la subsección anterior, el objeto a medir es la mesa. Las guías o tipos de intervención es el modo de colocar la cinta métrica sobre la mesa; el instrumento es la cinta métrica y, finalmente el operador es medidor que realiza la medición.

Esta investigación se centra en los elementos de los métodos de medición antes mencionados guías, instrumentos, y el operador.

1.2.3. Medición

Según Kaplan [100] los científicos realizan los siguientes tipos de observación:

- Entidades observables: Tipos de cosas que se pueden percibir a través de sus sentidos. Por ejemplo, las condiciones de las mesas de trabajo, como la cantidad, estado, tamaño. Términos que son fáciles y seguros de verificar.
- Entidades indirectamente observables: Cosas que requieren observaciones más sutiles, complejas e indirecta. Se requieren inferencias referidas a presuntas conexiones causales entre lo que es observado directamente. Por ejemplo, lugar de nacimiento podría ser observable indirectamente, podríamos preguntar a los participantes del estudio dónde nacieron, pero hay posibilidades de no tener ninguna observación directa de aquellas personas han nacido en el lugar que informan.
- Constructos: Abstracciones que no pueden ser observadas directamente como la experiencia o la calidad. Kaplan, se refiere como constructos a estas cosas más abstractas que miden los científicos. Los constructos pueden ser definidos en base a los observables.

En IS hay atributos que se miden directamente y otros atributos que son medidos indirectamente. Por ejemplo, el número de líneas de código es una medida directa mientras que la productividad de un desarrollador se mide en base al número de casos de prueba pasados sería una medida indirecta.

Según Fenton [52], medición es el proceso por el cual los números son asignados a los atributos de las entidades en el mundo real de tal manera que los describen de acuerdo a reglas claramente definidas. Una entidad podría ser un objeto como una persona o una especificación de software. Un atributo es una característica de la entidad, por ejemplo la edad de una persona o el tamaño de una especificación.

Los tipos de escala más comunes son las siguientes [21,53]:

- Nominal: Esta escala asigna un nombre o símbolo al atributo de la entidad. Se puede ver como una clasificación de entidades de acuerdo al atributo. Por ejemplo: Cuando usamos 1: para masculino, 2 para femenino. No hay una noción de magnitud asociada con el número o símbolo.
- Ordinal: Ordena las entidades de acuerdo a un criterio de ordenamiento. Ejemplos de criterios de orden son: "mayor que", "menor que", "más complejo que".
- Intervalo: Esta escala ordena los valores, hay la noción de distancia relativa entre dos entidades. Por ejemplo, las medidas de temperatura en Celsius o Fahrenheit, la fecha.
- Ratio: Esta escala proporciona orden, intervalos iguales y ratios (proporciones) entre entidades. Se usa esta escala cuando existe un valor cero significativo. La asignación de medidas debería empezar en cero e incrementarse a intervalos iguales conocidos como unidades. Por ejemplo, se usa esta escala para medir la temperatura en Kelvin o la duración de una fase de desarrollo.

Se puede medir cuán numerosa es una colección o la intensidad de una propiedad de un objeto. La técnica de medición depende de la propiedad del objeto a medir [24]. La técnica o instrumento de medición requieren de su teoría propia ya que no existe un medidor universal.

1.2.4. Fiabilidad

Kan [99] considera que los criterios de medición de calidad más importante son la fiabilidad y la validez.

- **Fiabilidad:** Se refiere a la consistencia de un número de mediciones repetidas. Cuando las medidas repetidas son altamente consistentes entonces el método de medida tiene un alto grado de fiabilidad.
- **Validez:** Es el criterio que se refiere a si la métrica refleja el significado real del concepto que se está operacionalizando. Se le denomina validez de constructo.

En el gráfico 1.2 presentamos la relación entre fiabilidad y validez. Los puntos pequeños representan las mediciones mientras que el punto central de diámetro grande y color negro representa el valor real que se pretende medir.

Figura 1.2: Fiabilidad y validez [36]

Es posible tener una medida con alta fiabilidad y baja validez. Esta medida podría ser consistente en obtener mala información. En otro caso se podría tener una medida con baja fiabilidad y alta validez, en donde los valores obtenidos fluctúan bastante. En el mejor de los casos tenemos una medida que tiene alta validez y alta fiabilidad. Esta medida produce resultados consistentes en aplicaciones repetidas.

Cuando los conceptos son más abstractos entonces la evaluación de la validez se torna difícil [99], la IS no está lo suficientemente madura como para contar con métodos válidos y confiables para medir los constructos de interés en el desarrollo del software (calidad, productividad, eficacia, eficiencia, etc). Esta falta de métodos estándar de medición puede impactar los experimentos y sus resultados. **Esta investigación se ocupa de comprender el impacto que tienen en los resultados los elementos de los métodos de medición usados en el experimento para medir las variables respuesta.**

El error en la medición es universal y estará siempre presente. Cualquier observación está compuesta por el valor real más un valor de error [36]. Las mediciones están sujetas a errores de tipo aleatorios y de tipo sistemáticos [24].

- **Error aleatorio:** Este error es causado por cualquier factor que afecte aleatoriamente a la medición de la variable. El error aleatorio no tiene efectos consistentes en toda la muestra. El error aleatorio depende del tamaño de la muestra: cuando ésta aumenta, el error disminuye.

- Error sistemático: Es causado por cualquier factor que afecte sistemáticamente a la medición de una variable. Los errores sistemáticos se mantienen en todos o en la mayoría de miembros de un grupo. A diferencia de los errores aleatorios, los errores sistemáticos tienden a ser consistentemente positivos o negativos.

1.3. Problema abordado

En los experimentos en IS se usan diferentes instrumentos de medición, tales como: cuestionarios, ambientes de desarrollo, software de medición, compiladores, etc., con el objetivo de operacionalizar las variables respuesta. En las mediciones con distintos instrumentos hemos observado diferencias importantes. Creemos que la instrumentación podría tener relación con las variaciones observadas en el valor de las variables respuesta. Esto implica que los resultados experimentales podrían ser interpretados de diferentes maneras dependiendo del instrumento utilizado, hasta el punto de obtener conclusiones contradictorias. Es probable que otros elementos de los métodos de medición, tales como las guías y los operadores, también ejercen una influencia indeseable. En consecuencia, creemos necesario estudiar los diferentes elementos que forman parte del método de medición y determinar el impacto que podrían tener en el resultado del experimento.

Las carencias descritas en el punto anterior y la limitada investigación sobre este tema en IS, ponen en evidencia la importancia de la problemática que aborda esta investigación. Los problemas a resolver son los siguientes:

- Falta o carencia de un proceso formal de operacionalización de constructos de las variables respuesta en los experimentos.
- Desconocimiento del impacto de la medición en el resultado de los experimentos.

En el proceso de medición de los experimentos, se pueden identificar los siguientes problemas que describimos a continuación:

- Instrumentos con errores: Los instrumentos utilizados en un experimento son susceptibles de errores, por ejemplo, si usamos casos de prueba para las mediciones, los errores en los casos de prueba pueden darse, de manera particular cuando éstos no han sido validados. Los errores en casos de prueba pueden originarse en el diseño o en la implementación del caso de prueba. Por ejemplo casos de prueba con valores esperados incorrectos. Un caso de prueba con error dará un valor erróneo de medida.
- Ausencia de un criterio generalmente aceptado para el diseño de los instrumentos de medida. En los casos de prueba, el diseñador no considera una técnica o estrategia conocida para crear la suite de casos de prueba, sino lo hace en base a su criterio propio. Por ejemplo, las suites de casos de prueba podrían elaborarse con distintas técnicas: particiones de clases de equivalencia, análisis de valores límite o simplemente ad hoc es decir sin ninguna técnica en particular.

Dentro de este mismo aspecto, tenemos redundancias en el diseño de los casos de prueba. Regularmente un caso de prueba se elabora para probar una condición de software. Pero en la práctica, hemos visto casos de prueba ad hoc que prueban varias veces una misma condición del software. Lo cual probablemente en un caso favorable, incrementaría el valor del número de casos de prueba pasados, afectando el valor de la variable respuesta.

Cuando nos referimos a casos de prueba, tenemos también ambigüedades relativas a las alternativas en el valor de la salida esperada del caso de prueba. Si un caso de prueba tiene varios valores alternativos de salida, quedará a criterio del medidor cuál de ellos dar por válido en cada experimento. Esto genera variaciones entre las mediciones. Por ejemplo, en el caso de una tarea experimental para buscar obstáculos, alternativa 1: cuando se encuentra N veces un mismo obstáculo (X), el programa reporta una sola vez el obstáculo (o). Alternativa 2: cuando se encuentra N veces un mismo obstáculo (o), el programa lo reporta N veces ($ooooo..N$). Como se ve en el ejemplo son salidas válidas distintas de un mismo caso de prueba, lo que quedaría a criterio del medidor y causaría diferencias entre mediciones.

- Errores en tiempo de ejecución del instrumento de medición. Por ejemplo, cuando se están pasando los casos de prueba por el código, se detiene la ejecución debido a un error en la ejecución de un caso de prueba en particular. El caso de prueba específico termina su ejecución anormalmente, eso provoca la terminación de la ejecución de toda la suite de casos de prueba de manera anormal. Esto sucede cuando se usa el lenguaje C++. Este tipo de errores son otra fuente de variación en las mediciones.
- Influencia del medidor en el código fuente: Para conectar los casos de prueba con el código fuente del sujeto, cada medidor realiza sus operaciones siguiendo su procedimiento y aplicando su criterio. En ciertas ocasiones el medidor hace cambios en el código fuente del sujeto. Estos cambios pueden ser de distintos tipos, según criterio del medidor. Las distintas intervenciones de los medidores pueden ser otra fuente de variación.
- Intervención en el código fuente de los casos de prueba: Usualmente el medidor interviene en el código fuente del sujeto, pero a veces es necesario hacer cambios en el código de los casos de prueba. Entre los cambios se pueden tener: creación de métodos, modificación de métodos etc. Dependiendo del tipo de cambios, esto es, la intervención realizada, los resultados de la medición pueden cambiar.
- Uso de librerías externas adecuadas: La medición a veces también depende del uso de las librerías externas adecuadas. Por ejemplo, en las mediciones en lenguaje Java se usa una librería externa (hamcrest-all-1.3). Si no se instala esta librería se tienen valores cero como resultado de la medición. Lo cual también puede ser el origen de variaciones en las mediciones.

Considerando los problemas señalados anteriormente relativos a la medición de experimentos en IS, es notorio que hay diversidad de aspectos de la medición que actualmente no se consideran y que dan lugar a diferencias importantes en la operacionalización de las variables respuesta.

1.4. Objetivos

La presente tesis doctoral se enmarca dentro del área de la IS Experimental; más concretamente en el análisis del impacto de los métodos de medición en los experimentos de IS. Las variables respuesta que se miden en los experimentos de IS hacen referencia con mucha frecuencia a aspectos no directamente observables, por ejemplo, efectividad con que realizan las tareas los usuarios, calidad del software, tamaño de la funcionalidad. Los experimentadores definen con frecuencia métricas ad hoc para dichas variables, sin conocer

su fiabilidad ni validez. Por ejemplo, tamaño del software medido en base a líneas de código, el tamaño de la funcionalidad medido en base a puntos de función, tamaño de funcionalidad en base a casos de uso.

En estas circunstancias, el uso de distintos instrumentos, guías u operadores de medición podría alterar sustancialmente los resultados de las mediciones. Estas observaciones se ven reflejados en lo que llamamos datos crudos de un experimento. Los cálculos del valor de las variables respuesta se hacen en base a los datos crudos, por lo tanto si se afectan las variables puede afectarse también la prueba de hipótesis. En consecuencia afectaríamos el resultado de la prueba y el resultado del experimento.

Esta investigación se plantea la siguiente pregunta de investigación:

PI1: ¿Cómo afecta a los resultados de los experimentos utilizar distintos casos de prueba?

PI2: ¿Cómo afecta a los resultados de los experimentos utilizar distintos tipos de intervención?

PI3: ¿Cómo afecta a los resultados de los experimentos utilizar distintos medidores?

1.5. Aproximación a la solución

Para realizar el estudio contamos con una familia de experimentos en el marco del proyecto Fidipro:ESEIL (Experimental Software Engineering Industrial Laboratory). En la familia de experimentos se estudia Test Driven Development (TDD) desde el aspecto de la calidad. Por lo tanto, contamos con un banco de pruebas que nos permitirá obtener mediciones para realizar el estudio.

De acuerdo a la experiencia del grupo de investigación en donde se desarrolla el estudio y a las primeras mediciones de los experimentos de la familia, se cuenta con una primera aproximación a un proceso de medición de experimentos de TDD. El proceso de medición involucra varios elementos distintos tales como instrumentos, medidores y manipulaciones.

Con el propósito de no limitarnos a la experiencia del grupo de investigación y conocer como operacionalizan otros experimentos similares, se ha revisando en la literatura científica reportes de experimentos en TDD que tienen como variable respuesta la calidad.

En el proceso de operacionalización de las variables respuesta se asignan métricas a los constructos. En la medición de experimentos en TDD se hace uso de casos de prueba de aceptación para operacionalizar la variable respuesta calidad. La medida se obtiene en base a los casos de prueba pasados. Se pasa la suite de casos de prueba por el código fuente de los sujetos y se cuentan los casos de prueba pasados.

En los casos de prueba de los experimentos se han encontrado aspectos como la ausencia de un criterio generalmente aceptado para diseñar los casos de prueba utilizados en la medición. Si los casos de prueba están incompletos no evaluarían correctamente el código y darían una medición equivocada.

Cada medidor realiza sus operaciones siguiendo su procedimiento y aplicando su criterio a la hora de conectar el instrumento de medida con el código fuente. Por lo tanto, la intervención del medidor en el código fuente podrían influir en los resultados de la medición.

Hemos seleccionado los elementos que creemos tienen mayor influencia en los resultados de la medición. Los casos de prueba, medidor y el tipo de intervención en el código fuente.

Para realizar el estudio vamos a operacionalizar de distintas maneras los experimentos de la familia TDD. Se harán las mediciones requeridas para el análisis de la influencia de los casos de prueba, luego las mediciones requeridas con distintos medidores y tipos de intervención para estudiar su influencia en los experimentos.

Para estudiar los casos de prueba vamos a contar con dos suites distintas de casos de prueba elaborados con una técnica diferente: Ad hoc y particiones de clases de equivalencia. Para la influencia de los medidores, contamos con los medidores que van a colaborar y realizamos las mediciones tanto para la medición sin protocolo y como para la medición con protocolo. Se implementarán distintos tipos de intervención para analizar su influencia en los resultados experimentales.

Mediante una estrategia de comparación, tomada de otras ciencias ya que no tenemos en IS. Vamos a determinar la incertidumbre de reproducibilidad en base al concepto del estándar ISO 5725, luego graficar las diferencias mediante Bland-Altman y por último calcular el Coeficiente de Correlación Intraclass para cada uno de los elementos de medición estudiados.

De existir estas diferencias nos plantearemos si los elementos de la medición de la calidad está afectando los resultados de los experimentos, adicionalmente cómo afectan las decisiones de los medidores a los resultados experimentales. Por último, daremos sugerencias a los experimentadores que les permitan obtener resultados más precisos en las mediciones de sus experimentos e intentaremos generalizar los resultados a otras poblaciones o situaciones de la IS.

1.6. Publicaciones

Se han publicado los siguientes artículos relacionados con la investigación de esta tesis:

- O. Dieste, A. Aranda, F. Uyaguari, B. Turhan, A. Tosun, D. Fucci, M. Oivo, N. Juristo, *Empirical Evaluation of the Effects of Experience on Code Quality and Programmer Productivity: An Exploratory Study*, Empirical Software Engineering Journal, , pp. 1-86, 2017.

Este artículo [43] recoge los experimentos Elektrobit, Playtech, Ericsson, UPV, UPM Seminario 2014 (medidos todos ellos por el doctorando). El paper discute el impacto de la experiencia en las variables respuesta de los experimentos.

- D. Fucci, G. Scanniello, S. Romano, M. Shepperd, B. Sigweni, F. Uyaguari, B. Turhan, N. Juristo, M. Oivo, *An External Replication on the Effects of Test-driven Development Using a Multi-site Blind Analysis Approach*, Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM'16, ACM. (Best Full Paper Award)

Este artículo [59] reporta el experimento de la Universidad de Basilicata (medido por el doctorando). Se propone la adopción en IS de la práctica de análisis ciego de datos para que el analista no influya en los resultados de los experimentos.

Se ha enviado el siguiente artículo:

- A. Santos, S. Vegas, F. Uyaguari, O. Dieste, B. Turham, N. Juristo *Increasing Validity through Replication: A TDD Experiment*

En este artículo se replica el experimento en TDD y se discute las amenazas a la validez que podrían explicar las diferencias en los resultados correspondientes a calidad en los experimentos en TDD. El experimento correspondiente es Seminario UPM 2014 (medido por el doctorando).

Se encuentran en elaboración los artículos:

- O. Dieste, F. Uyaguari, S. Vegas, A. Santos, N. Juristo, *Using Test Suites to Measure Response Variables: Does the Choice of Test Cases Have an Effect on Experiment Results?*

En este artículo se han considerado las mediciones de los experimentos Playtech y Ericsson (medidos por el doctorando). Las mediciones se han realizado con casos de prueba Ad hoc y particiones de clases de equivalencia. Este paper reporta las diferencias obtenidas en las mediciones con distintos instrumentos, además analiza las fuentes de varianza de los resultados y el impacto que tiene en los resultados. Para ello usa tres métodos: ISO 5725, Bland-Altman y CCI.

- F. Uyaguari, O. Dieste, S. Vegas, D. Fucci, I. Karac, N. Juristo, *Influencia del Medidor en los Resultados de la Medición y de los Experimentos en Test-Driven Development.*

En este artículo se reporta las diferencias obtenidas en las mediciones realizadas con distintos medidores. Se analiza las fuentes de varianza de los resultados y el impacto que tiene en los resultados experimentales. Para ello usarán los métodos: ISO 5725, Bland-Altman y CCI. Se han considerado las mediciones de los experimentos Playtech y FSecure. Las mediciones han sido realizadas por dos medidores que usan los mismos instrumentos de medida y lo aplican al código fuente de los mismos experimentos.

En un futuro cercano esperamos preparar los siguientes artículos:

- F. Uyaguari, O. Dieste, S. Vegas, N. Juristo, *Influencia del Tipo de Intervención en Código Fuente en los Resultados de la Medición y de los Experimentos en TDD.*

En este artículo se reportará las diferencias obtenidas en las mediciones realizadas a un mismo experimento con distintos tipos de intervención en el código fuente de los sujetos. Se analiza las fuentes de varianza de los resultados y el impacto que tiene en los resultados experimentales. Para ello usarán los métodos: ISO 5725 y CCI. Se han considerado las mediciones de los experimentos Playtech y FSecure. Las mediciones han sido realizadas por un mismo medidor interviniendo de varios modos en el código fuente: Sin intervención, sintáctica y semántica.

- F. Uyaguari, O. Dieste, S. Vegas, N. Juristo, *Influencia de la Operacionalización en Experimentos en Ingeniería del Software.*

En este artículo se reportará el análisis de la influencia de los instrumentos en los experimentos en IS. Se recopila lo obtenido en los papers anteriores respecto a los casos de prueba, medidores y tipos de intervención. Se analiza en general la influencia de la instrumentación en los experimentos. Se anotan sugerencias a tomar en cuenta para operacionalizar variables respuesta en los experimentos.

1.7. Estructura de este documento

La investigación presentada en este documento se ha organizado en los siguientes capítulos:

- **Capítulo 1: Introducción.** En este capítulo se da una visión general del área abordada y se explica sobre el problema de investigación. Fijamos los objetivos de investigación y ofrecemos una vista general de la solución.
- **Capítulo 2: Estado de la cuestión.** Realizamos una revisión de la literatura científica acerca de los estudios empíricos en TDD. Nos basamos en revisiones sistemáticas en TDD realizadas anteriormente. Seleccionamos los experimentos en TDD. Se analizan las diferencias instrumentales en los casos de prueba y en las tareas experimentales de los experimentos en TDD. También se revisa la literatura científica para la comparación de métricas.
- **Capítulo 3: Planteamiento del problema.** En este capítulo presentamos el método de investigación y la aproximación a la solución.
- **Capítulo 4: Familia ESEIL de experimentos sobre TDD.** La investigación se desarrolla en base a una familia de experimentos en TDD. Describimos el experimento base con su hipótesis, factores, tratamientos, variables respuesta, tareas experimentales y casos de prueba. Presentamos cada una de las replicaciones ejecutadas y la diferencia entre los diseños usados.
- **Capítulo 5: Medición de los experimentos de la familia estudiada: Operacionalización** En este capítulo presentamos el procedimiento seguido para obtener los datos para la investigación. Describimos las actividades: elaboración de la variable respuesta, definición de métrica, definición del instrumento de medición, implementación, verificación de instrumento, recepción del resultado experimental, aplicación del instrumento de medición y la obtención del resultado de la medición.
- **Capítulo 6: Medición de los experimentos de la familia estudiada: Tipos de intervención** Describimos los tipos de intervención en el código fuente, según el tipo de intervención tenemos: medición sin intervención, medición sintáctica y medición semántica. Por último describimos las mediciones realizadas en los experimentos.
- **Capítulo 7: Estrategias de comparación de resultados** Describimos la estrategia para hacer la comparativa de las mediciones con distintos instrumentos, medidores y tipos de intervención. Presentamos los métodos que vamos a utilizar en la comparativa tales como Incertidumbre de reproducibilidad, Bland-Altman, Coeficiente de correlación intraclase.
- **Capítulo 8: Estudio de las diferencias entre test suites** En este capítulo se analiza las diferencias entre las mediciones con distintos instrumentos, presentamos el análisis de reproducibilidad, se analiza el origen de las diferencias, el impacto en los resultados estadísticos y en los resultados experimentales.

- **Capítulo 9: Estudio de las diferencias entre medidores** Describimos el análisis sobre las mediciones realizadas con distintos medidores. Se presenta las diferencias entre las mediciones con distintos operadores. A continuación, se realiza el análisis de reproducibilidad y se muestran los gráficos de Bland-Altman y CCI. Por último se analiza el impacto de los medidores en los resultados experimentales.
- **Capítulo 10: Estudio de las diferencias entre tipos de intervención** En este capítulo se analiza diferencias de las mediciones con distintos tipos de intervención en el código fuente. El proceso de análisis es similar al análisis de las test suites y los medidores. Presentamos las diferencias, calculamos la incertidumbre de reproducibilidad, CCI y el impacto de los tipos de intervención en los resultados experimentales.
- **Capítulo 11: Discusión** En este capítulo se discute sobre los resultados principales de la investigación, los métodos de análisis y la generalización de resultados. Al final se elabora un cuadro que resume el impacto del análisis estadístico.
- **Capítulo 12: Amenazas a la validez** En este capítulo se describen las acciones realizadas durante la investigación para asegurar la validez del estudio respecto a la conclusión, constructo, generalización y validez interna.
- **Capítulo 13: Conclusiones y futuras líneas** Por último, tenemos un capítulo en donde damos una visión general en retrospectiva de la tesis, a continuación presentamos los resultados obtenidos, realizamos unas recomendaciones generales y anotamos las líneas futuras de investigación.

Capítulo 2

Estado de la cuestión

2.1. Estudios secundarios sobre TDD

La técnica de desarrollo Test Driven Development (TDD) se ha practicado desde los inicios de la programación [12] aunque TDD haya sido popularizada como una práctica de Extreme Programming [14]. TDD no es una técnica de testing, es una técnica de análisis y diseño, sirviendo para estructurar todas las actividades del desarrollo de software [12,13]. El desarrollador escribe los casos de prueba antes que el código fuente por lo tanto puede entender lo que el código necesita hacer, probar las preconcepciones de cómo funcionará el código y mejorar el diseño general del código de forma iterativa [13]. La técnica consiste en escribir la prueba unitaria para la funcionalidad que se quiere agregar, ejecutar todas las pruebas e identificar el nuevo fallo, escribir el código fuente suficiente para que pase el nuevo caso de prueba, ejecutar todas las pruebas, ver que todas sean exitosas y refactorizar para eliminar duplicaciones. El objetivo es llegar a tener código limpio que funcione [13].

TDD ha sido objeto de estudio por parte de los investigadores desde principios del nuevo siglo [104]. Se han realizado case studies, surveys y experimentos en el contexto académico e industrial. En ellos se compara TDD con las metodologías tradicionales desde diversos aspectos como la calidad y productividad entre otros. En lo relativo al interés de la industria por TDD, en una encuesta acerca de métodos ágiles realizada en el año 2012 [150], indica que TDD no es una metodología de desarrollo muy popular. De los 400 profesionales de más de 200 empresas, el 1,7 % de los encuestados lo reporta como utilizado [150], si bien vienen usando mayoritariamente otras prácticas ágiles [127].

Con el fin de identificar los experimentos existentes en TDD estudiamos las revisiones sistemáticas realizadas sobre TDD (Kollanus [104] en 2010, Causevic et. al [31] en 2011, Munir et. al [134] en 2014, Makinen & Munch [121] en 2014 y Bissi et. al [18] en 2016) así como el meta-análisis de Rafique & Misic [145] en 2013. En la tabla 2.1 presentamos un resumen de lo que indican las revisiones en TDD respecto a los resultados de los experimentos. Nos enfocamos en la influencia de TDD en calidad del producto software.

Tabla 2.1: Estudios secundarios sobre TDD

Estudio	Evento	Rango de años de estudios	Tipo de estudio secundario	N. estudios	Estudios empíricos contemplados	N. Experimentos	Calidad externa
Kollanus [104]	IEEE Seventh International Conference on the Quality of Information and Communications Technology, QUATIC 2010	2001-2010	Revisión sistemática de literatura	37	[17] [25] [26] [39] [42] [47] [48] [50] [58] [66] [67] [69] [72] [78] [89] [92] [85] [114] [115] [116] [120] [119] [124] [130] [136] [139] [146] [147] [154] [162] [163] [164] [169] [172] [174] [176] [177]	[25] [26] [42] [47] [48] [50] [58] [66] [67] [69] [72] [78] [89] [92] [115] [116] [120] [119] [130] [139] [169]	Se obtienen resultados no concluyentes e incluso contradictorios.
Causevic et. al [31]	IEEE Fourth International Conference on Software Testing, Verification and Validation, ICST 2011	2002-2010	Revisión sistemática de literatura	48	[1] [17] [25] [26] [27] [33] [38] [39] [46] [45] [50] [140] [57] [58] [67] [69] [72] [76] [79] [78] [89] [92] [85] [103] [106] [111] [113] [116] [117] [120] [118] [119] [122] [124] [128] [130] [131] [136] [153] [154] [155] [161] [162] [163] [164] [169] [170] [172]	[25] [26] [33] [45] [50] [57] [58] [67] [69] [72] [79] [78] [89] [92] [106] [116] [117] [120] [118] [119] [130] [169]	El autor encuentra 18 áreas de efecto de TDD y analiza 7 factores limitantes para la adopción de TDD en la industria. En el área de calidad de código el efecto de TDD es positivo.
Rafique & Mistic [145]	IEEE Transactions on Software Engineering, Vol. 39, No. 6, June 2013	2002-2011	Meta-análisis con respecto a calidad y productividad	25	[25] [42] [44] [47] [50] [58] [65] [69] [72] [78] [87] [101] [114] [115] [130] [136] [139] [138] [146] [164] [169] [172] [174] [175] [177]	[25] [42] [47] [50] [58] [65] [69] [72] [78] [87] [101] [115] [130] [139] [138] [169]	El autor concluye que TDD aumenta levemente la calidad externa en los experimentos académicos. En los experimentos industriales TDD permite un incremento más amplio de la calidad externa.
Munir et. al [134]	Information and Software Technology, Volume 56, Issue 4, 2014	2002-2012	Revisión sistemática considerando rigor y relevancia	41	[3] [5] [17] [25] [37] [38] [41] [42] [44] [45] [50] [54] [57] [58] [67] [69] [72] [77] [78] [89] [91] [86] [85] [105] [107] [115] [120] [119] [124] [125] [130] [136] [139] [138] [154] [162] [163] [164] [169] [171] [172]	[25] [42] [45] [50] [57] [58] [67] [69] [72] [77] [78] [89] [86] [105] [115] [120] [119] [130] [139] [138] [169]	En los estudios de la categoría B2(relevancia baja y rigor alto) no presentan diferencias TDD y tradicional. En la categoría C (relevancia baja y rigor) en uno favorece la calidad externa y en otro no presenta diferencias.

Tabla 2.1 – continuación

Estudio	Evento	Rango de años de estudios	Tipo de estudio secundario	N. estudios	Estudios empíricos con-templados	N. Experimentos	Calidad externa
Mäkiten & Münch [121]	Springer, Proceedings of the 6th International Conference Software Quality Days, SWQD 2014	2003-2012	Estudio comparativo de los efectos de TDD.	19	[17] [25] [42] [44] [66] [69] [72] [78] [89] [85] [120] [119] [124] [131] [136] [138] [169] [171] [172]	[25] [42] [66] [69] [72] [78] [89] [120] [119] [138] [169]	No se tienen resultados concluyentes en cuanto a calidad externa.
Bissi et. al [18]	Information and Software Technology 74, 2016	2003-2012	Revisión sistemática de literatura	27	[3] [4] [17] [25] [38] [47] [50] [66] [67] [69] [72] [89] [92] [86] [85] [120] [119] [124] [136] [139] [138] [154] [162] [164] [168] [169] [175]	[25] [47] [50] [66] [67] [69] [72] [89] [92] [86] [120] [119] [139] [138] [169]	Los resultados acerca de los efectos en calidad externa en los experimentos en la academia en algunos casos son contradictorios. En los experimentos en la industria TDD incrementa la calidad externa.

Los autores de los estudios secundarios han clasificado los estudios de diferentes maneras. Kollanus [104] clasifica por experimentos, case studies y surveys. Bissi et. al [18] clasifica los estudios en experimentos, case studies, cuestionarios, simulaciones y realiza el análisis de acuerdo al ámbito académico e industrial. Makinen & Munch [121] hace una comparativa de los estudios que los considera relevantes. Rafique & Misic [145] clasifica los estudios por el ámbito académico e industrial. Munir et. al [134] clasifica los estudios por rigor y relevancia.

A continuación nos centramos en los experimentos estudiados en cada una de las revisiones sistemáticas. En las tablas: 2.2 a 2.12, se indica para cada experimento el tratamiento usado como control y el resultado del experimento en lo relativo a calidad usando la siguiente simbología: (+) cuando el tratamiento TDD favorece el efecto, (-) cuando el tratamiento TDD disminuye el efecto, (/) cuando no hay diferencia en los efectos de los tratamientos, - no aplica al estudio.

2.1.1. Revisión sistemática de Kollanus [104]

Kollanus [104] en 2010 realiza una revisión sistemática de literatura donde analiza 40 estudios empíricos sobre TDD. 25 estudios son clasificados como experimentos, de los cuales 14 serían experimentos controlados de acuerdo a los autores. Hemos tomado en cuenta unicamente los experimentos controlados que estudian calidad, ya que el resto de estudios empíricos quedan fuera del alcance de esta investigación.

Tabla 2.2: Experimentos controlados estudiados en la revisión sistemática de Kollanus [104]

Experimento controlado	Control	Calidad externa
Canfora et. al (2006) [25]	TAC (Test After Coding)	-
Canfora et. al [26]	TAC (Test After Coding)	-
Erdogmus et. al (2005) [50]	ITL	(/)
Flohr & Schneider (2006) [58]	Testing clásico	-
George & Williams (2003) [66]	Cascada	-
George & Williams (2004) [67]	Cascada	(+)
Geras et. al (2004) [69]	Iterative Test Last (ITL)	-
Gupta & Jalote (2007) [72]	Iterative Test Last (ITL)	(+)
Huang & Holcombe (2009) [78]	Incremental Test Last	(/)
Madeyski (2005) [115]	Test Last Clásico	(-)
Muller & Hagner [130]	Desarrollo tradicional	(/)
Pancur et. al (2003) [139]	Iterative Test Last	(/)

En lo relativo a la influencia de TDD en la calidad externa, en dos experimentos ([67] y [72]), se obtiene que TDD incrementa la calidad y en un experimento ([115]), obtiene que TDD disminuye la calidad externa. En cuatro experimentos ([50], [78], [139] y [130]) TDD no hace diferencia en la calidad externa.

En resumen, los experimentos de esta revisión sistemática estudian la influencia de TDD en la calidad y da **resultados contradictorios**.

2.1.2. Revisión sistemática de Causevic et. al [31]

Causevic et. al [31] en el 2011 realizó una revisión sistemática de literatura en TDD. Analizaron 48 estudios, 25 de ellos son considerados experimentos. En la tabla 2.3 se presentan los experimentos de la industria. En la tabla 2.4 se listan los experimentos con profesionales y estudiantes. En la tabla 2.5 se presenta los experimentos en la academia.

Tabla 2.3: Experimentos controlados en la industria en Causevic et. al [31]

Experimento controlado	Control	Calidad de código	Observaciones
George & Williams (2003) [66]	Cascada	(+)	
Geras et. al (2004) [69]	Iterative Test Last (ITL)	(/)	
Janzen et. al (2007) [92]	-	-	En el experimento no considera calidad ni tiempo de desarrollo

Tabla 2.4: Experimentos controlados con profesionales y estudiantes en Causevic et. al [31]

Experimento controlado	Control	Calidad de código	Observaciones
Höfer & Philipp (2009) [76]	-	-	El experimento no considera calidad.
Müller & Höfer (2007) [131]	-	-	El experimento no considera calidad.

Los autores del estudio secundario [31] han clasificado los experimentos de acuerdo a los sujetos experimentales: profesionales y estudiantes. Los autores encuentran 18 áreas de efecto de TDD y analiza 7 factores limitantes para la adopción de TDD en la industria. En las áreas de efecto no menciona explícitamente calidad externa, pero menciona calidad del código producido por los desarrolladores. **La calidad del código de los desarrolladores en su mayoría da resultados positivos al aplicar TDD**, con excepción de un experimento que indica lo contrario [78].

Tabla 2.5: Experimentos controlados en la academia en Causevic et. al [31]

Experimento controlado	Control	Calidad de código	Observaciones
Domino et. al (2007) [45]	-	-	En el experimento no considera calidad.
Erdogmus et. al (2005) [50]	-	-	No da resultados de calidad
Filho (2006) [140]	-	-	En el estudio no considera calidad.
Flohr & Schneider (2005) [57]	-	-	Este experimento se reporta en Flohr & Schneider (2006) [58]
Flohr & Schneider (2006) [58]	Testing clásico	-	
Gupta & Jalote (2007) [72]	Cascada	-	
Huang & Holcombe (2009) [78]	Incremental Test Last	(-)	
Janzen & Saiedian (2006) [89]	Iterative Test Last (ITL)	-	
Kollanus & Isomottonen (2008) [106]	-	-	En el experimento no considera calidad.
Madeyski (2010) [119]	-	-	En el experimento no considera calidad.
Müller & Hagner (2002) [130]	Desarrollo tradicional	-	Efecto Negativo. No se reporta ningún efecto
Vu et. al (2009) [169]	Incremental TL	(+)	

2.1.3. Meta-análisis de Rafique & Misic [145]

El meta-análisis de Rafique & Misic [145] realizado en 2013 considera 25 artículos empíricos para analizar el impacto de TDD en la calidad del software. Los estudios son clasificados en industriales y académicos. De los 25 artículos extrae 37 experimentos para su análisis. En la tabla 2.6 se presentan los estudios que los autores consideran experimentos en la academia y en la tabla 2.7 los estudios considerados experimentos en la industria. En la columna calidad se anota el porcentaje de mejora que indican los autores del meta-análisis.

Tabla 2.6: Experimentos controlados en la academia según Rafique & Misic [145]

Experimento controlado	Control	Calidad (% mejora)	Observaciones
Desai et. al (2009) [42]	Cascada	3	
Edwards (2004) [48]	Cascada	46	
Erdogmus et. al (2005) [50]	ITL	-2	
Flohr & Schneider (2006) [58]	ITL	-	
George (2002) [65] E1	Cascada	16	
Gupta & Jalote (2007) E1 [72]	Cascada	16	
Gupta & Jalote (2007) E2 [72]	Cascada	-7	
Huang & Holcombe (2009) [78]	ITL	-3	
Janzen (2006) E2 [87]	ITL	-	
Janzen (2006) E3, E4 [87]	ITL	-	
Janzen (2006) E1, E5 [87]	ITL	-	
Kaufmann & Janzen (2003) [101]	Similar ITL	-	
Madeyski2005 E1 [115]	ITL	-27	
Madeyski2005 E2 [115]	ITL	-35	
Pancur and Ciglaric (2011)E2, E4 [138]	ITL	-6	
Pancur and Ciglaric (2011)E1, E3 [138]	ITL	-	
Pancur et. al (2003) [139]	ITL	-3	
Rahman (2007) [146]	Cascada	24	
Vu et. al (2009) [169]	Cascada	28	
Xu & Li (2009) [174]	Cascada	49	Es un case study según nosotros
Yenduri & Perkins (2006) [175]	Cascada	35	Es un case study
Zhang et. al (2006) [177]	Cascada	28	Experimento reportado en dos páginas

Los autores del meta-análisis concluyen que **TDD aumenta levemente la calidad externa en los experimentos académicos**. En los experimentos industriales **TDD permite un incremento más amplio de la calidad externa** en los experimentos industriales.

Tabla 2.7: Experimentos controlados en la industria según Rafique & Misic [145]

Experimento controlado	Control	Calidad (% mejora)	Observaciones
Canfora et. al (2006) [25]	Cascada	-	
Dogsa &Batic [44]	ITL	33	Es un case study
George (2002) [65] E2	Cascada	18	
Geras et. al (2004) [69]	ITL	-	
Lui & Chan (2004) [114]	Cascada	67	Es un case study según nosotros
Nagappan et. al (2008) E1 [136]	Cascada	62	Es un case study
Nagappan et. al (2008) E2 [136]	Cascada	76	Es un case study
Nagappan et. al (2008) E3 [136]	Cascada	90	Es un case study
Slyngstad et. al (2008) [164]	Cascada	33	Es un case study
Williams et. al (2003) [172]	Cascada	39	Es un case study

2.1.4. Revisión sistemática de Munir et. al [134]

Munir et. al [134] en 2014 realizó una revisión sistemática de literatura en donde analiza 41 estudios primarios de TDD. Los estudios son clasificados por rigor y relevancia. Asignan los valores alto y bajo al rigor y a la relevancia. El investigador demuestra cómo cambian las conclusiones sobre calidad al analizar por las cuatro categorías que se obtienen de combinar rigor alto/bajo con relevancia alta/baja. Dentro de cada categoría se encuentran experimentos, case study y surveys. Las categorías establecidas son: A, B1, B2 y C. La categoría A corresponde a estudios con rigor alto y relevancia alta TDD. La categoría B1, estudios con relevancia alta y rigor bajo. Categoría B2: relevancia baja y rigor alto. La categoría C: relevancia baja y rigor bajo.

En las categorías A y B1 no se tienen experimentos. En la categoría B2 están incluidos 18 experimentos ([119], [89], [120], [171], [115], [78], [50], [138], [72], [25], [169], [86], [130], [67], [91], [69], [42] y [58]). En la categoría C tenemos 6 experimentos ([107], [85], [45], [139], [106], [57]).

En la tabla 2.8 se presentan los experimentos de la categoría B2 que tienen como variable respuesta calidad. En la tabla 2.9 se presentan los experimentos de la categoría C.

Munir y colegas concluyen que en los estudios de la categoría B2: relevancia baja y rigor alto **no presentan diferencias en los resultados** para TDD y tradicional. En la categoría C: relevancia baja y rigor bajo muestran **resultados positivos y ninguna diferencia**.

Tabla 2.8: Experimentos de la categoría B2 según Munir et. al [134]

Experimento controlado	Control	Calidad	Observaciones
Canfora et. al (2006) [25]	Test After Coding (TAC)	-	
Desai et. al (2009) [42]	Test Last	(+)	
Erdogmus et. al (2005) [50]	ITL	-	
Flohr & Schneider (2006) [58]	ITL	-	
George & Williams (2004) [67]	Cascada	(+)	
Geras et. al (2004) [69]	ITL	-	
Gupta & Jalote (2007) [72]	Cascada	(+)	
Huang & Holcombe (2009) [78]	ITL	(/)	
Janzen & Saiedian (2006) [89]	ITL	-	
Janzen & Saiedian (2008) [86]	ITL	-	
Madeyski (2005) [115]	Test-Last clásico	(/)	
Madeyski (2010) [119]	ITL	(/)	
Madeyski & Szala [120]	ITL	-	
Muller & Hagner (2002) [130]	Desarrollo tradicional	(/)	
Pancur and Ciglaric (2011) [138]	Micro Iterative TL	(/)	
Vu et. al (2009) [169]	ITL	(/)	
Wilkerson et. al [171]	Inspección de código	(-)	Es un quasi experimento según autores del estudio primario

Tabla 2.9: Experimentos de la categoría C según Munir et. al [134]

Experimento controlado	Control	Calidad	Observaciones
Laranjeiro & Vieira (2009) [107]	-	-	Es un case study. No estudia calidad
Domino et. al (2007) [45]	-	-	No estudia calidad
Flohr & Schneider (2005) [57]	ITL	-	
Janzen & Saiedian (2008) [85]	-	-	No estudia calidad
Kollanus & Isomottonen (2008) [106]	-	-	No estudia calidad
Pancur et. al (2003) [139]	ITL	(/)	

2.1.5. Análisis comparativo de estudios empíricos de los efectos de TDD de Makinen & Munch [121]

Makinen & Munch [121] en 2014 realizaron un estudio comparativo de los efectos de TDD. En el estudio analizan un total de 19 estudios empíricos que consideran relevantes.

En la tabla 2.10 se muestran los experimentos que analizan la calidad externa.

Tabla 2.10: Experimentos controlados tomados en cuenta en Makinen & Munch [121]

Experimento controlado	Control	Calidad externa
Canfora et. al (2006) [25]	Test After Co- ding (TAC)	-
Desai et. al (2009) [42]	Test-Last	(/)
George & Williams (2003) [66]	Cascada	(+)
Geras et. al (2004) [69]	ITL	-
Gupta & Jalote (2007) [72]	Cascada	(/)
Huang & Holcombe (2009) [78]	ITL	(/)
Janzen & Saiedian (2006) [89]	ITL	(/)
Madeyski (2010) [119]	ITL	-
Müller & Höfer (2007) [131]	Desarrollo tradi- cional	(/)
Pancur and Ciglaric (2011) [138]	ITL	(/)
Vu et. al (2009) [169]	ITL	(/)

Considerando unicamente los experimentos, TDD tiene efecto estadísticamente positivo en calidad externa en un experimento [66]. Se indica que TDD no tiene efecto en calidad externa en siete experimentos ([131], [42], [72], [78], [89], [138] y [169]).

De acuerdo a los resultados de los experimentos, **no se tienen resultados concluyentes** en cuanto al efecto de TDD en calidad externa.

2.1.6. Revisión sistemática de Bissi et. al [18]

Bissi et. al [18] en su revisión sistemática analizaron 27 estudios primarios realizados entre 1999 y 2014. Clasifican los estudios en académicos, industriales y mixtos. 16 de los estudios primarios son experimentos correspondiendo con el 57,14 %. En la tabla 2.11 se muestran los experimentos en la academia y sus resultados. En la tabla 2.12 se presentan los experimentos en la industria.

En cuatro experimentos académicos TDD incrementa la calidad ([138], [69], [169] y [48]). En un experimento no hay diferencia ([72]) y en un experimento TDD disminuye la calidad ([139]). Los resultados acerca de los efectos en calidad externa en los experimentos en la academia en algunos casos son contradictorios. En los experimentos en la industria TDD incrementa la calidad externa ([66], [67] y [124]). **En calidad externa los resultados no son concluyentes.**

Tabla 2.11: Experimentos controlados en la academia según Bissi et. al [18]

Experimento controlado	Control	Calidad externa
Edwards (2004) [48]	Tradicional	(+)
Erdogmus et. al (2005) [50]	ITL	-
Geras et. al (2004) [69]	ITL	(+)
Gupta & Jalote (2007) [72]	Cascada	(/)
Janzen & Saiedian (2006) [89]	ITL	-
Janzen & Saiedian (2008) [86]	ITL	-
Pancur and Ciglaric (2011) [138]	Micro ITL	(+)
Pancur et. al (2003) [139]	ITL	(-)
Vu et. al (2009) [169]	ITL	(+)

Tabla 2.12: Experimentos controlados en la industria según Bissi et. al [18]

Experimento controlado	Control	Calidad
Canfora et. al (2006) [25]	TAC (Test After Coding)	-
George & Williams (2003) [66]	Cascada	(+)
George & Williams (2004) [67]	Cascada	(+)
Maximilien & Williams [124]	Ad-hoc Unit testing approach	(+)

2.2. Experimentos controlados sobre TDD

En el Anexo 1, Tabla A.1 aparecen los estudios empíricos primarios relativos a TDD. Este listado se elaboró tomando en cuenta todas las referencias bibliográficas de los estudios secundarios previamente mencionados. A este listado se han adicionado estudios empíricos encontrados no incluidos desde el año 2012 hasta la fecha de elaboración de este documento (Octubre de 2016). En la siguiente tabla 2.13 tenemos un listado de estudios empíricos no incluidos en las revisiones sistemáticas:

En la literatura se han reportado 114 estudios primarios acerca de TDD. De los 114 estudios 8 han sido reportados de manera repetida. Según los autores de los estudios primarios son experimentos 36 de ellos, de los cuales 29 pertenecen al contexto académico, 6 en el contexto industrial y uno mixto. 17 experimentos en TDD estudian calidad. La figura 2.1 muestra estos subconjuntos.

En la tabla A.1 del Anexo A se han marcado los experimentos con un color diferente a los otros estudios. En la tabla 2.14 se presenta el listado de los 17 experimentos que estudian calidad. Se ha descrito el control y los resultados de cada experimento en lo relativo a calidad. En la tabla 2.14 se usa la nomenclatura: E para Experimentos; EC para

Experimentos controlados; A para Académicos; I para Industria.

Figura 2.1: Estudios en TDD

Los experimentos tampoco llegan a resultados concluyentes, al igual que ocurría en los estudios secundarios que incluían diversos tipos de estudios empíricos, los resultados experimentales dan resultados contradictorios respecto al efecto de TDD en la calidad externa. Hemos revisado los experimentos y creemos que parte de la diferencia en sus resultados se debe, no solo a diferencias contextuales, sino también instrumentales. Consideramos diferencias contextuales aspectos donde se desarrollan los experimentos; por ejemplo, diferencias entre ámbito industrial y académico, diferencias asociadas a la experiencia de los sujetos, etc. Consideramos diferencias instrumentales las relativas a las distintas características de los instrumentos usados en el protocolo de los experimentos. Por ejemplo, la tarea empleada en los experimentos, los casos de prueba usados para las mediciones, la persona que extrae las métricas.

Hemos apreciado diferencias en los instrumentos utilizados en experimentos con el mismo objetivo no considerados en los análisis que realizan los autores de estudios secundarios. Por ejemplo, los casos de prueba que se usan en la medición de la calidad del código generado por los sujetos pueden variar. Nuestro trabajo pretende estudiar si los resultados de un experimento dependen del instrumento usado y por lo tanto contribuyen a las diferencias observadas entre los resultados de experimentos bastante semejantes.

Tabla 2.13: Estudios empíricos de TDD no incluidos en las revisiones sistemáticas

ID	Autor	Tipo de estudio	Industria / Academia	Revista / Conferencia
[PS11]	Causevic2012 [32]	Experimento	Academia	EASE 2012, IET
[PS12]	Causevic2012 [28]	Experimento	Academia	QUATIC 2012, IEEE
[PS13]	Causevic2013 [30]	Industria Empirical Study	Industria	CESI'14, IEEE
[PS30]	Fucci2013 [60]	Experimento controlado (Replicación)	Academia	ESEM 2013, ACM/IEEE
[PS31]	Fucci2014 [63]	Experimento controlado	Academia	ESEM'14, ACM
[PS32]	Fucci2014 [61]	Experimento controlado (Replication)	Academia	Article - Empirical Software Engineering - April 2014, Volume 19, Issue 2, pp 277-302
[PS33]	Fucci2015 [64]	Experimento	Academia	EASE '15, ACM
[PS34]	Fucci2015 [62]	Quasi Experimento	Industria	Information and Software Technology, Elsevier
[PS61]	Latorre2014 [109]	Quasi Experimento	Industria	IEEE Transactions on Software Engineering
[PS62]	Latorre2014 [110]	Experience Report	Industria	Empirical Software Engineering, Volume 19, Issue 3, Springer
[PS80]	Munir2014 [135]	Experimento	Industria	EASE '14, ACM
[PS86]	Ress2013 [148]	Case study	Industria	J. Technol. Manag. Innov. 2013, Volume 8, Special Issue ALTEC.
[PS87]	Salman2014 [152]	Experimento	Academia e Industria	Masther Thesis, U. de Oulu
[PS91]	Sfetsos2015 [156]	Case study	Academia	IISA 2015, IEEE
[PS106]	Aniche2015 [2]	Estudio cualitativo	Industria	Journal of the Brazilian Computer Society2015
[PS107]	Causevic2013 [29]	Experimento	Academia	2013 39th Euromicro Conference Series on Software Engineering and Advanced Applications
[PS108]	Fucci2016 [59]	Experimento	Academia	ESEM '16 , ACM
[PS109]	Hilton2016 [75]	Case study	Academia	Chapter: Agile Processes, in Software Engineering, and Extreme Programming Volume 251 of the series Lecture Notes in Business Information Processing pp 53-65, Springer
[PS110]	Mondal2014 [129]	Case study	Industria	Chapter Emerging Trends in Computing and Communication Volume 298 of the series Lecture Notes in Electrical Engineering pp 225-233
[PS111]	Percival2013 [142]	Survey	Academia	2013 46th Hawaii International Conference on System Sciences

Tabla 2.13 – continuación

ID	Autor	Tipo de estudio	Industria / Academia	Revista / Conferencia
[PS112]	Peters2014 [143]	Case study	Industria	Chapter - Agile Processes in Software Engineering and Extreme Programming. Volume 179 of the series Lecture Notes in Business Information Processing pp 294-301, Springer
[PS113]	Romano2016 [151]	Estudio cualitativo	Academia e Industria	EASE '16 Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering
[PS114]	Shelton2012 [160]	Case study	Industria	2012 IEEE Fifth International Conference on Software Testing, Verification and Validation

Tabla 2.14: Experimentos en TDD que estudian calidad

ID	Autor	Tipo	Ind. / Acad.	Revista / Conferencia	Control	Calidad externa	Observaciones
[PS6]	Bhadauria2009 [16]	E	A	PhD thesis, The University of Texas at Arlington	Tradicional (Test Last)	(+) TDD favorece la calidad, comparado con el método de desarrollo tradicional (TDD: 93.73, TL:76.06).	Estadísticamente significativa la diferencia.
[PS8]	Canfora2006 [25]	EC	I	ISESE'06, ACM	TAC (Test After Coding)	(+) TDD produce más aserciones que TAC (Test After Coding). (763 TDD vs. 668 TAC), p-level: 0.54772 pero no es estadísticamente significativo	
[PS11]	Causevic2012 [32]	E	A	EASE 2012, IET	Tradicional (Test Last)	(+) Test First da mejor calidad que Test Last. TF:83.94 %, TL: 78.28 %.	No es estadísticamente significativo
[PS19]	Desai2009 [42]	EC	A	SIGSE'09, ACM	Test-Last	(/) No hay diferencia significativa entre los que aplicaron Test-Last vs. TDD	
[PS23]	Edwards2003 [47]	E	A	Proceedings of the International Conference on Education and Information Systems Technologies and Application	no TDD (Tradicional)	(+) TDD favorece la calidad de código. Produce 45 % menos de defectos. Diferencia significativa.	
[PS25]	Erdogmus2005 [50]	EC	A	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING	ITL	(/) No hay diferencia. ITL: 0.85, TDD:0.83	
[PS30]	Fucci2013 [60]	EC	A	ESEM 2013, ACM/IEEE	ITL	(/) tamaño del efecto pequeño: 0.11, p-value: 0.53	No se rechaza la hipótesis nula.
[PS108]	Fucci2016 [59]	E	A	ESEM '16, ACM	ITL	(/) Diferencias muy pequeñas	No hay significancia estadística
[PS37]	George2004 [67]	E	I	Information and Software Technology, ACM	Cascada	(+) TDD favorece en 18 %	
[PS40]	Gupta2007 [72]	EC	A	ESEM 2007, IEEE	Cascada	(/) La calidad se ha incrementado por el esfuerzo en testing, no por la técnica de desarrollo	Anotamos la conclusión a la que han llegado los investigadores
[PS44]	Huang2009 [78]	EC	A	Information and Software Technology 51, Elsevier	Incremental Test Last	(/) Diferencia muy pequeña. TF: 38.0, TL:39.4, T-test:0.375	No se rechaza la hipótesis nula. No hay significancia estadística
[PS66]	Madyski2005 [115]	E	A	Conf. Softw. Eng.: Evol. Emerg. Technol.	Test-Last clásico	(-) Disminuye cuando se usa TDD.	
[PS78]	Muller2002 [130]	E	A	IEEE Proceedings - Software, Volume:149 Issue:5	Desarrollo tradicional	(/) TDD no incrementa la confiabilidad (reliability) a los programas.	Le llama reliability a los casos de prueba pasados. Bajo poder, igual a 0.645
[PS80]	Munir2014 [135]	E	I	EASE '14, ACM	ITL	(/) Similar número de casos de prueba pasados.	Resultados estadísticamente insignificantes
[PS82]	Pancur2003 [139]	EC	A	Eurocon 2003, IEEE	Iterative TL	(-) Pequeños beneficios a favor de ITL. TDD:92.6, ITL:95.1	Estadísticamente no hay diferencia significativa.

Tabla 2.14 – continuación

ID	Autor	Tipo	Ind.	Revista / Conferencia / Acad.	Control	Calidad externa	Observaciones
[PS83]	Pancur2011 [138]	EC	A	Information and Software Technology 53, Elsevier	Micro Iterative TL	(+) En cuanto a porcentaje de casos de prueba de aceptación pasados, el efecto de TDD es pequeño y en sentido positivo.	Los autores hacen un meta análisis. Estadísticamente ninguno de los efectos es significativo
[PS98]	Vu2009 [169]	E	A	2009 Sixth International Conference on Information Technology: New Generations, IEEE	Incremental TL	(+) ITL tiene 39% más defectos que TDD (número de defectos registrados)	

2.3. Diferencias instrumentales entre los experimentos de TDD

2.3.1. Operacionalización e instrumentación de variables respuesta en experimentos

La operacionalización de una variable respuesta se realiza mediante la medición. En la operacionalización se utilizan instrumentos de medición, tales instrumentos dependen del área en que estamos tratando. En nuestro caso medición de experimentos en IS.

Un procedimiento de medición tiene una serie de pasos a ejecutar. Este procedimiento se realiza con la ayuda de instrumentos de medición. Retomando el ejemplo ya visto en el capítulo de Introducción, si queremos medir la longitud de una mesa en centímetros: antes de ejecutar el procedimiento es necesario contar con el instrumento de medición, en este caso una cinta métrica de un determinado material cuya medida esté expresada en centímetros, a continuación se verifica la cinta, contrastándola con un patrón de medida. Una vez que contamos con el instrumento de medida vamos a ejecutar el procedimiento de medición: En primer lugar se reciben las mesas en el lugar en donde se va a medir, luego tendemos la cinta de medida sobre el largo de la mesa haciendo coincidir el inicio de la cinta con el inicio de la mesa y cubrir la cinta hasta el final de la mesa, por último, tomar la lectura indicada en centímetros en la cinta de medida.

Con el propósito de identificar los elementos del procedimiento de medición en nuestros experimentos en IS, hemos procedido a realizar la medición de la variable respuesta calidad de un experimento en TDD (Playtech). Este procedimiento de medición se muestra en la tabla 2.15. También se ha tomado otro experimento en TDD reportado en la literatura Erdogmus et al. al [50] para identificar los elementos que son parte de su procedimiento de medición, lo mostramos en la tabla 2.16.

Como se puede ver el procedimiento implica varios pasos que van desde la elaboración de la variable respuesta hasta la obtención del resultado. Aquí queremos hacer notar que el procedimiento de medición en nuestra área de experimentación y en IS en general puede implicar varios pasos como la elaboración de una métrica, elaboración de un instrumento de medición y la aplicación del instrumento de medida sobre el código fuente para obtener la medida. Los elementos que intervienen en este proceso, tales como protocolos, instrumentos de medida y los medidores serán objeto de nuestro estudio en esta investigación.

Tabla 2.15: Protocolo de Medición del Experimento TDD en Java (Playtech) - Operacionalización del constructo de la variable respuesta calidad

Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos MR	Implementación de instrumentos MP	Implementación de instrumentos BSK	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
Calidad externa del código fuente.	Ecuación 4.1	Definición del conjunto de casos de prueba para evaluar la calidad del código fuente con respecto a la especificación de cada tarea.	Suite de casos de prueba: Mars Rover 1.3,	Suite de casos de prueba Music Phone tests 1.2	Suite de casos de prueba Bowling Score Keeper 1.1	Se verifican las salidas esperadas de los casos de prueba, en base a una implementación de referencia. Luego se corrigen los casos de prueba con error.	El producto responde al código fuente de los sujetos y se encuentran en un repositorio de productos.	1. Copiar el código fuente de los sujetos en el repositorio del IDE. 2. Anotar el orden, que establece Junior, para la ejecución de los casos de prueba en el ambiente de pruebas. En base al orden de la ejecución de los casos de prueba, crear el formato para registrar las mediciones. 3. Abrir el proyecto a medir.	1. Registrar individualmente el estado de cada assertion, con el valor respectivo PASS, FAIL, ERR por cada User Story 2. Se obtiene el total PASS, FAIL, ERR por cada user story.
Subcaracterística accuracy del modelo de calidad ISO9126-1.			Formato en Excel para registrar las mediciones. (archivo metrics-Extraction(new_metrics).xls)	Formato en Excel para registrar las mediciones. (archivo metrics-Extraction(new_metrics).xls)	Formato en Excel para registrar las mediciones. (archivo metrics-Extraction(new_metrics).xls)			4. Cargar la suite de casos de prueba. 5. Cargar las librerías necesarias para la ejecución del suite de casos de prueba. 6. Modificar el código fuente en los casos que sea necesario: a) Nombres de métodos que tienen que estar con igual nombre que en el instrumento de medida. b) Formato de salidas (return) de funciones importantes que deben estar con el mismo número de parámetros que en el instrumento de medida. c) Modificación o creación de métodos constructores para que puedan conectar con los casos de prueba 7. Ejecutar el proyecto. Si contiene errores regresar al numeral 5 para verificar el cumplimiento de dichos pasos. 8. Guardar la salida de la ejecución en un fichero.	3. Se sunariza las divisiones de assertions PASS de los user story tackled para el total de assertions. Se divide para el número de user stroy tacle y se obtiene el porcentaje 3. Se sunariza las divisiones de assertions PASS de los user story tackled para el total de assertions. Se divide para el número de user stroy tacle y se obtiene el porcentaje 4. Las mediciones obtenidas son almacenadas en el fichero metricsExtraction. Los casos de prueba y los ficheros de la medición se encuentran en un Portal Web (Google Sites).

Tabla 2.16: Operacionalización del constructo de la variable respuesta calidad del Experimento Erdogmus et. al [50]

	Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
DESCRIPCIÓN	Calidad externa: Evaluación de la calidad mediante el uso de pruebas de aceptación de caja negra. La medida de la calidad se basa en defectos, contados story por story.	La calidad de una historia está dado por el porcentaje de asserts que pasan de la suite de casos de prueba de aceptación asociados. La calidad de cada historia se pondera por un indicador de dificultad de la historia basada en el número total de asserts del conjunto de casos de prueba de aceptación asociado. Una medida ponderada se calcula por cada sujeto del total de historias entregadas. El rango de la variable es de 0.5 a 1 (50 % a 100 % respectivamente)	Los casos de prueba de aceptación de caja negra fueron escritos por los investigadores.	Consiste de 105 métodos de prueba de aceptación de caja negra que constan de más de 350 aserciones. (Bowling Score Keeper). Los casos de prueba de aceptación son diferentes a los casos de prueba del programador.	El paper no especifica si se realizó un procedimiento de validación del instrumento de evaluación.	Se cuenta con el código fuente de los sujetos que firmaron su consentimiento. Los sujetos dejan su código en su CVS repository. Los sujetos trabajan en su assignment por varias semanas.	Para aplicar el instrumento de medida sobre el código fuente, se consideraron únicamente las historias de usuario entregadas por el sujeto. Al final del experimento, cada prueba de aceptación fue ejecutada automáticamente en el programa de los sujetos que terminaron el experimento.	Se obtiene el valor promedio por técnica (Test first TF y Test last TL).
QUIEN:			Prueba Bowling Score Keeper de Robert Martin. El problema fue dividido en varias historias de usuario.	JUnit, Java, Eclipse, CVS Repository			Se eliminaron los datos de los sujetos que contestaron que No aplicaron la técnica asignada.	
COMO:		No especifica como ha definido la métrica.	No indica si hubo alguna técnica para la definición del instrumento de evaluación.		No hay información si se realizó la validación del instrumento.		De manera automática se ejecutan los casos de prueba en cada programa del sujeto.	El programa, de manera automática.
CUANDO:							Cuando el sujeto termina el experimento.	Cuando el sujeto termina el experimento.

2.3.2. Casos de prueba

En los experimentos en TDD se usan casos de prueba de aceptación para medir calidad. Por ejemplo, para medir calidad externa se suele usar como medida el porcentaje de casos de prueba de aceptación pasados. Típicamente los investigadores generan una suite de casos de prueba que se pasa por el código fuente de los sujetos experimentales para extraer los datos crudos del experimento. En cada experimento podría ser distinta la técnica de elaboración de los casos de prueba dado que no se reporta sobre ello en la mayoría de los artículos. La suite de casos de prueba podría ser generada por los investigadores Ad hoc o mediante una técnica de testing (Por ejemplo: Equivalence Partitioning o Boundary Value Analysis). Al evaluar la calidad con distintos casos de prueba los resultados pueden ser distintos. Es decir la técnica con que se elaboraron los casos de prueba podrían tener influencia en los resultados de la medición y del experimento.

Con el propósito de conocer los casos de prueba utilizados en los experimentos en TDD hemos seleccionado los experimentos que miden calidad externa mediante la ejecución de casos de prueba de aceptación. Se han encontrado 17 experimentos que aparecen en la tabla 2.17. La tabla 2.17 resume la información aportada en los artículos sobre el modo de generar los casos de prueba. Unicamente en un experimento (5.88 %) el autor muestra los casos de prueba, George et. al [67]. En 9 experimentos (52.94 %) los autores indican quién o quienes elaboraron los casos de prueba (marcado en la tabla en gris). **En 6 experimentos (35.29 %) no se ofrece ninguna información acerca de cómo se obtuvieron los casos de prueba.** En ningún experimento reportan explícitamente el nombre de la técnica con que se elaboraron los casos de prueba que sirven de instrumento de medida para calidad.

Creemos que hay diferencias entre los casos de prueba y en general entre los instrumentos que los investigadores usan en los experimentos para medir el mismo constructo e incluso la misma métrica. Los autores ofrecen poca información sobre las características de los casos de prueba que han usado en sus mediciones. Consideramos que las características tales como la técnica con que se generaron los casos de prueba, la persona quién generó (investigador o sujeto experimental) son factores importantes y críticos que pueden influir en el resultado de la medición. Por lo tanto, su desconocimiento daría cabida a cierta incertidumbre en el resultado de la medición y por tanto, en los resultados del experimento.

Si se comparte las suites de casos de prueba, los protocolos de medición de la misma variable respuesta (por ej. calidad) para experimentos con el mismo objetivo (por ej. estudio de TDD) podría estandarizarse y ser utilizado en otros experimentos. En los reportes de los experimentos estudiados, en ningún reporte muestran los casos de prueba en el artículo, unicamente en un estudio los muestra en una tesis de master.

Los investigadores tampoco ofrecen información completa de cómo obtuvieron los casos de prueba, de donde los obtuvieron, quién los elaboró, ni la técnica para su elaboración.

En ninguno de los experimentos analizados se indica explícitamente si usan una técnica de generación de casos de prueba específica. Incluso en el 46 % de los experimentos analizados no usan una técnica sino simplemente el criterio del investigador. En consecuencia, tener suite de casos de prueba incompletas o poco estrictas, podría dar mediciones que no evalúen apropiadamente el código fuente de los sujetos. Al tener casos de prueba incompletos se perjudica en la medida a quienes implementaron código correcto ya que el instrumento de medida no evalúa por completo el código.

Tabla 2.17: Casos de prueba en los experimentos en TDD

ID	AUTOR	Industria / Academia	¿Están reportados?	¿Quién obtuvo los casos de prueba?	Técnica utilizada para la generación de los casos de prueba
[[PS19]	Desai2009 [42]	Academia	NO	El instructor crea la suite de casos de prueba	Se deduce Ad hoc
[[PS23]	Edwards2003 [47]	Academia	NO	El investigador desarrolló una suite de casos de prueba integral.	No indica
[[PS25]	Erdogmus2005 [50]	Academia	NO	Pruebas de aceptación de caja negra escritas por el investigador	No indica
[[PS30]	Fucci2013 [60]	Academia	NO	Elaborados por el investigador	Se deduce Ad hoc
[[PS31]	Fucci2014 [63]	Academia	NO	Elaborados por el investigador	Se deduce Ad hoc
[[PS32]	Fucci2014b [61]	Academia	NO	Se deduce elaborados por el investigador	Se deduce Ad hoc
[[PS33]	Fucci2015 [64]	Academia	NO	Se deduce elaborados por el investigador	Se deduce Ad hoc
[[PS108]	Fucci2016 [59]	Academia	NO	Se deduce elaborados por el investigador	Se deduce Ad hoc
[[PS37]	George2004 [67]	Industria	NO en paper, SI en la tesis. [65]	Fueron elaborados por el investigador	Se deduce ad hoc
[[PS38]	Geras2004 [69]	Industria	NO	Se tienen developer test cases, elaborados por el desarrollador y casos de prueba de cliente elaborados por el investigador	No indica
[PS66]	Madeyski2005 [115]	Academia	NO	No indica	No indica
[PS69]	Madeyski2007b [120]	Industria	NO	No indica	No indica
[PS78]	Muller2002 [130]	Academia	NO	No indica	Se deduce Ad hoc
[PS80]	Munir2014 [135]	Industria	NO	No indica	No indica
[PS82]	Pancur2003 [139]	Academia	NO	No indica	Se deduce Ad hoc
[[PS83]	Pancur2011 [138]	Academia	NO	Los elaboro el investigador	Se deduce Ad hoc
[PS98]	Vu2009 [169]	Academia	NO	No indica	No indica

2.3.3. Tarea experimental

A continuación pasamos a analizar lo que sucede con otro elemento instrumental de los experimentos sobre TDD. En la tabla 2.18 presentamos las características de la tarea experimental usada en cada uno de los 17 experimentos que miden calidad con casos de prueba pasados.

Según la tabla 2.18, en ocho experimentos (47.05 %) han usado Bowling Score Keeper (BSK) como tarea experimental. En cinco experimentos han usado otra tarea distinta a BSK. En cuatro artículos los investigadores reportan únicamente el nombre de la tarea mientras cuatro no indican cuál fue la tarea. Si los investigadores no reportan la tarea es imposible evaluar la validez externa de los resultados del experimento, tampoco es posible conocer las características de la tarea experimental y su influencia en los resultados experimentales.

Únicamente en un artículo (5.88 %) presentan explícitamente la especificación de la tarea experimental. En dos artículos (11.75 %) se puede ver la especificación a través de referencias a otros documentos que se encuentran en un repositorio de instrumentos experimentales.

En base a nuestra experiencia realizando experimentos en TDD, la granularidad de la especificación de los requisitos puede impactar en los resultados experimentales¹. Respecto a la característica de granularidad de la tarea, en nueve reportes de experimentos (52.94 %) comentan la granularidad de la especificación. En siete artículos (41.17 %), no aportan información acerca de la granularidad de la especificación de la tarea.

Tanto en los casos de prueba utilizados para medir calidad externa como en la tarea experimental, los investigadores reportan parcialmente lo relativo a la instrumentación. Sin embargo, se observa que las características de los instrumentos usados por cada investigador es distinto. En cada experimento podrían ser distintos los instrumentos: tarea experimental, nivel de granularidad de la especificación, casos de prueba usados en la medición, la técnica con que fueron elaborados los casos de prueba.

Como se ha visto en las revisiones, los experimentos en TDD en repetidas ocasiones dan resultados distintos y a veces resultados contradictorios. Algunos investigadores ([31], [145], [18]) buscan explicaciones en el contexto del experimento (industrial o académico), en la rigurosidad del estudio ([134]). No consideran que existen otros factores como la instrumentación que podrían estar afectando los resultados de los experimentos.

¹Karac I., Turhan B., Juristo N., A Controlled Experiment on The Impact of Task Description in the Context Test-driven Development, (por enviarse a TSE)

Tabla 2.18: Tarea Bowling Score Keeper (BSK²) en los experimentos en TDD

ID	AUTOR	Industria / Academia	¿Usa BSK?	¿Qué programa usa?	¿Muestra la especificación?	¿Está descompuesta la especificación?
[PS19]	Desai2009 [42]	Academia	No indica	Menciona que son 7 proyectos pero no especifica.	NO	No indica
[PS23]	Edwards2003 [47]	Academia	No indica	No indica	NO	No indica
[PS25]	Erdogmus2005 [50]	Academia	SI	BSK ²	NO	Se divide en problema en varias historias pequeñas
[PS30]	Fucci2013 [60]	Academia	SI	BSK	NO	13 historias de usuario
[PS31]	Fucci2014 [63]	Academia	SI	BSK	NO	13 historias de usuario
[PS32]	Fucci2014b [61]	Academia	SI	BSK	NO	Historias de usuario de diferente dificultad.
[PS33]	Fucci2015 [64]	Academia	SI	BSK	NO	BSK, compuesto de 13 historias de usuario.
[PS108]	Fucci2016 [59]	Academia	SI	BSK	NO	Deducimos que si
[PS37]	George2004 [67]	Industria	SI	Bowling Game	SI en la tesis, no en el paper	NO
[PS38]	Geras2004 [69]	Industria	NO	Program A - Registering a New Project. Program B - Recording Time Against a Project	SI	NO
[PS66]	Madeyski2005 [115]	Academia	NO	Finance-Accounting System	NO	No indica
[PS69]	Madeyski2007b [120]	Industria	NO	Develop a web-based system for academic institution.	NO	Indica que son 30 historias de usuario.
[PS78]	Muller2002 [130]	Academia	NO	GraphBase. Relacionado con gráficos.	NO	No indica

² Bowling Scorekeeper de Robert Martin [123]

Tabla 2.18 – continuación

ID	AUTOR	Industria / Academia	¿Usa BSK?	¿Qué programa usa?	¿Muestra la especificación?	¿Está descompuesta la especificación?
[PS80]	Munir2014 [135]	Industria	SI	BSK ³	SI en la página Web del experimento	Se observa la especificación dividida en 7 historias de usuario
[PS82]	Pancur2003 [139]	Academia	No indica	No indica	NO	No indica
[PS83]	Pancur2011 [138]	Academia	NO	Distributed database server with built-in data replication mechanism. implementation of a chat server	NO	No indica
[PS98]	Vu2009 [169]	Academia	No indica	No indica	NO	No indica

³ Bowling Scorekeeper de Robert Martin [123]

Capítulo 3

Planteamiento del problema y proceso de resolución

Para realizar la investigación contamos con una familia de experimentos en Test-Driven Development (TDD) que venimos replicando hace varios años en el marco del proyecto FidiPro ESEIL (Experimental Software Engineering Industrial Laboratory). TDD es una tecnología sobre la que existen varios experimentos a parte de nuestra familia. Calidad es una de las variables típicamente estudiadas. Como hemos visto en el Capítulo 2 de Estado de la Cuestión, las variables respuesta utilizadas en los experimentos en TDD han sido operacionalizadas con casos de prueba ad hoc realizados por diferentes investigadores y con procedimientos de medición que han sido insuficientemente reportados.

La metodología que se utilizará para la realización de esta investigación es totalmente empírica: estudiaremos el impacto de distintas operacionalizaciones en la familia de experimentos en TDD con el fin de entender en profundidad el impacto de la operacionalización en los resultados de los experimentos de IS. Esta tesis es un primer paso inductivo en la comprensión de este problema. Somos plenamente conscientes de que nuestra familia de experimentos de TDD son solamente un caso y que los resultados que obtengamos para esta familia no tienen porque cumplirse para otros experimentos. No obstante, creemos que trabajar en profundidad en un caso elevará el nivel de comprensión que se tiene en IS sobre los efectos de la operacionalización. Es éste nuestro objetivo, en absoluto dar con la respuesta general y definitiva. Tal objetivo es inalcanzable en la actualidad dada la nula investigación sobre esta temática en IS.

Las actividades a realizar para alcanzar nuestro objetivo serán las siguientes:

- **Comprobación de la existencia del problema.** A través del estudio de la literatura sobre los experimentos en TDD comprobaremos la consistencia de sus resultados. Así como la diversidad de operacionalización de las mismas variables respuesta. Los resultados de esta tarea se ha expuesto en el capítulo 2.
- **Medición de los experimentos de la familia TDD.** Medición de los mismos experimentos variando los elementos de la operacionalización que se estudian. Esta tarea se reporta en los capítulos 5 y 6.

- **Determinación de los efectos e interpretación del impacto** Se estudiará el impacto de los distintos elementos de los métodos de medida en los resultados de los experimentos. Para ello, se tomarán las mediciones obtenidas mediante variaciones de un elemento y se analizará su impacto. Usando los conceptos elaboraremos una interpretación sobre los efectos observados. Los resultados de esta tarea se discute en el capítulo 7.

Considerando la problemática planteada, se propone seguir el siguiente proceso de solución de acuerdo al esquema de la figura 3.1. En tal esquema se muestran las actividades para identificar los elementos de la operacionalización de constructos en IS hasta la determinación del impacto de los mismos en el resultado de los experimentos.

Tal y como hemos reportado en el capítulo anterior, en base a las revisiones sistemáticas de literatura de TDD, se obtuvieron los artículos de la literatura científica que reportan estudios empíricos de TDD. Se seleccionaron los experimentos en donde se estudia la variable respuesta calidad.

Con el propósito de identificar algunos aspectos de la operacionalización, en los experimentos identificados se analizó: casos de prueba, estudiando qué información se incluye en el reporte sobre cómo se generaron y quién lo hizo así como si se muestran o no; qué tarea experimental usan y qué información aportan sobre su especificación.

Adicionalmente, en base a las primeras mediciones realizadas y a los reportes de mediciones de otros experimentos de la literatura científica, se identificó el procedimiento de operacionalización de la variable respuesta calidad en los experimentos en TDD. El procedimiento se podría resumir en los siguientes pasos:

- Elaboración de la variable respuesta.
- Definición de la métrica.
- Definición del instrumento de medición.
- Implementación de instrumentos de medición.
- Verificación del instrumento de medición.
- Recepción del resultado de la tarea experimental.
- Procedimiento de aplicación del instrumento de medición sobre el producto.
- Obtención del resultado de la medición y entrega de los resultados

Una vez que contamos con la primera aproximación al protocolo de medición procedimos a realizar las mediciones necesarias para estudiar el primer elemento, los casos de prueba. Para ello realizamos la medición de un nuevo experimento de la familia TDD (Ericsson), esta vez aplicando una nueva suite de casos de prueba elaborada con la técnica PCdE.

Para que las mediciones con distintos casos de prueba sean comparables, es necesario fijar el medidor y el tipo de intervención en código. Para ello hemos introducido el concepto de tipo de intervención semántica. La intervención semántica es una alternativa a la intervención inicial o ad hoc con la que veníamos trabajando en un inicio, a esta se la denominó intervención sintáctica.

Figura 3.1: Esquema del proceso

Una vez identificados los elementos a estudiar: casos de prueba y tipos de intervención, a continuación se realizaron las mediciones adicionales necesarias para contar con los datos suficientes para realizar los análisis. Se midieron los experimentos: Playtech, ESPEL, FSecure, Ericsson y U. de Basilicata. Estas mediciones se indican en la tabla 3.1 En estas mediciones utilizamos suites de casos de prueba ad hoc y generados mediante particiones de clases de equivalencia (PCdE). Se realizaron los tres tipos de intervención: Sintáctica, Semántica y Sin intervención.

Tabla 3.1: Mediciones de los experimentos con distintos tipos de intervenciones

Experimento	Tipo de intervención	Casos de prueba
Playtech	Sin intervención	Ad hoc
	Sintáctica	
	Semántica	
	Sin intervención	PCdE
	Sintáctica	
	Semántica	
ESPEL	Sin intervención	Ad hoc
	Sintáctica	
	Sin intervención	PCdE
	Sintáctica	
FSecure	Sin intervención	Ad hoc
	Sintáctica	
	Semántica	
	Sin intervención	PCdE
	Sintáctica	
Ericsson	Sintáctica	Ad hoc
	Semántica	
	Sintáctica	PCdE
	Semántica	
U. de Basilicata	Sintáctica	Ad hoc

Aunque se tienen varias mediciones con distintos casos de prueba, para el análisis vamos a utilizar las mediciones correspondientes a los experimentos que son similares en sus diseños (Playtech y Ericsson). Los datos de las mediciones de FSecure que hemos obtenido se utilizarán para analizar el impacto del medidor. La medición del experimento de U. de Basilicata es parte la investigación que propone análisis ciego para que el analista no influya en los resultados del experimento.

Para analizar la influencia del elemento medidor, otros medidores tienen que medir los experimentos: Playtech, FSecure y Universidad del Sur de Dinamarca (USD). En la medición de estos experimentos se va a comparar los valores de la medición realizada por el doctorando con los valores obtenidos por un segundo medidor. Contamos con la colaboración los siguientes medidores: una estudiante del Máster en Software y Sistemas de UPM que midió el experimento Playtech (medidor B), un estudiante de doctorado de la Universidad de Oulu (medidor C) quién midió el experimento FSecure, el medidor C junto con una estudiante de doctorado de la Universidad de Oulu (medidor D) midieron el experimento en USD.

La medición del experimento en la USD ha sido medido utilizando un protocolo de medición por otros dos medidores, por un lado el doctorando y por otro lado los medidores C y D (conjuntamente). Un resumen de las mediciones en la que participan otros colaboradores de esta investigación se presentan en la tabla 3.2

Tabla 3.2: Mediciones realizadas por otros medidores

Experi- mento	Tipo de inter- vención	Casos de prueba	Medi- dor
Playtech	Sintáctica	PCdE	B
	Semántica		
FSecure	Sin intervención	Ad hoc	C
	Sintáctica		
	Semántica		
USD	Sintáctica / Semántica	Ad hoc	C/D

En los análisis del impacto de los medidores se utilizarán los datos de las mediciones de los experimentos Playtech y FSecure.

De esta manera hemos completado la obtención de los datos necesarios para realizar las comparaciones entre los distintos instrumentos, medidores y tipos de intervención. Durante este proceso, se ha visto la oportunidad de contar con datos para futuras investigaciones. Por ello contamos con la medición del experimento SDU, lo que servirá para analizar variaciones entre medidores utilizando un protocolo de medición.

También tenemos previsto en un futuro estudiar las variaciones para un mismo medidor, por este motivo se repitieron las mediciones de los experimentos Playtech y FSecure como indicamos en la tabla 3.3:

Tabla 3.3: Repetición de mediciones de los experimentos con el mismo medidor

Experi- mento	Tipo de inter- vención	Casos de prueba	Medi- dor	Observación
Playtech	Sin intervención	PCdE	F	Segunda medición
	Sintáctica			
	Semántica			
FSecure	Sin intervención	Ad hoc		
	Sintáctica			
	Semántica			

Tras haber realizado las mediciones procederemos a realizar los análisis. En la tabla 3.4 presentamos un listado de las mediciones y los correspondientes análisis en donde se usarán tales mediciones. Para ello tenemos la siguiente nomenclatura: A. Exp: Análisis de los efectos de la experiencia en la calidad y productividad. A. Cie: Análisis de ciego multisitio para que el analista no influya en los resultados. A.CP: Análisis del impacto de los casos de prueba. A. Med: Análisis del impacto del medidor. A.TI: Análisis del impacto de los tipos de intervención.

A continuación, se analiza el impacto de los casos de prueba en los resultados experimentales. Para ello seleccionamos las mediciones de dos experimentos: Playtech y Ericsson. Para analizar los experimentos fijamos el tipo de intervención en sintáctica, medidor F y variamos los tipos de casos de prueba.

Tabla 3.4: Mediciones de experimentos

Experi- mento	Medi- dor	Tipo de Inter- vención	Casos de prueba	Comen- tarios	Análisis				
					A. Exp.	A. Cie.	A. CP	A. Med.	A. TL.
Playtech	FU	Sin intervención	Ad-hoc						X
Playtech	FU	Sintáctica	Ad-hoc		X		X		X
Playtech	FU	Semántica	Ad-hoc						X
Playtech	FU	Sin intervención	PCdE						X
Playtech	FU	Sintáctica	PCdE				X	X	X
Playtech	FU	Semántica	PCdE					X	X
Playtech	FU	Sin intervención	PCdE	Segunda medición					
Playtech	FU	Sintáctica	PCdE	Segunda medición					
Playtech	FU	Semántica	PCdE	Segunda medición					
Playtech	EJ	Sintáctica	PCdE					X	
Playtech	EJ	Semántica	PCdE					X	
FSecure	FU	Sin intervención	Ad-hoc						
FSecure	FU	Sintáctica	Ad-hoc						
FSecure	FU	Semántica	Ad-hoc						
FSecure	FU	Sin intervención	Ad-hoc DF					X	X
FSecure	FU	Sintáctica	Ad-hoc DF					X	X
FSecure	FU	Semántica	Ad-hoc DF					X	X
FSecure	FU	Sin intervención	Ad-hoc	Segunda medición					
FSecure	FU	Sintáctica	Ad-hoc	Segunda medición					
FSecure	FU	Semántica	Ad-hoc	Segunda medición					
FSecure	FU	Sin intervención	PCdE						
FSecure	FU	Sintáctica	PCdE						
FSecure	DF	Sin intervención	Ad-hoc DF					X	X
FSecure	DF	Sintáctica	Ad-hoc DF					X	X
FSecure	DF	Semántica	Ad-hoc DF					X	X
ESPEL	FU	Sin intervención	Ad-hoc						
ESPEL	FU	Sintáctica	Ad-hoc						
ESPEL	FU	Sin intervención	PCdE						
ESPEL	FU	Sintáctica	PCdE						
Ericsson	FU	Sintáctica	Ad-hoc		X		X		
Ericsson	FU	Semántica	Ad-hoc						
Ericsson	FU	Sintáctica	PCdE				X		
Ericsson	FU	Semántica	PCdE						
UPM Se- minario	FU	Sintáctica	Ad-hoc		X				
UPV	FU	Sintáctica	Ad-hoc		X				
Elektrobit	FU	Sin intervención	Ad-hoc		X				
Elektrobit	FU	Sintáctica	Ad-hoc						

Tabla 3.4 – continuación

Experi- mento	Medi- dor	Tipo de Inter- vención	Casos de prueba	Comen- tarios	Análisis				
					A. Exp.	A. Cie.	A. CP	A. Med.	A. TI.
U.S. Di- namarca	FU	Semántica con Protocolo	Ad-hoc						
U.S. Di- namarca	DF/IK	Semántica con Protocolo	Ad-hoc						
PAF	FU	Sintáctica	Ad-hoc						
PAF	FU	Semántica	Ad-hoc						
U. de Ba- silicata 2015	FU	Sintáctica	Ad-hoc			X			

Para analizar el efecto los medidores, seleccionamos las mediciones de los experimentos: Playtech y FSecure. En estos experimentos fijamos los casos de prueba y variamos los medidores. En el experimento de Playtech se ha fijado los casos de prueba en PCdE y en el experimento FSecure se ha fijado los casos de prueba en Ad hoc.

Para analizar el efecto de los tipos de intervención, seleccionamos las mediciones de los experimentos: Playtech y FSecure. Ya que en estos experimentos tenemos los tres tipos de intervención. En estos experimentos fijamos los casos de prueba, el medidor y variamos el tipo de intervención.

Mediante el uso de tres estrategias de comparación se va a determinar las diferencias entre los resultados de la medición con distintos instrumentos. Se aplicará los métodos de ISO, Bland-Altman y Coeficiente de Correlación Intraclass para realizar la comparación. Los resultados de estos métodos son comparables entre sí y nos permitirán cuantificar las diferencias y buscar el origen de la variabilidad.

Capítulo 4

Familia ESEIL de experimentos sobre TDD

Las familias de experimentos son un conjunto de experimentos similares con un mismo objetivo que permiten combinar los resultados de varios experimentos con el propósito de aumentar el conocimiento sobre el proceso software [10]. Ciolkowski [34] indica que una familia de experimentos es más que una serie de experimentos replicados, se pueden introducir variaciones entre los diferentes experimentos y analizarlas.

La familia de experimentos que describimos es parte del proyecto Empirical Software Engineering Industry Lab (ESEIL) liderado por N. Juristo y financiado por TEKES¹ en su programa FiDiPro². En total se han ejecutado 11 experimentos realizados en distintos países en el ámbito industrial y académico.

Estas 11 replications comparten un objetivo común y son bastante similares entre ellas, pero incorporan diferencias de diseño y operacionalización con el objetivo de clarificar los resultados. Seis replications fueron realizadas en el ámbito industrial y cinco en el ámbito académico. Por tanto, esta familia es un banco de pruebas excepcional para estudiar el impacto de la operacionalización en los resultados experimentales. En el ámbito industrial participaron 97 profesionales como sujetos experimentales y en las universidades participaron 191 estudiantes. En total 288 sujetos experimentales. Los experimentos se ejecutaron a partir del año 2013 hasta el 2016. Las replications se ejecutaron en varios países: Finlandia, Malasia, Estonia, España, Ecuador, Italia y Dinamarca.

En los experimentos se han utilizado varios lenguajes de programación, dependiendo de la compañía o universidad hemos trabajado con Java, C++, PL SQL, C#. Los ambientes de desarrollo han sido diferentes versiones de Eclipse, PLSQL Developer y Visual Studio. Como framework de pruebas hemos usado JUnit, Google Test, Boost Test Library, UTPLSQL, Visual Studio Testing Tools e IntelliJ. En la tabla 4.1 presentamos un listado de las replications del experimento de TDD.

¹TEKES: Finnish Funding Agency for Technology and Innovation <https://www.tekes.fi/en>

²FiDiPro: Finland Distinguished Professor Programme <http://fidipro.fi>

Tabla 4.1: Familia ESEIL de experimentos en TDD

Industria / Academia	ID	Organización	Lugar	País	Fecha	Tipo de sujeto	Num. sujetos	Len-gua-je	IDE	Frame-work de pruebas
Industria	FS	F-Secure	Oulu, Kuala Lumpur y Helsinki	Finlandia, Malasia	Sep - Dec/2013	Profesionales (7-Oulu, 11-Kuala Lumpur, 6-Helsinki)	24	Java	Eclipse	JUnit
	EB	Elektrobit	Kajaani	Finlandia	Feb/2014	Profesionales	9	C++	Eclipse CDT under Linux	Google Test
	PT	Playtech	Tartu	Estonia	Abr/2014	Profesionales	17	Java	Eclipse	JUnit
	ERC	Ericsson	Jorvas	Finlandia	Abr/2015	Profesionales	20	C++	Eclipse, Vim	Boost
	MFR	MAPFRE RE	Madrid	España	Jun/2015	Profesionales	14	PL-SQL	PLSQL Developer	UTPLSQL
	PAF	PAF	Helsinki	Finlandia	Mar/2016	Profesionales	13	Java	Eclipse	JUnit, IntelliJ
Academia	UPM	Seminario UPM	Madrid	España	Mar/2014	Estudiantes de Master	16	Java	Eclipse Helios	JUnit
	UPV	UPV	Valencia	España	May/2014	Estudiantes de Master(MUII)	33	Java	Eclipse IDT	JUnit
	ESP	ESPEL	Latacunga	Ecuador	May/2014	Estudiantes de Master	14	Java	Eclipse	JUnit
	UdB	U. di Basilicata	Potenza	Italia	Abr/2015	Estudiantes de Master	21	Java	Eclipse	JUnit
	USD	University of Southern Denmark	Odense	Dinamarca	Nov/2015	12 estudiantes de Master y 59 estudiantes de pregrado.	71	C#	Visual Studio	Visual Studio Testing Tool

4.1. Descripción del experimento

En esta sección se van a describir las características que comparten todas las replicas, tales como los factores, variable respuesta y tareas.

La principal **pregunta de investigación** de los experimentos es:

RQ1: ¿TDD afecta la calidad del código comparado con la aproximación de desarrollo ITL?

En los experimentos tenemos la **hipótesis** correspondiente a la calidad. H10 y H11 corresponde a la calidad externa.

H10 (Hipótesis nula): No hay diferencia de los efectos de TDD e ITL en la calidad.

H11 (Hipótesis alternativa): Existe diferencia de entre los efectos de TDD e ITL en la calidad.

4.1.1. Factores y tratamientos

El factor que se tiene en los experimentos es la *técnica de desarrollo*. Los sujetos siguen una técnica de desarrollo para completar las distintas tareas experimentales. Los tratamientos son: Iterative Test Last (ITL) y Test Driven Development (TDD).

La técnica de desarrollo Iterative Test Last (**ITL**) es usada como control en nuestros experimentos. La técnica implica el desarrollo de un producto a través de una secuencia de pasos sucesivos de diseño e implementación [11]. Luego de la implementación de código fuente se escribe el caso de prueba.

El desarrollo iterativo incremental de software se ha practicado en proyectos desde mediados de los años 50 [108]. A mediados de los 70, Basili & Turner describen la clásica mejora iterativa como un enfoque de refinamiento escalonado de arriba hacia abajo para el desarrollo de software. La técnica comienza con una implementación inicial de un subproyecto seguido por la mejora gradual de implementaciones sucesivas para construir la implementación completa [11].

TDD consiste en escribir la prueba unitaria, ejecutar todas las pruebas y ver el nuevo fallo, escribir el código fuente, ejecutar todas las pruebas, ver que todos sean exitosos y refactorizar para eliminar duplicaciones. El objetivo es llegar a tener código limpio que trabaje [13].

En algunos experimentos se ha usado como control la técnica Test Last (**TL**). Cuando se aplica esta técnica no se refuerza que los participantes descompongan en tareas diminutas, al estilo TDD pero con prueba al final. En dos experimentos (Mapfre y PAF) se aplica como tratamiento control: A tu modo (YW) en lugar de ITL. En YW el sujeto usa su propia técnica de desarrollo.

En tres experimentos (Ericsson, Mapfre y USD) se tiene como factor al *nivel de granularidad* de las especificaciones con los tratamientos: Especificaciones de grano fino (Descompuesto en rebanadas) y Especificaciones de grano grueso (No descompuesto en rebanadas). El descomponer en rebanadas es una estrategia de descomposición de una tarea. Ejemplos de estrategias de descomposición bien conocidas son la descomposición funcional y la abstracción de datos subadyacentes a la orientación del objeto. La estrategia

de slicing requerida por TDD consiste en cortar verticalmente una tarea, aislando de las proyecciones del comportamiento requerido. La yuxtaposición de todos los cortes nos da el código requerido.

Dependiendo del diseño del experimento se han entregado a los participantes las especificaciones descompuestas en rebanadas o no descompuestas.

En la tabla 4.2 presentamos los factores y tratamientos aplicados en los experimentos de la familia.

Tabla 4.2: Factores y tratamientos por experimento

Factor	Tratamiento	Experimentos
Técnica de desarrollo	TDD	FS, EB, UPM, PT, UPV, ESP, ERC, MFR, UdB, USD, PAF
	ITL	PT, UPV, ERC, PAF
	TL	FS, EB, UPM, ESP, UdB, USD
	YW	MFR, PAF
Granularidad de especificación	Fino	ERC, MFR, USD
	Grueso	ERC, MFR, USD

4.1.2. Variable respuesta

En las repeticiones ejecutadas tenemos la variable respuesta: Calidad (QLTY) [49]. La **calidad** se refiere a la totalidad de las características del producto software desde un punto de vista externo, es la calidad cuando el software es ejecutado, el cual es medido y evaluado mientras se prueba en un ambiente simulado con datos simulados usando métricas externas [82].

El modelo de calidad del producto software ISO/IEC 25010:2011 [84] está compuesto de características y subcaracterísticas, la característica que nos interesa es la funcionalidad, la cual se refiere al grado en que un producto o sistema proporciona funciones que cumplen con las necesidades expresadas o implícitas cuando se utiliza en condiciones especificadas. La funcionalidad tiene subcaracterísticas y las presentamos en la tabla 4.3.

Tabla 4.3: Composición de la funcionalidad según ISO/IEC 25010:2011 [84]

Subcaracterística	Definición
Compleitud funcional	Grado en que el conjunto de funciones cubre todas las tareas y objetivos de usuario especificados.
Corrección funcional	Grado en que un producto o sistema ofrece los resultados correctos con el grado necesario de precisión requerido.
Pertinencia funcional	Grado en que las funciones facilitan la realización de tareas y objetivos especificados.

La subcaracterística Compleitud funcional se refiere a cuánto de la funcionalidad esperada ha sido entregada [74]. Esta subcaracterística no se interesa si la funcionalidad

entregada es correcta o apropiada, solo le interesa si está presente la funcionalidad en el producto.

La subcaracterística Corrección funcional, corresponde a la subcaracterística exactitud del estandar anterior ISO-9126 [74]. Exactitud, se refiere a la capacidad del producto software para proporcionar los resultados o efectos correctos o acordados con un grado necesario de precisión. Los requerimientos de exactitud con frecuencia están implícitos. En ISO/IEC TR 9126-2 [83] tenemos los siguientes ejemplos de métricas de calidad externa correspondientes a la subcaracterística exactitud:

- Diferencias entre los resultados actuales y esperados de las tareas ejecutadas durante la operación.
- Inconsistencia entre los procedimientos de operación actuales y los descritos en la operación manual.
- Resultados imprecisos o incorrectos causados por datos inadecuados, por ejemplo datos con muy pocos dígitos significativos para cálculos de precisión.

Por último, la tercera subcaracterística Pertinencia funcional hace referencia a que el producto software no es el fin en sí mismo, éste da soporte a los usuarios en hacer sus trabajos reales. El software es una herramienta para hacer el trabajo rápido y más confiable. Un ejemplo de adecuación funcional: al usuario se le presentan únicamente los pasos necesarios para completar una tarea, con exclusión de los pasos innecesarios.

La calidad total del código fuente de los sujetos la medimos como una combinación entre completitud y corrección funcional. Para calcular la calidad total se tiene la siguiente ecuación 4.1:

$$QLTY = \frac{\#Assert(PASS)}{\#Assert(ALL)} \quad (4.1)$$

$\#Assert(PASS)$ representa el número total de aserciones que pasan el conjunto de casos de prueba de aceptación de la tarea. $\#Assert(ALL)$ representa el número total de aserciones de los casos de prueba de la tarea experimental. En la tabla 4.4 presentamos la variable respuesta usados en cada experimento.

Tabla 4.4: Variables respuesta por experimento

Experimento	Variable respuesta	Descripción
FS, EB, UPM, PT, UPV, ESP, ERC, UdB, USD, PAF	Calidad	Porcentaje de aserciones pasadas sobre el total de aserciones creadas.
	Compleitud funcional	Porcentaje de historias de usuario que pasan por lo menos una aserción del conjunto de casos de prueba correspondiente a la historia de usuario.
MFR	Calidad total	Porcentaje de casos de prueba pasados sobre el total de casos de prueba creados.
	Compleitud funcional	Número de historias de usuario abordadas según lo indicado por el participante.

4.1.3. Tareas experimentales

Los sujetos aplican los tratamientos principalmente en dos problemas experimentales: Mars Rover API (MR) y Bowling Score Keeper (BSK). En algunos experimentos se usaron otras tareas experimentales: Music Phone (MP), Spreadsheet (SS) y Sudoku, así como tareas proporcionadas por la empresa donde se condujo el experimento. Tanto MR como BSK son ejercicios populares usados por la comunidad ágil. Las tareas usadas tienen distintos desafíos de programación y nos permitirá entender mejor la generalización de los resultados.

MR es un ejercicio de programación que requiere la implementación de una interfase pública para controlar el movimiento de un vehículo virtual en un grid con obstáculos. MR no implica la creación de una interfase de usuario. MR es un vehículo que se desplaza sobre un planeta virtual representado por un grid de dimensiones: $N \times N$. Cada celda del grid está referenciada por las coordenadas (X, Y) y podría contener o no, un solo obstáculo. El obstáculo de una celda impide el paso del vehículo. La implementación necesita manejar los casos en donde el vehículo se desplaza en el grid, supera los límites del planeta, encuentra obstáculos. Las posibles operaciones de MR son las siguientes:

- Inicialización del grid y asignación de obstáculos a las celdas correspondientes
- Movimientos forward y backward
- Giros a la derecha e izquierda
- Superar los límites del planeta
- Identificación y registro de un obstáculo.

Los sujetos reciben el documento con la especificación y una plantilla de código fuente con el objetivo de que los sujetos tenga una estructura de clases y métodos comunes, lo cual facilitaría el proceso de medición.

BSK es una versión modificada del Bowling Scorekeeper de Robert Martin [123]. En BSK, un frame está compuesto de dos tiros. Un juego está compuesto de un conjunto de frames. El objetivo de la tarea es calcular el marcador de un juego de bolos. La tarea no implica la creación de una interfase de usuario.

Se entrega a los sujetos la especificación completa y una plantilla de código fuente con el nombre de las clases y los métodos. Las operaciones principales de BSK son las siguientes:

- Adicionar un frame
- Adicionar un bonus
- Identificar cuando un frame es spare o strike
- Calcular un frame score
- Calcular el marcador del juego.

Todas las operaciones son importantes sin embargo la que demanda mayor esfuerzo es el cálculo del marcador del juego. Ya que depende del tipo de frame (regular, spare y strike) y de la posición del frame en el juego y si o no el próximo frame es strike.

Spreadsheet es una aplicación que simula el manejo de una hoja de cálculo, contiene celdas en las que se realizan operaciones matemáticas básicas y concatenación de cadenas (Strings). Las operaciones matemáticas son: suma, resta, multiplicación y división. En las operaciones se puede incluir referencias a otras celdas de la misma manera que en una hoja

de cálculo. Además el sujeto debe implementar el control de errores en las operaciones. Presentamos una descripción completa de la tarea Spreadsheet en el Anexo B.5

MP es una aplicación destinada a ejecutarse en un teléfono móvil compatible con GPS y con capacidad para MP3. Se asemeja a un sistema del mundo real con una arquitectura de tres niveles (interfaz gráfica de usuario, lógica de negocio y acceso a datos). El sistema consta de tres componentes principales que se crean y se acceden utilizando el patrón Singleton. El manejo es implementado usando el patrón Observer. La tarea se subdivide en 11 historias de usuario de bajo nivel. Se proporciona una descripción del código heredado, incluyendo las clases existentes, sus API y un diagrama de arquitectura del sistema. Presentamos una descripción completa de la tarea Music Phone en el Anexo B.6

Sudoku (**SK**) es un ejercicio de programación, de tipo green-field (carece de cualquier restricción impuesta por un proceso anterior). El ejercicio requiere el desarrollo de varias reglas de control contra una solución propuesta para un juego de Sudoku. Específicamente, el sujeto debe tratar con las operaciones de String y matriz y con la incorporación de toda la funcionalidad dentro de un único API. La tarea se subdivide en seis historias de usuario de bajo nivel. En el Anexo B.7 se proporciona una descripción completa de la tarea Sudoku.

En la tabla 4.5 presentamos un listado de los experimentos en donde se usó cada tarea experimental.

Tabla 4.5: Tareas experimentales por experimento

Tarea experimental	Experimentos
MR	FS, EB, UPM, PT, UPV, ESP, ERC, UdB, PAF
BSK	FS, EB, UPM, PT, UPV, ESP, ERC, UdB, USD, PAF
Spreadsheet	PAF
MP	FS, EB, UPM, PT, ESP
Sudoku	UPM, ESP
Propia de la empresa (DL)	MFR
Propia de la empresa (DC)	MFR

4.1.4. Casos de prueba de aceptación para obtención de métricas

Para realizar el proceso de extracción de la métrica calidad usamos casos de prueba. Los casos de prueba se pasan por el código fuente de los sujetos. En base a los casos de prueba pasados se aplica las fórmulas descritas en la sección 4.1.2 y se obtiene el valor de la medida. Contamos con una suite de casos de prueba de aceptación para cada tarea con el fin de estudiar el efecto de un elemento relevante de la instrumentación de métricas como son los casos de prueba. Para las tareas MR y BSK, se cuenta con dos suites de casos de prueba, generados con dos técnicas distintas: Ad hoc (AH) y PCdE. La tabla 4.6 presenta el total de casos de prueba y aserciones para los casos con PCdE y Ad hoc .

Las tablas 4.7 y 4.8 muestran el total de aserciones y casos de prueba por cada historia de usuario contenida en cada suite para MR y BSK.

Tabla 4.6: Total de casos de prueba y aserciones por cada suite de casos de prueba

MR				BSK			
N. Casos de prueba		N. Aserciones		N. Casos de prueba		N. Aserciones	
AH	PCdE	AH	PCdE	AH	PCdE	AH	PCdE
52	51	89	51	48	72	56	72

Tabla 4.7: Número de casos de prueba y aserciones Ad hoc

MR			BSK		
Subtarea	N. Tests	N. Aserciones	Subtarea	N. Tests	N. Aserciones
1	1	1	1	3	3
2	7	11	2	2	3
3	4	8	3	2	2
4	4	7	4	3	10
5	4	8	5	5	5
6	8	8	6	6	6
7	6	15	7	8	8
8	4	8	8	5	5
9	3	8	9	5	5
10	3	7	10	2	2
11	8	8	11	2	2
NA	NA	NA	12	3	3
NA	NA	NA	13	2	2
TOTAL:	52	89		48	56

Tabla 4.8: Número casos de prueba y aserciones con PCdE

MR			BSK		
Subtarea	N. Tests	N. Aserciones	Subtarea	N. Tests	N. Aserciones
0	4	4	1	4	4
1	15	15	2	5	5
2	2	2	3	6	6
3	2	2	4	4	4
4	2	2	5	7	7
5	2	2	6	7	7
6	9	9	7	4	4
7	1	1	8	3	3
8	1	1	9	4	4
9	1	1	10	7	7
10	1	1	11	13	13
11	1	1	12	1	1
12	10	10	13	3	3
NA	NA	NA	14	4	4
TOTAL:	51	51		72	72

Las suites de casos de prueba han sido implementados en el lenguaje del framework de pruebas correspondiente en cada experimento. En las tablas: 4.9, 4.11 y 4.10 presentamos el número de casos de prueba y aserciones para las tareas MP, Spreadsheet y Sudoku respectivamente.

Tabla 4.9: Número casos de prueba y aserciones de la tarea MP

MP		
Subtarea	N. Tests	N. Aserciones
1	4	4
2	3	12
3	3	12
4	4	4
5	7	17
6	8	12
7	7	17
8	1	1
9	2	11
10	2	13
11	4	20
TOTAL:	45	123

Tabla 4.10: Número casos de prueba y aserciones de la tarea Spreadsheet

Spreadsheet		
Subtarea	N. Tests	N. Aserciones
0	1	1
1	5	5
2	5	5
3	1	1
4	2	2
5	7	7
5B	4	4
6	1	1
7	1	1
8	1	1
9	7	7
9B	3	3
10	2	2
11	2	2
12	1	1
TOTAL:	43	43

Tabla 4.11: Número casos de prueba y aserciones de la tarea Sudoku

Sudoku		
Subtarea	N. Tests	N. Aserciones
1	1	1
2	3	3
3	1	1
4	1	1
5	1	1
6	4	6
TOTAL:	11	13

En la tabla 4.12 mostramos las suites de casos de prueba aplicados a cada experimento de la familia.

Tabla 4.12: Suite de casos de prueba por experimento

Tarea	Casos de prueba	Experimentos
MR	Ad hoc	FS, EB, UPM, PT, UPV, ESP, ERC, UdB, PAF
	PCdE	FS, PT, ERC
BSK	Ad hoc	FS, EB, UPM, PT, UPV, ESP, ERC, UdB, USD, PAF
	PCdE	FS, PT, ERC
MP	Ad hoc	FS, EB, UPM, PT, ESP
Sudoku	Ad hoc	UPM, ESP
Spreadsheet	Ad hoc	PAF

4.2. Operacionalización

Previo al experimento se lleva a cabo una encuesta demográfica dirigida a los sujetos antes de la ejecución del training en ITL. La encuesta se realiza con el propósito de obtener información más detallada de sus características. Los sujetos responden a preguntas relacionadas con su experiencia en programación, lenguaje de programación, pruebas unitarias, framework de pruebas y TDD, así como información acerca de estudios y cursos previos en TDD. Esta encuesta nos permite asignar mediante aleatorización estratificada los sujetos a tratamientos y tareas.

Se ha incentivado a los sujetos con el training gratis, lo cual ayuda a asegurar la participación de partners industriales. Esto ha atraído principalmente a participantes que no tienen ningún conocimiento de TDD. Los sujetos trabajan en las tareas experimentales en sesiones que se realizan únicamente en el laboratorio.

Los sujetos reciben un documento impreso con la especificación de la tarea experimental y un esqueleto de programa en Java o en el lenguaje correspondiente con el propósito

empiece lo más pronto posible la implementación de la tarea.

Habitualmente los experimentos de esta familia tienen una duración de 8 horas, distribuidos como se muestra en la tabla 4.13:

- Las primeras cuatro horas se hace el entrenamiento de los sujetos en el uso del framework de testing y se hacen ejercicios prácticos. El entrenamiento lo han realizado los investigadores B. Turhan y O. Dieste (dependiendo del experimento).
 - Cuando se aplica el tratamiento YW, el experimento empieza con los participantes implementado una tarea experimental mediante su modo habitual de desarrollo.
 - Después del entrenamiento, las tareas experimentales son implementadas por los sujetos y tienen una duración aproximada de tres horas sin descanso. De acuerdo al diseño del experimento las tareas experimentales MR, BSK son asignadas a los sujetos. El desarrollo del experimento ha sido supervisado, dependiendo del caso por D. Fucci, A. Tonsum y S. Vegas.
 - Al final del experimento los sujetos entregan su código fuente y se finaliza el experimento con el llenado de un cuestionario.
 - En la fase de operacionalización de los experimentos, las métricas fueron extraídas por D. Fucci, O. Dieste y F. Uyaguari, dependiendo del experimento.
- Las particularidades de cada una de las replicaciones se comentan en la siguiente sección.

Tabla 4.13: Cronograma del experimento FSecure (Planeado)

	DIA 1	DIA 2	DIA 3
15 minutos	Intro	Entrenamiento TDD	Tarea TDD - MP
15 minutos	Cuestionario demográfico		
60 minutos	Entrenamiento en UT		
15 minutos	Descanso		
75 minutos	ITL Kata (Randori)	ITL Kata (Randori)	
60 minutos	Lunch		
60 minutos	ITL Kata (Randori)	TDD Kata (Randori)	TDD Kata (Randori)
15 minutos	Descanso	Descanso	
135 minutos	Tarea ITL - MR API	Tarea TDD - BSK	Retrospectiva
30 minutos			

En la tabla 4.14 presentamos los nombres de las personas que desempeñaron cada rol en los experimentos.

Tabla 4.14: Roles por experimento

Industria / Academia	ID	Trainer	Supervisión	Medición	Re medición
Industria	FS	B. Turhan	N. Juristo	D. Fucci	F. Uyaguari
	EB	B. Turhan	N. Juristo	F. Uyaguari	
	PT	O. Dieste	N. Juristo	F. Uyaguari	
	ERC	O. Dieste	N. Juristo	F. Uyaguari	
	MFR	O. Dieste	N. Juristo	O. Dieste	
	PAF	O. Dieste	N. Juristo	F. Uyaguari	
Academia	UPM	B. Turhan	N. Juristo	F. Uyaguari	
	UPV	O. Dieste	N. Juristo	F. Uyaguari	
	ESP	O. Dieste	N. Juristo	O. Dieste	F. Uyaguari
	UdB	G. Scanniello	D. Fucci	F. Uyaguari	
	USD	I. Karac	B. Turhan	D. Fucci / I. Karac	F. Uyaguari

4.3. Replicaciones ejecutadas

En las tablas 4.15 y 4.16 presentamos las configuraciones específicas de cada replicación.

4.3.1. FSecure

El primer experimento de la familia fué ejecutado en la compañía F-Secure. En FSecure el experimento se realizó en tres localizaciones: Oulu, Kuala Lumpur y Helsinki. Este experimento tienen como base el experimento de Erdogmus et. al [50] ya replicado en [60] con ciertas modificaciones en el diseño. El diseño aplicado en el experimento base fue un factor y dos niveles con diseño entre sujetos (between-subjects). En FSecure usamos un diseño con un factor y dos niveles pero **inter-sujetos (within-subjects)**. El factor es la aproximación de desarrollo y los niveles o tratamientos: TL y TDD. Este mismo diseño se usa en EB y UPV.

El diseño inter-sujetos ofrece ventajas sobre el diseño del experimento base que fue entre-sujetos. Una de las fuentes más importantes de variación en los experimentos en IS es el nivel de habilidad de los sujetos. Esta variación puede ser controlada usando el diseño de tratamientos repetidos o inter-sujetos [157]. De esta manera las habilidades de cada sujeto impactan a todos los tratamientos de la misma forma [23] puesto que todos los sujetos aplican todos los tratamientos. En un diseño de tratamientos repetidos, cada sujeto se compara con él mismo, anulando su variabilidad inherente entre sujetos y de esta forma aumenta el poder [157]. En la tabla 4.17 se muestra el diseño del experimento FSecure.

Tabla 4.15: Configuraciones de los experimentos - Parte 1

Aspecto	FSecure	Elektrobit	Seminario UPM	Playtech	UPV	ESPEL
Tratamientos	TDD vs TL	TDD vs TL	TDD vs TL	TDD vs ITL	TDD vs ITL	TDD vs TL
Diseño	Un factor y dos niveles within-subjects	Un factor y dos niveles within-subjects	Un factor y dos niveles ABAB within-subjects	Dos factores tres niveles (2x3) within-subjects	Un factor y dos niveles within-subjects	Dos factores tres niveles (2x3) within-subjects
Tipo de sujetos	Profesionales	Profesionales	Informática, Estudiantes de Master	Profesionales	Estudiantes de Master	Estudiantes de Master
Cant. sujetos	24 individuos	9 individuos	16 individuos	17 individuos	33 individuos	14 individuos
Cant. unidades experimentales	72	27	57	50	60	34
Training	Curso ITL, TDD	Curso ITL, TDD	Curso de pruebas unitarias, TDD	Curso ITL, TDD	Curso ITL, TDD	Curso TDD
Duración del training	4 días / 7 horas	3 días / 6 horas	5 días / 6 horas	3 días / 11 horas	2 días	4 días / 20 horas
Lenguaje de programación	Java	C++	Java	Java	Java	Java
Ambiente de desarrollo	Eclipse	Eclipse CDT bajo Linux	Eclipse Helios	Eclipse	Eclipse IDE	Eclipse
Framework de pruebas	Junit	Google Test	Junit	Junit	Junit	Junit
Variables res-puesta	QLTY	QLTY	QLTY	QLTY	QLTY	QLTY
Tareas	MR, BSK, MP	MR, BSK, MP	MR, BSK, MP y Sudoku	MR, BSK, MP	MR, BSK	MR, BSK, MP y Sudoku
Duración de la tarea	2h 15min. BSK/MR y 3h MP	2h MR y BSK, 3h para MP	2h 15min. BSK/MR y 3h MP	2h 15min. BSK/MR y 3h MP.	2h 00min	2h 00min. con cierta flexibilidad

Tabla 4.16: Configuraciones de los experimentos - Parte 2

Aspecto	Ericsson	MAPFRE	U. de Basilicata	USD	PAF
Tratamientos	TDD vs ITL; Slicing vs. No Slicing	TDD vs YW; Slicing vs. No Slicing	TDD vs. TL	TDD vs. TL; CG vs. FG	YW, ITL, TDD
Diseño	Dos factores dos niveles (2x2) within-subjects con una blocking variable (tarea experimental)	Dos factores dos niveles (2x2) within-subjects con una blocking variable (tarea experimental)	Un factor dos niveles. Balanced crossover design.	Dos factores dos niveles (2x2) between-subjects con una blocking variable (nivel académico)	Un factor tres niveles within-subjects con una blocking variable (tarea experimental)
Tipo de sujetos	Profesionales	Profesionales	Estudiantes de Master y pregrado	Estudiantes de Master y estudiantes de pregrado	Profesionales
Cant. sujetos	20 individuos	14 individuos	21 individuos	71 individuos	13 individuos
Cant. unidades experimentales	40	28	40	71	28
Training	Curso de pruebas unitarias, ITL/Slicing, TDD	Curso de pruebas unitarias, Slicing, TDD	Curso de pruebas unitarias, TDD, refactoring.	Período 1 y período 2, Master y Pregrado	Curso de herramientas de testing, slicing, TDD
Duración del training	3 días / 9 horas	3 días	4 días / 3 horas cada día		3 días
Lenguaje de programación	C++	PL SQL	Java	C#	Java
Ambiente de desarrollo	Eclipse Vim	PL SQL Developer	Eclipse	Visual Studio	Eclipse
Framework de pruebas	Boost	UTPLSQL	Junit	VS Testing Tool	Junit, IntelliJ
Variables res-puesta	QLTY	QLTY	TEST, QLTY	QLTY	QLTY
Tareas	MR, BSK	DL, DC	MR, BSK	BSK	MR, BSK, SS
Duración de la tarea	2h 30min.	2h 00min	-	-	2h 00min

Tabla 4.17: Diseño del experimento FSecure

Día	Sesión	Tratamiento	Tarea		
			MR	BSK	MP
Día 1	Sesión 1	ITL	G1		
Día 2	Sesión 2	TDD		G1	
Día 5	Sesión 3	TDD			G1

En el experimento FSecure los 24 sujetos usan ITL en la primera sesión y TDD en la segunda y tercera sesión. El experimento fue realizado en 5 días ubicándose las tres sesiones experimentales en los días 1,3 y 5, con la siguiente distribución [49]:

- Día 1 (Sesión 1): ITL training e implementación de la tarea de control MR, usando ITL
- Día 2 (Sesión 2): TDD training e implementación de BSK usando TDD
- Día 3 y 4: Días libres para practicar TDD
- Día 5 (Sesión 3): Implementación de MP, tarea de mayor complejidad, usando TDD

El objetivo de embeber un experimento de 3 sesiones en un curso de 5 días era permitir que los asistentes trabajaran en los días 3 y 4 aplicando TDD en sus tareas diarias de programación. Este diseño permitiría estudiar la curva de aprendizaje de TDD y comparar la sesión 2 y 3.

Las tareas de MR y BSK tienen un nivel de complejidad equivalente. MR y BSK son las únicas consideradas dentro de la medición y análisis del experimento, dejando fuera MP por haber resultado difícil en exceso a los participantes. Desafortunadamente los asistentes al curso no dedicaron los días 3 y 4 a practicar con TDD, sino a recuperar los días anteriores en tareas como correo electrónico y reuniones. Por lo tanto, el objetivo de estudiar la curva de aprendizaje de TDD quedó trastocado y variado de contenido. Por lo tanto el diseño modificado del experimento que acabó midiéndose y analizando fue la parte con fondo blanco de la tabla 4.17.

Este diseño tiene una importante amenaza interna: confunde los tratamientos con las tareas (ITL con MR; TDD con BSK). Fuimos conscientes de esta amenaza desde el planteamiento del diseño, sin embargo nos vimos obligados a aceptarla por ser éste el primer experimento en empresa que realizamos. Recuérdese que el experimento estaba embebido en un curso de formación en TDD y que éste fue el único modo que encontramos de reclutar participantes. Nos pareció demasiado arriesgado dividir a los asistentes en grupos diferentes realizando tareas diferentes. Con el tiempo, como veremos en las secciones siguientes, la experiencia nos permitió aventurarnos en diseños experimentales que claramente no encajan con un curso de entrenamiento para profesionales.

4.3.2. Elektrobit

El experimento Elektrobit también tiene un diseño de un factor dos niveles inter-subjects. Dada la mala experiencia con los días 3 y 4 del experimento en FS, reducimos el experimento de Elektrobit a 3 días. En la tabla 4.18 presentamos el diseño del experimento.

En el experimento Elektrobit participan 9 individuos y se tiene la siguiente distribución [49]:

Tabla 4.18: Diseño del experimento Elektrobit

Día	Sesión	Tratamiento	Tarea		
			MR	BSK	MP
Día 1	Sesión 1	ITL	G1		
Día 2	Sesión 2	TDD		G1	
Día 3	Sesión 3	TDD			G1

- Día 1: ITL training e implementación de la tarea de control MR, usando ITL
- Día 2: TDD training e implementación de BSK usando TDD
- Día 3: Implementación de MP (tarea compleja) usando TDD

A partir de este experimento no se usará MP, se usarán únicamente las tareas MR y BSK.

4.3.3. UPV

En el experimento en la Universidad Politécnica de Valencia (UPV) se aplica de nuevo el diseño de un factor dos niveles inter-sujetos con tarea y tratamiento confundidos. A pesar de las limitaciones conocidas de este diseño, el objetivo de replicar en UPV era comparar el rendimiento entre profesionales y estudiantes, por lo tanto la réplica debía ser idéntica. El experimento se realizó en tres días al igual que en Elektrobit. En UPV sin embargo se usa la tarea experimental Sudoku en lugar de MP dada la excesiva complejidad que los participantes de FSecure y Elektrobit encontraron en esta tarea. Por otro lado, dado que la sesión 3 no estaba siendo medida ni analizada en FSecure y Elektrobit, este cambio no resulta impactante en los resultados experimentales. En la tabla 4.19 presentamos el diseño del experimento UPV.

Tabla 4.19: Diseño del experimento UPV

Día	Sesión	Tratamiento	Tarea		
			MR	BSK	Sudoku
Día 1	Sesión 1	ITL	G1		
Día 2	Sesión 2	TDD		G1	
Día 3	Sesión 3	TDD			G1

En el experimento participan 33 individuos y la distribución del experimento es de la siguiente forma:

- Día 1: ITL training e implementación de la tarea de control MR, usando ITL
- Día 2: TDD training e implementación de BSK usando TDD
- Día 3: Implementación de SK usando TDD.

La tarea SK tiene menor dificultad que MP. Por problemas durante el almacenamiento del código fuente, esta tarea no fue medida y tampoco fue considerada en el análisis.

4.3.4. UPM Seminario 2014

En este experimento se desacopla la tarea experimental del tratamiento de modo que se pueda apreciar el efecto de TDD [49]: Diferentes sujetos trabajarán en diferentes tareas en la misma aproximación de desarrollo. Por lo tanto se trata de un diseño de un factor y dos niveles **ABAB inter-subjects** [49]. La tarea pretendía ser una variable de bloque dado que la cantidad de sujetos no permitía estudiar a fondo la influencia de la tarea. En la tabla 4.20 se presenta el diseño del experimento.

El experimento se realizó en 5 días y tiene la siguiente configuración:

- Día 1: Unit testing training. 4 sujetos implementan MR, 3 BSK, 5 MP y 4 Sudoku usando ITL
- Día 2: TDD training. Ejercicios con TDD.
- Día 3: 4 sujetos implementan MR, 5 BSK, 3 MP y 4 Sudoku usando TDD. Ejercicios con TDD.
- Día 4: 4 sujetos implementan MR, 4 BSK, 3 MP y 5 Sudoku usando ITL. Ejercicios con TDD.
- Día 5: 4 sujetos implementan MR, 4 BSK, 5 MP y 3 Sudoku usando TDD

Tabla 4.20: Diseño del experimento Seminario UPM

Día	Sesión	Tratamiento	Tarea			
			MR	BSK	MP	SK
Día 1	Sesión 1	TL	G1	G2	G3	G4
Día 3	Sesión 2	TDD	G2	G1	G4	G3
Día 4	Sesión 3	TL	G3	G4	G1	G2
Día 5	Sesión 4	TDD	G4	G3	G2	G1

Con este diseño los sujetos aplican los tratamientos dos veces. De esta manera se aumenta la sensibilidad del experimento, los efectos del tratamiento de un sujeto son medidos con respecto al promedio de todos los tratamientos. Los sujetos pueden aplicar las técnicas más de una vez, por lo tanto podrían incrementar su performance durante la segunda aplicación del tratamiento.

4.3.5. Playtech

En este experimento se aplica un diseño con **dos factores y tres niveles (2x3) inter-sujetos**. Los factores son la aproximación de desarrollo y la tarea experimental. La cantidad de sujetos del experimento debería permitir estudiar la tarea como factor. Para ello redefinimos las tareas de nuevo a dos en comparación con UPM. La aproximación de desarrollo usadas son: ITL y TDD. Las tareas son: MR y BSK. De esta manera se puede estudiar cómo afecta la aproximación de desarrollo y como influye la tarea experimental en la calidad así como la existencia de interacción entre ellas. En la tabla 4.21 presentamos el diseño del experimento.

El experimento de Playtech se realizó en tres días, en el grupo G1 se tienen 8 participantes, en el grupo G2 9 participantes y se usó la siguiente distribución:

Tabla 4.21: Diseño del experimento Playtech

Día	Sesión	Tratamiento	Tarea		
			MR	BSK	MP
Día 1	Sesión 1	ITL	G1	G2	
Día 2	Sesión 2	TDD	G2	G1	
Día 3	Sesión 3	TDD			G1 y G2

- Día 1: ITL training e implementación de la tarea de control usando ITL. Grupo 1: MR, Grupo 2: BSK.
- Día 2: TDD training e implementación de la tarea experimental usando TDD. Grupo 1: BSK, Grupo 2: MR.
- Día 3: ITL y TDD training e implementación de la tarea compleja usando TDD. Grupo 1: MP, Grupo 2: MP.

4.3.6. ESPEL

En el experimento en la Escuela Politécnica del Ejército - sede Latacunga (ESPEL) se replicó de nuevo el experimento de UPM utilizando de nuevo el diseño con dos factores y tres niveles inter-sujetos. Sin embargo en esta ocasión el diseño es ABB. El factor secundario: tarea experimental fue contrabalanceado en las cuatro sesiones experimentales para evitar confundir los factores tarea experimental y aproximación de desarrollo.

En la tabla 4.22 presentamos el diseño del experimento.

Tabla 4.22: Diseño del experimento ESPEL

Día	Sesión	Tratamiento	Tarea			
			MR	BSK	MP	SK
Día 2	Sesión 1	TL	G1	G2	G3	G4
Día 3	Sesión 2	TDD	G2	G1	G4	G3
Día 4	Sesión 3	TDD	G3	G4	G1	G2

El experimento se ejecutó en 4 días, participaron 14 estudiantes en total asignados a los cuatro grupos y la distribución ha sido la siguiente:

- Sesión 1: Unit testing training y Slicing training
- Sesión 1: Slicing training e implementación de la tarea de control usando ITL (MR,BSK, MP, SK)
- Sesión 2: TDD training e implementación de la tarea usando TDD (MR,BSK, MP, SK)
- Sesión 3: TDD training e implementación de la tarea usando TDD (MR,BSK, MP, SK)

4.3.7. Universidad de Basilicata

El experimento en la Universidad de Basilicata [59] es una replicación del experimento de Playtech. Se trata de un estudio **crossover, de un factor con dos niveles**. En el diseño cruzado (crossover) los sujetos reciben la secuencia de tratamientos. La aproximación del desarrollo tiene los tratamientos Test Last y TDD. En la tabla 4.23 presentamos el diseño del experimento.

Tabla 4.23: Diseño del experimento U. de Basilicata

Día	Sesión	Tarea	Tratamiento	
			TDD	TL
Día 1	Sesión 1	MR	Grupo A	Grupo B
Día 2	Sesión 2	BSK	Grupo B	Grupo A

El experimento se realiza en dos días, participan 21 individuos quienes reciben entrenamiento en TDD y pruebas unitarias, con una duración de 20 horas. La distribución del experimento ha sido la siguiente:

- Día 1: Los sujetos del grupo A implementan la tarea experimental MR con TDD y los sujetos del grupo B implementan MR con ITL.
- Día 2: Los sujetos del grupo A implementan la tarea experimental BSK con ITL y los sujetos del grupo B implementan BSK con TDD.

4.3.8. Ericsson

En el experimento de Ericsson se acomete el estudio de un segundo factor relacionado con TDD: el nivel de granularidad de la especificación de las tareas experimentales. Dados los resultados de los experimentos anteriores, empezamos a sospechar que se obtenían resultados especialmente favorables a TDD porque su especificación (que habíamos heredado de experimentos anteriores a esta familia) tenía una granularidad especialmente fina que podría favorecer la aplicación de TDD.

El experimento Ericsson es un diseño de dos factores con **dos niveles (2x2) inter-sujetos** (Tabla 4.24). Los factores son la aproximación de desarrollo y el nivel de definición de la tarea. Las aproximaciones de desarrollo son ITL y TDD. El nivel de definición de la tarea es granularidad fina y granularidad gruesa. La granularidad fina es una descomposición de las tareas experimentales en pequeños incrementos. El factor granularidad de la especificación permitirá determinar la influencia del nivel de la definición de la tarea en el resultado del experimento.

En el diseño de Ericsson se tienen dos tareas experimentales: MR y BSK. La tarea experimental es una variable de bloque dentro de nuestro diseño, para que no sea una fuente de variación de los resultados.

Tabla 4.24: Diseño del experimento Ericsson

Día	Sesión	Tratamiento	Nivel de especificación de la tarea			
			Granularidad fina		Granularidad gruesa	
			MR	BSK	MR	BSK
Día 1	Sesión 1	ITL	G1, G2	G3, G4	G5, G6	G7, G8
Día 2	Sesión 2	TDD	G4, G8	G2, G6	G3, G7	G1, G5

El experimento se desarrolló en tres días, participaron 20 individuos y la configuración fue la siguiente:

- Día 1: Training unit testing
- Día 2: Training ITL/Granulación fina de la especificación. Implementación de la tarea experimental con ITL. Grupo 1: MR, Grupo 2: BSK.
- Día 3: Training TDD. Implementación de la tarea experimental con TDD. Grupo 1: BSK, Grupo 2: MR.

4.3.9. University of Southern Denmark

El propósito del experimento es analizar los efectos de la granularidad de la especificación de los requerimientos. El experimento de University of Southern Denmark (USD) utiliza un diseño de **dos factores dos niveles entre-sujetos**. Los factores son: Aproximación del desarrollo y granularidad de la especificación. La aproximación del desarrollo es Test Last y TDD. La granularidad de los requerimientos es: fina y gruesa. Granularidad gruesa (CG del inglés Coarse-grained) y granularidad fina (FG del inglés Fine-grained).

En este experimento se tienen al nivel titulación como variable de bloque. El nivel de titulación puede ser: undergraduate students, graduate students. La tarea experimental es BSK. Presentamos el diseño en la tabla 4.25.

Tabla 4.25: Diseño del experimento USD

Período	Sesión	Curso	BSK			
			TDD		TL	
			Gran. fina	Gran. gruesa	Gran. fina	Gran. gruesa
Período 1	Sesión 1	Undergraduate	G1	G2	G3	G4
Período 2	Sesión 2	Graduate	G5	G6	G7	G8

En el experimento se realizó en dos períodos académicos, participaron 71 individuos:

- Período 1: Undergraduate students, implementan la tarea experimental BSK. Grupo 1: usa Test Last con CG, Grupo 2: Test last con FG, Grupo 3: TDD con CG, Grupo 4: TDD con FG.
- Período 2: Graduate students, implementan la tarea experimental BSK: Grupo 5: usa Test Last con CG, Grupo 6: Test last con FG, Grupo 7: TDD con CG, Grupo 8: TDD con FG.

4.3.10. Mapfre

El experimento de Mapfre es similar al experimento de Ericsson. Se aplica un diseño de **dos factores con dos niveles inter-sujetos**. Los factores son, como en Ericsson, aproximación de desarrollo y nivel de definición de la tarea. Mapfre incorpora una novedad, es el primer experimento que estudia el tratamiento: A tú modo (YW del inglés Your way). Hasta Mapfre no habíamos logrado comparar TDD con el modo de desarrollo habitual de los participantes (que al fin y al cabo es la comparación relevante para las empresas). De nuevo, la razón inicial de no haberlo hecho era la interferencia que podría tener en el curso comenzar con 2 horas en los que los participantes simplemente desarrollan como saben. Habíamos hecho un intento en Ericsson pero se malogró cuando descubrimos al comienzo del curso la falta de pericia de la mayoría de los participantes en el IDE y el framework de pruebas. Así pues, el diseño inicial se abortó y lo convertimos en tiempo real en dos niveles (ITL y TDD) es la aproximación de desarrollo propia de la empresa Mapfre. Las aproximaciones de desarrollo son: A tú modo y TDD. En la tabla 4.26 presentamos el diseño del experimento:

Tabla 4.26: Diseño del experimento Mapfre

Día	Sesión	Tratamiento	Nivel de especificación de la tarea			
			Granularidad fina		Granularidad gruesa	
			DL	DC	DL	DC
Día 1	Sesión 1	YW	G1	G3	G2	G4
Día 2	Sesión 2	TDD	G4	G2	G3	G1

En el experimento participaron 14 individuos, se ejecutó en 3 días distribuidos de la siguiente forma:

- Día 1: Training en conceptos de herramientas de testing. Desarrollo de la tarea aplicando la aproximación A tu modo.
- Día 2: Entrenamiento en especificación de tareas de grano fino.
- Día 3: TDD Training. Implementación de la tarea experimental con TDD

También es novedoso en Mapfre que las tareas experimentales son propias de la empresa. Esta era también una antigua pretensión de nuestros experimentos en empresas. Sin embargo, hasta Mapfre, no habíamos logrado que una empresa se comprometiera en los experimentos lo suficiente como para invertir tiempo en elegir y especificar unas tareas de juguete que rememoren las tareas habituales de sus desarrolladores. Al igual que el tratamiento. Al tener un experimento con tareas cercanas a la empresa, elevan drásticamente la relevancia de los resultados para la compañía. Las tareas experimentales para este experimento son: Data loading (DL) y Data checking (DC). Para cada una de las tareas se tiene las especificaciones de granularidad fina y gruesa.

En el experimento de MAPFRE tenemos la siguiente ecuación para calidad 4.2.

$$TotalQuality_{MAPFRE} = \frac{Percentage_of_successful_test_cases}{Total_number_of_test_cases} \quad (4.2)$$

4.3.11. PAF

En el experimento PAF pretendimos estudiar conjuntamente los tres niveles trabajados en la familia para el factor técnica de desarrollo: YW, ITL y TDD. Por lo tanto se aplicó un diseño de **un factor tres niveles inter-sujetos**. El factor es la aproximación de desarrollo. Los niveles son YW, ITL, TDD.

La tarea experimental es considerada como variable de bloque para que no sea fuente de variación de los resultados. El tamaño de la muestra no nos permitió estudiarla como factor. Las tareas experimentales son: MR, BSK y Spreadsheet. En la tabla 4.27 presentamos el diseño del experimento:

Tabla 4.27: Diseño del experimento PAF

Día	Sesión	Tratamiento	Tarea		
			BSK	MR	SS
Día 1	Sesión 1	YW	G1	G3	G2
Día 2	Sesión 2	ITL	G3	G2	G1
Día 3	Sesión 3	TDD	G2	G1	G3

En el experimento participaron 13 individuos. El experimento se realizó en tres días y la distribución es la siguiente:

- Día 1: Conceptos de herramientas de testing. Los sujetos implementan la tarea experimental usando la aproximación de desarrollo: YW. Grupo 1: BSK, Grupo 2: SS, Grupo 3: MR.
- Día 2: Slicing training. Los sujetos implementan la tarea experimental usando la aproximación de desarrollo ITL. Grupo 1: SS, Grupo 2: MR, Grupo 3: BSK.
- Día 3: TDD training. Los sujetos implementan la tarea experimental usando la aproximación de desarrollo TDD. Grupo 1: MR, Grupo 2: BSK, Grupo 3: SS.

En este experimento se perdieron datos ya que no fue bien entendida la importancia de que los sujetos deben estar presentes en todas las tareas. Se planeó contar con 14 sujetos, 13 asistieron. 4 de ellos asistieron a una sola sesión. Del grupo 3 únicamente asistió 1 sujeto. Con pocos participantes es difícil detectar un posible efecto aunque este exista.

4.4. Diferencias de diseños

En la tabla 4.28 se presentan los diseños en la familia ESEIL de experimentos en TDD. Los primeros experimentos realizados fueron: FSecure, Elektrobit y UPV. Los experimentos tienen el propósito de estudiar el efecto del tratamiento TDD en la calidad del producto. Los tres usan el diseño de un factor con dos niveles inter-sujetos. En el experimento Seminario UPM se modifica el diseño a un factor dos niveles within-subjects ABAB, con el propósito de desacoplar la tarea experimental para apreciar mejor el efecto de TDD y ampliar el rango de tareas estudiadas.

En los experimentos Playtech y ESPEL adicionalmente a los efectos del tratamiento TDD, se analiza el efecto de la tarea experimental en la calidad. Aunque se mantienen las especificaciones originales de MR y BSK (heredadas de experimentos anteriores en TDD).

En el experimento Ericsson y USD se analiza los efectos de TDD y también los efectos de un segundo factor, nivel de definición de la tarea: granulación fina o granulación gruesa, que se cree tiene efectos en los resultados del experimento. En el experimento de Mapfre se hace un experimento similar a Ericsson, se tiene como tratamiento la granularidad de la especificación. La tarea experimental es una variable de bloque para que no sea una fuente de variabilidad en los resultados del experimento. En el experimento UdB se aplica el diseño de un factor con dos niveles, diseño cruzado.

Para finalizar, tenemos el experimento de PAF se trabaja con un nuevo tratamiento del factor aproximación del desarrollo. Inicialmente teníamos ITL y TDD. ITL es el control. En este experimento se trabaja con el control YW.

Tabla 4.28: Diseños usados en la familia de experimentos en TDD

ID	Lugar	Diseño	Factor 1	Factor 2	Blocking variable	Baseline
FS	F-Secure	Within-subjects	Aproximación de desarrollo: ITL y TDD	NA	NA	Erdogmus et. al[50]
EB	Elektrobit	Within-subjects	Aproximación de desarrollo: ITL y TDD	NA	NA	FS
UPV	UPV	Within-subjects	Aproximación de desarrollo: ITL y TDD	NA	NA	FS y EB
UPM	UPM Seminario	ABAB within-subjects	Aproximación de desarrollo: ITL y TDD	NA	Tarea experimental: MR, BSK, MP, Sudoku	
PT	Playtech	Within-subjects	Aproximación de desarrollo: ITL y TDD	Tarea experimental: MR, BSK, MP	NA	
ESP	ESPEL	Within-subjects ABBB	Aproximación de desarrollo: ITL y TDD	Tarea experimental: MR, BSK, MP, Sudoku	NA	PT
UdB	U. di Basilicata	Balanced crossover design.	Aproximación de desarrollo: Test last, TDD.	NA	Se deduce que es la tarea experimental: MR, BSK	PT
ERC	Ericsson	Within-subjects con una blocking variable (tarea experimental)	Aproximación de desarrollo: ITL, TDD.	Nivel de definición de la tarea: Slicing, No slicing.	Tarea experimental: MR, BSK.	
USD	University of Southern Denmark	Between-subjects con una blocking variable (nivel académico)	Aproximación de desarrollo: Test last, TDD.	Granularidad de los requerimientos: Coarse-grained (CG) y Fine-grained (FG).	Nivel académico: graduate, undergraduate.	ERC
MFR	Mapfre	Within-subjects con una blocking variable (tarea experimental)	Aproximación de desarrollo: YW, TDD.	Nivel de definición de la tarea: Slicing, No slicing.	Se deduce que es la tarea experimental: Data loading (DL) y Data checking (DC)	
PAF	PAF	Within-subjects con una blocking variable (tarea experimental)	Aproximación de desarrollo: YW, ITL, TDD.	NA	Tarea experimental: MR, BSK, SS	

Capítulo 5

Medición de los experimentos de la familia estudiada: Operacionalización

En el presente capítulo se describe cómo se ha realizado el proceso de medición de código fuente de los sujetos experimentales. El proceso cubre desde la elaboración de la variable respuesta hasta la obtención de los valores de calidad por cada sujeto. El procedimiento de medición utilizado en nuestros experimentos en TDD se presenta en la figura 5.1.

El procedimiento contempla los siguientes pasos:

1. Selección de la métrica.
 - Elaboración de la variable respuesta.
 - Definición de métrica
2. Preparación del instrumento de medición.
 - Definición del instrumento de medición.
 - Implementación.
 - Verificación.
3. Medición.
 - Recepción del resultado de la tarea experimental.
 - Aplicación del instrumento de medición sobre el producto.
 - Obtención de valores de la variable respuesta.

Este procedimiento fue obtenido retrospectivamente estudiando las primeras mediciones de la familia de experimentos. En la tabla 5.1 se presenta el procedimiento para la medición del experimento FSecure. El procedimiento de medición del experimento Playtech fue presentado en el capítulo 2, en la tabla 2.15. En el anexo C, sección C.1 presentamos el protocolo seguido para la mediciones de los experimentos restantes: Elektrobit, Seminario UPM, UPV y ESPEL.

Este mismo procedimiento de medición fue aplicado a la operacionalización de algunos experimentos de la literatura con el fin de comprobar el nivel de estandarización y reporte

del proceso de medición. En las tablas 5.2 y 5.3 presentamos el procedimiento de medición del experimento reportado en [65]. Anteriormente, en la tabla 2.16 del capítulo 2 presentamos el procedimiento de medición del experimento reportado en [50]. En el Anexo C, sección C.2 se tienen los procedimientos de medición de los experimentos reportados en [25], [69], [48] y [42].

Figura 5.1: Procedimiento de medición de los experimentos en TDD

5.1. Selección de la métrica

5.1.1. Elaboración de la variable respuesta

Una entidad software se puede clasificar en proceso, producto o recurso, en el experimento de TDD se mide un aspecto del producto software resultante de la tarea experimental.

Tabla 5.1: Protocolo de medición del experimento TDD en Java (FSecure) - Operacionalización del constructo de la variable respuesta calidad

Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos MR	Implementación de instrumentos MP	Implementación de instrumentos BSK	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
Calidad externa del código fuente.	Ecuación 4.1	Definición del conjunto de casos de prueba para evaluar la calidad del código fuente con respecto a la especificación de cada tarea.	Suite de test de casos: Mars Rover 1.3,	Suite de test de casos Music Phone tests 1.2	Suite de test de casos Bowling Score Keeper	Se verifican las salidas esperadas de los casos de prueba, en base a una implementación de referencia. Luego se corrigen los casos de prueba con error.	El producto corresponde al código fuente de los sujetos y se encuentran en un repositorio de productos.	1. Copiar el código fuente de los sujetos en el repositorio del IDE. 2. Anotar el orden, que establece Junit, para la ejecución de los casos de prueba en el ambiente de pruebas. En base al orden de la ejecución de los test cases, crear el formato para registrar las mediciones. (Este paso es para la medición sin plugin)	1a. En el caso de Sudoku el plugin ofrece directamente la medida de calidad. 1b. Registrar individualmente el estado de cada aserción, con el valor respectivo PASS, FAIL, ERR por cada User Story
Subcaracterística del modelo de calidad ISO9126-1.			Formato en Excel para registrar las mediciones. (archivo metrics-Extraction-(new-metrics)-.xls)	Formato en Excel para registrar las mediciones. (archivo metrics-Extraction-(new-metrics)-.xls)	Plugining junit.jar			3. Abrir el proyecto a medir. 4. Cargar la suite de casos de prueba. 5. Cargar las librerías necesarias para la ejecución del suite de casos de prueba.	2. Se obtiene el total PASS, FAIL, ERR por cada user story.

Tabla 5.1 – continuación

Elaboración de la variable res-puesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos MR	Implementación de instrumentos MP	Implementación de instrumentos BSK	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
			Java runtime environment 1.7	Java runtime environment 1.7	Java runtime environment 1.7			6. Modificar el código fuente en los casos que sea necesario: a) Nombres de métodos que tienen que estar con igual nombre que en el instrumento de medida. b) Formato de salidas (return) de funciones importantes que deben estar con el mismo número de parámetros que en el instrumento de medida. c) Modificación o creación de métodos constructores para que puedan conectar con los test cases 7. Ejecutar el proyecto. Si contiene errores regresar al numeral 5 para verificar el cumplimiento de dichos pasos. 8. Guardar la salida de la ejecución en un fichero. En el caso de BSK la medición se realiza con el plugin de forma que se obtiene directamente la medida de Calidad.	3. Se resume las divisiones de aserciones PASS de los user story tackled para el total de aserciones. Se divide para el número de user story tackled y se obtiene el porcentaje
			Librería: hamcrest-all-1.3 IDE: ECLIPSE	Librería: commons-lang3-3.1 IDE: ECLIPSE	IDE: ECLIPSE			4. Las mediciones obtenidas son almacenadas en el fichero metricsExtraction. Los casos de prueba y los ficheros de la medición se encuentran en un Portal Web (Google Sites).	

Tabla 5.2: Operacionalización del constructo de la variable respuesta calidad del experimento de George, con estudiantes [65]

	Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
DESCRIPCIÓN:	Calidad de código extema	Número de casos de prueba funcionales pasados de prueba de caja negra).	Los casos de prueba funcional usados para la evaluación fueron desarrollados por el investigador. No especifica la técnica usada para el diseño de los casos de prueba.	7 casos de prueba de caja negra para la evaluación del código de los desarrolladores estudiantes.	En el paper no se indica si se realizó alguna verificación del instrumento de evaluación.	Código fuente de los sujetos (En Java), estudiantes, correspondiente a 30 pares TDD y 37 pares no TDD.	Se aplica el instrumento de medición al código fuente del sujeto. De un total de 30 TDD pairs, 8 fueron eliminados de la evaluación debido a las siguientes razones: ? 3 grupos fueron eliminados ya que tenían un solo miembro de equipo (no consistente con la práctica de pair programming). ? El código fuente de 5 grupos no compilaba o no ejecutaba.	Se toma la medida entre 0 y 7, correspondiente a los casos de prueba pasados. El investigador registra la medición en una hoja con los siguientes campos: TDD/Non TDD, ID sujeto, Número de casos de prueba pasados, Listado de casos de prueba específicos pasados. (apéndice D de la tesis de Master)
QUIEN:			Investigador				Investigador	Investigador
COMO:		No sé indica como se definió la métrica.			No se indica si se verificó el instrumento.			
CUANDO:								

Tabla 5.3: Operacionalización del constructo de la variable respuesta calidad del experimento George, con profesionales [65]

	Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
DESCRIPCIÓN:	Calidad de código externa	Número de casos de prueba funcionales pasados (casos de prueba de caja negra).	Los casos de prueba funcional usados para la evaluación fueron desarrollados por el investigador. El paper no especifica la técnica usada para el diseño de los casos de prueba. Los casos de prueba se tienen en la tesis, prueban funcionalidad y entradas inválidas.	20 casos de prueba de caja negra. Mirando los casos de prueba que se anexan en la tesis de Master se deducen que corresponden con 20 aserciones Java, JUnit	El paper no hace referencia a la realización de la verificación del instrumentos de evaluación.	Se cuenta con el código fuente de 24 sujetos, en Java, profesionales que participan en el experimento. 6 pares TDD y 6 pares no TDD.	El investigador ejecuta los casos de prueba en el código fuente del sujeto. El paper no indica que se hayan realizado modificaciones en el código de los sujetos.	Se toma la medida, en el rango entre 0 y 20, contando el número de casos de prueba pasados. El investigador registra las mediciones en una tabla con los siguientes campos: TDD/Non TDD, ID sujeto, Número de casos de prueba pasados, Listado de casos de prueba específicos pasados. (apéndice D de la tesis de Master).
QUIEN:			Investigador.				Investigador	Investigador
COMO:			No especifica una técnica usada.		No especifica si hubo verificación.			
CUANDO:								

Fenton [53] recomienda que la obtención de medidas sean orientadas por los objetivos, es decir que no se tomen las medidas más fáciles de obtener sino las que resulten de un análisis de las necesidades de mejora o evaluación que se quiera hacer en el proyecto. Para ello aplicamos el enfoque Goal Question Metric (GQM) de Basili [7]. El enfoque GQM permite definir los objetivos del experimento, los cuales vendrían orientados por la hipótesis de la investigación. En nuestro caso pretendemos medir la calidad del software, la cual es una medida indirecta y por tanto se necesita asociarla a una medida directa. En nuestros experimentos la medida directa sobre la que se basa la medida de calidad son los casos de prueba pasados. De acuerdo al estandar ISO/IEC 25010, las subcaracterísticas a medir serían la completitud funcional y la corrección funcional, el primero evaluando que el código del sujeto cubra las funciones especificadas y que los resultados esperados sean correctos y precisos de acuerdo a lo requerido en las especificaciones de las tareas experimentales.

De acuerdo al enfoque GQM, se plantea un conjunto de preguntas que permitan caracterizar el objetivo, a continuación se definen las métricas que darán respuesta a las preguntas planteadas. GQM nos permite llegar del objetivo a las métricas. La elaboración de la variable respuesta es un procesos iterativo, que se repite hasta alcanzar la métrica adecuada que nos dé una medida del aspecto que estamos evaluando.

Basili y Rombach [7] sugieren determinar el objetivo del experimento según se muestra en la Tabla 5.4:

Tabla 5.4: Definición del objetivo de calidad

Propósito	Especificar el propósito de la medición	Evaluar
Issue	Asunto a medir	Calidad
Objeto (proceso)	Objeto o proceso a medir.	Producto resultante de la tarea experimental.
Punto de vista	Punto de vista desde el que es tomada la medida.	Experimentador

A continuación se formulan las preguntas que intentan caracterizar el objeto de medida con respecto al aspecto de calidad determinado y desde el punto de vista del experimentador.

- Objetivo: Valorar la corrección y completitud funcional funcional.
- Pregunta: ¿Cuánta cantidad de efectos correctos o pactados produce el código del sujeto?
- Métrica de la variable respuesta en el experimento TDD: Número de casos de prueba pasados exitosamente o número de aserciones pasadas exitosamente.

5.1.2. Definición de la métrica

La variable respuesta nos ha indicado qué vamos a medir, para lo cual vamos a poner otro ejemplo. Suponemos que se desea medir líneas de código (LOC), pero no se ha definido qué es una LOC y cómo lo vamos a medir. Si falta esta información, el medidor no sabría, por ejemplo, si un comentario cuenta o no, si una línea en blanco cuenta o no, etc. Es decir, nótese que para un constructo tan básico como LOC existen multitud de alternativas que

deben elegirse si se desea tener una definición unívoca de la métrica. Al igual que en el ejemplo de las LOC, se debe definir qué se considera una línea de código, en el caso de los casos de prueba pasados (que es la medida directa para medir calidad) se debe definir qué entendemos por caso de prueba pasado: ¿se considera pasado cuando el cien por ciento de los pasos (que componen el caso) pasan? ¿si falla un paso entonces falla todo el caso de prueba? o alguna alternativa intermedia, concretamente tenemos tres opciones básicas:

- Si falla un paso, entonces falla el caso de prueba completo y por lo tanto se considera no pasado el caso de prueba completo.
- Si pasa un porcentaje de pasos del caso de prueba, el caso de prueba se considera pasado dependiendo del porcentaje de pasos que fallan. Es decir, se define un umbral por encima del cual el caso de prueba se considera no pasado. Por ejemplo, se define el umbral 50 % que significa que para que un caso de prueba se considere pasado necesita que un 51 % de los pasos que componen el caso hayan pasado satisfactoriamente.
- Que la medida básica sobre la que se asientan las mediciones sea el paso y no el caso de prueba.

Cuando se trabaja con la métrica Casos de Prueba Pasados, se consideran los pasos que pasan exitosamente (verificando que no exista diferencia entre la salida real y la salida esperada que ha resultado de la ejecución del paso), se cuentan y se aplica el umbral seleccionado 100 % (Todos los pasos han sido superados satisfactoriamente) o cualquier nivel inferior para obtener el valor de la métrica directa (Casos de prueba pasados).

Para las medidas indirectas, es necesario definir y elaborar los parámetros de la fórmula. Tal medida debe tener una definición unívoca, en base a los estándares si los hay. Por ejemplo, una medida indirecta que usamos en los experimentos de TDD es la calidad del código que viene definida por la fórmula 4.1 (definida en el capítulo 4) y que enuncia que la calidad de un código para calcular tenemos en cuenta únicamente las historias de usuario abordadas por el desarrollador y, para éstos, su calidad es la cantidad de aserciones pasadas; la calidad del código es el promedio de la calidad de las historias de usuario abordadas.

La definición de la métrica, al igual que la definición de la variable respuesta, es un proceso iterativo de refinamiento hasta obtener el resultado deseado. Por ejemplo, en la familia de experimentos ESEIL las métricas han evolucionado. Por ejemplo para calidad parcial, el parámetro historias de usuario abordadas a veces lo hemos medido directamente (preguntando a los participantes: Playtech, MAPFRE). Es importante resaltar las dificultades en una familia de experimentos el proceso iterativo de mejora de las métricas. Se necesita un equilibrio entre mejorar la métrica y mantener la antigua de modo que los resultados de la nueva replicación sea comparable con las anteriores. Una heurística que hemos aplicado en la familia ESEIL ha sido duplicar las métricas según se mejora. Es decir, contar en todos los experimentos tanto con la métrica antigua como con la mejorada. Por ejemplo, tanto Playtech como MAPFRE, las historias de usuario abordadas pueden medirse al estilo FSecure o con la mejora de medirse directamente. La parte oscura de esta heurística y el aumento de la complejidad es la dificultad del mantenimiento de los datos (con varias versiones por cada experimento).

5.2. Preparación del instrumento de medición

Con el propósito de estudiar el impacto de la instrumentación en los resultados experimentales se generaron dos suites de casos de prueba para medir la calidad de código fuente de cada tarea. Para la generación de los casos de prueba tenemos los siguientes pasos: Definición, implementación y verificación del instrumento de medición,

5.2.1. Definición del instrumento de medición

Los casos de prueba son los instrumentos que utilizamos para operacionalizar la variable respuesta en los experimentos de la familia ESEIL. Para realizar la comparativa de la operacionalización con distintos instrumentos se generaron dos suites de casos de prueba: Ad hoc y PCdE.

5.2.1.1. Casos de prueba Ad hoc

El término “Ad hoc” se refiere a una solución específica, elaborada para un problema preciso. Los casos de prueba Ad hoc son los primeros casos que se generaron con el propósito de medir los experimentos de la familia ESEIL. Los casos de prueba correspondientes a la tarea MR fueron generados desde cero por el investigador D. Fucci después de desarrollar la solución MR.

En la tabla 5.5 presentamos como ejemplo los casos de prueba Ad hoc generados para los movimientos básicos de la tarea MR. Los movimientos básicos consisten en ejecutar los comandos: forward (f), backward (b), left (l) y right (r) para mover el Rover.

Cada caso de prueba tiene un identificador (C.P. ID), por ejemplo, el caso de prueba con ID: 2.4, tiene el nombre: *theRoverShouldChangeFacingThenGivenATurnCommandAfterLanding*, el cual prueba que el Rover cambie de orientación cuando se le da un comando de giro luego de aterrizar en su posición inicial. El caso tiene 2 aserciones, cada aserción tiene un ID: 2.4.1 y 2.4.2. Vamos a tomar la segunda aserción. Los valores para las variables de posición inicial del Rover en la aserción 2.4.2 son: 0 en X, 0 en Y y “W” en Facing (orientación del Rover), para ejecutar los movimientos del Rover se ingresa el comando: “rr” (lo cual significa que el Rover va a girar dos veces a la derecha), el grid no contiene obstáculos (variable Obstáculos vacía “”) y los resultados esperados son “0,0,E”, lo cual corresponde a la posición final del rover: 0 en X, 0 en Y y “E” en el Facing.

La suite completa de casos de prueba Ad hoc correspondiente a la tarea MR se presentan en el Anexo E sección E.1.1. En la tabla 4.7 del capítulo 4 presentamos el número de aserciones por cada caso de prueba correspondiente a la suite Ad hoc.

En lo relativo a los casos de prueba Ad hoc para la tarea BSK. Los casos de prueba originales fueron elaborados y usados en el experimento [50]. En el experimento [60] se toman los casos de prueba originales y se prescinde los casos de prueba que se ocupan de validación de entradas del usuario y presentación de salidas. Originalmente la suite tenía 105 pruebas y fue reducido a 56 pruebas [60]. En nuestros experimentos trabajamos con la versión simplificada de 56 aserciones.

Los casos de prueba Ad hoc correspondiente a la tarea BSK se presentan en el Anexo E sección E.1.2.

Tabla 5.5: Casos de prueba Ad hoc para probar movimientos básicos de Mars Rover

US ID	C.P. ID	Nombre del caso de prueba	Assert ID	X	Y	Fa- cing	Comando	Obstáculos	Resultados espe- rados
1	1.1	theRoverStaysAtOriginOnceLandedAnd- ExecutedAnEmptyCommand ()	1.1.1	0	0	"N"	""	"	"(0,0,N)"
2	2.1	theRoverFacesSouthAfterLandingAnd- ExecutingATurnCommandToTimesInA- Row	2.1.1	0	0	"N"	"rr"	"	"(0,0,S)"
2	2.1		2.1.2	0	0	"S"	"ll"	"	"(0,0,N)"
2	2.2	theRoverFacesWestAfterLandingAndExe- cutingALeftCommandThreeTimesInA- Row	2.2.1	0	0	"N"	"lll"	"	"(0,0,E)"
2	2.3	theRoverMovesForwardOnceLanding	2.3.1	0	0	"N"	"f"	"	"(0,1,N)"
2	2.3		2.3.2	0	1	"N"	"f"	"	"(0,2,N)"
2	2.4	theRoverShouldChangeFacingThenGiven- ATurnCommandAfterLanding	2.4.1	0	0	"N"	"l"	"	"(0,0,W)"
2	2.4		2.4.2	0	0	"W"	"rr"	"	"(0,0,E)"
2	2.5	theRoverMovesBackwards- OnceLanding	2.5.1	0	0	"N"	"fb"	"	"(0,0,N)"
2	2.6	theRoverIsFacingTheSameDirectionAfter- ExecutingFourTimesTheSameTurnCom- mand	2.6.1	0	0	"N"	"llll"	"	"(0,0,N)"
2	2.6		2.6.2	0	0	"N"	"rrrr"	"	"(0,0,N)"
2	2.7	theRoverFacesEastAfterLandingAnd- ExecutingARightCommandThreeTimesInA- Row	2.7.1	0	0	"N"	"rrr"	"	"(0,0,W)"

5.2.1.2. Casos de prueba con Particiones de Clases de Equivalencia

Con el propósito de analizar el impacto de los instrumentos en los resultados experimentales se elaboró una suite adicional de casos de prueba. Esta segunda suite de casos de prueba fue elaborada con la técnica PCdE.

La técnica se basa en que las entradas y salidas de un componente se puede dividir en clases, de acuerdo a las especificaciones. Por lo tanto, el resultado de probar un solo valor de partición de equivalencia se considera representativo de la partición completa [22].

Los casos de prueba para ambas tareas (MR y BSK) fueron elaborados por la estudiante del Master en Software y Sistemas de la UPM, E. Jaimes en su proyecto de fin de Master.

Para elaborar los casos de prueba se consideró hacer el testing de las tareas MR y BSK con pruebas de nivel funcional, basado en la especificación de requisitos o técnica de caja negra. En un inicio se tomaron en cuenta las técnicas de PCdE, análisis de valores límite y la técnica combinatoria pair-wise. Luego de realizar un análisis de las técnicas tomando en cuenta las características de las tareas Mars Rover y BSK, se decidió usar la técnica PCdE ya que ofrece un balance razonable entre el cubrimiento ofrecido y la cantidad de casos de prueba.

En la tabla 5.6 se presentan a manera de ejemplo 51 clases de equivalencia correspondientes a la tarea MR. Estas clases fueron identificadas en los movimientos básico del Rover: forward (f), backward (b), left (l), right (r). En la tabla 5.7 tenemos algunos casos de prueba generados con las clases de equivalencia. Por ejemplo, el caso de prueba con US ID (ID de la historia de usuario): 4, C.P. ID (ID del caso de prueba): 13, clases de equivalencia a las que pertenece el valor de la entrada y salida del caso de prueba: 1, 7, 13, 19, 28 y 41, ancho del grid: 1, alto del grid: 1, valor inicial X: 4, valor inicial Y: 0, facing (orientación) inicial: "W", comando para mover el Rover: "r", Obstáculos: "" (grid sin obstáculos), valor esperado del caso de prueba ("4,0,N").

En el Anexo E presentamos la suites completa, la cual fue generada con la técnica PCdE. En la sección E.2.2 están los casos de prueba de la tarea MR y y en la sección E.2.4 los casos de prueba correspondientes a la tarea BSK. En el mismo anexo, en la sección E.2.1 tenemos las clases de equivalencia definidas para la elaboración de los casos de prueba de MR y en la sección E.2.3 lo correspondiente a BSK. En la tabla 4.8 del capítulo 4 presentamos el número de aserciones por cada caso de prueba correspondiente a la suite Ad hoc.

Tabla 5.6: Definición de clases de equivalencia de los movimientos básicos (f, b, l, r) de la tarea MR

Condición externa	ID Clase	Clases válidas	ID Clase	Clases no válidas
1. Ancho del grid	1	Entero positivo > 0	2	ancho = 0
			3	ancho < 0
			4	Decimal
			5	Carácter
			6	Null
2. Alto del grid	7	Entero positivo > 0	8	Alto=0
			9	Alto <0
			10	Decimal
			11	Carácter
			12	Null
3. Posición inicial Rover coordenada X	13	Entero positivo tal que: $X \geq 0$ y $X < \text{Ancho del grid}$	14	Entero < 0
			15	Entero $\geq$ ancho del grid
			16	Decimal positivo y/o negativo
			17	Carácter
			18	Null
4. Posición inicial Rover coordenada Y	19	Entero positivo tal que: $Y \geq 0$ y $Y < \text{Alto del grid}$	20	Entero < 0
			21	Entero $\geq$ alto del grid
			22	Decimal positivo y/o negativo
			23	Carácter
			24	Null
5. Facing inicial del Rover	25	Carácter N	33	Valor numérico
	26	Carácter S	34	Carácter diferente a válido
	27	Carácter E	35	Más de un carácter válido
	28	Carácter W	36	Null
	29	Carácter n		
	30	Carácter s		
	31	Carácter e		
	32	Carácter w		
6. Array de caracteres de comandos	37	Sin movimiento	49	String de carácter diferente a válido
	38	Carácter f	50	Valor numérico
	39	Carácter b	51	Combinación carácter válido y no válido
	40	Carácter l	52	Null
	41	Carácter r		
	42	Carácter F		
	43	Carácter B		
	44	Carácter L		

Tabla 5.6 – continuación

Condición externa	ID Clase	Clases válidas	ID Clase	Clases no válidas
	45	Carácter R		
	46	Combinación aleatoria de caracteres válidos minúscula (fblr)		
	47	Combinación aleatoria de caracteres válidos mayúscula (FBLR)		
	48	Combinación aleatoria de caracteres válidos mayúscula y minúscula (FfBbLlRr)		

5.2.2. Implementación del instrumento de medición

A menudo el instrumento de medición es un ente abstracto que necesita una implementación para ser utilizado. En nuestro caso los casos de prueba necesitan ser implementados. Los casos de prueba son implementados usando una u otra herramienta (JUnit, Google Tests, Boost Test Library) de acuerdo a la tecnología que usan los sujetos para desarrollar la tarea experimental. Como se ha visto en la tabla 4.15 la familia ESEIL cubre un rango amplio de tecnologías (IDE y framework de pruebas) lo cual ha supuesto un importante incremento de la dificultad de los instrumentos de medida.

En el proceso se han usado los siguientes elementos:

- Suite de casos de prueba para medir cada tarea experimental (MR, BSK).
- Java Runtime Environment 1.7 o superior.
- Framework de pruebas, por ejemplo: JUnit, Boost, Google Test.
- Librerías externas: *hamcrest-all-1.3*, *commons-lang3-3.1*

Los framework de pruebas con los que hemos trabajado son similares en cuanto su funcionamiento, es decir los casos de prueba implementados deben recibir unos datos de entrada y los valores de salida obtenidos se comparan con los valores de la salida esperados. Sin embargo, hay diferencias que se han debido tomar en cuenta respecto al lenguaje que estamos utilizando, por ejemplo, el lenguaje Java permite asignar el valor nulo a una variable a diferencia de C++ que no lo permite, en tales casos nos adaptamos a las restricciones que nos impone el lenguaje.

5.2.3. Casos de prueba con PCdE - MR

Tabla 5.7: Casos de prueba con PCdE de los movimientos básicos de la tarea MR

US ID	C.P. ID	C 1	C 2	C 3	C 4	C 5	C 6	Ancho	Alto	X	Y	Facing inicial	Comando	Obstáculos	Resultado esperado
0	24	2	7	13	19	27	37	0	3	0	2	"E"	""	""	Excepción Ancho de grid invalido
0	25	3	7	13	19	25	37	-3	3	2	2	"N"	""	""	Excepción Ancho de grid invalido
0	29	1	8	13	19	25	37	10	0	7	0	"N"	""	""	Excepción Alto de grid invalido
0	30	1	9	13	19	26	37	3	-1	1	1	"S"	""	""	Excepción Alto de grid invalido
1	02	1	7	13	19	25	37	2	2	0	0	"N"	""	""	"(0,0,N)"
1	03	1	7	13	19	26	37	10	10	9	9	"S"	""	""	"(9,9,S)"
1	04	1	7	13	19	27	37	2	8	1	7	"E"	""	""	"(1,7,E)"
1	05	1	7	13	19	28	37	10	1	4	0	"W"	""	""	"(4,0,W)"
1	06	1	7	13	19	29	37	3	5	1	4	"n"	""	""	"(1,4,N)"
1	07	1	7	13	19	30	37	2	5	0	2	"s"	""	""	"(0,2,S)"
1	08	1	7	13	19	31	37	2	8	1	7	"e"	""	""	"(1,7,E)"
1	09	1	7	13	19	32	37	5	5	4	0	"w"	""	""	"(4,0,W)"
1	34	1	7	14	19	25	37	10	6	-9	5	"N"	""	""	Excepción Posicion inicial X fuera del grid
1	35	1	7	15	19	26	37	10	10	10	9	"S"	""	""	Excepción Posicion inicial X fuera del grid
1	39	1	7	13	20	26	37	20	10	5	-5	"S"	""	""	Excepción Posicion inicial Y fuera del grid
1	40	1	7	13	21	25	37	100	100	1	100	"N"	""	""	Excepción Posicion inicial Y fuera del grid
1	45	1	7	13	19	34	37	1	2	0	0	"X"	""	""	Excepcion Facing invalido.
1	46	1	7	13	19	35	37	10	10	9	9	"EW"	""	""	Excepcion Facing invalido.
1	47	1	7	13	19	36	37	6	3	1	1	null	""	""	Excepción Facing inválido.
2	10	1	7	13	19	25	38	2	2	0	0	"N"	"F"	""	"(0,1,N)"
2	14	1	7	13	19	26	42	10	10	1	1	"S"	"F"	""	"(1,0,S)"
3	11	1	7	13	19	26	39	10	10	9	7	"S"	"B"	""	"(9,8,S)"
3	15	1	7	13	19	27	43	6	3	5	2	"E"	"B"	""	"(4,2,E)"
4	13	1	7	13	19	28	41	10	1	4	0	"W"	"T"	""	"(4,0,N)"
4	17	1	7	13	19	25	45	1	1	0	0	"N"	"R"	""	"(0,0,E)"
5	12	1	7	13	19	27	40	2	8	1	7	"E"	"I"	""	"(1,7,N)"
5	16	1	7	13	19	28	44	3	2	2	1	"W"	"L"	""	"(2,1,S)"

5.2.4. Verificación del instrumento de medición

Los casos de pruebas preparados para la medición debe funcionar adecuadamente durante la medición. Para la verificación del correcto funcionamiento de los casos ejecutándose en el ambiente de pruebas usamos implementaciones de referencia de las tareas (MR y BSK). Las implementaciones de referencia son realizaciones de las tareas experimentales desarrolladas por los investigadores en el laboratorio. Las implementaciones de referencia consideran todas las especificaciones de las tareas y están libres de errores ya que contamos con todo el tiempo requerido para desarrollarlas de forma incremental e ir las probando paralelamente. Esta tarea fue desarrollada por el profesor O. Dieste y el autor de esta tesis.

Pasamos los casos de prueba por las implementaciones de referencia en el ambiente de pruebas preparado para la medición. Se verifican que las salidas esperadas de los casos de prueba, sean iguales que las salidas de la implementación de referencia. En este proceso se identificaron a menudo errores en las implementaciones de los casos de prueba. Luego de corregir los errores se obtuvieron los casos de prueba estándares que se aplicaron en las mediciones de todos los experimentos de la familia.

Este proceso se realizó para los casos de prueba estándares generados para el entorno de Java, para los lenguajes que se usaron posteriormente se transcribieron los casos de prueba al respectivo lenguaje sin la ayuda de la implementación de referencia.

5.3. Medición

5.3.1. Recepción del resultado de la operación experimental

Los sujetos experimentales implementan las tareas en el laboratorio durante el experimento. Los investigadores que participan en las sesiones experimentales son los encargados de tomar el código y almacenarlo en un repositorio. El medidor a menudo no participa en las sesiones experimentales obteniendo el código fuente para la medición del repositorio. A modo de ejemplo y como ilustración de la forma de los datos crudos, en el Anexo 2 presentamos el código fuente del sujeto FSK06 del Experimento FSecure correspondiente a la implementación de la tarea MR en Java.

Para realizar la medición es necesario copiar el código fuente en el workspace del ambiente de desarrollo integrado (IDE).

Algunas veces no se tiene el proyecto del sujeto, sino únicamente se cuenta con los ficheros individuales que contienen el código fuente, es decir los ficheros con la extensión .java. En este caso para realizar la medición se necesita crear un proyecto en una carpeta, preferiblemente en donde se encuentran los ficheros del código fuente del sujeto. Esto ha ocurrido unas pocas veces y se debe a qué se pide a los sujetos que suban los datos a un repositorio, sin embargo algunos sujetos no lo hacen correctamente.

A modo de respaldo, se solicita a los sujetos que envíen el código fuente en ficheros comprimidos .zip, .rar los cuales se deben descomprimir usando Winrar o Winzip. Es recomendable eliminar los espacios en blanco del nombre de las carpetas. Por ejemplo, en lugar del nombre “Playtech 2014” usar el nombre “Playtech2014” (sin espacios en blanco). En la tarea MP el compilador de Eclipse no encuentra una librería si el nombre de la carpeta contiene espacios en blanco.

En algunas ocasiones el sujeto no entrega ningún fichero de ningún tipo (su carpeta se encuentra vacía) o entrega el esqueleto de programa tal como lo recibió al inicio de la

sesión y no escribió nada de código. En este caso será necesario tomarlo en cuenta para reportarlo más adelante en la medición.

Esta fase es la más breve de todas. Usualmente al medidor le toma 30 minutos descargar el código de todos los sujetos de un experimento, copiarlo al workspace de su ambiente de desarrollo y realizar la verificación de la copia. Para realizar esta actividad es necesario coordinar con los investigadores que están ejecutando el experimento para que indiquen en qué momento están listos los códigos en el repositorio para proceder a su medición. En el caso de los experimentos en empresa hemos descubierto que la rapidez en la entrega de los resultados experimentales es crítico. Desafortunadamente en los primeros experimentos (F-Secure principalmente) la empresa y los participantes perdieron interés en el experimento y sus resultados tras pocas semanas. Aprendimos que la medición y análisis de los experimentos de empresa no pueden hacerse con los tiempos habituales de la academia.

5.3.2. Aplicación del instrumento de medición

Para medir la calidad del código fuente se cuenta con una suite de casos de prueba. El medidor pasa los casos de prueba por el código fuente de cada uno de los sujetos que han participado en el experimento. En la Figura 5.2 se muestra un esquema con los elementos en el proceso de aplicación de los casos de prueba.

Figura 5.2: Aplicación del instrumento de medición al código fuente

El código fuente que desarrollan los sujetos no siempre coincide con la estructura de código sugerida en las tareas, lo que dificulta conectarlo a la suite de casos de prueba, por

esta razón es necesario realizar modificaciones en el código entregado por los sujetos para lograr la conexión. Por ejemplo, a veces el nombre de los métodos que utiliza el sujeto o el número de parámetros del constructor que utiliza es distinto al sugerido. El medidor tiene que intervenir para realizar pequeñas modificaciones que permitan conectar el código fuente a la suite de casos de prueba si no se quiere pecar de rigurosidad exagerada y obtener unos datos plagados de ceros. Esta tarea, en cierto modo subjetiva, fue una de las motivaciones para realizar esta investigación.

No se trata de agregar código nuevo en el código del sujeto. Las intervenciones del medidor se hacen generalmente con el propósito de conectar los casos de prueba a la lógica del código del sujeto de tal forma que se pueda medir todo el código de los sujetos experimentales, si no habría estos ajustes o intervenciones tendríamos código fuente no medido, generalmente algún error de un tipo ajeno al código fuente del sujeto impide su medición.

En nuestra investigación las primeras mediciones de los experimentos en Java se realizaron manualmente. En la medición manual, el medidor transcribe el resultado de la ejecución de cada caso de prueba a una hoja de cálculo. Esta transcripción de los datos puede dar lugar a errores debido al procedimiento manual. Como solución, se elaboró un plugin que nos ha servido para realizar las mediciones en el lenguaje Java, ambiente de desarrollo Eclipse, correspondientes a las tareas MR y BSK. La medición automática es mediante un programa que ejecuta los casos de prueba sobre el código y calcula automáticamente las métricas de cada sujeto sin la intervención del medidor. Este plugin ha sido desarrollado en el marco del proyecto Fidipro ESEIL.

En la tabla 5.8 mostramos los ambientes de desarrollo utilizados para las mediciones. En dicha tabla presentamos también el lenguaje y el framework de pruebas que se usan en la medición. Además anotamos los experimentos que se han medido en cada ambiente de desarrollo.

Tabla 5.8: Ambientes de pruebas para las mediciones

ID	Lenguaje/IDE/Testing framework	Experimento	Tipo
1	Java 1.8, Eclipse Kepler Release 2, Junit 4	FS, UPM, PT, UPV	Manual
2	Java 1.8, Eclipse Kepler Release 2, Junit 4	FS, PT, ESP, UdB, PAF	Automático
3	C++, Eclipse CDT under Linux, Google Test	EB	Manual
4	C++, Eclipse, Boost	ERS	Manual
5	C#, Visual Studio, VS Testing Tool	USD	Manual

Tenemos cinco ambientes de desarrollo en donde se realizaron las mediciones. Ahora describimos el proceso de aplicación de los casos de prueba al código fuente para cada ambiente de desarrollo:

5.3.2.1. Ambiente de desarrollo 1: Java, Eclipse - Manual

Este es el ambiente de desarrollo utilizado en las primeras mediciones de la investigación. El lenguaje es Java 1.7, entorno de desarrollo Eclipse, framework de pruebas JUnit 4. En la preparación del ambiente de desarrollo integrado (IDE) es necesario incorporar la siguiente librería a Eclipse: *hamcrest-all-1.3*, es un framework para testing, lo utilizamos cuando se mide la tarea MR. Permite definir matchers de clases personalizadas para hacer las comparaciones entre valores obtenidos y valores esperados.

Para iniciar la medición, se abre el proyecto a medir en el ambiente de desarrollo. Luego se copia la suite de casos de prueba dentro del proyecto. Verificar que la versión de Java y JUnit del proyecto de medición corresponda con 1.7 y 4 respectivamente.

A continuación se ejecutan los casos de prueba. En la figura 5.3 presentamos la pantalla en donde se visualiza la ejecución de los casos de prueba. En la parte izquierda se tiene un listado de los casos de prueba. En la parte central de la pantalla está el código fuente del sujeto. En la parte inferior (Consola) se encuentra un reporte obtenido de la ejecución de cada aserción.

Figura 5.3: Visión parcial ejemplo del entorno de desarrollo 1: Java, Eclipse - Manual

El resultado de la ejecución de cada aserción puede tener uno de los siguientes valores:

- PASS: Pasa la aserción.
- ERR: Error en la ejecución de la aserción.
- FAIL: No pasa la aserción.

Con el propósito de transcribir los resultados, se elabora una hoja de cálculo electrónica. En las filas escribimos el nombre del caso de prueba junto con la aserción ya indicado en la sección 5.2.1. En las columnas escribimos el identificador de cada sujetos experimental (Por ejemplo: PTE001, PTE002, etc.). El medidor obtiene el resultado de la ejecución de cada aserción, por cada sujeto. El resultado se escribe en la hoja de cálculo.

En la figura 5.4 presentamos una visión parcial ejemplo de la medición para el experimento UPV en el ambiente de desarrollo 1.

US	CP	CASO DE PRUEBA	ASERCIÓN	UPV0 1	UPV0 2	UPV0 3	UPV0 4	UPV0 5	UPV0 6	UPV0 7	UPV0 8	UPV0 9	UPV0 0	UPV1 1	UPV1 2	UPV1 14	UPV1 15	UPV1 16	UPV1 7	UPV1 8
13	13.1	testBestScoresAllSparesContent	13.1.1	PASS	ERR	PASS	ERR	ERR	ERR	ERR	PASS	PASS	ERR	ERR	ERR	PASS	PASS	ERR	ERR	ERR
13	13.2	testBestScoresAllStrikesContent	13.2.1	PASS	ERR	PASS	ERR	PASS	ERR	ERR	PASS	PASS	ERR	PASS	ERR	PASS	PASS	ERR	ERR	ERR
2	2.1	testFrameScore	2.1.1	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	ERR	PASS
2	2.1		2.1.2	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	ERR	PASS
2	2.2	testAllPinsFrameScore	2.2.1	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	ERR	PASS
4	4.1	testGetFrame	4.1.1	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
4	4.1		4.1.2	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
4	4.1		4.1.3	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
4	4.1		4.1.4	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
4	4.1		4.1.5	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
4	4.1		4.1.6	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
4	4.2	testAddFrame	4.2.1	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
4	4.2		4.2.2	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
4	4.3	testGameScore	4.3.1	PASS	ERR	ERR	PASS	PASS	ERR	PASS	PASS	FAIL	PASS	PASS	PASS	PASS	ERR	PASS	FAIL	PASS
4	4.3		4.3.2	PASS	ERR	ERR	PASS	PASS	ERR	PASS	PASS	FAIL	PASS	PASS	PASS	PASS	ERR	PASS	FAIL	PASS
8	8.1	testMultipleStrikesInTheMiddleZeroBonifForm	8.1.1	PASS	ERR	PASS	PASS	PASS	PASS	PASS	PASS	FAIL	PASS	PASS	PASS	PASS	FAIL	PASS	PASS	FAIL
8	8.2	testStartWithAMultipleStrikesContent	8.2.1	PASS	ERR	PASS	PASS	PASS	PASS	PASS	PASS	FAIL	PASS	PASS	PASS	PASS	FAIL	FAIL	PASS	FAIL
8	8.3	testStartWithAMultipleStrikesForm	8.3.1	PASS	ERR	PASS	PASS	PASS	PASS	PASS	PASS	FAIL	PASS	PASS	PASS	PASS	FAIL	FAIL	PASS	FAIL
8	8.4	testMultipleStrikesInTheMiddleForm	8.4.1	PASS	ERR	PASS	PASS	PASS	PASS	PASS	PASS	FAIL	PASS	PASS	PASS	PASS	FAIL	FAIL	PASS	FAIL

Figura 5.4: Resultado de las aserciones en la hoja de cálculo para UPV

En la tabla de la hoja de cálculo que se muestra en la imagen tenemos el resultado de cada aserción luego de pasarlo por el código de cada sujeto. La primera columna muestra el número de historia de usuario (US) a la que pertenece el caso de prueba, la segunda columna muestra el identificador del caso de prueba (CP), la tercera columna muestra el nombre del caso de prueba, la cuarta columna es el identificador de la aserción y las columnas restantes contienen los resultados de los casos de prueba por cada sujeto (una columna por sujeto). El identificador del sujeto se indica en la primera fila de la columna. Se usan colores para identificar cada resultado: verde para PASS, amarillo para ERR y rojo para FAIL.

Es necesario considerar que el código fuente que desarrollan los sujetos no siempre ha sido desarrollado con la estructura de clases y métodos establecidos en la plantilla. En algunos casos el código fuente tiene defectos, lo cual dificulta conectarlo a los casos de prueba. Tomando en cuenta esta dificultad, este tipo de medición toma aproximadamente entre 30 minutos y unas pocas horas por sujeto. El tiempo dependerá del número de aserciones que tiene la suite de casos de prueba y la calidad del código fuente. El código fuente sin errores se mide más rápido que un código fuente con errores.

5.3.2.2. Ambiente de desarrollo 2: Java-Eclipse Automático

Es un ambiente de desarrollo similar al anterior. El código fuente que se mide está en lenguaje Java, el framework de pruebas es JUnit y el IDE es Eclipse. La principal diferencia está en el plugin adicional que se instala en este ambiente de desarrollo.

Para instalar el plugin es necesario copiar el fichero *junit.jar* en la carpeta *c:\eclipse_Hacked\plugins\org.junit_4.11.0.v201303080030*

Al ejecutar los casos de prueba, el plugin (*jUnit.jar*) cuenta automáticamente el número de aserciones pasadas, falladas y con error por cada historia de usuario y aplica la fórmula para obtener la métrica de calidad. Este procedimiento automatizado permite la extracción de la métrica de forma más rápida.

Para realizar la medición automática es necesario incluir el siguiente fichero (*AllTest.java*) en el proyecto del sujeto que está midiendo. El código del fichero *AllTest.java*

llama al código que contiene los casos de prueba. En la Figura 5.5 tenemos el código para llamar a los casos de prueba de la tarea BSK.

Hemos modificado de las líneas 14 a la 17, en donde escribimos los nombres de los archivos de las clases que contienen el código fuente: *US01.class*, *US02.class*, *US03.class*, *US04.class*, *US05.class*, *US06.class*, *US07.class*, *US08.class*, *US09.class*, *US10.class*, *US11.class*, *US12.class*, *US13.class*.

```
1
2 package oulu.fsecure.tdd.day2;
3
4 import org.junit.Statistical;
5 import org.junit.runner.JUnitCore;
6 import org.junit.runner.Result;
7 import org.junit.runner.notification.Failure;
8
9 class AllTestsPluginOuluBSK {
10
11 public static void main(String[] args) {
12     Statistical es= Statistical.getInstance();
13     es.initSendStatics();
14     Result result = JUnitCore.runClasses(US01.class ,
15     US02.class , US03.class ,US04.class ,US05.class ,US06.class ,
16     US07.class ,US08.class , US09.class ,US10.class ,US11.class ,
17     US12.class ,US13.class );
18
19     for (Failure failure : result.getFailures()) {
20         System.out.println(failure.toString());
21     }
22 }
23 }
```

Figura 5.5: Código fuente del fichero: *AllTests.java*

En el IDE Eclipse, en la ventana Explorador de proyectos, para ejecutar el programa damos un clic con el botón derecho en el proyecto y seleccionamos: Run as / Java application. Luego seleccionamos la aplicación que deseamos ejecutar: *AllTestPlugin* y damos un clic en OK.

En la figura 5.6 presentamos una imagen del ambiente de desarrollo 2 con una visión parcial ejemplo de la medición automática .

En la parte izquierda de la pantalla se encuentra el Explorador del proyecto con los ficheros del código fuente y casos de prueba. Allí mismo se encuentra el fichero con el resultado de las métricas, al final del listado con el nombre: *TddStatics.txt*

Figura 5.6: Visión parcial ejemplo del entorno de desarrollo 2: Java-Eclipse Automático

En el ejemplo de la figura anterior, se ha realizado la medición de la tarea MR del sujeto FSK06 del experimento FSecure. Los resultados de la medición se encuentran en la ventana central.

Este entorno podríamos considerarlo como el más rápido, ya que una medición podría tomar unos pocos minutos en el mejor de los casos, hasta unos 30 minutos.

5.3.2.3. Ambiente de desarrollo 3: C++, Eclipse, Google Test

Este ambiente de desarrollo se ejecuta sobre Oracle VM Virtual Box versión 5.0. Se ha creado una máquina virtual con el sistema operativo Ubuntu 14.04 LTS. Para ingresar al sistema usamos el usuario: fidipro y contraseña: fidipro. El workspace se encuentra en la siguiente carpeta: */home/fidipro/MEASUREMENT_workspace*

El ambiente de desarrollo usado es Eclipse IDE for C/C++ Developer, versión: Kepler Service Release 1. El framework de pruebas es Google Test.

Para configurar el IDE: abrir el proyecto, dar un clic con el botón derecho en el proyecto, seleccionar Properties. Seleccionar C/C++ Build. Luego dar un clic en Manage Configurations. Seleccionar Unit Test, a continuación dar un clic en Set Active. Por último, seleccionar Apply.

Para compilar y generar el fichero ejecutable seleccionar Build All en el menú principal. En el Explorador del proyecto dentro de la carpeta UnitTest deberá aparecer el fichero ejecutable. Dar un clic con el botón derecho en el fichero ejecutable y seleccionar Run as / Local C C++ Application.

En la figura 5.7 presentamos pantalla con una visión parcial ejemplo de la medición de la tarea MR del sujeto EBKAJ13 perteneciente al experimento Elektrobot.

Figura 5.7: Visión parcial ejemplo del entorno de desarrollo 3: C++, Google Test

En la parte izquierda de la figura se encuentra el Explorador del proyecto. Dentro de la carpeta UnitTest se encuentra con un icono azul el fichero MarsRoverAPI - [x86/le]. En la parte central se encuentra la consola. En la consola se visualiza el resultado de la ejecución de los casos de prueba. En el ejemplo, se han ejecutado 51 pruebas de aceptación de la suite de casos de prueba de la tarea Mars Rover. En el reporte de la consola, se presenta cada aserción ejecutada y su resultado: OK y FAIL.

El entorno C++ es especialmente antipático para la medición puesto que al ejecutar los casos de prueba se tiene un error que interrumpe la ejecución normal de los casos de prueba subsiguientes. En este caso se debe deshabilitar el caso de prueba con error y se vuelve a compilar y ejecutar todos los casos.

La medición es de tipo manual, es decir se deben transcribir los resultados de las aserciones a la hoja de cálculo. La medición toma desde 30 o 40 minutos hasta varias horas. El peor caso sucede cuando se tiene que probar caso por caso, debido a errores en tiempo de ejecución, en esta situación una medición puede tomar dos horas aproximadamente por un sujeto.

5.3.2.4. Ambiente de desarrollo 4: C++, Eclipse, Boost

Este ambiente de desarrollo es similar al anterior, se ejecuta sobre una máquina virtual Oracle VM Virtual Box versión 5.0. Se ha creado una instalación con el sistema operativo Ubuntu 14.04 LTS. Para ingresar al sistema usamos el usuario: fidipro. El workspace se encuentra en la siguiente carpeta: `/home/fidipro/MEASUREMENT_workspace`

El ambiente de desarrollo es Eclipse IDE for C/C++ Developer, versión: Kepler Service Release 1. La diferencia con el ambiente de desarrollo anterior (3) está en el framework de pruebas, en este caso trabajamos con Boost Test Library 1.61.0

En este ambiente hemos creado un proyecto de medición con los casos de prueba en C++ Boost.

Para la medición, abrimos el proyecto del sujeto y copiamos los ficheros que contienen código fuente al proyecto de medición. Seleccionamos Project / Build All. En la ventana Project Explorer damos clic con el botón derecho en el proyecto y seleccionamos Run as / Local C++ Application.

En la figura 5.8 presentamos una pantalla con una visión parcial ejemplo de la medición de la tarea MR del sujeto HERHL05 del experimento Ericsson.

Figura 5.8: Visión parcial ejemplo del entorno de desarrollo 4: C++, Boost

En el explorador tenemos el proyecto de medición: *boost_test_mr_1*. El fichero con los casos de prueba tiene el nombre: *boost_test_mr.cpp*. El código fuente del sujeto está en los ficheros *MarsRover.cpp* y *MarsRover.h*

En la parte inferior de la pantalla, se tiene la consola en donde se muestra el resultado de la ejecución de cada aserción. Además se muestra el total de fallos (failures) ocurridos en la ejecución de la suite de casos de prueba.

Luego de pasar los casos de prueba, el resultado de cada aserción se copia a la hoja de cálculo. En la hoja de cálculo se calcula el valor de la calidad.

5.3.2.5. Ambiente de desarrollo 5: C#, Visual Studio (VS), VS Testing Tool

El ambiente de desarrollo 5 fue usado para medir el experimento de la University of Southern Denmark. El lenguaje es C#, el ambiente de desarrollo es Microsoft Visual Studio Community 2015. Los casos de prueba fueron implementados en lenguaje C# por el investigador D. Fucci de la Universidad de Oulu.

En primer lugar se abre el proyecto desarrollado por los sujetos en el ambiente Visual Studio. A continuación, se copian los ficheros de los casos de prueba al proyecto del sujeto experimental. En la figura 5.9 presentamos una captura de pantalla como una visión parcial ejemplo de la medición del código fuente de la tarea BSK del sujeto 75 del experimento en la University of Southern Denmark.

Figura 5.9: Visión parcial ejemplo del entorno de desarrollo 5: Visual Studio

Para visualizar el explorador de casos de prueba se debe seleccionar Test / Windows / Test Explorer en el Visual Studio.

Como se observa en la figura 5.9, en la parte izquierda de la pantalla se encuentra el Test Explorer. Se observa el listado de casos de prueba. Los casos de prueba pasados están con un icono verde, los casos de prueba fallados con color rojo y con color amarillo tenemos los casos de prueba no ejecutados (contienen la sentencia *Ignore*).

Cuando se pasan los casos de prueba, el resultado de cada aserción se transcribe a la hoja de cálculo electrónica. Los valores de calidad son calculados en la hoja de cálculo.

5.3.3. Obtención de los valores de la variable respuesta

Al aplicar el instrumento de medida al producto software se obtienen los datos crudos. Los datos crudos son el listado de los test cases pasados y sus resultados: pasado, fallado y con error (PASS, FAIL y ERR) para cada uno de los participantes. A partir de los datos crudos se obtienen los resultados de la medición que son los valores de las variables respuesta.

Para obtener el valor de la variable respuesta calidad aplicamos una fórmula puesto que es una medida indirecta. El valor de calidad se obtiene en base a un cálculo que considera los casos de prueba pasados. El valor de las medidas se obtiene aplicando las fórmulas del capítulo 4.

En los ambientes de desarrollo 1, 3, 4 y 5 los cálculos con las fórmulas los realizan automáticamente en la hoja de cálculo. En el ambiente de desarrollo 2, las ecuaciones son calculadas automáticamente por la aplicación y se muestran en la consola. Los valores de las medidas obtenidas automáticamente son transcritas a una hoja de cálculo para pasarlos a los investigadores que se encargan de la fase de análisis del experimento.

Cada sujeto para cada tarea tienen un valor. En la figura 5.10 tenemos una imagen del formato de la hoja de cálculo con datos de las mediciones del experimento Playtech,

tareas experimentales MR y BSK. Por cada sujeto y tarea experimental se presentan los siguientes valores: Número de historias abordadas, calidad, tratamiento, sesión.

ID	MARS ROVER				BOWLING SCOREKEEPER			
	#tst	QLTY	_TRATA	SESION	#tst	QLTY	_TRATAM	SESION
PTE001	11,00	100,00	ITL	1	13,00	89,29	TDD	2
PTE002	9,00	71,91	ITL	1	3,00	12,50	TDD	2
PTE003	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A	#N/A
PTE004	6,00	48,31	ITL	1	11,00	76,79	TDD	2
PTE005	0,00	0,00	TDD	2	1,00	1,79	ITL	1
PTE006	6,00	42,70	TDD	2	9,00	83,93	ITL	1
PTE007	0,00	0,00	TDD	2	10,00	62,50	ITL	1
PTE008	3,00	15,73	ITL	1	1,00	1,79	TDD	2
PTE009	0,00	0,00	TDD	2	13,00	96,43	ITL	1
PTE010	0,00	0,00	TDD	2	13,00	87,50	ITL	1
PTE011	2,00	10,11	TDD	2	11,00	91,07	ITL	1
PTE012	11,00	89,89	ITL	1	10,00	53,57	TDD	2
PTE013	11,00	100,00	ITL	1	13,00	98,21	TDD	2
PTE014	11,00	89,89	TDD	2	13,00	96,43	ITL	1
PTE015	11,00	77,53	TDD	2	11,00	91,07	ITL	1
PTE016	9,00	50,56	ITL	1	11,00	87,50	TDD	2
PTE017	3,00	11,24	TDD	2	2,00	5,36	ITL	1
PTE018	#N/A	#N/A	ITL	1	8,00	71,43	TDD	2

Figura 5.10: Visión ejemplo del resultado de una medición

Las variables de los resultados de la medición son los siguientes:

- #tst: Número de historias de usuario abordadas.
- QLTY: Calidad del producto.

Como los diseños de esta familia son siempre medidas repetidas, los sujetos aparecen N (tantos como tratamientos) veces en el resultado de la medición.

La hoja de cálculo contiene columnas que vienen del diseño (TRATAM: y SESION) para que el analista conozca a que tratamiento y sesión corresponden los valores.

El valor que aparece como #N/A significa que el dato no está disponible (Por ejemplo, debido a que el sujeto no entregó el código fuente). 0 significa que el sujeto ha obtenido 0 en la medición correspondiente a esa variable respuesta.

Cuando la medición es manual, la obtención del resultado de la medición de la variable respuesta en la hoja de cálculo y la verificación de los valores podría tomar entre 2 y 3 horas. Ya que se debe verificar que las fórmulas de la hoja de cálculo estén correctas para el número de sujetos del experimento que se mide.

Para cada fichero, en la primera página de la hoja de cálculo se tienen los datos de la medición: Experimento, fecha del experimento, lenguaje, tratamientos, medidor, tipo de casos de prueba, tipo de intervención, tipo de medición (manual o automática) y fecha de medición tal como se muestra en la figura 5.11 como un ejemplo de la hoja informativa de la medición del experimento Playtech.

Empresa/Universidad:	Playtech
Fecha del experimento:	2014
Lenguaje:	Java
Tipo:	TDD vs. ITL
Medidor:	Fernando
Tipo de caso de prueba:	Ad hoc
Tipo de intervención:	Sintáctica
Tipo de medición (manual/automática):	Manual
Fecha de medición:	Entre Agosto/2014 y Septiembre/2014
Notas:	
	0: Significa que el sujeto obtiene 0 en la medición correspondiente.
	NA: El dato no está disponible (Ej. El sujeto no proporcionó el código)

Figura 5.11: Visión ejemplo de una hoja de información de una medición

Los ficheros que contienen la hoja de cálculo resultante de las mediciones de las variables respuesta han sido almacenados en un repositorio al que se accede mediante el siguiente URL:

<https://sites.google.com/site/fidiproeseil/repository>

Este repositorio contiene una hoja de cálculo por cada medición. Los ficheros de las mediciones están clasificados por experimentos. Para acceder al repositorio hay que dar un clic en el enlace “Repository of experimental results” en la página principal. En la figura 5.12 presentamos una imagen a manera de ejemplo de una parte del almacenamiento de las mediciones del experimento Playtech.

Figura 5.12: Visión de la interfase correspondiente al repositorio

Capítulo 6

Medición de los experimentos de la familia estudiada: Tipos de intervención

Para estudiar la influencia del método de medición se debe, obtener mediciones realizadas en distintas condiciones con el propósito de compararlas. El medidor intenta conectar los casos de prueba al código de los sujetos. Sin embargo, existen errores que impiden conectar los casos de prueba. Por este motivo es necesario hacer manipulaciones en el código para conectar los casos de prueba y extraer las métricas. De no hacerse manipulaciones en el código fuente del sujeto tendríamos una cantidad alta de ceros y valores muy bajos en las mediciones debido a que el código bueno del sujeto no estaría siendo medido por la presencia de errores no atribuibles a la implementación del sujeto. Esto influiría directamente en el análisis y en el resultado del experimento, dando resultados de baja fiabilidad.

Existen diversos niveles de manipulación que el medidor puede llegar a hacer según el tipo de intervención. Hemos clasificado en: sin intervención, intervención sintáctica e intervención semántica. A continuación, se describen los detalles de cada tipo de medición.

6.1. Medición sin intervención

En este tipo de medición, el código de los sujetos es medido tal como es entregado, sin intervención alguna por parte del medidor. Aunque estén presentes errores menores en el código fuente, estos no son corregidos. Sin embargo, a veces el sujeto modifica de manera deliberada la configuración del ambiente de desarrollo del experimento, lo cual hace que no se ejecute el proyecto y la medición sea cero. El medidor debería considerar las siguientes operaciones necesarias en algunos casos para asegurar que el ambiente de desarrollo para la medición esté configurado correctamente:

- Verificar que están cargadas las librerías del lenguaje de programa y del framework de pruebas.
- Verificar que las librerías externas estén cargadas.
- Verificar que los valores iniciales de las variables sean los adecuados.
- Verificar que los nombres de las carpetas del proyecto estén escritas correctamente.

- Verificar que las directivas de compilación estén de acuerdo a la versión del lenguaje que estamos utilizando.

Para realizar la medición es necesario **verificar que las librerías** de Java Runtime Environment (JRE) versión 6 ó 7 y JUnit se encuentren cargadas en el proyecto de medición con el propósito de compilar el código fuente del sujeto. De no haberse cargado la librería va a dar error el momento de iniciar la compilación resultando valores iguales a cero en las medidas. Se debe verificar que se haya incluido la versión 4 de JUnit en las librerías del proyecto de forma que se puedan ejecutar los casos de prueba, cuando no se tiene cargada la librería no se ejecutan los casos de prueba. En la figura 6.1 se muestra las librerías JRE y JUnit cargadas en el IDE Eclipse.

Figura 6.1: Verificación del nombre de las librerías JRE y JUnit

El código fuente del sujeto está contenido en un proyecto mientras que el código fuente de los casos de prueba está contenido en otro proyecto distinto. El medidor agrega los ficheros de los casos de prueba al proyecto del sujeto para tener todo junto y ejecutar el proyecto. Los ficheros de los **casos de prueba y del código fuente deben estar contenidos en el mismo paquete**. Si los casos de prueba hacen referencia a otro paquete hay que cambiar la referencia al paquete donde está el código fuente. Por ejemplo, en el experimento FSecure, como se muestra en la figura 6.2, el código fuente de los casos de prueba (archivo US01.java) tiene la siguiente declaración: *package training.tdd.day1*; la cual debe coincidir como en este caso con el nombre del paquete (*training.tdd.day1*) del código del sujeto. De no hacerse de esta manera, se tendrá error y no se podrá medir el código fuente, obteniéndose valores cero en los resultados.

Figura 6.2: Verificación del nombre del paquete importado

La medición de algunas **tareas requieren ciertas librerías externas** para ejecutar correctamente los casos de prueba, concretamente, cuando se mide manualmente la tarea MR es necesario que la librería exterior *hamcrest-all-1.3* esté incluida en el proyecto de medición. Si no está incluida la librería dará error al ejecutar los casos de prueba. Las librerías se descargan y se copian al directorio correspondiente.

A diferencia de la medición manual, cuando se mide automáticamente la tarea MR no debe incluirse la librería. En este caso es necesario verificar que la librería *hamcrest-all-1.3* **no** esté incluida en el proyecto de medición, ya que el plugging contiene la librería. Si se mide automáticamente, con la librería *hamcrest-all-1.3* cargada en el proyecto de medición automática, el resultado de la medición es siempre un valor igual a 0.

En algunas tareas los **casos de prueba asumen cierta configuración inicial**, lo cual implica que los sujetos, cuando asignan valores a las variables, estén de acuerdo con la configuración establecida, sin embargo ciertos sujetos asumen configuraciones distintas. Cuando el sujeto asume una configuración distinta los casos de prueba no pasarán. El medidor verifica esta configuración en el código fuente del sujeto y, si está incorrecta, lo interviene escribiendo la configuración correcta para que los casos de prueba se conecten adecuadamente al código.

Por ejemplo, los casos de prueba Ad hoc consideran que el Rover empiece en la posición (0,0,N), pero cuando se mide con los casos de prueba con PCDE se considera que el Rover empieza en una posición establecida por el caso de prueba, por lo tanto el medidor debe escribir el método *goToStart(int x1, int y1, String facing)* que lleva al Rover a una posición establecida en los parámetros: *x1*, *y1*, *facing*. El método *goToStart* le da una posición inicial al Rover en el grid, a partir de la posición inicial el Rover empezará a ejecutar los movimientos. El código fuente del método *goToStart* lo presentamos en el Anexo F, sección F.1.

Los sujetos tienen una plantilla para implementar cada tarea. En la plantilla se sugieren nombres de métodos. Los casos de prueba asumen que estos métodos están implementados en el código de los sujetos. Sin embargo los sujetos tienen la libertad de crear métodos y asignarles un nombre según su criterio. Los casos de prueba hacen referencia a métodos del código de los sujetos. El medidor interviene en el código del sujeto **creando los métodos con el nombre sugerido**. Estos métodos generalmente manejan datos de la implementación del sujeto. Si no se realiza esta intervención los casos de pruebas fallan debido a que los métodos no se encuentran y son necesarios para la medición.

Por ejemplo, en la medición de la tarea BSK con la suite de casos de prueba PCdE, el medidor debe incluir los siguientes métodos en la clase Bowling: *public int getSize()* y *public boolean findBonus()*, el primero obtiene el número de frames del juego y el segundo método indica si el objeto bonus es nulo. Esta intervención es para conocer si el sujeto está implementado los bonus con una variable adicional, de esta manera el medidor conocerá como medir el bonus del juego: como una variable (bonus) o como un frame (frame número 11). De no hacerse esta intervención es posible que tengamos errores en la medición del bonus, la medición resultaría imprecisa. El código fuente de los métodos mencionados lo mostramos en el Anexo F, sección F.1.

Los elementos utilizados por el sujeto durante el experimento tienen una configuración establecida. Sin embargo esta **configuración inicial de los elementos de desarrollo es alterada** por los sujetos. Esta alteración podría hacerse de manera deliberada. Si el sujeto altera la estructura de archivos o la configuración original del ambiente de desarrollo

IDE, se va a tener errores en la medición sin obtener los valores para la variable respuesta.

Las carpetas y los archivos de un proyecto tienen diferentes propiedades tales como el nombre, la extensión, visible u oculto, etc. Cuando se realiza una medición se asumen una configuración inicial de las propiedades de los ficheros y carpetas que almacenan el proyecto del sujeto. Los nombres de las carpetas no deben tener espacios, los ficheros no deben estar ocultos. Si no se cuenta con esta configuración de carpetas y ficheros la medición dará errores, como resultado tendremos los valores de la medición igual a cero. Por ejemplo, en el experimento FSecure, en las tareas MR y BSK, se observó en la carpeta que almacena el proyecto del sujeto, el fichero `.classpath` tenía la propiedad “Oculto”. Esto impide la normal ejecución de los casos de prueba. Para quitar la propiedad oculto, es necesario eliminar la propiedad no deseada desde el sistema operativo. A continuación, el medidor podría ejecutar el proyecto con JUnit y registrar los valores de la medición obtenidos.

El ambiente de desarrollo del experimento viene configurado de manera única para todos los sujetos que participan en el experimento. La configuración del IDE tiene muchos parámetros respecto a distintos aspectos tales como las directivas de compilación. Mediante una directiva de compilación se le indica al compilador que versión de lenguaje va a utilizar en la compilación. Por ejemplo, en el experimento Ericsson algunos sujetos han configurado el ambiente de desarrollo Eclipse para compilar con C++2011. Esto se realiza escribiendo la directiva de compilación `-std=c++11` en el parámetro *Other dialect flags*. Los sujetos experimentales han usado sentencias C++2011 que son extrañas para el ambiente de medición sugerido (versión C++03). El uso de sentencias exclusivas de otro estándar produce error en la medición, resultando cero el valor de las medidas. Este evento está asociado a experimentos en lenguaje C++, indistintamente si es industria o academia, puntualmente en dos sujetos de un experimento en C++ que usaron sentencias C++2011.

Las operaciones antes mencionadas se han realizado en general en la mayoría de experimentos tanto en los académicos como en los profesionales.

6.2. Medición con Intervención Sintáctica

Para medir con intervención sintáctica, el medidor debe aplicar en primer lugar las operaciones para la medición sin intervención, con el propósito de asegurar inicialmente la ejecución del proyecto del sujeto.

Tras estas intervenciones básicas, para la medición con intervención sintáctica, el medidor realiza intervenciones necesarias para evitar errores menores presentes en el código fuente del sujeto y traducirse en resultados de medición con valores igual a cero o resultados con valores de medición inferiores a la medida real, ya que el código afectado por el error menor no pasa los casos de prueba. Las operaciones para realizar una intervención sintáctica son las siguientes:

- Dar el nombre sugerido a las clases del código fuente del sujeto.
- Asignar nombres sugeridos a los métodos del código del sujeto.
- Elaboración de las variables que retornan los resultados de un método.
- Modificación del formato de la salida de los métodos.
- Creación de los constructores con los parámetros sugeridos.
- Creación de métodos sugeridos faltantes en el código del sujeto.
- Asignar valores iniciales sugeridos a las variables.

- Eliminar la generación de valores aleatorios dirigidos a los casos de prueba.

El sujeto recibe una plantilla con clases y variables básicas, a partir del cual desarrolla su propio código fuente. En esta plantilla las clases principales tienen un nombre sugerido. Sin embargo, el sujeto a veces modifica estos nombres de clases. Cuando los nombres de clases del código fuente de los sujetos son distintos al nombre sugerido, la conexión con los casos de prueba falla, por lo tanto el resultado de la medición va a ser cero. Esto sucede ocasionalmente en los experimentos con estudiantes, profesionales y en los diferentes ambientes de desarrollo. La corrección implica que el medidor entre en el código fuente del sujeto y **renombrar la clase con el nombre sugerido**.

Por ejemplo, el sujeto UPV17, en la tarea BSK del experimento UPV, ha dado el nombre `FrameBowling` a la clase con nombre sugerido `Frame` y el nombre `Game` a la clase con nombre sugerido `BowlingGame`. El medidor interviene nombrando las clases con los nombres sugeridos, de esta manera se favorece la conexión con los casos de prueba.

En la plantilla de clases que recibe el sujeto, también se encuentran los métodos sugeridos para cada una de las clases. El sujeto también tiene la libertad de modificar los nombres de los métodos sugeridos al igual que el nombre de las clases sugeridas. Algunos sujetos modifican el nombre de los métodos, esto se ha observado en todos los experimentos, sin embargo si se cambia el nombre de los métodos se dificulta la conexión con los casos de prueba ya que el código fuente de los casos de prueba usa los nombres sugeridos para llamar a los métodos del código de los sujetos. En consecuencia se afecta el resultado de la medición, dando un valor igual a cero o menor al valor real de la medición. El medidor interviene en el código fuente de los sujetos **asignando el nombre sugerido a los métodos**.

Por ejemplo, en la tarea BSK, los nombres sugeridos de los métodos se denominan `getThrow1()`, `getThrow2()`, `score()`. En el código fuente del sujeto PTE009 del experimento Playtech se observa que el sujeto ha asignado nombres distintos a dos métodos: `int getFirst()`, `int getSecond()` en lugar de `getThrow1()`, `getThrow2()`, respectivamente. El medidor interviene en el código fuente del sujeto asignando el nombre sugerido, de no hacerse esta intervención el código fuente el valor de la medida podría ser cero o inferior al valor real.

Ahora tenemos un ejemplo con la tarea MR, el nombre sugerido del método que devuelve la posición del Rover se denomina `executeCommand(String)`. El sujeto FSOU2806 en la tarea MR del experimento FSecure le ha dado un nombre distinto al método: `void process(String commands)`. El medidor cambia el nombre del método `process` y lo pone como `executeCommand`, de esta forma permitiría conectar el código a los casos de prueba y obtener la medida.

A veces los métodos creados por los sujetos no devuelven un valor o devuelven un valor nulo, esto ocurre con frecuencia en la tarea MR en comparación con otras tareas. Si un método no devuelve el valor calculado, no se tiene un valor que se pueda comparar con el valor esperado en los casos de prueba, por lo tanto el valor de la medición dará cero o un valor inferior al real. Esta corrección implica que el medidor **elabore la variable que retornará el valor resultante de los cálculos realizados por el método**. Realizar esta intervención demanda poco tiempo, posiblemente algunos minutos, sin embargo permitirá medir bloques de código buenos.

Como un ejemplo, en la tarea MR el return del método `public String executeCommand(String command)` no devuelve un valor con el resultado del método. En lugar de devolver un valor `String` devuelve un valor nulo. En este caso, el medidor analiza si el

sujeto ha elaborado los elementos que debería devolver el método (en este caso una cadena String que contiene la posición del Rover), entonces implementa código fuente necesario que para que el método retorne el valor de la posición del Rover. En el Anexo F, sección F.2, presentamos un ejemplo.

Algunas veces los valores que devuelven los métodos escritos por los sujetos, están en un formato distinto al requerido en las especificaciones, entonces no conectan con los casos de prueba. Esto provoca que los casos de prueba no se conecten y el resultado de la medición sea un valor igual a cero. El medidor interviene en el código fuente del sujeto realizando el **cambio del formato de salida del método** a lo establecido en las especificaciones. Este tipo de intervención se ha realizado en particular en la tarea MR. En el Anexo F, sección F.3 presentamos ejemplos.

En la plantilla de clases sugerida que recibe el sujeto, las clases tienen constructores. Sin embargo, el sujeto modifica o crea nuevos constructores distintos a los sugeridos. Para que se realice la medición, los constructores de las clases deben tener los parámetros establecidos en las clases sugeridas. Si cambian los parámetros del constructor no se podría realizar la medición de manera correcta ya que los constructores referidos en los casos de prueba serían distintos a los existentes en el código del sujeto. El resultado de la medición sería igual a cero. El medidor interviene en el código fuente realizando la **creación de los constructores con los parámetros sugeridos**, poniéndolos de acuerdo a los constructores sugeridos, de esta manera el código se conectará adecuadamente a los casos de prueba.

En la tarea MR el constructor sugerido de la clase principal debe tener el formato *MarsRover(int, int, String)*. En la tarea BSK, los constructores de las clases deben ser: *BowlingGame()*, *Frame(int, int)*.

Tenemos el ejemplo del sujeto PTE011 en la tarea MR del experimento Playtech. El constructor del sujeto tiene seis parámetros (*public MarsRover(int x, int y, String obstacles, int startRow, int startColumn, char startDirection)*) en lugar de tres parámetros como se había sugerido. El medidor escribe el constructor con tres parámetros de acuerdo al código estándar (*public MarsRover(int x, int y, String obstacles)*). El medidor entra en el código del sujeto y crea el nuevo constructor con tres parámetros lo cual permite conectar el código a los casos de prueba.

En la plantilla de código fuente que recibe el sujeto están todas las clases sugeridas con los métodos sugeridos. En varios experimentos hemos visto que a veces el sujeto elimina completamente un método. Si un método sugerido no existe entonces no es posible conectar el código fuente a los casos de prueba por lo tanto la medición será igual a cero. El medidor debe entrar en el código del sujeto y **crear el método sugerido faltante** con el propósito de medir el código bueno escrito por el sujeto.

Por ejemplo, en el código fuente del sujeto ERHL07 del experimento Ericsson, de la tarea BSK, el método sugerido *addFrame(Frame f)* no existe en el código del sujeto. En lugar de dicho método, el sujeto ha implementado un método *addFrame(int firstThrow, int secondThrow)*. Sin embargo, en el código de los casos de prueba hacemos referencia al método sugerido *addFrame(Frame f)* para pasar un Frame al juego. Por lo tanto al haber esta diferencia, el código no va a conectar con los casos de prueba. El medidor podría crear el método *addFrame(Frame f)* en el código fuente del sujeto y adaptarlo al código del sujeto. De este modo se permitirá que los casos de prueba pasen por el código del sujeto sin problema y así obtener una mejor medición.

En la plantilla de código fuente que recibe el sujeto también se incluyen variables de las clases con sus valores iniciales. El sujeto es libre de modificar la configuración inicial establecida en lo relativo a las variables. A veces esta intervención del sujeto en los valores iniciales de las variables no permite conectar el código adecuadamente por lo tanto se obtienen medidas que generalmente dan valores de medición igual a cero. El medidor entra en el código y **configura las variables con los valores iniciales sugeridos**.

Por ejemplo, en el código fuente del sujeto FSK06 en la tarea MR del experimento FSecure. La posición inicial del Rover se guarda en la posición *xStart* e *yStart*. De manera incorrecta el sujeto ha asignado los valores del tamaño del grid (*x* e *y*) a la posición inicial del Rover. De esta forma no pasa ningún caso de prueba. El medidor entra al código del sujeto y asigna el valor 0 a las variables *xStart* e *yStart* para mantener la configuración inicial y obtener los valores de la medición.

Usualmente los sujetos escriben sus casos de prueba para probar el funcionamiento de su código fuente. Los casos de prueba reciben datos de entrada fijos definidos por el sujeto. Sin embargo algunos sujetos generan valores aleatorios que se ingresan en los casos de prueba. La generación de los valores aleatorios a veces reemplaza los valores de los casos de prueba utilizados en la medición. Por este motivo se generan errores en el momento de realizar la medición y los resultados de la medición se ven alterados. El medidor debe entrar al código fuente del sujeto y **eliminar la generación de los valores aleatorios** para lograr la ejecución de la suite de casos de prueba durante la medición.

Por ejemplo en el código del sujeto 48 correspondiente a la tarea BSK del experimento de la Universidad del Sur de Dinamarca (SDU), tiene un bloque de código en el constructor de la clase BowlingGame que genera valores aleatorios para los tiros del juego (*random = new Random(); firstThrow = random.Next(11); secondThrow = random.Next(11 - firstThrow);*). El medidor considera que los valores de los tiros se deben generar desde la suite de casos de prueba, por lo tanto elimina las sentencias que generan los valores aleatorios del código fuente de los sujetos.

Los sujetos son libres de adoptar o no la plantilla sugerida para empezar su implementación de código fuente. Si el sujeto tiene una estructura completamente diferente a la plantilla de código fuente sugerida, la interpretación de la estructura de nuevas clases y métodos del código del sujeto se deja para realizarlo después mediante otro tipo de intervención que lo explicaremos más adelante (intervención semántica).

Hasta aquí se han descrito un conjunto de tipos de intervenciones en el código fuente del sujeto, en donde el medidor entra en el código y realiza ciertos cambios que permitirán el paso de los casos de prueba por bloques de código válidos que estaban siendo impedidos de medirse por errores principalmente relativos a modificaciones realizadas por el sujeto a la plantilla sugerida. Adicionalmente a los tipos de intervenciones anotadas anteriormente a continuación tenemos ciertas consideraciones.

Cuando se interviene en el código fuente del sujeto a veces es necesario eliminar o modificar código del sujeto. En lugar de eliminar el código original de los sujetos es preferible **colocar en comentarios el código original**. Esto permitiría tener una bitácora de las intervenciones realizadas por el medidor en el código fuente de los sujetos. Esto debe tomarse en cuenta en todos los experimentos ya que esta actividad de intervención se realiza permanentemente durante la medición.

Cuando el medidor realiza cambios en el código del sujeto debe tener cuidado sobre el código que modifica o adiciona. El código a medir es el código que el sujeto ha escrito

para resolver la tarea asignada. El **medidor no debe implementar código nuevo** para funcionalidades no abordadas por el sujeto. Por ejemplo, en la tarea BSK, si los sujetos no han implementado código para manejar los bonus, el medidor no debería hacerlo por el sujeto ya que estaría alterando el experimento.

Cuando se ejecuta código fuente con defectos suelen ocurrir errores en tiempo de ejecución. Cuando pasamos los casos de prueba por código fuente en C++, un error puede romper la ejecución de todos los casos de prueba. Los errores de tipo *memory access violation* de C++ hacen que se interrumpa la ejecución de toda la suite de casos de prueba. De esa forma tenemos una medición incompleta dando como resultado una medición con valores incorrectos. El medidor debe **comentar los casos de prueba que den error en tiempo de ejecución** para que continúen ejecutándose los casos de prueba subsiguientes y se complete la medición.

Por ejemplo en el código del sujeto ERHL11 del experimento Ericsson en la tarea BSK tenemos este tipo de errores, para evitar que se ejecute un caso de prueba que da error y se rompa la ejecución de toda la suite de casos de prueba, comentamos el código que no queremos que se ejecute con los caracteres: `/* */`. Este procedimiento realizamos para cada caso de prueba que da error en la medición.

6.3. Medición con Intervención Semántica

Según hemos avanzado en la medición de los experimentos se ha visto la necesidad de llegar a un nivel más detallado de análisis del código fuente del sujeto con el propósito de comprender mejor la solución que el desarrollador da a la tarea experimental. En este sentido la necesidad de intervenir ya no es relativa a cambios menores sino en ocasiones se requiere intervenciones mayores con el propósito de ajustar la programación del sujeto a la estructura de clases y métodos sugerida. Se trata de obtener una medición que evalúe cada parte de código implementada por el sujeto pero sin imponer la aproximación a la solución que tienen los investigadores. Los desarrolladores pueden ser muy imaginativos y originales en sus soluciones sorprendiendo a los investigadores. Dar por inválidos estos códigos sólo por no adoptar la solución esperada no parece justo.

A esta intervención la hemos denominado intervención de tipo semántica, pues implica cambios en las abstracciones del código para este tipo de intervención. Se realizan las siguientes actividades con el propósito de identificar problemas en el código que se pueden intervenir y permitan medir el código completo:

- Revisión de las clases del código fuente del sujeto.
- Revisión de cada uno de los casos de prueba que fallan.
- Revisión de errores con la ayuda del depurador del ambiente de desarrollo.

6.3.1. Revisión de las clases del código fuente

Aunque el sujeto recibe una plantilla de clases y métodos para iniciar su implementación, unos pocos sujetos no usan la plantilla, hace su implementación a su modo sin considerar las clases, métodos y variables sugeridas. Esto dificulta la medición ya que los casos de prueba hacen referencia a la estructura de clases sugerida. Al no conectar el código con los casos de prueba tendremos resultados de la medición igual a cero. El propósito de esta revisión es que el medidor comprenda la solución que ha dado el sujeto a la tarea

experimental, comprender el motivo de los casos de prueba fallados y descubrir el origen de los fallos que se presentan durante la medición.

Si las clases del programa del sujeto son distintas a la sugerida es necesario tener en cuenta las clases sugeridas por cada tarea. Si el sujeto cambia las clases es difícil conectarlas a los casos de prueba, por lo tanto el resultado de la medición dará cero aunque el sujeto haya implementado código bueno. El medidor intenta **escribir el código fuente del sujeto dentro de las clases sugeridas** para conectarlo a los casos de prueba.

En la tarea BSK, la plantilla se compone de dos clases: Frame, y BowlingGame. En MR se debería contar con una sola clase: MarsRover, sin embargo algunos sujetos resuelven la tarea con clases distintas.

Por ejemplo, el sujeto ERHL02, experimento Ericsson, tarea BSK. El sujeto no usa la estructura de dos clases: Frame y BowlingGame. En lugar de ello usa un solo fichero en donde escribe algunos métodos. Para intervenir en este código el medidor copia el código del sujeto a las clases sugeridas y ejecuta los casos de prueba.

A veces al pasar el código del sujeto por la suite de casos de prueba, el resultado es que fallan todos los casos de prueba, sin embargo el código fuente pasa de manera exitosa por los casos de prueba creados por el sujeto. El medidor debe **revisar los casos de prueba pasados y que han sido escritos por el sujeto**, de esta manera comprenderá mejor el formato en qué está ingresando los datos a los casos de prueba de la medición. A continuación, el medidor entra en el código y realiza las modificaciones para que el formato de los valores de salida del código sean similares al formato de los datos que ingresan a la suite de casos de prueba. Si no hace esta operación es posible que el resultado de la medición sea igual a cero aunque el sujeto haya escrito código bueno con pequeñas diferencias en el ingreso de los datos.

Por ejemplo, en la suite de casos de prueba de la tarea BSK se requiere que los tiros del Bowling sean cargados mediante dos valores enteros correspondientes al Tiro 1 y Tiro 2 del juego del Bowling, pero el sujeto hace la carga de los tiros mediante un arreglo de enteros. Tenemos el ejemplo concreto correspondiente al sujeto FSK07, en la tarea BSK del experimento FSecure. En primera instancia el medidor ejecutó la suite de casos de prueba y no conectaron con el código fuente del sujeto sin embargo los casos de prueba elaborados por el sujeto si pasaron. A continuación, abrimos el código fuente y observamos que los casos de prueba son cargados en un arreglo de enteros llamado *scores* (*public Frame(int[] scores)*) no está en el formato sugerido. El medidor entra en el código fuente y hace los cambios en los métodos, de manera que se manejen los tiros en formato de dos enteros y puedan conectarse a los casos de prueba (*public Frame(int i, int j)*). El medidor hace las modificaciones en el código de manera que el tipo de datos se ajuste al de los casos de prueba (valores enteros), de esta manera se ha corregido el ingreso de los tiros del Bowling y se podrá conectar a la suite de casos de prueba.

6.3.2. Revisión de errores con la ayuda del depurador

En algunas ocasiones se observa que el sujeto ha desarrollado código fuente, pero al pasar la suite de casos de prueba da error en todos los casos. Se revisa el código fuente con la ayuda del depurador del ambiente de desarrollo. Cuando se hace la depuración, se han encontrado referencias a valores nulos, manejo incorrecto un arreglo. Si bien son pequeños errores del sujeto, el medidor podría corregirlos de manera que en la medición se permita el paso de los casos de prueba por los bloques de código bueno que ha desarrollado el sujeto.

Estos casos son raros dentro de los experimentos medidos sin embargo la intervención permite obtener medidas distintas de cero.

Para que las variables puedan ser usadas deben ser declaradas de manera correcta. Usualmente el sujeto con la ayuda del compilador detecta este tipo de errores durante el desarrollo de la tarea. Sin embargo ocurre que los errores de declaración de variables se detectan el momento de realizar la medición. Estos pequeños errores impiden la medición de los bloques de código buenos dando cero como resultado de la medición. El medidor entra en el código y arregla el error haciendo una **declaración correcta de la variable** que permita realiza la medición del código bueno del sujeto.

Tenemos el ejemplo del sujeto FSK08 en la tarea MR del experimento FSecure. Con la ayuda del depurador se localizó un defecto en la sección de la declaración de variables. La variable *recordedObstacles* no ha sido inicializada (*private ArrayList< String > recordedObstacles;*). El medidor interviene y corrige el error en la declaración e inicialización de la variable con la siguiente sentencia: *private ArrayList< String > recordedObstacles = new ArrayList< String >();*. De esta manera se ejecutaron los casos obteniendo una medida distinta de cero.

A veces los sujetos hacen uso de arreglos de datos, pero en su manejo hacen referencia a una posición nula. Durante la medición, cuando se pasan los casos de prueba este tipo de errores hace que todos los casos de prueba den como resultado error, obteniéndose un valor cero en la medición aunque el sujeto tenga código que parece bueno. El medidor entra en el código y con la ayuda del depurador identifica el error, que usualmente suele ser al inicio en la carga de datos. El medidor **corrige el error de manejo de los arreglos** para que se puedan pasar los casos de prueba por el código bueno del sujeto.

Por ejemplo, en la tarea BSK, en lo relativo al manejo de Frames. Los sujetos definen 10 Frames, el índice de referencia de los frames en el lenguaje Java, va de 0 a 9. El código del sujeto hace referencia a un Frame número 10, lo cual es incorrecto y da como resultado un error. Si a criterio del medidor se puede corregir y es un solo error que se corrige con una pequeña intervención que permitirá medir bloques de código buenos entonces el medidor interviene y corrige este error, esta operación permitirá obtener una medida distinta de cero.

6.3.3. Revisión de los casos de prueba que fallan

Con el propósito de asegurar la ejecución de todos los casos de prueba hacemos una revisión de los resultados de cada caso de prueba, dando énfasis en los casos de prueba fallados. El medidor revisa los valores esperados y obtenidos de los casos de prueba no pasados, tratando de deducir el motivo por el que no pasaron. La intervención se hace cuando se tienen resultados buenos, alternativos a los especificados en los casos de prueba.

Revisamos cada uno de los casos que no han pasado (fail) o han dado error. En la figura 6.3 se muestra a manera de ejemplo una vista parcial de los casos de prueba ejecutados en el IDE Eclipse. En un cuadro color rojo están marcados los casos de prueba que han dado Fail. En cuadro color verde están marcados los casos que han dado Error.

Los casos de prueba ejecutados que dan el resultado Fallado o Error podrían relacionarse con las siguientes situaciones:

- Un caso de prueba podría fallar porque la subtarea correspondiente no fue implementada por el sujeto. El sujeto no tuvo la intención de implementar la subtarea. En

este caso el medidor no interviene.

- La ejecución de un caso de prueba podría fallar porque la historia de usuario correspondiente fue implementada de manera incompleta o con defectos. Por ejemplo, en la tarea MR, el código fuente del sujeto está implementado para manejar correctamente el comando forward (f), pero el sujeto no ha implementado correctamente el comando backward (b). Otro ejemplo de defectos en el manejo del Rover es que no está realizando correctamente el incremento de los contadores cuando ejecuta el comando Forward. En este caso el medidor no interviene.
- Los resultados pueden presentarse en varios formatos válidos, en especial cuando hay cierta ambigüedad. Sin embargo los casos de prueba verifican una sola salida esperada. A veces un caso de prueba falla porque el sujeto tiene una alternativa de resultado válida. Aquí el medidor sí interviene en el código.

Tenemos los siguientes ejemplos en donde interviene el medidor en el código de los sujetos. El medidor luego de revisar los casos de prueba que fallan ha encontrado alternativas válidas de resultados.

Figura 6.3: Vista parcial de de casos de prueba que dan Error o Falla en el IDE Eclipse

En la tarea MR en el manejo de obstáculos. De acuerdo a la suite de casos de prueba, cuando el Rover encuentra varias veces el mismo obstáculo se espera que lo reporte una sola vez de la siguiente forma: $(\text{"(50,51,S)(50,50)"})$, el obstáculo está identifica-

do por sus coordenadas X e Y en el grid $(50,50)$. Sin embargo algunos sujetos, en su código reportan el mismo obstáculo tantas veces (2) como lo encuentran, de esta forma: $(50,51,S)(50,50)(50,50)$. En este caso el medidor considera este resultado como una **alternativa válida** por lo tanto es un caso de prueba pasado.

En la tarea BSK, cuando se implementa el frame Bonus. En algunos casos los sujetos lo implementan como un Frame adicional en el arreglo de Frames y en otros casos lo implementan en un Frame independiente llamado Bonus. El medidor podría considerar ambos casos como válidos y podría incluir el código que sea necesario para que se pueda medir el manejo de bonus de ambas formas. En el anexo F, sección F.4 presentamos este ejemplo.

La suite de casos de prueba con PCdE verifica el control de entradas inválidas a diferencia de la suite de casos de prueba ad hoc que no lo hace. El medidor luego de pasar los casos de prueba por el código, verifica que el sujeto ha implementado el control de entradas inválidas. Para el control de entradas inválidas el sujeto lo puede hacer: mediante el manejo de Excepciones o mediante control de flujo con comparaciones de los valores ingresados y presentando mensajes en pantalla en caso de obtener datos inválidos. El medidor debe considerar los distintos tipos de controles en entradas inválidas, para ello entra al código y revisa el código del sujeto.

En la tarea MR, cuando la entrada es un comando inválido, según los casos de prueba con PCdE se espera que haya un control de errores de la entrada inválida. Sin embargo, que el Rover no haga ningún movimiento ante una entrada inválida y se quede en la posición actual se considera un resultado alternativo válido y el medidor da por pasado el caso de prueba.

En el anexo F, sección F.5 presentamos este ejemplo.

6.4. Experimentos tratados

En la tabla 6.1 presentamos un listado con las mediciones realizadas necesarias para alcanzar los objetivos de la investigación. El orden del listado de mediciones es de acuerdo al nombre de los experimentos.

Los datos que se muestran para cada medición son los siguientes:

- Experimento: Nombre del experimento
- Medidor: Iniciales del nombre de la persona que realizó la medición.
- Tipo de intervención: Intervención del medidor en el código fuente: Sin intervención, sintáctica y semántica.
- Casos de prueba: Técnica con que se elaboraron los casos de prueba: Ad hoc (estándares), Ad hoc DF (estándares de Davide), PCdE
- Comentarios: Comentarios de la medición. En algunos casos se realizó una segunda medición, con un intervalo de tiempo entre las dos mediciones
- Tipo de medición: Tipo de medición de acuerdo al procedimiento manual o automático.
- Fecha de medición: Mes y año de medición.

Tabla 6.1: Mediciones de los experimentos

Experimento	Mediador	Tipo de Intervención	Casos de prueba	Comentarios	Tipo de medición	Fecha Med.	A. Exp.	A. Cie.	A. CP	A. Med.
Playtech	FU	Sin intervención	Ad-hoc		Automático	Jun/2015				
Playtech	FU	Sintáctica	Ad-hoc		Manual	Ago/2014 a Sep/2014	X		X	
Playtech	FU	Semántica	Ad-hoc		Automático	Jun/2015				
Playtech	FU	Sin intervención	PCdE		Automático	Jun/2015				
Playtech	FU	Sintáctica	PCdE		Automático	Abr/2015			X	X
Playtech	FU	Semántica	PCdE		Automático	Jun/2015				X
Playtech	FU	Sin intervención	PCdE	Segunda medición	Automático	Dic/2015				
Playtech	FU	Sintáctica	PCdE	Segunda medición	Automático	Dic/2015				
Playtech	FU	Semántica	PCdE	Segunda medición	Automático	Dic/2015				
Playtech	EJ	Sintáctica	PCdE		Automático	Nov/2014				X
Playtech	EJ	Semántica	PCdE		Automático	Nov/2014				X
FSecure	FU	Sin intervención	Ad-hoc		Automático	Jul/2015				
FSecure	FU	Sintáctica	Ad-hoc		Manual, excepto BSK	Sep/2014				
FSecure	FU	Semántica	Ad-hoc		Automático	Nov/2015				
FSecure	FU	Sin intervención	Ad-hoc DF		Automático	May/2016				X
FSecure	FU	Sintáctica	Ad-hoc DF		Manual, excepto BSK	May/2016				X
FSecure	FU	Semántica	Ad-hoc DF		Automático	May/2016				X
FSecure	FU	Sin intervención	Ad-hoc	Segunda medición	Automático	Dic/2015				
FSecure	FU	Sintáctica	Ad-hoc	Segunda medición	Automático	Dic/2015				
FSecure	FU	Semántica	Ad-hoc	Segunda medición	Automático	Dic/2015				

Tabla 6.1 – continuación

Experimento	Mediador	Tipo de Intervención	Casos de prueba	Comentarios	Tipo de medición	Fecha Med.	A. Exp.	A. cie.	A. CP	A. Med.
FSecure	FU	Sin intervención	PCdE		Automático	Jul/2015				
FSecure	FU	Sintáctica	PCdE		Automático	Jun/2015				
FSecure	DF	Sin intervención	Ad-hoc DF		Automático	Nov/2015				X
FSecure	DF	Sintáctica	Ad-hoc DF		Automático	Nov/2015				X
FSecure	DF	Semántica	Ad-hoc DF		Automático	Nov/2015				X
ESPEL	FU	Sin intervención	Ad-hoc		Automático	Jun/2015				
ESPEL	FU	Sintáctica	Ad-hoc		Automático	Ene/2015				
ESPEL	FU	Sin intervención	PCdE		Automático	Jun/2015				
ESPEL	FU	Sintáctica	PCdE		Automático	Abr/2015				
Ericsson	FU	Sintáctica	Ad-hoc		Manual	Abr/2015	X		X	
Ericsson	FU	Semántica	Ad-hoc		Manual	Ago/2015				
Ericsson	FU	Sintáctica	PCdE		Manual	May/2015			X	
Ericsson	FU	Semántica	PCdE		Manual	May/2015				
UPM Seminario	FU	Sintáctica	Ad-hoc		Manual excepto Sudoku	Sep/2014	X			
UPV	FU	Sintáctica	Ad-hoc		Manual	Sep/2014	X			
Elektrobit	FU	Sin intervención	Ad-hoc		Manual	Jul/2014	X			
Elektrobit	FU	Sintáctica	Ad-hoc		Manual	Mar/2016				
U.S.Dinamarca	FU	Semántica con Protocolo	Ad-hoc		Manual	Mar/2016				
U.S.Dinamarca DF/IK		Semántica con Protocolo	Ad-hoc		Manual	Feb/2016				
PAF	FU	Sintáctica	Ad-hoc		Automático	Mar/2016				
PAF	FU	Semántica	Ad-hoc		Automático	Abr/2016				
U. de Basilea 2015	FU	Sintáctica	Ad-hoc		Automático	Jul/2015		X		

- A. Exp.: Indica si la medición fue usada en el análisis de la influencia de la experiencia en los resultados experimentales.
- A. Cie.: Indica si la medición fue usada en la replicación en donde se realizó el blind análisis.
- A. CP: Indica si la medición fue usada en el análisis de la influencia de los casos de prueba.
- A. Med.: Indica si la medición fue usada en el análisis de la influencia del medidor.

Las mediciones fueron realizadas en el período comprendido entre Agosto de 2014 y Mayo de 2016. En total se tienen 42 mediciones, correspondientes a 10 experimentos. 7 mediciones (17 %) se realizaron en 2014, 27 mediciones (64 %) en 2015 y 8 mediciones (19 %) en 2016. 36 mediciones (86 %) fueron realizadas por el autor de esta tesis y 6 mediciones por otros investigadores. 28 mediciones (67 %) se realizaron con casos de prueba Ad hoc y 14 mediciones (33 %) con PCdE. 29 mediciones (69 %) se realizaron de manera automática y 13 mediciones (31 %) de forma manual. Se realizaron 11 mediciones (26 %) sin intervención, 18 mediciones (43 %) con intervención sintáctica, 11 mediciones (26 %) con intervención semántica y 2 mediciones (5 %) con intervención Semántica con protocolo.

Las primeras mediciones con intervención sintáctica en Java fueron realizadas manualmente (Playtech, FSecure, Seminario UPM, UPV), las mediciones restantes en Java se realizaron de manera automática. Las mediciones en los experimentos en lenguajes distintos a Java se realizaron manualmente. Las mediciones se obtuvieron de acuerdo a los requerimientos de información para cada uno de los siguientes análisis:

Para realizar el análisis de la experiencia en los resultados experimentales se tomaron en cuenta los experimentos industriales: Playtech, Ericsson y Elektrobit; y con los experimentos académicos Seminario UPM y UPV. En todos ellos se aplicaron los test cases Ad hoc y el tipo de intervención fue sintáctico.

El experimento de la Universidad de Basilicata es la replicación de un experimento en TDD, donde se aplica análisis ciego para realizar la comparativa con el experimento original [59].

Para el análisis de la influencia de los casos de prueba se tienen los siguientes experimentos: Playtech y Ericsson. Se aplicó la intervención sintáctica. Para realizar la comparativa se aplicaron dos tipos de casos de prueba Ad hoc y PCdE.

En el análisis de la influencia de los medidores se contó con la colaboración de otros medidores. Con E. James se midió el experimento Playtech, aplicando casos de prueba con PCdE, se aplicaron dos tipos de intervención: Sintáctica y Semántica. Con D. Fucci se midió el experimento FSecure, aplicando casos de prueba Ad hoc en las mediciones sin intervención, sintáctica y semántica. El experimento U.S. Dinamarca fue medido con el uso de un protocolo de medición, fue medido por D. Fucci e I. Karac de la U. de Oulu, quienes hicieron una medición y para la comparativa se tiene la medición del autor de esta tesis. Adicionalmente, con el propósito de realizar en el futuro una comparativa entre las mediciones del mismo medidor, se tienen experimentos medidos en dos ocasiones por el autor de esta tesis, tales experimentos son Playtech y FSecure, el primero con PCdE y el segundo con FSecure.

Capítulo 7

Estrategias de comparación de resultados

Con el propósito de realizar la comparación de los instrumentos, medidores y tipos de intervención se han seleccionado unas estrategias que nos permitirán evaluar el acuerdo entre mediciones y determinar las fuentes de variación cuando medimos calidad.

Según el área científica, ciencias físicas o ciencias sociales, existen distintas estrategias para obtener el acuerdo entre métodos de medición. En IS no tenemos un marco referencial o estudios similares generalmente aceptados. Por lo tanto, en esta investigación tomamos los métodos usados en otras ciencias para determinar si hay acuerdo entre medidas cuando usamos distintos instrumentos, medidores o tipos de intervención.

Para realizar la evaluación se han seleccionado tres estrategias. En primer lugar, el International Standard Organization (ISO) proporciona los conceptos básicos de medición y el concepto de reproducibilidad. En base a ello, calculamos la incertidumbre de reproducibilidad. Mediante el gráfico de Bland Altman visualizamos el acuerdo entre medidas, por último, calculamos el coeficiente de correlación intraclass (CCI).

7.1. Aproximación a la comparación

Dependiendo del área científica o de la tradición de la investigación, existen varias estrategias para comparar instrumentos de medición o, más generalmente, métodos de medición. Aunque son en gran parte equivalentes, los investigadores de diversas áreas conocen sólo de algunos de ellos. Por ejemplo, los ingenieros están probablemente familiarizados con los enfoques defendidos por JCGM (Join Committe for Guides in Metrology)¹ e ISO. Mientras que las ciencias médicas y de la conducta, el gráfico de Blant-Alman y los coeficientes de correlación intraclass son ampliamente usados. En esta sección, revisaremos esas estrategias con los detalles necesarios para su aplicación.

La teoría de la medición es una rama bien establecida de las matemáticas, con implicaciones de gran alcance tanto en ciencias naturales como en ingeniería, incluyendo IS [53]. Aparecen conceptos básicos y definiciones relevantes en el VIM ² (Vocabulaire International

¹El JCGM está compuesto por ocho organizaciones, entre ellas la ISO(International Standard Organization) y la IEC(International Electrotechnical Commision)

²ISO 3534-1 [81] es una norma que proporciona definiciones similares en la mayoría de aspectos a VIM. Otros organismos, por ejemplo, las agencias nacionales de normalización pueden haber definido su propio

de Metrologie) [95].

Una medición (2.1)³ es el “proceso de obtención experimental de uno o más valores de cantidad” de un mensurando (2.3). En nuestro caso el mesurando es el código fuente en Java o C++ que satisface unas especificaciones de requisitos. Para nuestros propósitos, esos “valores de cantidad” son equivalentes al concepto de resultado de la medición (2.9), aunque en algunos campos puede haber diferencias sutiles entre ambos conceptos.

Una medición se realiza de acuerdo con un método de medición (2.5), el cual describe la “organización lógica de las operaciones utilizadas en la medición”. Entre otros componentes, un método de medición incluye un procedimiento de medición y un instrumento de medición.

Un procedimiento de medición (2.6) es una descripción detallada de una medición, típicamente destinada a un operador humano. Un instrumento de medición (3.1) es un “dispositivo utilizado para hacer mediciones, solo o en conjunto con uno o más dispositivos suplementarios”.

La evaluación del error de medición (sistemático y aleatorio) se realiza en dos condiciones: de repetibilidad y de reproducibilidad. En nuestro caso vamos a trabajar con la reproducibilidad. En la presente tesis doctoral, el mismo medidor no ha medido el mismo objeto más de una vez. La razón principal reside en que, para realizar la medición, es necesario manipular tanto el código como los casos de prueba para que las abstracciones coincidan y la prueba pueda ejecutarse. Estas manipulaciones no pueden “borrarse” de la memoria del medidor. Esto da lugar a que mediciones realizadas próximas en el tiempo produzcan resultados similares, esto es, la varianza de repetibilidad $S_r = 0$.

En un estudio de reproducibilidad se investiga las fuentes de variación de las mediciones. Entre tales fuentes de variación tenemos las llevadas a cabo en diferentes “*condiciones de medición, a partir de un conjunto de condiciones que incluye diferentes ubicaciones, operadores, mediciones[instrumentos], y mediciones replicadas en objetos iguales y similares*”. La incertidumbre debida a la varianza entre-laboratorios. Por lo tanto, teniendo en cuenta que en esta investigación trabajamos con distintos instrumentos, medidores y tipos de intervención usaremos el concepto de reproducibilidad.

Un método complementario que utilizamos es el **gráfico de Bland-Altman**, el cual fue creado como un enfoque alternativo al uso de los coeficientes de correlación. Junto con el CCI, son utilizados ampliamente en las ciencias médicas y conductuales. El gráfico Bland-Altman es un método subjetivo que evalúa el acuerdo entre dos métodos [19]. Debido a que la terminología ISO y Bland-Altman son un poco distintas para un mismo concepto, usaremos la terminología establecida en ISO en todo el análisis.

Bland-Altman usa la desviación estándar de las diferencias para calcular la incertidumbre de reproducibilidad. Tanto en ISO como en Bland-Altman calculamos la incertidumbre de reproducibilidad, por lo tanto es viable la comparativa entre los resultados de ambos métodos.

El **CCI** es una medida de evaluación de consistencia o reproducibilidad. Utilizado principalmente en las ciencias conductuales.

Los valores del CCI pueden oscilar entre 0 y 1, donde 0 indica ausencia de reproducibilidad y 1 la reproducibilidad absoluta de la medición con distintos instrumentos, medidores o tipos de intervención. El valor del CCI para una reproducibilidad satisfactoria es arbitrario,

vocabulario, estrechamente relacionado con VIM

³Incluimos el número de definición VIM entre paréntesis para referencia.

pero en general podría decirse que sobre los 0.75 representa una excelente reproducibilidad, entre 0.4 y 0.75 una reproducibilidad entre regular y buena, por debajo de 0.4 una reproducibilidad baja [56]. En IS no se tienen estudios previos que indiquen que valor de CCI representa una reproducibilidad excelente.

7.2. Exactitud

Para el propósito de esta investigación, el concepto de accuracy es particularmente relevante. Exactitud (2.13) es la “proximidad del acuerdo entre un valor de una cantidad medida y un valor de cantidad real de un mensurando”

Exactitud tiene dos componentes según VIM ⁴, en la Figura 7.1 tenemos: veracidad y precisión. Veracidad (2.14) es la “proximidad del acuerdo entre el promedio de un número infinito de valores de cantidad de medida repetidas y un valor de cantidad de referencia”. Precisión (2.15) es la “proximidad del acuerdo entre [...] los valores de cantidad medidos obtenidos por mediciones repetidas sobre sujetos iguales o similares bajo condiciones especificadas”.

Figura 7.1: Componentes de exactitud según VIM

La veracidad tiene su origen en la presencia de errores sistemáticos (2.17), también conocido como sesgo, por ejemplo: pérdida de calibración. A su vez, la precisión surge de la presencia de errores aleatorios (2.19), por ejemplo, errores de lectura del operador.

La evaluación del error de medición (sistemático y aleatorio) se realiza en dos circunstancias, esto es, condiciones de repetibilidad y reproducibilidad.

La fiabilidad (Repetibilidad en la terminología de ISO) (2.20) es la incertidumbre (2.26) de un conjunto de medidas realizadas usando “*el mismo procedimiento de medición, mismos operadores, mismo [instrumento] de medición, mismas condiciones operativas, misma ubicación [...] en objetos iguales o similares sobre un corto período de tiempo*”. El GUM⁵ (Guide to the expression of Uncertainty in Measurement) [94] usa la desviación estándar s_r de un conjunto de mediciones $X = \{x_1, x_2, \dots, x_n\}$ como base para definir el concepto de repetibilidad:

⁴Esta figura asume que la distribución del error es gaussiana, lo que es habitual pero no necesariamente cierto en todas las circunstancias (atribución: SV1XV bajo licencia CC-BY-SA-3.0)

⁵ISO 5725 [80] es el equivalente estándar ISO. Existen varias diferencias entre GUM e ISO 5725, pero son irrelevantes para nuestros propósitos y no se discutirán

$$s_r = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2} \quad (7.1)$$

s_r estima la variabilidad σ_r alrededor del promedio del conjunto de medidas X , es decir, el componente precisión de la exactitud. s_r no se ve afectado por el cambio de ubicación, es decir, $Var(X) = Var(X + k)$, siendo k una constante arbitraria. Significa que la veracidad no juega ningún papel en la determinación de la incertidumbre.

La reproducibilidad (2.24) es la incertidumbre de un conjunto de medidas llevadas a cabo en diferentes “condiciones de medición, a partir de un conjunto de condiciones que incluye diferentes ubicaciones, operadores, mediciones [instrumentos], y mediciones replicadas en objetos iguales y similares”. GUM no aborda la reproducibilidad. ISO 5725 define la reproducibilidad de una manera restringida como la incertidumbre debida a la varianza entre-laboratorios. Sin embargo, esta definición puede ser fácilmente extrapolada a otras condiciones de medición, tales como diferentes instrumentos de medición, el cual es uno de los objetivos de esta investigación.

ISO 5725 define la reproducibilidad como la desviación estándar s_R , calculada como se muestra en la ecuación 7.2:

$$s_R^2 = s_L^2 + s_r^2 \quad (7.2)$$

s_r es la incertidumbre de repetibilidad discutida anteriormente, mientras s_L es la incertidumbre debido a diferencias aleatorias entre laboratorios. En lo que sigue a continuación, usaremos el símbolo s_M , que significa la incertidumbre debida a diferencias aleatorias entre los instrumentos en otras condiciones de medición (nuestro problema en concreto), en lugar de s_L .

s_M y s_r pueden ser estimados usando el modelo lineal [51,141] mostrado en la ecuación 7.3. Debido a la existencia de dos fuentes de variabilidad s_M and s_r , el modelo de la ecuación 7.3 corresponde con un diseño Split-plot, siendo M un factor aleatorio, y donde:

$$y = m + M + e \quad (7.3)$$

donde:

- y es el valor resultado de la medición.
- m es el intercept.
- M representa la incertidumbre $N(0, \sigma_M)$ debido al instrumento de medición en los resultados de la medición.
- e representa la incertidumbre que no se puede asignar a ningún otro componente del modelo lineal. En todos los casos cubiertos por la norma ISO, $e \sim N(0, \sigma_r)$, es decir, equivale a incertidumbre de repetibilidad. Sin embargo, como ya hemos indicado, en nuestro caso obviaremos la incertidumbre de repetibilidad. Ello implica unos ligeros cambios a la ecuación 7.3, que se describirá en los capítulos siguientes.

Cuando s_R es calculado como se indica en las ecuaciones 7.2 y 7.3, representa la desviación estándar de las diferencias entre mediciones obtenidas por los instrumentos en otras condiciones de medida bajo comparación.

7.3. Incertidumbre expandida

s_R y s_r son incertidumbres estándar (2.30). Las diferencias de medición exceden s_R y s_r bastante a menudo. Por esta razón las incertidumbres ampliadas (2.35) R y r proporcionan información más significativa sobre el grado en que las mediciones pueden diferir.

Cuando solamente se comparan dos instrumentos⁶, las incertidumbres expandidas son los límites del intervalo de confianza de la diferencia de medias entre las mediciones $\bar{d}$ en el nivel $1 - \alpha$. Más específicamente, si tenemos dos conjuntos de mediciones $M_1 = \{x_{11}, x_{12}, \dots, x_{1n}\}$ y $M_2 = \{x_{21}, x_{22}, \dots, x_{2n}\}$, entonces $\bar{d}$ es calculado como:

$$\bar{d} = \frac{1}{n} \sum_{i=1}^n (x_{1i} - x_{2i}) \quad (7.4)$$

El intervalo de confianza para $\bar{d}$ es:

$$\bar{d} \pm t_{(1-\alpha/2)} * s_R * \sqrt{2} \quad (7.5)$$

con $(n - 1)$ grados de libertad. $k = t_{(1-\alpha/2)}$ se conoce como factor de cobertura (2.38). Cuando $n \geq 30$, puede usarse la aproximación normal. Típicamente, $\alpha = 0.95$, y por lo tanto $k = Z_{(1-\alpha/2)} = 1.96$. Las incertidumbres ampliadas son definidas como se indica en la ecuación 7.6:

$$\begin{aligned} r &= 1.96 \times s_r * \sqrt{2} \\ R &= 1.96 \times s_R * \sqrt{2} \end{aligned} \quad (7.6)$$

7.4. Representación gráfica

Cuando se comparan solo dos instrumentos de medición, la reproducibilidad puede visualizarse usando un gráfico Bland-Altman [19]. Este gráfico representa los valores medios obtenidos por cada instrumento o condición de medida en función de su diferencia. Como se observa en la figura 7.2, se muestra el promedio de las variables AH y EP en función de su diferencia. En los gráficos usamos rangos de -100 a 100 para presentar los datos. Cada punto representa la diferencia entre los valores medidos con dos instrumentos distintos.

Líneas horizontales se trazan en⁷ la diferencia media $\bar{d}$ y $\bar{d} \pm R = \bar{d} \pm 1.96 \times s_R \times \sqrt{2}$. El diagrama de dispersión así obtenido muestra no sólo las incertidumbres expandidas, sino también la distribución de las diferencias, permitiendo un examen más profundo, por ejemplo, sobre las relaciones entre la media y las diferencias [19].

⁶La interpretación con más de dos medidas es menos evidente, pero no difiere esencialmente de lo indicado

⁷El gráfico Bland-Altman usa un nivel de cobertura $k = 2$, pero nosotros usaremos $k = 1.96$ para evitar reportar resultados ligeramente diferentes dependiendo del método de comparación usado

Figura 7.2: Gráfico de Bland-Altman

7.5. Medición en las ciencias del comportamiento

En las ciencias físicas y la ingeniería, los mensurandos (/los elementos a medir) tienen una realidad física, por ejemplo, longitudes, duraciones, etc. La veracidad de una medición no puede ser establecida perfectamente; sin embargo, la existencia de valores de referencia garantiza que la veracidad no exceda ciertos límites.

En las ciencias del comportamiento⁸, los mensurados, con pocas excepciones, carecen de realidad física. Por ejemplo, aspectos tales como calidad o productividad, o incluso características más físicas tales como líneas de código, son constructos hipotéticos, no directamente observables.

Con el propósito de hacer observables a los constructos, tienen que ser operacionalizados [157]. Sin embargo, al ser inobservables, no se pueden definir con precisión los valores de referencia. Los valores de referencia se definen habitualmente en función de un método o instrumento de medición dado como referencia estándar. Sin embargo, dicho valor de referencia no comparte las propiedades de los valores de referencia para sistemas físicos; en particular, la escala de medición es arbitraria. En consecuencia, la veracidad se convierte en un concepto discutible.

La ausencia de valores de referencia en las ciencias del comportamiento es menos severa de lo que parece. Hemos discutido anteriormente que la veracidad no juega un papel importante en la determinación de la exactitud de los métodos de medición en las ciencias físicas. Las ecuaciones 7.1 y 7.3 no dependen de los valores de referencia, y pueden ser aplicados a métodos de medición arbitrarios.

Las ecuaciones 7.1 y B.1 dependen del tipo de escala utilizado. Por ejemplo, si un método de medición utiliza una escala de 1 a 5, mientras que otro método destinado al mismo constructo usa la escala 0 %-100 %, no se pueden comparar fácilmente. Del mismo

⁸Las ciencias de la salud tienen áreas con características similares a las naturales, por ejemplo, la presión sanguínea, y las ciencias del comportamiento, por ejemplo, medición del dolor.

modo, las ecuaciones 7.1 y B.1 pueden aplicarse solo a variables de razón (ratio); No pueden usarse variables ordinal/nominal. Las ciencias del comportamiento, en particular la psicología, utiliza dichos tipos de variables con frecuencia.

En lo que respecta a la terminología, en las ciencias del comportamiento, el término fiabilidad es usado en lugar de precisión cuando las variables tienen intervalo/ratio; cuando las variables son nominal/ordinal, el término más frecuente es agreement [15]. Los términos consistencia y conformidad pueden también ser encontrados como sinónimos de precisión y veracidad [133]. Sin embargo, el uso de todos estos términos es bastante inconsistente en la literatura. Por esta razón, utilizaremos la terminología VIM y GUM.

En cuanto a los procedimientos de comparación, las ciencias del comportamiento se basan en el concepto de validez convergente. Validez convergente es “el grado en que una medida se correlaciona con otras medidas del mismo constructo” [149]. Esta idea general se implementa de maneras distintas.

La estrategia de comparación más conocida es el coeficiente de correlación de Pearson r (o las contrapartes no paramétricas en caso de no normalidad)⁹. r es una instancia del coeficiente de correlación interclase, que evalúa la relación entre dos variables diferentes, por ejemplo, los resultados de dos instrumentos de medición, independientemente de los elementos que están midiendo. r tiene varios inconvenientes [19,112] y está generalmente aceptado que r debe utilizarse únicamente con fines exploratorios (y con la debida precaución).

Hoy en día, los coeficientes de correlación intraclase (ICC) son las opciones preferidas para comparar instrumentos de medición. Hay varios tipos de ICC¹⁰. Para nuestro problema de investigación, el correcto es el Modelo 3 (diseño factorial mixto)¹¹ [73,133]. El Modelo 3 evalúa la variabilidad dentro de los items (código de los sujetos) y lo compara con la variabilidad total, incluyendo el producido por los instrumentos de medición. La ventaja de ICC con comparación con r es precisamente la separación de ambas fuentes de varianza, que r no puede lograr.

El Modelo 3 hace las siguientes suposiciones:

- El conjunto de items $I = \{1, 2, \dots, n\}$ es una muestra aleatoria de la población objetivo.
- Los instrumentos o condiciones de medición M son los únicos que interesan.
- Cada item en I ha sido medido por todos los instrumentos o condiciones de medición M .

Cuando se cumplen estos supuestos, el Modelo 3 se define como:

$$\rho = \frac{s_I^2}{s_I^2 + s_e^2} \quad (7.7)$$

donde s_I^2 es la varianza dentro de los items (intraclase), y s_e^2 es la varianza del error (interclase). s_I^2 y s_e^2 se puede obtener fácilmente a partir del modelo lineal:

⁹Enfoques relacionados, como α de Cronbach, abordan una cuestión diferente a la similitud entre los métodos de medición y no puede utilizarse para resolver nuestro problema de investigación

¹⁰Consultar [73] para una descripción en profundidad. Un algoritmo de selección está disponible en la página 24

¹¹También conocido como ICC(3, 1) en algunos paquetes estadísticos como SPSS®.

$$y = m + M + I + e \quad (7.8)$$

donde m es el intercept, M el factor fijo que representa a los instrumentos de medición, I el factor aleatorio que representa el conjunto de items y e el random error. La heterogeneidad de los items en I es el origen de la varianza intraclase σ_i . La varianza (interclase), debido a los instrumentos de medida M , se asigna al término del error $e \sim N(0, \sigma_e)$.

La similitud entre las ecuaciones 7.3 y 7.8 es notable. De hecho, debido a las características particulares de nuestro problema, en los Capítulos 8-10 veremos que las ecuaciones 7.3 y 7.8 son equivalentes.

Capítulo 8

Estudio de las diferencias entre test suites

8.1. Características generales de los datos

8.1.1. Experimentos

Para este estudio, utilizaremos dos experimentos realizados en industria (que denominaremos PT y EC para ocultar la identidad de las empresas). Estos experimentos están descritos en la Sección 4.3.

8.1.2. Mediciones

Tal y como indica la Tabla 8.1, hemos realizado 148 mediciones sobre 74 códigos, desarrollados por 37 sujetos experimentales, usando dos test suites: Ad hoc y PCdE. Todas las mediciones han sido realizadas por el mismo medidor ("Measurer 1").

Tabla 8.1: Características de los experimentos que usaremos en el estudio de las diferencias entre test suites

Experimento	# Sujetos	# Códigos	# Mediciones	Diseño
PT	17	34	68	within-subjects
EC	20	40	80	within-subjects
Todos los experimentos	37	74	148	within-subjects

8.2. Análisis exploratorio

En esta sección, exploraremos los datos desde distintas perspectivas (gráficos de densidad, correlaciones, diagramas de dispersión y modelos lineales), con el objetivo de determinar desde una perspectiva exploratoria si las mediciones obtenidas mediante las distintas test suites difieren, y en qué grado.

8.2.1. Distribución de los datos

Una estrategia sencilla que permite evaluar la similaridad entre las mediciones tomadas por los distintos medidores es comparar la distribución de las mediciones usando, por ejemplo, gráficos de densidad. Si las mediciones son similares, los gráficos de densidad obtenidos por los medidores tendrán un aspecto similar. Lo inverso no es cierto, esto es, gráficos de densidad similares pueden proceder de conjuntos de mediciones muy diferentes¹. No obstante, la intención del presente texto es tanto didáctica como expositiva, por lo que optamos por un acercamiento progresivo a la respuesta a la pregunta "Son similares las mediciones realizadas por distintas test suites?". Análisis posteriores explorarán otras dimensiones de los datos.

La Fig. 8.1 muestra los gráficos de densidad para PT y EC. Hemos optado por generar los gráficos de densidad separados por experimento para evitar mezclas espúreas, aunque la distribución de los datos en ambos experimentos es muy similar. La distribución de las mediciones realizadas con las test suites Ad hoc y PCdE no podrían ser más diferentes. La test suite Ad hoc tiende de generar mediciones en todo el rango de posibles calidades (entre 0 y 100%). Por el contrario, la test suite PCdE produce mediciones muy bajas, inferiores en prácticamente todos los casos al 50 %. Aunque debemos esperar a realizar el resto de análisis, la Fig. 8.1 sugiere que las mediciones generadas por ambas test suites son **muy distintas**. Probablemente, dicha diferencia entre mediciones debe tener un impacto considerable en el análisis estadístico de los experimentos PT y EC.

Figura 8.1: Gráficos de densidad para PT y EC

8.2.2. Correlaciones

Una condición necesaria, pero no suficiente [19,112], para que las mediciones obtenidas por distintos medidores sean similares es que las mediciones estén correladas. Dado que la

¹Supongamos un caso sencillo: $X_1 = \{a, b\}$ y $X_2 = \{b, a\}$ poseen la misma distribución, aunque los datos están invertidos.

variable QLTY es de tipo escala, se aplicaremos la correlación de Pearson. La Tabla 8.2 muestra las correlaciones entre las mediciones obtenidas con las test suites Ad hoc y PCdE. En ambos casos, las correlaciones son altas y estadísticamente significativas. En conjunción con el gráfico de densidad de la Fig. 8.1, las correlaciones sugieren que las mediciones Ad hoc y PCdE varían (crecen o decrecen) conjuntamente, pero sus valores absolutos difieren, esto es, **existe un sesgo entre ambos tipos de mediciones**.

Tabla 8.2: Correlaciones entre test suites para los experimentos PT y EC

Experiment	r	$p - value$
PT	0.67	<0.001
EC	0.82	<0.001

El sesgo es fácilmente visible en un diagrama de Bland-Altman. Realizaremos dicho análisis más tarde. Por el momento, centraremos nuestra atención en los gráficos de dispersión, mostrados en la Fig. 8.2.

Estos gráficos de dispersión son equivalentes a las correlaciones indicadas en la Tabla 8.2 (nótese que $r = \sqrt{R^2}$), pero poseen la ventaja frente al coeficiente de correlación que la disposición de los puntos (esto es, las mediciones obtenidas mediante las test suites Ad hoc y PCdE) son visibles. En el caso de que ambas test suites coincidieran en todas sus mediciones, los puntos deberían situarse encima de la línea roja (concordancia perfecta). Es evidente que éste no es el caso. Prácticamente todos los puntos se sitúan muy por debajo de la línea roja, lo que indica que las mediciones obtenidas con la test suite Ad hoc son netamente mayores que las obtenidas con PCdE (nótese que la recta de regresión es $\sim EP = 0,3AH$). Tal diferencia entre mediciones debería reflejarse de alguna forma en el análisis estadístico.

Figura 8.2: Gráficos de dispersión para PT y EC (recta de regresión en azul, concordancia perfecta en rojo).

Tabla 8.3: Análisis de la variable respuesta QLT_Y para el experimento PT

	AH	EP
(Intercept)	68,45 (12,33) ^{***}	22,07 (4,63) ^{***}
TreatmentTDD	−20,45 (9,99) [*]	−7,65 (3,34) [*]
TaskMR	−22,28 (9,99) [*]	6,27 (3,34)
GroupMR → BSK	13,38 (14,93)	−,89 (5,87)
AIC	321.33	260.47
Num. obs.	34	34
Var: Group:Subject (Intercept)	522.07	98.69
Var: Residual	845.18	94.57

*** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

8.2.3. Análisis estadístico de experimentos

Las diferencias entre las mediciones Ad hoc y PCdE quedan confirmadas mediante el análisis estadístico de los experimentos PT (Tabla 8.3) y EC (Tabla 8.4). En general, los efectos de todos los factores cambian cuando se utiliza una u otra medición:

- El *intercept* cambia de valor, especialmente en PT, donde pasa de 68.45 a 22.07. Esto implica una reducción de $\frac{68,45}{22,07} = 3,1 \sim 3$, esto es, la razón entre Ad hoc y PCdE que muestran los gráficos de dispersión. En EC, el cambio es menor, pero también sustancial.
- La *Tarea* también cambia bruscamente de valor, hasta el punto de que con Ad hoc la calidad de $BSK > MR$, pero con PCdE $MR > BSK$. En PT, el cambio también afecta a la significación estadística.
- El *Grupo* también cambia de valor en ambos experimentos, llegando incluso a cambiar de signo en PT. La significación estadística también se ve afectada.

En contrapartida, el factor *Tratamiento* no parece verse afectado, o lo hace en mucha menor medida. Aunque cambia fuertemente de valor, ni el signo ni la significación estadística parecen verse afectados.

Tabla 8.4: Análisis de la variable respuesta QLTY para el experimento EC

	AH	EP
(Intercept)	27,32 (10,09)**	13,86 (4,80)**
TreatmentTDD	16,81 (9,70)	2,78 (4,14)
TaskMR	-12,53 (9,70)	4,35 (4,14)
GroupMR → BSK	23,37 (10,46)*	3,37 (5,38)
AIC	373.98	319.38
Num. obs.	40	40
Var: Group:Subject (Intercept)	76.63	59.32
Var: Residual	941.32	171.18

*** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

8.3. Análisis de reproducibilidad

Dado que el mismo objeto sólo se ha medido una una vez con cada test suite, los datos usados para el análisis de reproducibilidad tienen la estructura de un experimento split-plot, donde cada código corresponde con un *whole plot*, y las test suites con *subplots*. La determinación de la reproducibilidad puede realizarse, matemáticamente, de varias formas. En nuestro caso, utilizaremos el modelo mixto (en notación *lmer*) indicado en la Eq. 8.1.

$$QLTY = Instrument + (1|Item) \quad (8.1)$$

siendo el "Instrument" la test suite basada en Ad hoc o PCdE.

La Tabla 8.5 muestra el resultado del análisis de reproducibilidad de acuerdo a la Eq. 8.1. La fila "Var: Residual" representa la variabilidad adscribible a la test suite, mientras que la fila "Var: Item (Intercept)" representa la variabilidad adscribible al código. La *varianza de reproducibilidad* se calcula a partir de la variabilidad adscribible a la test suite y tiene el valor $S_R = \sqrt{389,25} = 19,73$, mientras que la *incertidumbre expandida* es $1,96 \times \sqrt{2} \times 19,73 = 54,69$. Esto significa que la diferencia entre dos mediciones tomadas por dos medidores oscilará, en el 95 % de los casos, en $\pm 54,69$ puntos porcentuales. Se trata de un valor, a nuestro juicio, muy elevado.

Hemos creído ilustrativo analizar la reproducibilidad por experimento, ya que la variabilidad en los resultados del análisis estadístico parecen más importantes en PT que en EC.

Tabla 8.5: Análisis conjunto de reproducibilidad

	Modelo de la Eq. 8.1
(Intercept)	46,77 (3,27)***
InstrumentEP	-26,81 (3,24)***
Var: Item (Intercept)	400.80
Var: Residual	389.25

*** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

La Tabla 8.6 muestra los resultados. Efectivamente, PT exhibe una *incertidumbre expandida* más elevada que EC, que podría estar asociada a los cambios en el análisis estadístico mostrado en la Tabla 8.3.

Tabla 8.6: Análisis de reproducibilidad para los experimentos PT y EC

Experimento	Varianza de reproducibilidad	Incertidumbre expandida
PT	22,26	$\pm 61,7$
EC	16,85	$\pm 46,7$

8.4. Gráficos de Bland-Altman

El análisis de reproducibilidad realizado en la Sección 8.3 proporciona de forma abstracta el rango (la *incertidumbre expandida*) en que dos mediciones tomadas por dos medidores pueden diferir. En esta sección, obtendremos los mismos resultados, pero de modo gráfico.

8.4.1. Gráfico conjunto de los experimentos PT y EC

La Fig. 8.3 muestra el gráfico Bland-Altman para los experimentos PT y EC, conjuntamente. Este gráfico es la contrapartida visual del análisis de reproducibilidad conjunto descrito en la Sección 8.3, cuando sólo se comparan dos entidades (test suites, en este caso). De hecho, puede comprobarse que el efecto de "InstrumentEP" en la Tabla 8.5 es el mismo (salvo signo) que el sesgo mostrado en el gráfico de Bland-Altman, y los límites de acuerdo coinciden con el *sesgo* $\pm$ *incertidumbre expandida*.

La ventaja del diagrama de Bland-Altman es, como se ha indicado anteriormente, su carácter gráfico. Puede observarse perfectamente la distribución de las diferencias entre mediciones. Dos características son apreciables:

- Pácticamente todos los puntos (/diferencias entre mediciones) son superiores a cero. Esto respalda lo ya observado en los diagramas de dispersión, donde indicábamos que las mediciones Ad hoc eran generalmente mayores que las mediciones PCdE.
- Las diferencias entre mediciones no están distribuidas alrededor de la línea central. Se observa una clara tendencia a que las diferencias sean mayores cuando mayor son los valores de QLTY. Esto respalda lo observado en los diagramas de densidad, respecto a la relativamente uniforme distribución de las mediciones Ad hoc, y la distribución sesgada a la derecha de las mediciones de PCdE.

8.4.2. Gráficos diferenciados por experimento

Las características anteriores siguen siendo visibles si se grafican los datos de PT y EC por separado, tal y como muestra la Fig. 8.4. Esto implica que las particularidades de las mediciones, con casi total seguridad, tienen su origen no en virtud del experimento (PT o EC) donde son recogidas, sino a causa de las test suites utilizadas.

Figura 8.3: Gráficos de Bland-Altman para los experimentos PT y EC conjuntamente

Figura 8.4: Gráficos de Bland-Altman para los experimentos PT y EC, por separado

8.4.3. Estudio gráfico de posibles variables influyentes

En el análisis estadístico reportado en la Sección 8.2.3, hemos observado que las test suites Ad hoc y PCdE producen que ciertos factores del análisis estadístico cambien de forma sustancial sus tamaños de efecto. Una forma de profundizar en las razones por las cuales se producen estas variaciones es estudiar, de forma separada, los gráficos de Bland-Altman² para cada uno de los niveles de los factores afectados.

8.4.3.1. Factor *Tarea*

Las Fig. 8.5 y 8.6 muestran las mediciones que genera cada test suite, separadas por tarea experimental (MR y BSK). En ambos casos, puede observarse que la línea central (esto es, el sesgo) para MR y BSK no coinciden. Pongamos como ejemplo la Fig. 8.5, que representa los datos de PT. El sesgo para MR es 17.77, mientras que para BSK es 47.08. Es una diferencia de casi 30 puntos. Esta misma diferencia coincide, *grosso modo*, con el cambio en el tamaño de efecto (de -22.28 a 6.27) para *Tarea* en la Tabla 8.3. Lo mismo ocurre, aproximadamente, con EC.

Figura 8.5: Gráficos de Bland-Altman para los experimentos PT, separados por tarea

²Este mismo estudio podría haberse realizado en el marco del análisis de reproducibilidad. Lo presentamos en el marco de los gráficos de Bland-Altman debido a la ventaja del aparato visual para entender los efectos de las variables bajo estudio.

Figura 8.6: Gráficos de Bland-Altman para los experimentos EC, separados por tarea

8.4.3.2. Factor *Grupo*

Las Figs. 8.7 y 8.8 muestran las mediciones que genera cada test suite, separadas por grupo ($MR \rightarrow BSK$ y $BSK \rightarrow MR$). Puede observarse fácilmente que los sesgos relativos parecen ser los causantes de las variaciones en el análisis estadístico. A modo de ejemplo, volvamos a centrar la atención en la Fig. 8.7, correspondiente a PT. El sesgo entre los grupos $MR \rightarrow BSK$ y $BSK \rightarrow MR$ es de unos 14 puntos, que corresponde casi exactamente con el cambio en el tamaño de efecto (de 13.38 a -.89) para *Grupo* en la Tabla 8.3. Lo mismo ocurre, aproximadamente, con EC.

8.4.3.3. Posibles razones de las variaciones

La conclusión más evidente, a la vista de los datos obtenidos, es que la variación en el análisis estadístico mostrado en la Sección 8.2.3 se debe a los sesgos diferenciales de las test suites respecto a los niveles de ciertos factores, en concreto, *Tarea* y *Grupo*. En contrapartida, el papel de la *incertidumbre expandida* está menos claro. En capítulos siguientes seguiremos intentando aclarar la influencia, si existe, de la *incertidumbre expandida* en los análisis estadísticos.

Figura 8.7: Gráficos de Bland-Altman para los experimentos PT, separados por grupo

Figura 8.8: Gráficos de Bland-Altman para los experimentos EC, separados por grupo

8.5. ICC

El ICC es $\rho = \frac{s_i^2}{s_m^2 + s_i^2}$, siendo s_m^2 la varianza adscribible a los medidores y s_i^2 la varianza adscribible a los items (/código). El ICC puede calcularse directamente usando los datos obtenidos durante en análisis de reproducibilidad en la Sección 8.3. De este modo, $\rho = \frac{400,8}{389,25 + 400,8} = 0,51$.

Habitualmente, se considera que existe una alta concordancia entre instrumentos de medición cuando $\rho \geq 0,75$ [55]. El valor de ICC que hemos obtenido es mucho menor. Esto implica que, desde el punto de vista del ICC, **las mediciones obtenidas por las distintas test suites son bastante diferentes entre sí.**

8.6. Conclusiones

En esta sección expondremos los hallazgos que hemos obtenido en este capítulo, acerca de los efectos de la test suite en los resultados de la medición.

8.6.1. Respecto a la influencia de las test suites

Los resultados de los análisis indican claramente que

las test suites construidas con las estrategias "Ad hoc" y "Partición de equivalencia" generan mediciones completamente distintas.

Las mediciones Ad hoc producen valores sistemáticamente más altos de calidad que las mediciones PCdE. Además, los análisis estadísticos realizados con ambos tipos de mediciones arrojan resultados que varían en gran medida: el *intercept* y todos los factores (excepto el *Tratamiento*) se ven afectados.

Respecto a los factores, hemos podido comprobar cómo las variaciones en los tamaños de efecto están directamente relacionadas con sesgos diferenciales entre niveles.

Por ejemplo, si un nivel X de un factor posee un sesgo de +20, y otro nivel Y un sesgo de +10, los 10 puntos de diferencia se trasladan casi directamente al efecto del nivel X . A medida que el sesgo entre niveles aumenta, también aumenta el tamaño de efecto, lo que altera el análisis estadístico, hasta el punto de que el efecto de un factor puede llegar a invertirse.

No hemos podido verificar fehacientemente los motivos del cambio de valor del *intercept*. Lo más probable es que se deba al sesgo entre test suites a nivel de experimento (esto es, a los sesgos reportados en la Fig. 8.3, o al *slope* de la recta de regresión. Sin embargo, la traslación del sesgo no es tan clara en este caso, y el *slope* sólo es un multiplicador aproximado. Sin duda, las variaciones en el *intercept* se ven afectadas por los sesgos subyacentes que afectan a la *Tarea* y el *Grupo*.

El hallazgo más importante de los análisis realizados concierne al *Tratamiento*. Éste no parece verse alterado por los cambios en las mediciones. La razón de ello es, en nuestra opinión, que las test suites sólo interaccionan con el código de los sujetos, pero no con la estrategia de programación. El mismo código, ya sea desarrollado usando YW, ITLD o TDD, resulta en la misma medición.

El *Tratamiento*, al ser transparente para las test suites, no se ve afectado.

8.6.2. Motivos de la diferencia entre mediciones generadas por Ad hoc y PCdE

Las test suites Ad hoc y PCdE, al tratarse de artefactos, pueden ser estudiados para tratar de determinar el origen de las diferencias entre mediciones. Este estudio puede

realizarse desde distintos puntos de vista. Por ejemplo, podrían relacionarse los casos de prueba entre test suites y tratar de determinar disparidades y redundancias. Lo primero es sencillo, pero lo segundo no, ya que determinar que un caso de prueba es redundante exige la existencia de criterios de construcción de casos de prueba, los cuales para Ad hoc no están disponibles.

Por este motivo, hemos optado por una aproximación más empírica. En lugar de analizar teóricamente los casos de prueba, hemos ejecutado las test suites sobre códigos de referencia MR y BSK que implementan fragmentos de la funcionalidad requerida. Hemos realizado tantas ejecuciones como combinaciones independientes de funcionalidad que MR y BSK podrían poseer. En cada ejecución, registramos la calidad real del código (que conocemos de antemano, ya que se trata de un código de referencia) y la calidad medida por la test suite.

Las Figs. 8.9 y 8.10 muestran los resultados de este estudio. En cada gráfico, la línea diagonal representa el caso en que las test suites proporcionan exactamente el valor de calidad correcto. La calidad correcta se representa en el eje horizontal "maxQLTY", y se determina a partir de la funcionalidad implementada en el código de referencia. La calidad medida por las test suites se representa en el eje vertical QLTY. En nuestra opinión, está ocurriendo lo siguiente:

- Las dos test suites proporcionan calidades parecidas para MR, aunque la test suite PCdE tiende a generar valores ligeramente menores. Esto coincide con el relativamente reducido sesgo observado en las Figs. 8.5 y 8.6.
- Los resultados de las mediciones de BSK son completamente distintas para Ad hoc y PCdE. En el primer caso, aunque existen fuertes oscilaciones por encima y debajo de la línea diagonal, la distribución de mediciones envuelve razonablemente a ésta. Sin embargo, para PCdE la situación es completamente distinta. Las mediciones tienden a formar una franja horizontal, esto es, las mediciones generadas son siempre muy bajas y, lo que es más importante, más bajas sistemáticamente que las obtenidas con Ad hoc. Estos resultados son completamente concordantes con los sesgos de las Figs. 8.5 y 8.6.

Figura 8.9: Calidades generadas por las test suites utilizando el código de referencia MR

Figura 8.10: Calidades generadas por las test suites utilizando el código de referencia BSK

En nuestra opinión, las bajas calidades obtenidas por la test suite PCdE, en particular con BSK, se debe a un problema de expectativas incumplidas. Las test suites basadas en PCdE asumen que el código debe poseer un cierto control de errores, a los cuales van dirigidos un gran número de casos de prueva (dos de cada tres). En un código donde el control de errores no sea requerido, como es el caso de MR y BSK en los experimentos que estamos analizando, todos los casos de prueba de condiciones excepcionales fallan.

Una posible alternativa podría ser exigir a los sujetos experimentales que implementen control de errores en sus códigos. Sin embargo, ésta no es necesariamente la solución correcta. La Fig. 8.11 muestra cómo se comporta la test suite PCdE en BSK en dos condiciones: a) que el código de referencia implemente el control de errores, o b) retirando los

casos de prueba de condiciones excepcionales de la test suite PCdE. Los resultados son evidentes. Cuando el código de referencia implementa el control de errores (parte izquierda de la Fig. 8.11), los resultados de medición exhiben un sesgo fuertemente positivo, causada esta vez porque los casos de prueba de condiciones excepcionales pasan sistemáticamente. Si se eliminan los los casos de prueba de condiciones excepcionales (parte derecha de la Fig. 8.11), las mediciones se aproximan mucho más a la línea diagonal que marca la concordancia entre la calidad esperada y la calidad medida.

En conclusión:

el diseño de las test suites de medición no es una tarea sencilla. Las consideraciones teóricas (no cabe duda que PCdE es una aproximación sistemática, mientras que Ad hoc no lo es) pueden no ser adecuadas debido a incumplimiento de expectativas *u otras consideraciones*. Las test suites de medición deberían ser pilotadas, posiblemente usando códigos de referencia.

Figura 8.11: Calidades generadas por las test suites PCdE utilizando el código de referencia BSK con control de errores (izquierda) y sin casos de prueba de condiciones excepcionales (derecha)

8.6.3. Respecto a los tipos de análisis

Todos los métodos de análisis han sido congruentes en señalar la disparidad entre las mediciones obtenidas mediante las test suites Ad hoc y PCdE. El diagrama de Bland-Altman ha sido especialmente informativo, ya que permite obtener los sesgos de las mediciones fácilmente. Esta misma información podría obtenerse mediante el análisis de reproducibilidad, pero los diagramas de Bland-Altman tienen la ventaja de proporcionar información visual, y no puramente numérica.

Nos ha resultado en cierta medida sorprendente no poder relacionar la *incertidumbre expandida* con ninguno de los efectos negativos de las mediciones en el análisis estadístico.

Sólo los sesgos tiene una relación clara. Una posibilidad es que la *incertidumbre expandida* pudiera motivar los cambios en las significaciones estadísticas, aunque lo creemos poco probable. La significación estadística está fundamentalmente ligada a la variabilidad de las mediciones, no a la variabilidad de la diferencia entre dos mediciones, que es lo que mide la *incertidumbre expandida*. Una *incertidumbre expandida* alta podría venir motivada, por ejemplo: a) que tanto las mediciones procedentes de las test suites Ah hoc y PCdE posean variabilidades similares (y probablemente moderadas), o b) porque la test suite Ah hoc presente una variabilidad alta, y PCdE una baja (o viceversa). En el primer caso, la significación estadística podría ser cualquiera, mientras que en el segundo, el análisis de las mediciones Ad hoc *tendería* a ser no significativo, mientras que el análisis de PCdE tendería a ser significativo (o viceversa). Ello implica que:

independientemente del valor de la *incertidumbre expandida*, los análisis estadístico podrían ser más o menos estables, en función de las características particulares de cada medición.

Capítulo 9

Estudio de las diferencias entre medidores

9.1. Características generales de los datos

9.1.1. Experimentos

Para este estudio, utilizaremos tres experimentos, dos de ellos realizados en industria (que denominaremos FS y PT para ocultar la identidad de las empresas) y otro en academia (que denominaremos DI por paralelismo con los anteriores). Estos experimentos están descritos en la Sección 4.3.

9.1.2. Mediciones

Tal y como indica la Tabla 9.1, hemos realizado 570 mediciones sobre 285 códigos, desarrollados por 116 sujetos experimentales, usando dos test suites: Ad hoc y PCdE y cuatro tipos de intervención: Sin intervención, sintáctica, semántica, y semántica con protocolo. 3 medidores han participado en las mediciones.

Tabla 9.1: Características de los experimentos que usaremos en el estudio de las diferencias entre medidores

Experimento	# Sujetos	# Medidores	# Mediciones	Diseño
FS	24	2	288	within-subjects
PT	17	2	132	within-subjects
DI	75	2	150	between-subjects
Todos los experimentos	116	3	570	-

9.2. Análisis exploratorio

9.2.1. Distribución de los datos

La Fig. 9.1 muestra los gráficos de densidad para FS, PT y DI (de izquierda a derecha, respectivamente). Puede observarse claramente que los gráficos correspondientes a

ambos medidores se superponen casi exactamente. Esto indica que las distribuciones de **las mediciones son muy similares intra-experimentos, y bastante parecidas entre-experimentos.**

Figura 9.1: QLTy density plot for FS, PT and DI

9.2.2. Correlaciones

La Tabla 9.2 muestra los resultados, divididos por experimentos. No se ha calculado una correlación conjunta debido a que existen varios medidores, y las correlaciones son bi-variadas. No obstante, la Tabla 9.3 muestra las correlaciones entre los medidores 1 y 2, y los medidores 1 y 3. En todos los casos, las correlaciones son muy altas [35] y estadísticamente significativas.

Tabla 9.2: Correlaciones entre medidores para los experimentos FS, PT y DI

Experiment	r	$p - value$
FS	0.84	<0.001
PT	0.75	<0.001
DI	0.91	<0.001

Tabla 9.3: Correlaciones entre los medidores 1, 2 y 3

Medidores	r	$p - value$
Medidores 1 y 2	0.86	<0.001
Medidores 1 y 3	0.86	<0.001

La Fig. 9.2 muestra los gráficos de dispersión para FS, PT y DI (de izquierda a derecha, respectivamente). Los puntos se encuentran relativamente alejados de la línea roja, que indica la concordancia perfecta entre medidores. No obstante, la recta de regresión (en azul) se acerca mucho a la línea roja. Todo ello sugiere que las mediciones individuales pueden variar, pero **la tendencia es que los medidores obtengan mediciones muy próximas entre sí.**

Figura 9.2: Gráficos de dispersión para FS, PT y DI (recta de regresión en azul, concordancia perfecta en rojo).

Tabla 9.4: Análisis de la variable respuesta QLTY para el experimento FS

	Measurer 1	Measurer 2
(Intercept)	26,66 (5,24)***	23,52 (4,98)***
TreatmentTDD	24,52 (3,58)***	24,75 (3,54)***
AIC	1337.48	1331.92
Num. obs.	144	144
Var: Subject (Intercept)	505.68	444.35
Var: Residual	462.58	451.35

*** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

9.2.3. Análisis estadístico de experimentos

La similitud entre mediciones sugerida en las secciones anteriores es confirmada por el análisis estadístico de los experimentos FS (ver Tabla 9.4), PT (Tabla 9.5) y DI (Tabla 9.6). Cada tabla presenta los resultados del análisis, divididos por medidor. La estructura de las tablas es distinta, debido a los diferentes diseños de los experimentos FS, PT y DI. No obstante, en todos los casos, los resultados de los análisis son similares entre medidores: los efectos no cambian de signo, y la significación estadística se mantiene, con pocas excepciones. Esto sugiere que **las mediciones realizadas sobre el mismo código por distintos medidores son muy similares entre sí**.

El único caso donde se observa una alteración significativa es en el factor *Tarea* en PT, que experimenta una busca caída en el tamaño de efecto entre los medidores 1 y 3, lo que a su vez produce que desaparezca la significación estadística. También se observan pequeños cambios en el *intercept*. Esto sin duda está motivado por los sesgos entre medidores, al igual que hemos observado anteriormente para las test suites en el Capítulo 8. Esto se comprobará utilizando los diagramas de Bland-Altman posteriormente.

Tabla 9.5: Análisis de la variable respuesta QLTY para el experimento PT

	Measurer 1	Measurer 3
(Intercept)	28,84 (5,05)***	23,87 (3,86)***
TreatmentTDD	-13,19 (4,10)**	-9,57 (3,21)**
TaskMR	15,16 (4,10)***	3,62 (3,21)
GroupMR → BSK	1,25 (6,25)	4,92 (4,73)
AIC	560.44	529.27
Num. obs.	66	66
Var: Group:Subject (Intercept)	94.50	51.02
Var: Residual	270.14	166.29

*** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

Tabla 9.6: Análisis de la variable respuesta QLTY para el experimento DI

	Measurer 1	Measurer 2
(Intercept)	47,76 (4,48)***	38,70 (4,61)***
TreatmentTL	-6,00 (6,47)	-4,11 (6,65)
R ²	.01	.01
Adj. R ²	-.00	-.01
Num. obs.	75	75

*** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

9.3. Análisis de reproducibilidad

Dado que el mismo medidor no ha medido el mismo objeto más de una vez, los datos usados para el análisis de reproducibilidad tienen la estructura de un experimento split-plot, donde cada código corresponde con un *whole plot*, y los medidores con un *subplot*.

La Tabla 9.7 muestra el resultado del análisis de reproducibilidad para todos los medidores conjuntamente. La fila "Var: Residual" representa la variabilidad adscribible al medidor, mientras que la fila "Var: Item (Intercept)" representa la variabilidad adscribible al código. La *varianza de reproducibilidad* se calcula a partir de la variabilidad adscribible al medidor y tiene el valor $S_R = \sqrt{131,17} = 11,45$, mientras que la *incertidumbre expandida* es $1,96 \times \sqrt{2} \times 11,45 = 31,75$. Esto significa que la diferencia entre dos mediciones tomadas por dos medidores oscilará, en el 95 % de los casos, en $\pm 31,75$ puntos porcentuales.

La Tabla 9.8 muestra el análisis indicado en la Eq. 8.1 para los experimentos FS, PT y DI, por separado. La *incertidumbre expandida* es ligeramente menor en los experimentos DI y PT que en FS. El promedio de las *incertidumbres expandidas* coincide *grosso modo* con la *incertidumbre expandida* global.

La información que el análisis de reproducibilidad proporciona es diametralmente opuesto a los resultados del análisis exploratorio mostrado en la Sección 9.2. En el análisis exploratorio, la impresión obtenida fue que las mediciones obtenidas por distintos medidores diferían sólo ligeramente. Sin embargo, el análisis de reproducibilidad muestra que

Tabla 9.7: Análisis conjunto de reproducibilidad

	Modelo de la Eq. 8.1
(Intercept)	38,43 (1,72)***
MeasurerMeasurer 2	-4,39 (1,08)***
MeasurerMeasurer 3	-8,47 (1,93)***
Var: Item (Intercept)	712.78
Var: Residual	131.17

*** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

Tabla 9.8: Análisis de reproducibilidad para los experimentos FS, PT y DI

Experimento	Varianza de reproducibilidad	Incertidumbre expandida
FS	13,15	$\pm 36,44$
PT	9,85	$\pm 27,3$
DI	8,54	$\pm 23,67$

las mediciones difieren en un grado considerable. Existen casos en los que dos mediciones sobre el mismo código pueden diferir en $\pm 31,75$ puntos porcentuales. Esta diferencia **es realmente considerable** pero, sin embargo, **no parece afectar al análisis** de los experimentos FS, PT y DI.

9.4. Gráficos de Bland-Altman

9.4.1. Gráfico conjunto de los tres medidores

El gráfico de Bland-Altman sólo está definido para la comparación de dos entidades (instrumentos, medidores, etc.). Cuando existen más de dos entidades (medidores en este caso), debemos acudir a formulaciones alternativas que, por cierto, son bastante escasas en la literatura.

En nuestro caso, utilizaremos la aproximación de Jones et al. [97]. El gráfico, mostrado en la Fig. 9.3, representa en el eje de abscisas la medida de las mediciones por cada entidad (código, en nuestro caso). En el eje de ordenadas se representan, dos a dos, todas las diferencias entre medidores. Esto es en todo similar al gráfico Bland-Altman original, aunque en este último sólo se pueden mostrar las diferencias entre dos medidores.

A diferencia del gráfico Bland-Altman, los puntos en la Fig. 9.3 están centrados en cero, y no en la media de diferencias entre dos medidores, ya que en el caso de más de dos medidores, dicha media no existe (por ejemplo, con tres medidores, existen tres posibles diferencias medias). Por esta misma razón, los límites del intervalo de confianza al 95 % son distintos, ya que están centrados en cero; sin embargo, el ancho del intervalo permanece inalterado.

Ya más concretamente, la Fig. 9.3 muestra con claridad que existen considerables diferencias entre mediciones, tal y como la *incertidumbre expandida* calculada en la Sección 9.3 indicaba. La mayoría de los puntos están concentrados en una franja de aproximadamente 20-25 puntos desde cero. Sin embargo, un número considerable de los mismos diverge en

Figura 9.3: Gráfico de Bland-Altman (modificado) para los tres medidores conjuntamente

mucha mayor medida, alcanzando diferencias de hasta 50 puntos.

Los límites del gráfico son $\pm 31,75$, y coinciden exactamente (salvo redondeos) con la *incertidumbre expandida* reportada anteriormente. Nótese que los límites del gráfico corresponden con los límites del intervalo de confianza al 95 %, por lo que una no pequeña fracción de los puntos pueden estar fuera (como es el caso) de dichos límites. No obstante, la mayor parte de los puntos están incluidos entre los límites $\pm 31,75$. Por este motivo, la Fig. 9.3 es perfectamente concordante, como no podía ser de otra manera, con el análisis de reproducibilidad reportado en la Sección 9.3.

9.4.2. Gráficos diferenciados por experimento

Decíamos anteriormente que la mayoría de los puntos están concentrados en una franja de aproximadamente 20-25 puntos desde cero, mientras que sólo una minoría está a mayores distancias. La razón es la existencia de distintas "poblaciones" de mediciones (los tres experimentos, y los tres medidores). La creación de gráficos diferenciados por experimento, mostrados en la Fig. 9.4, permite visualizar dichas "poblaciones" y su impacto en términos de *incertidumbre expandida*.

Los gráficos para FS, PT y DI difieren, aunque no excesivamente. En lo que respecta al sesgo, la Fig. 9.4 muestra que son muy pequeños. Los valores oscilan entre el 3 % y el 8 %. Estos sesgos corresponden aproximadamente con la variación del *intercept* en los análisis estadísticos reportados en la sección anterior.

En lo que respecta a la *incertidumbre expandida*, el gráfico de FS muestra las mayores divergencias entre mediciones. Estas divergencias, junto con otras más pequeñas pero observables en el gráfico de PT, son las que provocan esa minoría de mediciones con ~ 50 puntos de diferencia observados en la Fig. 9.4. Dichas divergencias también tienen su reflejo en la *incertidumbre expandida*, ya que el gráfico de FS es el que presenta los límites más amplios. Por el contrario, el gráfico de DI es el que presenta menores divergencias y límites más estrechos.

Figura 9.4: Gráficos de Bland-Altman para los experimentos FS, PT y DI

Los valores de los límites conciden, en todos los casos con la *incertidumbre expandida* reflejada en la Tabla 9.8. Nótese que las diferencias en los valores de la Tabla 9.8 y la Fig. 9.4 se deben a que los gráficos de Bland-Altman individuales no están centrados en cero. Simplemente sumando(/restando) la línea central de los gráficos Bland-Altman al límite correspondiente, se obtiene la *incertidumbre expandida* apropiada.

9.4.3. Estudio gráfico de posibles variables influyentes

Cabrá preguntarse si existe una razón, aparte de la propia aleatoriedad, por la cual las *incertidumbres expandidas* difieran entre FS, PT y DI. Los tres experimentos son repeticiones cercanas, pero difieren en algunos aspectos concretos:

- **Los medidores:** FS y DI fueron medidos por los medidores 1 y 2, mientras que PT por los medidores 1 y 3. Dado que FS posee la mayor *incertidumbre expandida*, y DI la menor, parece poco probable que éstas puedan achacarse a los medidores. Esto es especialmente importante para nuestros propósitos. Al menos en el conjunto de datos que manejamos, **no apreciamos que existan medidores mejores o peores**.
- **Las tareas experimentales:** FS y PT son experimentos *within* con dos tareas experimentales (MR y BSK), mientras que DI es un experimento *between* que sólo utiliza una tarea experimental (BSK). Podría ocurrir los medidores fueran sensibles a las tareas que miden. Para comprobar esta hipótesis, hemos generado gráficos de Bland-Altman para FS y PT, separando MR y BSK, los cuales se muestran en las Figs. 9.5 y 9.6.

En el caso de FS, la *incertidumbre expandida* es mayor en BSK que en MR. En PT, ocurre al contrario. **No se parece razonable concluir, por lo tanto, que *incertidumbre expandida* produzcan algún impacto en el resultado del análisis estadístico**. Sin embargo, llama la atención las diferencias entre sesgos en los gráficos de las Figs. 9.5 y 9.6. El sesgo de MR en PT (esto es, el valor de la línea central del gráfico de Bland-Altman) es el mayor de todos ($13,14 \gg \{3,14, 2,91, 1,45\}$). Además, **este sesgo posee un valor parecido a la variación del tamaño de efecto de MR** en el análisis estadístico del experimento PT (Véase la Sección 9.2.3).

- **Tipo de intervención:** Los experimentos FS y PT han sido medidos utilizando distintos tipos de intervención ("Sin intervención", "Sintáctica" y "Semántica"), mien-

tras que en el experimento DI se ha utilizado una medición "Sintáctico-Semántica" apoyada por protocolo. El protocolo podría hacer más uniformes las mediciones obtenidas por los medidores. Para comprobar esta hipótesis, hemos confeccionado gráficos comparando los medidores en función del tipo de intervención usado, los cuales se muestran en la Fig. 9.7.

Aunque no disponemos de medidas tomadas por los medidores 1 y 3 usando protocolo, parece apreciarse que cuanto menor sea la manipulación del código de los sujetos experimentales realizada por los medidores, menor la diferencia entre mediciones. Esto era esperable: la medición Semántica, que proporciona el mayor grado de libertad al medidor, posee una *incertidumbre expandida* más alta que la medición Sintáctica.

Adicionalmente, la existencia de protocolos de medida mejora la similitud entre mediciones. Nótese que tanto la medición Semántica como la Sintáctica están asociada a *incertidumbres expandidas* más altas que la medición Sintáctico-Semántica con protocolo.

- **Entrenamiento:** Los experimentos fueron realizados en el orden FS, PT y, finalmente, DI. Teniendo en cuenta que la *incertidumbre expandida* es mayor en FS que PT, no podemos descartar que la práctica adquirida por los medidores contribuya a mejorar la consistencia entre medidas.

Figura 9.5: Gráficos de Bland-Altman para el experimento FS, diferenciando las tareas MR y BSK

9.5. ICC

El ICC es $\rho = \frac{s_i^2}{s_m^2 + s_i^2}$, siendo s_m^2 la varianza adscribible a los medidores y s_i^2 la varianza adscribible a los ítems (/código). El ICC puede calcularse directamente usando los datos obtenidos durante en análisis de reproducibilidad en la Sección 9.3. De este modo, $\rho =$

Figura 9.6: Gráficos de Bland-Altman para el experimento PT, diferenciando las tareas MR y BSK

Figura 9.7: Gráficos de Bland-Altman por tipo de intervención

$\frac{712,78}{131,17+712,78} = 0,84$. Habitualmente, se considera que existe una alta concordancia entre medidores cuando $\rho \geq 0,75$ [55]. El valor de ICC que hemos obtenido supera este valor crítico, lo que implica que, desde el punto de vista del ICC, **las mediciones obtenidas por los distintos medidores son muy similares**.

9.6. Conclusiones

En esta sección expondremos los hallazgos que hemos obtenido en este capítulo. Para ello, nos apoyaremos también en los hallazgos del Capítulo 8 acerca de los efectos de la test suite en los resultados de la medición.

9.6.1. Respecto a la influencia del medidor

El análisis de reproducibilidad muestra que las mediciones realizadas por distintos medidores pueden variar sustancialmente. Sin embargo, dicha variación no tiene un impacto equivalente en el análisis estadístico de los experimentos. Las mediciones obtenidas por los distintos medidores están, en general, fuertemente correladas y se aproximan mucho a la bisectriz del primer cuadrante, que indica la concordancia perfecta.

Al menos en los experimentos FS, PT y DI que hemos utilizado como banco de pruebas, que un medidor u otro realice la medición no parece que afecte de modo relevante al análisis estadístico de los experimentos, independientemente de la *incertidumbre expandida* asociada a un par de medidores.

En PT se observa un fuerte sesgo negativo asociado a la *Tarea* MR. Pueden existir varios motivos para ello, pero no podemos aventurar cuál es correcto. Sea cual sea el motivo, pasa necesariamente por algún tipo de influencia en los medidores. El sesgo de MR en PT está asociado a mediciones sistemáticamente más altas realizadas por el "Measurer 1" (alternativamente, más bajas realizadas por el "Measurer 3")¹.

Afortunadamente, fuerte sesgo negativo asociado a la *Tarea* MR se reparte aleatoriamente entre *Tratamientos*, de la misma forma que en el Capítulo anterior dedicado al efecto de las test suites (ver Sección 8). Ello provoca que:

el mayor o menor rigor de los medidores hacia una *Tarea* concreta no ejerce impacto alguno sobre los *Tratamientos*, esto es, el factor *Tratamiento* no se ve afectado en el análisis estadístico de los experimentos.

Como medida precautoria adicional,

los medidores deberían estar cegados respecto al *Tratamiento* con el que cada código que miden ha sido generado.

9.6.2. Respecto a los tipos de análisis

El análisis de reproducibilidad y los gráficos de Bland-Altman muestran que las diferencias entre mediciones puede ser muy acusada, de 30 puntos o más. Sin embargo, una elevada *incertidumbre expandida* no implica necesariamente un impacto en el análisis estadístico de los experimentos. Parece más importante prestar atención al

sesgo entre medidores, ya sea general o existente sólo entre ciertos niveles de algún factor, ya que dicho sesgo se tralada de forma muy marcada al *intercept* o al tamaño de efecto de los niveles correspondientes.

¹En nuestra opinión, el sesgo entre mediciones observadas en PT se debe probablemente a un mayor rigor que el "Measurer 3" ha aplicado a la hora de medir. El hecho que se se circunscriba a MR es una prueba adicional, ya que el "Measurer 3" se encontraba especialmente incómodo midiendo esta tarea. No obstante, ésto es sólo una opinion subjetiva del autor.

9.6.3. Respecto a los tipos de interención

La evidencia respecto al impacto del tipo de intervención es limitada, debido a que ha sido estudiada en conjunción con el efecto de los medidores. Un análisis en profundidad de los tipos de intervención será realizado en el capítulo siguiente. No obstante, parece que:

La similitud entre mediciones aumenta cuando se limita la capacidad de los medidores para modificar el código de los sujetos (a efectos de medición), o bien cuando el rango de posibles modificaciones en el código está reflejado en un protocolo de actuación para los medidores.

Capítulo 10

Estudio de las diferencias entre tipos de intervención

10.1. Características generales de los datos

10.1.1. Experimentos

Para este estudio, utilizaremos dos experimentos realizados en industria (que denominaremos FS y PT para ocultar la identidad de las empresas). Estos experimentos están descritos en la Sección 4.3.

10.1.2. Mediciones

Las características generales de estos estudios se muestran en la Tabla 10.1. Ambos experimentos FS y PT han sido medidos por los medidores 1 y 2, utilizando los casos de prueba AH. En los dos experimentos se han utilizado los tres tipos de intervención: Sin intervención, sintáctica y semántica. En total, se han recabado 387 mediciones, realizadas sobre 129 códigos realizados por 41 sujetos experimentales.

Tabla 10.1: Características de los experimentos que usaremos en el estudio de las diferencias entre tipos de intervención

Experimento	# <i>Sujetos</i>	# <i>Códigos</i>	# <i>Intervenciones</i>	# <i>Medidores</i>	# <i>Mediciones</i>	Diseño
FS	24	96	3	2	288	Within
PT	17	33	3	1	99	Within
All experiments	41	129	3	2	387	-

10.2. Análisis exploratorio

10.2.1. Distribución de los datos

La Fig. 10.1 muestra la distribución de las mediciones, divididas por experimento, tal y como se ha hecho en secciones anteriores. Se observa claramente como el tipo de intervención "Sin intervención" genera valores de QLTY más bajos que los tipos de intervención "Sintáctico" y "Semántico". A su vez, el tipo de intervención "Sintáctico" produce valores

de QLTY menores que el "Semántico". Esto era esperable, ya que el tipo de intervención sintáctico permite realizar ligeras modificaciones en el código de los sujetos experimentales que, en una gran cantidad de casos, aumenta el número de aserciones pasadas. Lo mismo puede afirmarse del tipo de intervención semántico.

Es destacable la superposición de las distribuciones de las mediciones, sobre todo en FS. Si se confirma lo que hemos observado en anteriores gráficos de densidad (por ejemplo, el referido a los medidores en la Sección 9.2.1), las mediciones obtenidas utilizando los tres tipos de intervención **podrían ser bastante similares** y no ejercer efectos perniciosos sobre los análisis estadísticos de los experimentos FS y PT.

Figura 10.1: Diagrama de densidad para FS y PT

10.2.2. Correlaciones

La Tabla 10.2 muestra las correlaciones de Pearson y los correspondientes p-valores para los tres tipos de intervención. Las correlaciones son bastante altas en todos los casos, estadísticamente significativas, y numéricamente muy próximas. El menor valor ($r = 0,65$) corresponde con la correlación entre la intervención "Sin intervención" y la intervención "Semántica". Esta baja correlación confirma lo que ya se entreveía en la Fig. 10.1, respecto a la diferencia existente entre la distribución de mediciones para las intervenciones "Sin intervención" y "Semántica".

Tabla 10.2: Correlaciones entre tipos de intervención

	Sin intervención	Sintáctica
Sintáctica	$r = 0.72$ $p\text{-value} < 0.001$	-
Semántica	$r = 0.65$ $p\text{-value} < 0.001$	$r = 0.86$ $p\text{-value} < 0.001$

La prudencia sugiere que, además de calcular las correlaciones a nivel general, calculemos las correlaciones a nivel de experimento. Esto nos permitirá identificar si algún experimento posee características particulares que permitan, en su caso, explicar por qué diferentes mediciones producen distintos análisis estadísticos. Las Tablas 10.3 y 10.4 muestran las correlaciones entre tipos de intervención para FS y PT, respectivamente. Los valores de las correlaciones siguen el mismo patrón que en la Tabla 10.2, tanto en FS como PT. No obstante, las correlaciones son más bajas en PT que en FS. De nuevo, esto era también esperable, ya que el solapamiento entre las distribuciones de los tipos de intervención es menor para PT que para FS (véase Fig. 10.1).

Tabla 10.3: Correlaciones entre tipos de intervención para FS

	Sin intervención	Sintáctica
Sintáctica	$r = 0.76$	-
	$p\text{-value} < 0.001$	-
Semántica	$r = 0.71$	$r = 0.9$
	$p\text{-value} < 0.001$	$p\text{-value} < 0.001$

Tabla 10.4: Correlaciones entre tipos de intervención para PT

	Sin intervención	Sintáctica
Sintáctica	$r = 0.65$	-
	$p\text{-value} < 0.001$	-
Semántica	$r = 0.52$	$r = 0.77$
	$p\text{-value} = 0.0018$	$p\text{-value} < 0.001$

Las correlaciones por sí solas no son suficientes para determinar la similitud entre mediciones, tal y como ya hemos indicado anteriormente en la Sección 9.6. Los gráficos de dispersión han resultado bastante informativos a este respecto. La Fig. 10.2 muestra los gráficos de dispersión para los tres posibles pares de tipos de intervención. En general, las mediciones están muy dispersas. La dispersión es mayor cuando el tipo de intervención "Sin intervención" está implicado. No obstante, ello se debe exclusivamente a las calidades "cero" (véanse los puntos alineados verticalmente con el cero) que en muchos casos produce la medición "Sin intervención". Excluyendo los "ceros", la distribución de los puntos es similar en los tres casos, tal y como se observa en la Fig. 10.3. Es interesante observar que las rectas de regresión (líneas azules) no sufren variaciones sustanciales en ninguno de sus parámetros entre las Figs. 10.2 (con "ceros") y 10.3 (sin "ceros").

Existe un buen número de mediciones situadas sobre la línea roja (concordancia perfecta entre mediciones), pero una proporción al menos similar está considerablemente alejada de dicha línea roja. Esto produce que la recta de regresión (línea azul) se aparte considerablemente de la línea roja. El menor grado de ajuste se produce entre la intervención "Sin intervención" y la intervención "Semántica", como ya hemos indicado anteriormente.

La mayor parte de los puntos se localizan por encima de la línea roja (esto es, en la esquina superior izquierda del gráfico). Esto simplemente confirma lo que ya sugerían los diagramas de densidad, esto es, que el tipo de intervención "Semántico" produce valores de QLTY más altos que el tipo de intervención "Sintáctico", y éste a su vez produce valores mayores que "Sin intervención".

Los capítulos anteriores nos han mostrado que cuando existe un sesgo entre dos mediciones, los análisis estadísticos se ven alterados. En el caso de la comparación entre medidores (Capítulo 9), el sesgo negativo del "Measurer 3" produjo un cambio en el tamaño de efecto para la *Tarea*. El mismo escenario está ocurriendo, en este caso, debido a los sesgos de las mediciones entre tipos de intervención y, en consecuencia, **deberíamos observar un impacto parecido en los factores experimentales**. Quizás tenga lugar una excepción en el caso de los tipos de intervención "Sintáctica" y "Semántica" debido a la proximidad de la recta de regresión (línea azul) y la recta de concordancia perfecta (línea roja).

Figura 10.2: Correlaciones entre tipos de intervención

Figura 10.3: Correlaciones entre tipos de intervención, excluyendo ceros

Al igual que en capítulos anteriores, hemos creado gráficos de dispersión a nivel de experimento, los cuales se muestran en las Figs. 10.4 (FS) y 10.5 (PT). Las tendencias no se diferencian demasiado de los gráficos globales mostrados en la Fig. 10.2. Es apreciable que los intervalos de confianza son más anchos en PT (zona gris alrededor de la línea azul), y los coeficientes de determinación (R^2) más pequeños, lo que se traduce en rectas de regresión más horizontales que en FS. Esto está probablemente causado por el escaso número de sujetos experimentales (17 sujetos en PT, frente a 24 en FS), lo que empeora el ajuste de la recta de regresión.

Figura 10.4: Correlaciones entre tipos de intervención en FS

Figura 10.5: Correlaciones entre tipos de intervención en PT

10.2.3. Análisis estadístico de experimentos

Las Tablas 10.5 y 10.5 muestran los resultados de los análisis estadísticos para FS y PT, respectivamente, divididos por tipo de intervención. En el caso de FS, el único cambio apreciable es el aumento del valor del *intercept* a medida que el tipo de intervención se hace más complejo. En contrapartida, el efecto del tratamiento permanece esencialmente constante.

En PT, se puede observar el mismo incremento del *intercept*, pero los restantes factores también se ven alterados en cierta medida:

- El efecto del *Tratamiento* TDD disminuye a medida que el tipo de intervención se hace más complejo, aunque no se produce un cambio de signo. Asimismo, se observa un cambio en la significación estadística entre los tipos de intervención "Sin intervención" y "Sintáctico" - "Semántico". Esto representa una novedad; hasta el momento, el **Tratamiento no se había visto alterado**.
- La *Tarea* MR se comporta de una forma muy parecida al *Tratamiento* TDD.
- El *Grupo* $MR \rightarrow BSK$ pasa de tener un efecto negativo con el tipo de intervención "Sin intervención", a un valor positivo con los tipos de intervención "Sintáctico" y "Semántico".

Tabla 10.5: Análisis de la variable respuesta QLTY para el experimento FS

	Sin intervención	Sintáctica	Semántica
(Intercept)	14,87 (5,17)**	26,92 (5,32)***	33,47 (5,58)***
TreatmentTDD	25,28 (4,39)***	25,28 (3,82)***	23,34 (3,82)***
AIC	894.19	877.28	879.47
Num. obs.	96	96	96
Var: Subject (Intercept)	409.84	502.86	571.61
Var: Residual	462.63	351.05	349.42

*** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

Tabla 10.6: Análisis de la variable respuesta QLTY para el experimento PT

	Sin intervención	Sintáctica	Semántica
(Intercept)	50,69 (12,10)***	68,45 (11,86)***	83,33 (8,15)***
TreatmentTDD	-16,32 (12,71)	-25,03 (9,31)**	-22,59 (7,60)**
TaskMR	-13,95 (12,71)	-17,71 (9,31)	-23,63 (7,60)**
GroupMR → BSK	-6,73 (12,71)	17,96 (14,88)	16,49 (9,41)
AIC	312.43	308.10	288.85
Num. obs.	33	33	33
Var: Subject:Group (Intercept)	.00	571.16	130.52
Var: Residual	1317.53	695.42	467.84

*** $p < 0,001$, ** $p < 0,01$, * $p < 0,05$

La modificación del valor del *intercept* se debe, sin duda, a los sesgos entre tipos de intervención, como ya hemos observado anteriormente (véase, por ejemplo, el aumento del *intercept* en el experimento DI en el Capítulo 9). Estos sesgos son probablemente elevados (ello se podrá observar con precisión tanto en el análisis de reproducibilidad como en los diagramas de Bland-Altman) pero todos los diagramas de dispersión muestran cierta separación entre las líneas azul y roja), lo que a su vez provoca una variación del *intercept* en la misma proporción.

10.3. Análisis de reproducibilidad

Al igual que en los casos anteriores, cada código corresponde con un *whole plot*, y los tipos de intervención con *subplots*. El resultado del análisis se muestra en la Tabla 10.8. La *incertidumbre expandida* es $1,96 \times \sqrt{2} \times 16,96 = 47,02$, mucho más amplia que en el caso de los medidores y comparable a la *incertidumbre expandida* de las test suites.

La Tabla 10.8 muestra el análisis de reproducibilidad para FS y PT por separado. La *incertidumbre expandida* en FS es un tanto menor que en PT, aunque no demasiado. En términos absolutos, la *incertidumbre expandida* de FS es similar a la obtenida en el caso de los medidores (véase Tabla 9.8) y, como en aquel caso, el análisis estadístico no se ve afectado.

Tabla 10.7: Análisis conjunto de reproducibilidad

	Modelo de la Eq. ??
(Intercept)	28,78 (2,94) ^{***}
InterventionSyntactic	14,73 (2,11) ^{***}
InterventionSemantic	22,15 (2,11) ^{***}
Var: Item (Intercept)	828.01
Var: Residual	287.73

^{***} $p < 0,001$, ^{**} $p < 0,01$, ^{*} $p < 0,05$

La *incertidumbre expandida* de PT es bastante alta, comparable a la obtenida en el análisis de reproducibilidad de las test suites (véase Tabla 8.6). Por sí misma, la *incertidumbre expandida* no fue la causa principal de las alteraciones del análisis estadístico de los experimentos PT y EC en el Capítulo 8. La causa fundamental, en nuestra opinión, fue el tratamiento diferencial que los casos de prueba AD hoc y PCde realizan a las tareas MR y BSK. Es más; el único cambio observado en el análisis de los medidores en el capítulo 9 estuvo en relación a un sesgo de los medidores respecto a la tarea MR en PT. Este sesgo no venía asociado a una amplia *incertidumbre expandida*, sino a un sesgo diferencial de MR en PT.

En otras palabras: en análisis de reproducibilidad ha tenido un éxito limitado en determinar cuándo dos mediciones arrojan análisis estadísticos distintos. Estudiaremos inmediatamente a continuación, utilizando los diagramas Bland-Altman, si de nuevo son los sesgos, y no la *incertidumbre expandida*, la que permite explicar las alteraciones en los análisis estadísticos de los experimentos.

Tabla 10.8: Análisis de reproducibilidad para los experimentos FS y PT

Experimento	Varianza de reproducibilidad	Incertidumbre expandida
FS	14,66	$\pm 40,63$
PT	21,28	$\pm 58,97$

10.4. Gráficos de Bland-Altman

10.4.1. Gráfico conjunto de los tres tipos de intervención

Comenzaremos el análisis de Bland-Altman con un diagrama para los tres tipos de intervención conjuntamente, el cual se muestra en la Fig. 10.6. No resulta sencillo leer este diagrama debido a que, en este caso, los puntos aparecen muy agrupados. No obstante, es fácilmente observable un aspecto de "triángulo". Entre los valores 0 y 60 de QLTY, aproximadamente, las mediciones tienden a diverger, en especial las originadas con los tipos de intervención "Sin intervención" y "Sintáctica". Esto confirma visualmente la amplia *incertidumbre expandida* de los datos (47.02 puntos porcentuales).

Al existir más de dos tipos de intervención, el diagrama de la Fig. 10.6 no puede mostrar el sesgo. A este respecto, hay dos soluciones posibles: a) Utilizar los tamaños de efecto

reportados en la Tabla 10.7, que corresponden con los sesgos de cada tipo de intervención en relación a 'Sin intervención'. b) Construir diagramas de Bland-Altman para cada par de tipos de intervención por separado. La segunda de las opciones es preferible, ya que muestra tanto los sesgos entre tipos de intervención como las *incertidumbres expandidas*, y reduce además el número de puntos totales en cada diagrama, facilitando la interpretación.

Los diagramas para cada par de tipos de intervención por separado se muestran en la Fig. 10.7. Salta inmediatamente a la vista los fuertes sesgos entre los tipos de intervención "Sin intervención" y "Sintáctica" o "Semántica". Los valores de los sesgos son proporcionales (no necesariamente coincidentes) a las variaciones del *intercept* en las Tablas 10.5 y 10.6. La Fig. 10.7 también proporciona la *incertidumbre expandida* en cada caso. Puede apreciarse claramente que las mayores *incertidumbres expandidas* están asociadas a los pares "Sin intervención" - "Sintáctica" y "Sin intervención" - "Semántica", como va venimos indicando.

Figura 10.6: Gráfico de Bland-Altman (modificado) para los tres tipos de intervención conjuntamente

Figura 10.7: Gráficos de Bland-Altman para los tipos de intervención, a pares

10.4.2. Gráficos diferenciados por experimento

Siguiendo la misma operativa que en capítulos precedentes, hemos construido gráficos de Bland-Altman diferenciados por experimento, tal y como puede observarse en las Figs. 10.8 y 10.9. El análisis diferenciado por experimento es relevador, como viene siendo habitual:

- En el caso de FS, los tres gráficos de la Fig. 10.8 son relativamente uniformes. Se caracterizan por sesgos no demasiado elevados, e *incertidumbres expandidas* solapadas para cada par de tipos de intervención. Esto produce que:
 - Los sesgos se transmitan al *intercept* del análisis estadístico.
 - Dado que las mediciones **deberían** distribuirse aleatoriamente, el solapamiento entre los límites de acuerdo produce que los efectos de los factores en el análisis estadístico no se vean alterados.
- PT posee mayores sesgos que FS, e *incertidumbres expandidas* muy altas. Los gráficos de la Fig. 10.9 distan de ser parecidos entre sí.
 - Los sesgos se transmitan al *intercept* del análisis estadístico, aunque con menor precisión que en el caso de FS.
 - Los sesgos entre niveles de los factores produce variaciones en los tamaños de efecto.

Un aspecto adicional que se observa en las Figs. 10.8 y 10.9 es la distribución no uniforme de las mediciones. Compárense las Figs. 10.8 y 10.9 entre si, y ambas con las Figs. 9.4 y ???. Se observa claramente que en aquellos casos donde las mediciones se distribuyen alrededor de la línea central a lo largo de todo el eje de las X (esto es, Figs. 10.8 y 9.4), los análisis estadísticos no se ven afectados. Por el contrario, cuando ésto no ocurre (Figs. 10.9 y ???), los análisis estadísticos sí se ven afectados. Es posible que los cambios observados en la significación estadística estén relacionados con la aleatoriedad de la distribución de las diferencias entre mediciones.

Figura 10.8: Gráficos de Bland-Altman para los tipos de intervención, a pares, para el experimento FS

Figura 10.9: Gráficos de Bland-Altman para los tipos de intervención, a pares, para el experimento PT

10.4.3. Estudio gráfico de posibles variables influyentes

Las principales variaciones en el análisis estadístico del experimento PT afectan al factor *Grupo*. En todos los casos que hemos observado hasta el momento, esta variación siempre se ha explicado mediante un sesgo normalmente observable en el diagrama de Bland-Altman.

Las Figs. 10.10 y 10.11 muestran los gráficos Bland-Altman para cada uno de los grupos experimentales ($MR \rightarrow BSK$ y $BSK \rightarrow MR$) de PT por separado. Puede observarse claramente la diferencia de los sesgos entre los pares de intervenciones para $MR \rightarrow BSK$ y $BSK \rightarrow MR$. En $MR \rightarrow BSK$, los sesgos tienden a ser mucho más negativos que en $BSK \rightarrow MR$. Ello explica el efecto negativo de "Group $MR \rightarrow BSK$ " en el análisis de PT (véase Tabla 10.2.3).

Figura 10.10: Gráficos de Bland-Altman para los tipos de intervención, a pares, para el experimento PT y grupo $MR \rightarrow BSK$

10.5. ICC

El ICC es $\rho = \frac{s_i^2}{s_m^2 + s_i^2}$, siendo s_m^2 la varianza adscribible a los tipos de intervención y s_i^2 la varianza adscribible a los items (/código). Usando los datos obtenidos durante en

Figura 10.11: Gráficos de Bland-Altman para los tipos de intervención, a pares, para el experimento PT y grupo $BSK \rightarrow MR$

análisis de reproducibilidad, $\rho = \frac{828,01}{287,73+828,01} = 0,74$. Habitualmente, se considera que existe una alta concordancia entre medidores cuando $\rho \geq 0,75$ [55]. El valor de ICC que hemos obtenido prácticamente alcanza este valor crítico, lo que implica que, desde el punto de vista del ICC, **las mediciones obtenidas por los distintos medidores son muy similares**.

10.6. Conclusiones

En esta sección expondremos los hallazgos que hemos obtenido en este capítulo. Para ello, nos apoyaremos también en los hallazgos de capítulos anteriores.

10.6.1. Respecto a la influencia del tipo de intervención

Modificar el código de los sujetos experimentales para conectar de forma adecuada las test suites parecía, en principio, una operación fuertemente subjetiva que debería provocar fuertes sesgos, amplias incertidumbres y, en consecuencia, variaciones en el análisis estadístico de los experimentos. Sin embargo, nuestros resultados indican lo contrario. Los tipos de intervención más subjetivos ("Sintáctico" y "Semántico") son razonablemente parecidos entre sí, a diferencia del tipo "Sin intervención", que genera mediciones muy distintas a las dos anteriores. En consecuencia:

Las mediciones realizadas con tipos de intervención "Sintáctica" y "Semántica" pueden usarse para realizar mediciones. El tipo de intervención "Sin intervención" debe evitarse.

Nótese que la utilización de los tipos de intervención "Sintáctica" y "Semántica" también producen que el *Tratamiento* no se vea afectado. Esto representa una razón adicional para usar dichos tipos de intervención, y no el tipos de intervención "Sin intervención".

10.6.2. Respecto a los tipos de análisis

En este capítulo hemos entendido mejor el impacto del sesgo en el análisis estadístico:

- El sesgo global, que afecta a todos los niveles de todos los factores de un experimento, se transmite de forma directa al *intercept* del modelo lineal usado para análisis.
- Los sesgos diferenciales (ej: que ocurren en MR, pero no en BSK) se transmiten al factor, aumentando o disminuyendo el tamaño de efecto del nivel correspondiente.

El diagrama de Bland-Altman es el instrumento con el que más fácilmente se aprecian los sesgos globales o entre niveles de los factores, aunque el análisis de reproducibilidad puede producir exactamente los mismos resultados.

Finalmente,

hemos sido **incapaces de identificar** qué impacto posee una mayor o menor *incertidumbre expandida* en el resultado del análisis estadístico de los experimentos.

Posiblemente, no tenga ninguno.

Capítulo 11

Discusión

11.1. Resultados principales

Los análisis realizados en los Capítulos 8-10 han puesto de manifiesto que los componentes de los métodos de medición (en nuestro caso: las test suites, medidores, y tipos de intervención) ejercen distinta influencia en los resultados de las mediciones, así como en los resultados de los análisis estadísticos. La Tabla 11.1 resume los principales hallazgos obtenidos.

En primer lugar, debemos discutir **qué componentes de los métodos de medición pueden suponer un riesgo para la validez** de los análisis estadísticos de experimentos, que es uno de los objetivos principales de la presente Tesis Doctoral. Al menos en TDD, que es el área a la cual pertenecen los experimentos utilizados en los análisis, nuestras observaciones indican que:

- Las test suites ejercen un gran impacto tanto en las mediciones como en los análisis estadísticos de experimentos.
- El tipo de intervención "Sin intervención" ejerce un gran impacto tanto en las mediciones, y un impacto más limitado en los análisis estadísticos de experimentos.
- Los medidores y los tipos de intervención "Sintáctica" y "Semántica" son relativamente inocuos.

En otras palabras:

que la medición de los resultados de un experimento sea realizado por uno u otro medidor, realizando distintos tipos de manipulación en el código de los sujetos con la finalidad de conectar la test suite de medición, **no cambia las conclusiones del experimento.**

Es conveniente indicar que el impacto de las test suites y del tipo de intervención "Sin intervención" **está limitado a experimentos *within-subjects***. En todos los casos que hemos analizado, los problemas con las mediciones y posterior análisis surgen de un comportamiento diferencial de las test suite y medidores respecto a las tareas MR y BSK. La utilización de dos tareas sólo es necesaria en experimentos *within-subjects*, donde los

sujetos aplican varios tratamientos sobre distintas tareas. En el caso de los experimentos *between-subjects*, donde sólo existe una tarea experimental, el comportamiento diferencial no es posible, por lo que los efectos perniciosos de las test suites y del tipo de intervención "Sin intervención" se desvanecen.

También es importante resaltar que, por la misma razón indicada anteriormente, los tratamientos (por ejemplo: ITLD, TDD) no se ven afectados por los componentes de los métodos de medición. Un mismo código, ya sea perteneciente a MR o a BSK, puede ser desarrollado usando ITLD o TDD. Cualquier influencia en las mediciones de MR o BSK, *al menos en principio*, deberían distribuirse equitativamente entre los tratamientos ITLD y TDD, resultado éstos inalterados en el análisis estadístico. Por consiguiente:

aunque la mayoría de los experimentos en TDD no reporten detalles respecto a los test cases usados, medidores, o tipos de intervención, sus resultados respecto a los tratamientos utilizados (ITLD, TDD, u otros) **son probablemente fiables** (al menos, en lo que respecta a la influencia del método de medición).

11.2. Métodos de análisis de diferencias entre mediciones

A lo largo de la presente Tesis Doctoral, hemos utilizado una diversidad de métodos de análisis de mediciones. Muchos de ellos son equivalentes (análisis de reproducibilidad, diagramas de Bland-Altman, ICC), y proporcionan una medida de la varianza entre mediciones que puede ser después utilizada para calcular límites de acuerdo (esto es, la *incertidumbre de reproducibilidad*). Otros, como la correlación de Pearson y los diagramas de dispersión apoyados por las rectas de correlación, proporcionan una información ligeramente distinta, como es el grado de variación conjunta de las mediciones.

El método que más efectivo ha resultado para determinar cuándo dos mediciones distintas pueden provocar cambios en los análisis estadísticos de experimentos ha sido el diagrama de Bland-Altman. Sin embargo, de forma bastante sorprendente, la *incertidumbre de reproducibilidad* no resulta útil, mientras que los sesgos son mucho más informativos. En honor a la verdad, los métodos indicados anteriormente no están diseñados para identificar el impacto de las mediciones en los análisis estadísticos de experimentos. Sin embargo, nosotros esperábamos que aspectos como la *incertidumbre expandida* tuviesen un reflejo más o menos directo en el análisis estadístico. Estábamos, obviamente, equivocados, ya que:

el análisis estadístico no se ve afectado cuando los componentes del método de medición se comportan de forma consistente respecto a todos los niveles de todos los factores. Esto puede comprobarse estudiando la reproducibilidad de las mediciones separadamente para cada nivel de cada factor (no globalmente, o a nivel de experimento).

Dos ejemplos ayudarán a entender la afirmación anterior. Sabemos que el tipo de intervención "Sintáctica" es más rigurosa que el tipo de intervención "Semántica", lo que provoca que los valores de QLTY sean menores en el primer caso que en el segundo. Esto es perfectamente visible en la Fig. 10.2 del Capítulo 10. Pues bien; esta diferencia no produce una modificación del efecto de los factores en el análisis estadístico, debido a que

todos los códigos, independientemente de si se han desarrollado con ITLD o TDD, o si implementan MR o BSK, están igualmente afectados por los bajos valores de *QLTY*. El único impacto en el análisis estadístico se manifiesta en el *intercept*, que sube o baja en proporción al sesgo entre mediciones, el cual se muestra en la Fig. 10.7.

Por el contrario, el "Measurer 3", durante la medición de PT, ha sido un tanto más riguroso que el "Measurer 1" durante la medición de MR, como se puede observar en la Fig. ?? del Capítulo 9. Esto provoca que el efecto de MR disminuya en proporción al sesgo, como se muestra en la Fig. 9.6.

En ambos casos, la *incertidumbre expandida* no juega ningún papel relevante.

Obviamente, existe una razón matemática que justifica la afirmación y ejemplos anteriores. Un modelo lineal, que es la técnica estadística habitualmente utilizada en el análisis de experimentos, puede interpretarse como una suma de vectores en un espacio n -dimensional [20, p.137–138]. En nuestro caso, los vectores serían *QLTY* (el resultado de la suma) y *Tratamiento*, *Tarea* y *Grupo* (los sumandos). Estos vectores son perfectamente visibles cuando se construye la descomposición matricial de conjunto de datos (véase un ejemplo en [20, p.1148]), y dicha suma es lo que se refleja en ecuación que define todos los modelos lineales (por ejemplo, la Eq. 8.1). En consecuencia:

si dos conjuntos de mediciones X e Y pueden relacionarse linealmente como $Y = aX + b$, la utilización de una medición y otra no cambia el resultado del análisis estadístico, ya que la aplicación de una transformación lineal a los vectores no modifica la relación entre los mismos. El único cambio que se puede producir es el re-escalamiento del *intercept* y de los tamaños de efecto en una proporción $aX + b$.

El problema surge cuando las mediciones no se relacionan linealmente de una única forma, esto es, el caso del comportamiento diferencial de las test suite y medidores respecto a las tareas MR y BSK indicado anteriormente. En ese caso, ciertas coordenadas de los vectores se ven alteradas de modo distinto a otras, cambiando completamente el resultado de la suma y, por ende, de los resultados del análisis.

Naturalmente, tampoco podemos olvidar en este punto el grado de ajuste (el coeficiente de determinación R^2 de la relación lineal entre mediciones:

Cuando el coeficiente de determinación R^2 entre dos mediciones es muy bajo (alrededor de 0.40, según nuestras observaciones), el análisis estadístico puede variar, aunque exista una relación lineal consistente entre todos los pares de mediciones.

En este último caso, la relación lineal entre mediciones es tan inexacta que el error se transmite al análisis estadístico de forma no predecible.

11.3. Generalización

Cualquier generalización se debe realizar con gran prudencia, y más todavía en nuestro caso, ya que sólo hemos analizado datos provenientes de unos pocos experimentos en TDD. No obstante, se pueden realizar algunas inferencias que, probablemente, tienen aplicabilidad general:

- La medición utilizado test suites está automatizada en gran parte. Los medidores deben conectar el código de los sujetos con las test suites utilizando una serie de tipos de intervención predefinidos, pero una vez completado este paso, la medición se realiza automáticamente. Este proceso es equivalente, *grosso modo*, a una medición en química: se prepara la muestra a medir, y se introduce en un aparato especial de medición. Aunque los tipos de intervención (al igual que la preparación de la muestra en química) implican una cierta subjetividad, ésta se mantiene dentro de ciertos límites. Por lo tanto, recomendamos que independientemente del área científica particular:

el método de medición debe automatizarse lo máximo posible.

- La automatización se consigue, en los experimentos en TDD, mediante la utilización de test suites. Sin embargo, tal y como se ha reportado en el Capítulo 8, las test suites son el componente del método de medición que más problemas ha generado debido a su comportamiento diferencial respecto a las tareas MR y BSK. La pregunta sería: ¿puede evitarse dicho comportamiento diferencial?

A priori, opinamos que si puede evitarse, de dos formas posibles:

- Realizando un experimento piloto, que permita comprobar la reproducibilidad de las mediciones y, en su caso, ajustar las test suites.
- Realizando un análisis de reproducibilidad utilizando código de referencia. Este código podría consistir en variantes de las tareas MR y BSK, con ciertas *user stories* implementadas y otras no.

La segunda de las opciones parece preferible, por tres motivos:

1. Requiere menor esfuerzo.
2. Permite ejercer un mayor control sobre las mediciones.
3. No es necesario comparar dos mediciones al existir un valor de referencia (la QLTY asociada al código de referencia) disponible.

En otras palabras:

el instrumento de medición (las test suites) no puede construirse en base a fundamentos teóricos exclusivamente, sino que tiene que ser validado empíricamente, mediante un análisis de reproducibilidad.

- Un modo de mejorar la uniformidad de las mediciones pasa necesariamente por eliminar la subjetividad en la aplicación de los tipos de intervención. Como hemos indicado anteriormente en la Sección 11.1, la intervención "Sin intervención" posee disfunciones, por lo que esto no es una opción. Ignoramos si este es el caso en otras áreas, pero es muy probable. Los exámenes representan un símil creíble: ¿Cuántas

veces hemos calificado inicialmente con cero el ejercicio de un estudiante que, en una lectura más profunda, es satisfactorio o incluso brillante? ¿O cuántas veces hemos rechazado un artículo en una revisión que, finalmente, ha sido aceptado? El mero examen superficial no parece la mejor estrategia de evaluación posible.

En el área de TDD, una forma de eliminar la subjetividad del medidor pasaría por obligar a los sujetos a respetar el API de las tareas experimentales. Las modificaciones del API son, en gran medida, el origen de los tipos de intervención. Sin embargo, obligar a un sujeto a respetar un API con el que no está de acuerdo es cuestionable. Sería quizás preferible limitar el acceso de los medidores al código de los sujetos experimentales, permitiéndoles acceso únicamente al API. Esto pondría coto a las decisiones subjetivas que pudieran tomar.

En general, lo que estamos sugiriendo es:

aplicar ocultamiento en la realización de mediciones. Los medidores deberían ser agnósticos frente al código de los sujetos (o cualesquiera producto resultante de las sesiones experimentales), el tratamiento aplicado o, incluso, los resultados de medición. Esto minimizaría la subjetividad de la medición y la aparición de sesgos.

- En la Sección 11.1 concluíamos que *el análisis estadístico no se ve afectado cuando los componentes del método de medición se comportan de forma consistente respecto a todos los niveles de todos los factores*. Una forma mucho más sencilla de comprobar si el análisis estadístico se ve afectado sería realizar los análisis utilizando las distintas mediciones y comparar los resultados. Sin embargo:

desaconsejamos que los experimentadores tomen decisiones *post-hoc*. Cualquier decisión debe preceder al análisis estadístico de los resultados (excluyendo, naturalmente, el estudio del ajuste de los modelos usando AIC o BIC, por citar dos ejemplos).

Tabla 11.1: Impacto de los elementos estudiados

Test	Relación entre medidas	Impacto en el análisis estadístico	Motivos de los cambios en el análisis estadístico	¿Pueden usarse indiferentemente en experimentos?
Test suites	Existen amplias divergencias:	Importante:	Dos motivos:	Si, en experimentos <i>between-subjects</i> donde sólo se utilice una <i>Tarea</i> (MR o BSK). En experimentos <i>within-subjects</i> no puede confiarse en ningún caso en los resultados del factor <i>Tarea</i> ni ninguna de sus interacciones.
	a) Sesgos $\sim 30\%$.	a) Fuertes variaciones en el <i>intercept</i> .	a) La test suite basada en PE asigna QLTY más bajas que AH. Esto produce la disminución del <i>intercept</i> .	
	b) Incertidumbres expandidas $\sim 60\%$.	b) Fuertes variaciones en el factor <i>Tarea</i> y <i>Grupo</i> .	b) La test suite basada en PE penaliza a BSK más que MR. Esto produce que el efecto de MR (en comparación con la línea base de BSK) aumente.	
	c) $R^2 \sim 50\%$.	c) El factor <i>Tratamiento</i> no se ve afectado.	c) La medición de MR utilizado casos de prueba AH y PE no es equivalente, lo que provoca cambios de menor entidad en el factor <i>Grupo</i> .	

Tabla 11.1 – viene de la página anterior

Relación entre medidas		Impacto en el análisis estadístico	Motivos de los cambios en el análisis estadístico	¿Pueden usarse indiferentemente en experimentos?
Medidor	Las divergencias no son tan acusadas:	Prácticamente nulo.	En el experimento PT, el "Measurer 3" se comporta de forma muy estricta midiendo MR.	Si, en experimentos <i>between-subjects</i> medidos por un sólo medidor. En experimentos <i>between-subjects</i> medidos por un sólo medidor existe un moderado riesgo de que los resultados del factor <i>Tarea</i> y sus interacciones estén sesgados.
	a) Sesgos $\sim 10\%$.	Sólo la <i>Tarea</i> MR disminuye ligeramente de valor en PT.	En lo que respecta a BSK, el "Measurer 1" y el "Measurer 3" se comportan de forma similar. Esto provoca una disminución (sesgo) del valor de MR respecto a BSK que se refleja de igual modo en el análisis estadístico.	
	b) Incertidumbres expandidas $\sim 30\%$.			
	c) $R^2 > 0,60$, generalmente.			
Tipo de intervención	Existen amplias divergencias:	Importante:	Dos motivos:	Los tipos de intervención "Sintáctica" y "Semántica" pueden ser usados en experimentos <i>between-subjects</i> y <i>within-subjects</i> . El tipo de intervención "Sin intervención" no debería ser usado hasta que se comprenda mejor.
	a) Sesgos $\sim \{14\%, 22\%\}$, según el caso.	a) Fuertes variaciones en el <i>intercept</i> .	a) El sesgo causado por los tipos de intervención se traslada al <i>intercept</i> , al igual que en el caso de las test suites.	
	b) Incertidumbres expandidas $\sim \{40\%, 60\%\}$, según el caso.	b) Fuertes variaciones en el factor <i>Grupo</i> .	b) Las variaciones en el factor <i>Grupo</i> ocurre únicamente en el tipo de intervención "Sin intervención". Su origen no está claro.	
	c) En algún caso, $R^2 < 50\%$.	c) El resto de factores no se ven afectados.		

Capítulo 12

Amenazas a la validez

En este capítulo, discutimos las amenazas a la validez de esta investigación, de acuerdo con [157]. Dado que la presente Tesis Doctoral no consiste en un experimento o quasi-experimento, deberemos realizar una cierta adaptación de los conceptos de validez.

12.1. Validez de la conclusión

La validez de la conclusión define en qué medida las conclusiones obtenidas son estadísticamente válidas. En este contexto, debería entenderse como *la objetividad con que las conclusiones se derivan de los datos analizados*.

Hemos tomado diversas precauciones para asegurar la validez interna:

- Para asegurar la validez de conclusión, hemos utilizaron varios métodos de comparación, en particular los conceptos establecidos en el estándar ISO 5725, y demás estándares relacionados
- Las conclusiones están basadas en casi un millar de mediciones tomadas en más de 1,100 mediciones distintas.
- La captura y análisis de datos se ha automatizado en gran medida para evitar problemas de transcripción y manipulación. Por ejemplo, las mediciones se obtienen automáticamente mediante un software especial integrado en Eclipse.

12.2. Validez interna

La validez interna hace referencia a la capacidad de obtener conclusiones acerca del efecto de la variable independiente sobre la dependiente. En nuestro caso, debería entenderse como *el grado en que las conclusiones obtenidas (por ejemplo, el fuerte impacto de las test suites en el análisis estadístico) se derivan de realmente de las mediciones realizadas*.

Hemos tomado diversas precauciones para asegurar la validez interna:

- Todas las mediciones se han tomado de forma independiente, a partir del código fuente original creado por los sujetos experimentales. Esto es, no se ha reutilizado código de tal forma que se puedan producir dependencias no deseadas entre mediciones.

- Los medidores han actuado independientemente, sin que existiese comunicación entre ellos.
- Para las mediciones, el autor empleó sesiones de 8 horas de medición durante dos años. Normalmente las sesiones fueron alternadas con revisión de lectura científica / escritura de la tesis. Los restantes medidores trabajaron a tiempo parcial. Por este motivo, es poco probable que se hayan producido efectos de fatiga.
- Los datos utilizados en la presente tesis no son las primeras mediciones obtenidas por los medidores 1 y 2. Con anterioridad, dichos medidores ya habían medido otros experimentos. PT fue el primer experimento medido por el medidor 3. Esto podría contribuir a las diferencias entre las mediciones del medidor 1 y 3 descritas en el Capítulo 9.

No obstante, la mayoría de las mediciones usadas en esta tesis han sido adquiridas por los medidores 1 y 2. Por ello, es poco probable que se hayan producido efectos de aprendizaje que puedan haber alterado las medidas, aunque ello no pueda descartarse totalmente, como hemos indicado en el Capítulo 9.

12.3. Validez de constructo

La validez de constructo determina en qué grado las variables representan correctamente los constructos teóricos. Esto es, la validez de constructo determina en qué medida las conclusiones obtenidas están ligadas a las operacionalizaciones de las variables independientes y dependientes utilizadas, y no a los constructos subyacentes.

Hemos tomado diversas precauciones para asegurar la validez de constructo:

- En el presente estudio, hemos estudiado las diferencias entre las mediciones de la variable respuesta Calidad. Sólo hemos reportado la segunda, para evitar ser reiterativos, pero conviene reiterar que las conclusiones son similares.
- Para cada componente del método de medición, esto es, las variables independientes, estudiamos distintas variantes:
 - Dos tipos de estrategias de construcción de casos de prueba.
 - Cuatro medidores distintos.
 - Tres tipos de intervención.

12.4. Validez externa

La validez externa hace referencia a la posibilidad de generalización de los resultados de una investigación a otras poblaciones, situaciones o momentos temporales.

En general, las investigaciones y, en especial, las Tesis Doctorales poseen un carácter muy local y son de difícil generalización. No obstante, hemos intentado mejorar la validez externa de la forma siguiente:

- Para incrementar la posibilidad de generalización de resultados hemos trabajado con mediciones de experimentos realizados por sujetos profesionales representativos de la industria.

- Hemos trabajado con suites de casos de prueba representativas, elaboradas por personas distintas. La primera test suite llamada Ad hoc se utilizó en la medición de experimentos en la industria correspondiente al proyecto ESEIL. La segunda suite fue elaborada en una trabajo de fin de Master utilizando la técnica PCdE bajo la supervisión de uno de los Directores de esta tesis.
- Hemos trabajado con medidores representativos: tres estudiantes de Doctorado con experiencia en el área del software y una estudiante de Master.

Capítulo 13

Conclusiones y futuras líneas

En este capítulo se presentan las conclusiones obtenidas de la investigación y las futuras líneas de investigación. El objetivo de la investigación ha sido estudiar el impacto de los componentes de los métodos de medición de la variable respuesta calidad en el valor de las mediciones y en el resultado de los análisis estadístico de experimentos en TDD.

13.1. Visión general

Durante el proceso experimental, la operacionalización de las variables respuesta de los experimentos es muy dependiente del investigador. Cada investigador operacionaliza a su manera, sin que existan protocolos o recomendaciones acerca de cómo llevar a cabo el el proceso de operacionalización.

La presente Tesis Doctoral se enmarca en la investigación experimental en TDD. Después de revisar la literatura científica relacionada, hemos podido constatar que la inmensa mayoría de los trabajos experimentales no ofrecen información detallada sobre la forma en que han operacionalizado sus variables respuesta. Un aspecto especialmente importante que está ausente en la mayoría de reportes experimentales es el método de medición. Por ejemplo, los experimentos de TDD no indican el modo en que las test suites de medición han sido construidas. Asimismo, no es habitual que dichas test suites estén disponibles para verificación independiente. Como es esperable dada esta situación, otros detalles típicamente obviados en muchos experimentos de TDD e IS en general, tales como el medidor y el tipo de intervención, no sin ni siquiera mencionados.

Con el propósito de estudiar el impacto de los componentes del método de medición en el análisis estadístico de experimentos, hemos utilizado como banco de pruebas la familia de experimentos en TDD generada en el marco del proyecto ESEIL FiDiPro. Estos experimentos, realizados tanto en industria como en academia, nos han permitido contar con las mediciones para llevar a cabo la investigación.

13.2. Actividades

Los experimentos del proyecto ESEIL no se diferenciaban, desde el punto de vista del protocolo de operacionalización, de otros experimentos en TDD disponibles en la literatura. Al no existir un método de medición, iniciamos la investigación definiendo uno, tomando como referencia los procedimientos utilizados en otros experimentos realizados en el seno

del grupo de investigación. Este método de medición ha sido definido y contrastado con los procedimientos utilizados en otros experimentos en TDD con el propósito de validarlo para otros experimentos distintos a los de la familia ESEIL.

En el método de medición se identificaron un conjunto de elementos relevantes, tales como los test suites, entorno de desarrollo, medidores, tipo de intervención en el código fuente, métrica, framework de pruebas, compiladores de código, entre otros. De todos ellos seleccionamos los elementos que creemos tienen mayor influencia en los resultados de las mediciones y que han sido objeto de nuestro estudio: test suites, medidores, y tipo de intervención en código fuente.

El objetivo de la investigación ha sido determinar cómo influyen los componentes del método de medición utilizados en los resultados de la medición y el análisis estadístico de los experimentos. Para lograr este objetivo, realizamos un comparaciones entre distintas alternativas de cada componente, utilizando diversas estrategias: correlaciones, análisis de reproducibilidad, gráficos de Bland-Altman, e ICC. De este modo determinamos las diferencias entre mediciones, su origen, y el impacto de dichas diferencias en los resultados del análisis estadístico de los experimentos.

- Para estudiar las test suites, realizamos las mediciones de dos experimentos con dos suites de casos de prueba: Ad hoc y PCdE. Las suites de casos de prueba difieren entre sí en la técnica con que fueron elaborados. La primera suite fue desarrollada para la necesidad de medir los experimentos y para ello no se utilizó ninguna técnica específica (Ad hoc). La segunda suite fue elaborada mediante la técnica de particiones de clases de equivalencia. Para su estudio, mantenemos fijo el medidor y el tipo de intervención.
- Durante el desarrollo de la investigación se determinó que las mediciones podrían realizarse con tres tipos de intervención en el código fuente de los sujetos experimentales. Las intervenciones son cambios que hace el sujeto para favorecer la conexión de las test suites con el código:
 - Cuando el medidor entra en el código del sujeto y hace cambios menores para favorecer la conexión con las test suites, denominamos a este tipo de intervención como intervención sintáctica.
 - Cuando el medidor considera la intención del sujeto experimental y hace cambios más importantes en el código, denominamos a este tipo de intervención como intervención semántica.
 - Por último una medición también puede realizarse sin realizar ninguna intervención en el código fuente.

Para realizar el análisis de la influencia del tipo de intervención en el código fuente hemos aplicado estas intervenciones en la medición de dos experimentos, manteniendo fijo el medidor y las test suites.

- En fases iniciales de la investigación, en un primer experimento se realizó la operacionalización con dos medidores y se identificaron distintos resultados en ciertas mediciones puntuales. Estas diferencias observadas en un inicio motivaron a estudiar la influencia del medidor. Se consideró que el criterio del medidor a la hora de hacer las intervenciones en el código podría influir en los resultados de la medición.

Se contó con la participación de tres medidores para realizar las mediciones de los experimentos:

- Inicialmente, los medidores mantuvieron fijo las test suites y los tipo de intervención.
- En una segunda fase, se realizó la medición de un experimento con dos medidores utilizando un protocolo de medición con el propósito de que los medidores sigan unos pasos y ciertas consideraciones para hacer las mediciones, con el propósito de analizar la influencia de los medidores usando un protocolo de medición.

13.3. Resultados

Los resultados que hemos obtenido indican, como era esperable, que cada componente del método de medición ejerce una influencia distinta en las mediciones y en el resultado de los análisis estadísticos de experimentos:

- Las test suites impactan fuertemente en tanto en las mediciones como en el resultado de los análisis estadísticos de experimentos. Afortunadamente, el impacto se circunscribe a factores relacionados con la tarea experimental. El tratamiento experimental, en la medida en que es independiente de la tarea experimental, no se ve afectado.
- Los medidores no poseen demasiada influencia en las mediciones ni en el resultado de los análisis estadísticos de experimentos.
- Los tipos de intervención sintáctico y semántico impactan fuertemente en las mediciones, pero no afectan al resultado de los análisis estadísticos de experimentos. La "No intervención" si tiene impacto en el análisis estadístico, aunque al igual que las test suites, no parece afectar al tratamiento.
- La utilización de un protocolo para la ejecución de un tipo de intervención en la práctica mejora la similitud entre mediciones tomadas por distintos medidores.

Aunque se deduce de lo indicado anteriormente, es conveniente destacar que en ninguno de los casos estudiados, el tratamiento (ITLD o TDD) se ha visto afectado por los componentes del método de medición. Esto implica que, con bastante seguridad, podemos confiar en los resultados de los experimentos en TDD publicados en la literatura. Esta confianza aumenta incluso cuando los experimentos son *between-subjects*, ya que este diseño es muy resistente a cualquier influencia indeseada de los componentes del método de medición.

13.4. Recomendaciones generales

El mayor problema que hemos detectado concierne a las test suites. En un principio, parecía que el origen del problema era el modo de construcción Ad hoc de los casos de prueba, problema que sería solucionado con un modo de construcción más formal basado en PCdE.

Sin embargo, estábamos equivocados. Por una parte, un modo determinado de construcción de las test suites no es intrínsecamente malo, sino su correspondencia con las

expectativas del investigador y del sujeto experimental. Si los investigadores y sujetos experimentales no esperan que se controlen explícitamente los casos excepcionales en el código, una test suite basada en PCdE arrojará resultados mucho mas bajos que una test suite Ad hoc construida ignorando dichos casos excepcionales. Por otra, es muy sencillo que las test suites no respeten una propiedad básica de las métricas conocida como *representational condition*: códigos con mayores funcionalidades debe arrojar valores de QLTY proporcionalmente mayores. Esto no ocurre en general con la test suite basada en PCdE.

La principal recomendación surgida de esta investigación es que las test suites (y los instrumentos de medición, en general), no pueden ser diseñados atendiendo a consideraciones teóricas únicamente, y deben ser verificados en la práctica, a ser posible en el marco de un análisis de reproducibilidad.

Debemos confesar que ha resultado sorprendente constatar el reducido impacto de los medidores y los tipos de intervención, con la excepción de la "No intervención", en el análisis estadístico. Para ser fieles a la verdad, esperábamos justamente lo contrario. Esto libera de mucha presión a los investigadores, ya que no necesitan establecer controles específicos para asegurar la validez de los datos recabados por uno u otro medidores aplicando algún tipo de manipulación en el código de los sujetos experimentales. En cualquier caso, aconsejamos que se aplique ocultamiento de los medidores respecto al código de los sujetos, resultados de medición y tratamientos, con la finalidad de evitar decisiones subjetivas. La utilización de protocolos para los tipos de intervención es beneficiosa para aumentar la similitud de las mediciones y, por lo tanto, también es aconsejable.

13.5. Líneas futuras de investigación

La presente investigación deja muchos flancos abiertos, que han sido imposibles de afrontar por falta de tiempo y recursos. Citaremos sólo las líneas futuras más importantes, algunas de los cuales esperamos abordar en el futuro inmediato:

- Las recomendaciones que hemos proporcionado se basan en una inferencia a partir de los resultados obtenidos. Sin embargo, como esta misma Tesis Doctoral muestra claramente, las inferencias son falibles (véase el caso de las test suites, por ejemplo). Para hacer más robustos los hallazgos de la tesis, sería necesario repetir este estudio con otra familia de experimentos, preferiblemente en un área distinta de IS.
- Nos ha resultado imposible crear un esbozo de protocolo del proceso de operacionalización de variables respuesta para experimentos en IS. Ni siquiera hemos podido realizar un estudio teórico acerca del mismo.
- Queda pendiente determinar cómo un experimentador puede decidir si un instrumento de medición (por ejemplo, la test suite Ad hoc, o la test suite PCdE) es fiable. Los pasos del análisis de reproducibilidad para un instrumento de medición deberían también protocolizarse.
- Hemos recomendado el ocultamiento para disminuir la subjetividad de los medidores cuando aplican distintos tipos de intervención. Sin embargo, otro camino distinto para mejorar la validez de las mediciones sería utilizar distintos medidores, tipos de intervención e incluso test suites, y promediar los resultados ya sea mediante medias simples o ponderadas.

Bibliografía

- [1] 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.*, pages 227–243. Springer US, 2005.
- [2] M. Aniche and M. A. Gerosa. Does test-driven development improve class design? a qualitative study on developers’ perceptions. *Journal of the Brazilian Computer Society*, 21(1):15, 2015.
- [3] 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 (ICSTW)*, pages 469–478. IEEE, 2010.
- [4] 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 (SBES)*, pages 1–10. IEEE, 2012.
- [5] S. Bannerman and A. Martin. A multiple comparative study of test-with development product changes and their effects on team speed and product quality. *Empirical Software Engineering*, 16(2):177–210, 2011.
- [6] E. G. Barriocanal, M.-A. S. Urbán, I. A. Cuevas, and P. D. Pérez. An experience in integrating automated unit testing practices in an introductory programming course. *SIGCSE Bull.*, 34(4):125–128, 2002.
- [7] V. R. Basili. Software modeling and measurement: the goal/question/metric paradigm. *Encyclopedia of Software Engineering*, 1994.
- [8] V. R. Basili and H. D. Rombach. The TAME project: towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, 1988.
- [9] V. R. Basili, R. W. Selby, and D. H. Hutchens. Experimentation in software engineering. *IEEE Transactions on Software Engineering*, (7):733–743, 1986.
- [10] V. R. Basili, F. Shull, and F. Lanubile. Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25(4):456–473, 1999.

- [11] V. R. Basili and A. J. Turner. Iterative enhancement: A practical technique for software development. *IEEE Transactions on Software Engineering*, SE-1(4):390–396, 1975.
- [12] K. Beck. Aim, fire. *IEEE Software*, 18(5):87–89, 2001.
- [13] K. Beck. *Test-driven development: by example*. Addison-Wesley Professional, 2003.
- [14] K. Beck and C. Anders. *Extreme programming explained: embrace change*. Addison-Wesley Professional, 1999.
- [15] J. Beckstead. Agreement, reliability, and bias in measurement: Commentary on Bland and Altman (1986; 2010). *International Journal of Nursing Studies*, 48(1):134–135, 2011.
- [16] V. S. Bhadauria. *To test before or to test after—An experimental investigation of the impact of Test Driven Development*. PhD thesis, The University of Texas at Arlington, 2009.
- [17] 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*, pages 356–363. ACM, 2006.
- [18] 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*, 74:45 – 54, 2016.
- [19] J. M. Bland and D. G. Altman. Statistical methods for assessing agreement between two methods of clinical measurement. *The Lancet*, 327(8476):307 – 310, 1986.
- [20] G. E. Box, J. S. Hunter, and W. G. Hunter. *Estadística para investigadores: Diseño, innovación y descubrimiento*. Reverté, 2008.
- [21] L. Briand, K. E. Emam, and S. Morasca. On the application of measurement theory in Software Engineering. *Empirical Software Engineering*, 1(1):61–88, 1996.
- [22] British Computer Society BCS SIGIST. *Standard for Software Component Testing*. Number Draft 3.4. 2001.
- [23] R. E. Brooks. Studying programmer behavior experimentally: The problems of proper methodology. *Commun. ACM*, 23(4):207–213, 1980.
- [24] M. Bunge. *Buscar la filosofía en las ciencias sociales*. Siglo XXI, 1999.
- [25] 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*, ISESE '06, pages 364–371. ACM, 2006.
- [26] 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.*, pages 383–388. Springer, 2006.
- [27] L. Cao and B. Ramesh. Agile requirements engineering practices: An empirical study. *IEEE Software*, 25(1):60–67, 2008.
- [28] A. Cauevic, S. Punnekkat, and D. Sundmark. Quality of testing in Test Driven Development. In *2012 Eighth International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 266–271, Sept 2012.
- [29] A. Cauevic, S. Punnekkat, and D. Sundmark. TDDHQ: Achieving higher quality testing in Test Driven Development. In *2013 39th Euromicro Conference on Software Engineering and Advanced Applications*, pages 33–36, Sept 2013.
- [30] A. Causevic, R. Shukla, and S. Punnekkat. Industrial study on Test Driven Development: Challenges and experience. In *Proceedings of the 1st International Workshop on Conducting Empirical Studies in Industry, CESI '14*, pages 15–20. IEEE Press, 2013.
- [31] 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*, pages 337–346, March 2011.
- [32] A. Causevic, D. Sundmark, and S. Punnekkat. Test case quality in Test Driven Development: A study design and a pilot experiment. In *16th International Conference on Evaluation Assessment in Software Engineering (EASE 2012)*, pages 223–227, May 2012.
- [33] L.-R. Chien, D. J. Buehrer, C.-Y. Yang, and C.-M. Chen. An evaluation of TDD training methods in a programming curriculum. In *IEEE International Symposium on IT in Medicine and Education, 2008. ITME 2008.*, pages 660–665, Dec 2008.
- [34] M. Ciolkowski, F. Shull, and S. Biffl. A family of experiments to investigate the influence of context on the effect of inspection techniques. In *Proceedings of the 6th International Conference on Empirical Assessment in Software Engineering, EASE*, pages 48–60, 2002.
- [35] J. Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Routledge, 1988.
- [36] Columbia Center for Teaching and Learning. http://ccnmtl.columbia.edu/projects/qmss/measurement/validity_and_reliability.html, visitado 12 de Enero de 2017.
- [37] L. Crispin. Driving software quality: How Test-driven Development impacts software quality. *IEEE Software*, 23(6):70–71, 2006.
- [38] L.-O. Damm and L. Lundberg. Results from introducing component-level test automation and Test-Driven Development. *Journal of Systems and Software*, 79(7):1001 – 1014, 2006.

- [39] L.-O. Damm and L. Lundberg. 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*, pages 187–199. Springer, 2007.
- [40] P. Davies. *Mind of God: The Scientific Basis for a Rational World*. Simon and Schuster, 1993.
- [41] C. Desai, D. Janzen, and K. Savage. A survey of evidence for Test-Driven Development in academia. *SIGCSE Bull.*, 40(2):97–101, 2008.
- [42] 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, SIGCSE '09*, pages 148–152. ACM, 2009.
- [43] O. Dieste, A. M. Aranda, F. Uyaguari, B. Turhan, A. Tosun, D. Fucci, M. Oivo, and N. Juristo. Empirical evaluation of the effects of experience on code quality and programmer productivity: an exploratory study. *Empirical Software Engineering*, pages 1–86, 2017.
- [44] T. Dogša and D. Batič. The effectiveness of Test-Driven Development: an industrial case study. *Software Quality Journal*, 19(4):643–661, 2011.
- [45] M. A. Domino, R. W. Collins, and A. R. Hevner. Controlled experimentation on adaptations of pair programming. *Information Technology and Management*, 8(4):297–312, 2007.
- [46] 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*, SIGMIS CPR '03, pages 44–51. ACM, 2003.
- [47] 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*, volume 3, 2003.
- [48] S. H. Edwards. Using software testing to move students from trial-and-error to reflection-in-action. *SIGCSE Bull.*, 36(1):26–30, 2004.
- [49] Entregables del Proyecto FIDIPRO:Eseil. <https://sites.google.com/site/fidiproeseil/documents>, visitado 01 de Noviembre de 2016.
- [50] H. Erdogmus, M. Morisio, and M. Torchiano. On the effectiveness of the Test-First approach to programming. *IEEE Transactions on Software Engineering*, 31(3):226–237, 2005.
- [51] M. Feinberg. *Chemometrics in Environmental Chemistry – Applications*. Springer-Verlag Berlin Heidelberg, 1995.

- [52] N. Fenton. Software measurement: a necessary scientific basis. *IEEE Transactions on Software Engineering*, 20(3):199–206, 1994.
- [53] N. Fenton and J. Bieman. *Software metrics: a rigorous and practical approach*. CRC Press, 2014.
- [54] M. Ficco, R. Pietrantuono, and S. Russo. Bug localization in test-driven development. *Adv. Soft. Eng.*, 2011:2:1–2:18, 2011.
- [55] J. Fleiss. *Design and Analysis of Clinical Experiments*. Wiley, 2011.
- [56] J. L. Fleiss. *Design and analysis of clinical experiments*. John Wiley & Sons, 1986.
- [57] T. Flohr and T. Schneider. An XP experiment with students – setup and problems. In *Product Focused Software Process Improvement: 6th International Conference, PROFES 2005, Oulu, Finland, June 13-15, 2005. Proceedings*, pages 474–486. Springer Berlin Heidelberg, 2005.
- [58] T. Flohr and T. Schneider. Lessons learned from an XP experiment with students: Test-first needs more teachings. In *Product-Focused Software Process Improvement: 7th International Conference, PROFES 2006, Amsterdam, The Netherlands, June 12-14, 2006. Proceedings*, pages 305–318. Springer Berlin Heidelberg, 2006.
- [59] D. Fucci, G. Scanniello, S. Romano, M. Shepperd, B. Sigweni, F. Uyaguari, B. Turhan, N. Juristo, and M. Oivo. An external replication on the effects of Test-Driven Development using a multi-site blind analysis approach. In *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '16*, pages 3:1–3:10. ACM, 2016.
- [60] D. Fucci and B. Turhan. A replicated experiment on the effectiveness of Test-First Development. In *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '13*, pages 103–112. ACM, Oct 2013.
- [61] D. Fucci and B. Turhan. On the role of tests in test-driven development: a differentiated and partial replication. *Empirical Software Engineering*, 19(2):277–302, 2014.
- [62] 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*, 68:82 – 97, 2015.
- [63] D. Fucci, B. Turhan, and M. Oivo. Impact of process conformance on the effects of test-driven development. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14*, pages 10:1–10:10. ACM, 2014.
- [64] 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, EASE '15*, pages 25:1–25:6. ACM, 2015.

- [65] B. George. *Analysis and Quantification of Test-Driven Development Approach*. Master's thesis. North Carolina State University, 2002.
- [66] B. George and L. Williams. An initial investigation of Test Driven Development in industry. In *Proceedings of the 2003 ACM Symposium on Applied Computing, SAC '03*, pages 1135–1139. ACM, 2003.
- [67] B. George and L. Williams. A structured experiment of Test-Driven Development. *Information and Software Technology*, 46(5):337 – 342, 2004.
- [68] A. Geras. *The effectiveness of Test-Driven Development*. Master's thesis. University of Calgary, 2004.
- [69] A. Geras, M. Smith, and J. Miller. A prototype empirical evaluation of Test Driven Development. In *Proceedings. 10th International Symposium on Software Metrics, 2004.*, pages 405–416, Sept 2004.
- [70] W. W. Gibbs. Software's chronic crisis. *Scientific American*, 271(3):72–81, 1994.
- [71] E. Grant and H. Sackman. An exploratory investigation of programmer performance under on-line and off-line conditions. *IEEE Transactions on Human Factors in Electronics*, HFE-8(1):33–48, 1967.
- [72] 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, ESEM 2007*, pages 285–294, Sept 2007.
- [73] K. Gwet. *Handbook of Inter-Rater Reliability. The Definitive Guide to Measuring the Extent of Agreement Among Raters*. Advanced Analytics, LLC, 4 edition, 2014.
- [74] A. M. J. Hass. *Guide to advanced software testing*. Artech House, 2008.
- [75] M. Hilton, N. Nelson, H. McDonald, S. McDonald, R. Metoyer, and D. Dig. TDDViz: Using software changes to understand conformance to Test Driven Development. In *Agile Processes, in Software Engineering, and Extreme Programming: 17th International Conference, XP 2016, Edinburgh, UK, May 24-27, 2016, Proceedings*, pages 53–65. Springer International Publishing, 2016.
- [76] 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: 10th International Conference, XP 2009, Pula, Sardinia, Italy, May 25-29, 2009. Proceedings*, pages 33–42. Springer Berlin Heidelberg, 2009.
- [77] L. Huang and M. Holcombe. Empirical assessment of test-first approach. In *Testing: Academic Industrial Conference - Practice And Research Techniques (TAIC PART'06)*, pages 197–202, Aug 2006.
- [78] L. Huang and M. Holcombe. Empirical investigation towards the effectiveness of test first programming. *Information and Software Technology*, 51(1):182 – 194, 2009.

- [79] 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, 2007. TAICPART-MUTATION 2007*, pages 82–88, Sept 2007.
- [80] International Standards Organization. *Accuracy (trueness and precision) of measurement methods and results – Parts 1 to 6*. Number ISO 5725. 1994.
- [81] International Standards Organization. *Statistics-Vocabulary and Symbols – Part 1: General statistical terms and terms used in probability*. Number ISO 3534-1. 2014.
- [82] ISO. *ISO/IEC 9126-1: Information Technology - Software product quality - Part 1: Quality model*, 2000.
- [83] ISO. *ISO/IEC 9126-2: Software Engineering - Product quality - Part 2: External metrics*, 2002.
- [84] ISO. *ISO/IEC 25010: Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*, 2011.
- [85] D. Janzen and H. Saiedian. Does Test-Driven Development really improve software design quality? *IEEE Software*, 25(2):77–84, 2008.
- [86] D. Janzen and H. Saiedian. Test-driven learning in early programming courses. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '08, pages 532–536. ACM, 2008.
- [87] D. S. Janzen. *An empirical evaluation of the impact of test-driven development on software quality*. PhD thesis, University of Kansas, 1993.
- [88] D. S. Janzen and H. Saiedian. Test-driven development: Concepts, taxonomy, and future direction. *Computer*, 38(9):43–50, 2005.
- [89] 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)*, pages 141–148, April 2006.
- [90] D. S. Janzen and H. Saiedian. Test-driven learning: Intrinsic integration of testing into the cs/se curriculum. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '06, pages 254–258. ACM, 2006.
- [91] D. S. Janzen and H. Saiedian. A leveled examination of Test-Driven Development acceptance. In *29th International Conference on Software Engineering (ICSE'07)*, pages 719–722, May 2007.
- [92] 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)*, pages 89–96, July 2007.
- [93] R. Jeffries and G. Melnik. Guest editors' introduction: TDD—the art of fearless programming. *IEEE Software*, 24(3):24–30, 2007.

- [94] Joint Committee for Guides in Metrology. *Evaluation of measurement data – Guide to the expression of uncertainty in measurement*. Number JCGM 100:2008. 2008.
- [95] Joint Committee for Guides in Metrology. *International vocabulary of metrology – Basic and general concepts and associated terms*. Number JCGM 200:2012. 2012.
- [96] C. G. Jones. Test-driven Development goes to school. *J. Comput. Sci. Coll.*, 20(1):220–231, 2004.
- [97] M. Jones, A. Dobson, and S. O’Brian. A graphical method for assessing agreement with the mean between multiple observers using continuous measures. *International Journal of Epidemiology*, 40(5):1308, 2011.
- [98] N. Juristo and A. M. Moreno. *Basics of software engineering experimentation*. Kluwer Academic Publishers, 2001.
- [99] S. H. Kan. *Metrics and models in software quality engineering*. Addison-Wesley Longman Publishing Co., Inc., 2002.
- [100] A. Kaplan. *The conduct of inquiry*. Transaction Publishers, 1973.
- [101] R. Kaufmann and D. Janzen. Implications of test-driven development: A pilot study. In *Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA ’03, pages 298–299. ACM, 2003.
- [102] K. Keefe, J. Sheard, and M. Dick. Adopting xp practices for teaching object oriented programming. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52*, ACE ’06, pages 91–100. Australian Computer Society, Inc., 2006.
- [103] 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*, ICSE ’06, pages 544–550. ACM, 2006.
- [104] S. Kollanus. Test-driven development - still a promising approach? In *2010 Seventh International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 403–408, Sept 2010.
- [105] 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*, ITiCSE ’08, pages 124–127. ACM, 2008.
- [106] 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*, Koli ’08, pages 25–31. ACM, 2008.
- [107] N. Laranjeiro and M. Vieira. Extending Test-Driven Development for robust web services. In *Second International Conference on Dependability, 2009. DEPEND ’09.*, pages 122–127, June 2009.

- [108] C. Larman and V. R. Basili. Iterative and incremental developments. a brief history. *Computer*, 36(6):47–56, 2003.
- [109] R. Latorre. Effects of developer experience on learning and applying Unit Test-Driven Development. *IEEE Transactions on Software Engineering*, 40(4):381–395, 2014.
- [110] R. Latorre. A successful application of a Test-Driven Development strategy in the industrial environment. *Empirical Software Engineering*, 19(3):753–773, 2014.
- [111] L. Layman, L. Williams, and L. Cunningham. Motivations and measurements in an agile case study. *Journal of Systems Architecture*, 52(11):654 – 667, 2006.
- [112] J. Lee, D. Koh, and C. Ong. Statistical evaluation of agreement between two methods for measuring a quantitative variable. *Computers in Biology and Medicine*, 19(1):61 – 70, 1989.
- [113] N. F. LeJeune. Teaching Software Engineering practices with Extreme Programming. *J. Comput. Sci. Coll.*, 21(3):107–117, 2006.
- [114] 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: 5th International Conference, XP 2004, Garmisch-Partenkirchen, Germany, June 6-10, 2004. Proceedings*, pages 219–222. Springer Berlin Heidelberg, 2004.
- [115] 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.*, page 113, 2005.
- [116] 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*, pages 278–289. Springer Berlin Heidelberg, 2006.
- [117] 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*, pages 207–221. Springer Berlin Heidelberg, 2007.
- [118] L. Madeyski. Impact of Pair Programming on thoroughness and fault detection effectiveness of unit test suites. *Software Process: Improvement and Practice*, 13(3):281–295, 2008.
- [119] L. Madeyski. The impact of Test-First programming on branch coverage and mutation score indicator of unit tests: An experiment. *Information and Software Technology*, 52(2):169 – 184, 2010.
- [120] L. Madeyski and Ł. Szala. 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*, pages 200–211. Springer Berlin Heidelberg, 2007.

- [121] 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: 6th International Conference, SWQD 2014, Vienna, Austria, January 14-16, 2014. Proceedings*, pages 155–169. Springer International Publishing, 2014.
- [122] 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*, pages 13–22. Springer Berlin Heidelberg, 2009.
- [123] C. Martin. *Advanced Principles, Patterns and Process of Software Development*. Prentice Hall, 2001.
- [124] E. M. Maximilien and L. Williams. Assessing test-driven development at ibm. In *Proceedings. 25th International Conference on Software Engineering, 2003*, pages 564–569, May 2003.
- [125] K. McDaid, A. Rust, and B. Bishop. Test-driven development: Can it work for spreadsheets? In *Proceedings of the 4th International Workshop on End-user Software Engineering, WEUSE '08*, pages 25–29. ACM, 2008.
- [126] 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.*, pages 481–488, May 2005.
- [127] A. T. Misirli, H. Erdogmus, N. Juristo, and O. Dieste. Topic selection in industry experiments. In *Proceedings of the 2Nd International Workshop on Conducting Empirical Studies in Industry, CESI 2014*, pages 25–30. ACM, 2014.
- [128] V. B. Mišić. Perceptions of Extreme Programming: An exploratory study. *SIGSOFT Softw. Eng. Notes*, 31(2):1–8, 2006.
- [129] S. Mondal and P. P. Das. Effectiveness of Test-Driven Development as an SDLC Model: A case study of an elevator controller design. In *Emerging Trends in Computing and Communication: ETCC 2014, March 22-23, 2014*, pages 225–233. Springer India, 2014.
- [130] M. M. Muller and O. Hagner. Experiment about Test-First programming. *IEE Proceedings - Software*, 149(5):131–136, 2002.
- [131] M. M. Müller and A. Höfer. The effect of experience on the Test-Driven Development process. *Empirical Software Engineering*, 12(6):593–615, 2007.
- [132] M. M. Muller and W. F. Tichy. Case study: Extreme Programming in a university environment. In *Proceedings of the 23rd International Conference on Software Engineering, 2001. ICSE 2001*, pages 537–544, May 2001.
- [133] R. Müller and P. Büttner. A critical discussion of intraclass correlation coefficients. *Statistics in Medicine*, 13:2465–2476, 1994.

- [134] H. Munir, M. Moayyed, and K. Petersen. Considering rigor and relevance when evaluating Test Driven Development: A systematic review. *Information and Software Technology*, 56(4):375–394, 2014.
- [135] H. Munir, K. Wnuk, K. Petersen, and M. Moayyed. An experimental evaluation of Test Driven Development vs. Test-last Development with industry professionals. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, EASE '14, pages 50:1–50:10. ACM, 2014.
- [136] 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*, 13(3):289–302, 2008.
- [137] L. W. Neuman. *Social research methods: Qualitative and quantitative approaches*. Pearson Education Limited, 2002.
- [138] M. Pancur and M. Ciglaric. Impact of Test-Driven Development on productivity, code and tests: A controlled experiment. *Information and Software Technology*, 53(6):557 – 573, 2011.
- [139] 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*, volume 2, pages 83–86 vol.2, Sept 2003.
- [140] W. P. Paula Filho. Quality Gates in Use-case Driven Development. In *Proceedings of the 2006 International Workshop on Software Quality*, WoSQ '06, pages 33–38. ACM, 2006.
- [141] F. Pavese. ISO 5725:1994. In *European Network for Business and Industrial Statistics pre-Conference Workshop on "Measuring Uncertainty in the GUM (VIM) vs ISO 5725 (ISO 3534) Approaches: Which Statistical Viewpoint to Choose?*, ENBIS-15, 2009.
- [142] J. Percival and N. Harrison. Developer perceptions of process desirability: Test Driven Development and Cleanroom Compared. In *2013 46th Hawaii International Conference on System Sciences (HICSS)*, pages 4800–4809, Jan 2013.
- [143] H. Peters, C. Knieke, O. Brox, S. Jauns-Seyfried, M. Krämer, and A. Schulze. A Test-Driven Approach for Model-Based Development of powertrain functions. In *Agile Processes in Software Engineering and Extreme Programming: 15th International Conference, XP 2014, Rome, Italy, May 26-30, 2014. Proceedings*, pages 294–301. Springer International Publishing, 2014.
- [144] S. L. Pfleeger. Experimental design and analysis in software engineering. *Annals of Software Engineering*, 1(1):219–253, 1995.
- [145] Y. Rafique and V. B. Misic. The effects of Test-Driven Development on external quality and productivity: A meta-analysis. *IEEE Transactions on Software Engineering*, 39(6):835–856, 2013.

- [146] S. M. Rahman. Applying the TBC method in introductory programming courses. In *2007 37th Annual Frontiers In Education Conference - Global Engineering: Knowledge Without Borders, Opportunities Without Passports*, pages T1E–20–T1E–21, Oct 2007.
- [147] A. Rendell. Effective and pragmatic Test Driven Development. In *Agile Conference, 2008. AGILE '08*, pages 298–303, Aug 2008.
- [148] A. P. Ress, R. de Oliveira Moraes, and M. S. Salerno. Test-Driven Development as an innovation value chain. *Journal of Technology Management and Innovation*, 8:10 – 10, 02 2013.
- [149] R. Robins, R. Fraley, and R. Krueger. *Handbook of Research Methods in Personality Psychology*. Guilford Publications, 2009.
- [150] P. Rodríguez, J. Markkula, M. Oivo, and K. Turula. Survey on agile and lean usage in finnish software industry. In *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '12*, pages 139–148. ACM, 2012.
- [151] S. Romano, D. Fucci, G. Scanniello, B. Turhan, and N. Juristo. Results from an ethnographically-informed study in the context of Test Driven Development. In *Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering, EASE '16*, pages 10:1–10:10. ACM, 2016.
- [152] I. Salman. *Students versus Professionals as Experiment Subjects: An Investigation on the Effectiveness of TDD on Code Quality*. Master’s Thesis. University of Oulu, Finland, 2014.
- [153] O. Salo and P. Abrahamsson. An iterative improvement process for agile software development. *Software Process: Improvement and Practice*, 12(1):81–100, 2007.
- [154] 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*, pages 5–14, Aug 2007.
- [155] P. Sfetsos, L. Angelis, and I. Stamelos. Investigating the Extreme Programming System—an empirical study. *Empirical Software Engineering*, 11(2):269–301, 2006.
- [156] P. Sfetsos, L. Angelis, and I. Stamelos. Prioritized Test-Driven reverse engineering process: A case study. In *2015 6th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pages 1–6, July 2015.
- [157] W. Shadish, T. Cook, and D. Campbell. *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin Company, 2002.
- [158] M. Shaw. Prospects for an engineering discipline of software. *IEEE Software*, 7(6):15–24, 1990.
- [159] M. Shaw. Continuing prospects for an engineering discipline of software. *IEEE Software*, 26(6):64–67, 2009.

- [160] W. Shelton, N. Li, P. Ammann, and J. Offutt. Adding criteria-based tests to Test Driven Development. In *2012 IEEE Fifth International Conference on Software Testing, Verification and Validation*, pages 878–886, April 2012.
- [161] L. B. Sherrell and J. J. Robertson. Pair programming and agile software development: Experiences in a college setting. *J. Comput. Sci. Coll.*, 22(2):145–153, 2006.
- [162] 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*, ESEM 2007, pages 275–284, Sept 2007.
- [163] 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*, pages 143–156. Springer Berlin Heidelberg, 2008.
- [164] 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.*, pages 214–223, Oct 2008.
- [165] J. Spacco and W. Pugh. Helping students appreciate Test-driven Development (TDD). In *Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications*, OOPSLA '06, pages 907–913. ACM, 2006.
- [166] W. Trochim and J. Donnelly. *The Research Methods Knowledge Base*. Cengage Learning, 2006.
- [167] B. Turhan, L. Layman, M. Diep, H. Erdogmus, and F. Shull. How effective is Test-Driven Development. In *Making Software: What Really Works, and Why We Believe It*, pages 207–217. OReilly Press, 2010.
- [168] I. Turnu, M. Melis, A. Cau, A. Setzu, G. Concas, and K. Mannaro. Modeling and simulation of open source development using an agile practice. *Journal of Systems Architecture*, 52(11):610 – 618, 2006.
- [169] 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*, pages 229–234, April 2009.
- [170] 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.
- [171] 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*, 38(3):547–560, 2012.

- [172] 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.*, pages 34–45, Nov 2003.
- [173] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Springer Science & Business Media, 2012.
- [174] S. Xu and T. Li. Evaluation of Test-Driven Development: An academic case study. In *Software Engineering Research, Management and Applications 2009*, pages 229–238. Springer, Berlin, Heidelberg, 2009.
- [175] S. Yenduri and L. A. Perkins. Impact of using Test-Driven Development: A case study. In *Software Engineering Research and Practice*, pages 126–129, 2006.
- [176] 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*, pages 84–87, 2001.
- [177] 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*, pages 211–212. Springer Berlin Heidelberg, 2006.

Apéndice A

Estudios empíricos en TDD

Describimos cada una de las columnas de la tabla sobre estudios empíricos en TDD que se presenta a continuación: la primera columna contiene el código ID, el cual está compuesto por el prefijo PS y un número secuencial ([PS1]..[PS105]). En la segunda columna está el apellido del primer autor del estudio junto con el año de publicación, la tabla está ordenada por esta columna. En las columnas posteriores está el número de referencia del estudio usado en cada revisión sistemática. En la siguiente columna está el tipo de estudio, de acuerdo a lo que dice cada autor. En la penúltima columna tenemos el contexto académico o industrial y finalmente el nombre de la revista o conferencia en donde ha sido publicado el artículo.

Tabla A.1: Estudios empíricos en TDD

ID	Autor	Bi [18]	Mu [134]	Ma [121]	Ca [32]	Ko [104]	Tu [167]	De [41]	Ra [145]	Tipo estudio (autor)	Academia/ Industria	Revista / conferencia
[PS1]	Abrahamsson2005 [1]				[8]					Case study	Industria	Chapter Book. Business Agility and Information Technology Diffusion Volume 180, Springer
[PS2]	Aniche2010 [3]	[40]	[72]							Questionnaire	Academia e Industria	ICSTW 2010, IEEE Computer Society.
[PS3]	Aniche2012 [4]	[38]								Questionnaire	Industria	SBES 12, IEEE Computer Society.
[PS4]	Bannerman2011 [5]		[60]							Case study	Estudios de la industria y academia	Empirical Software Engineering 2011, Springer.
[PS5]	Barriocanal2002 [6]							[2]		Experience Report	Academia	SIGCSE Bulletin 2002, ACM.
[PS6]	Bhadauria2009 [16]									Experiment	Academia	PhD thesis, The University of Texas at Arlington
[PS7]	Bhat2006 [17]	[33]	[64]	[12]	[9]	[9]				Case study	Industria	ISESE'06, ACM.
[PS8]	Canfora2006 [25]	[32]	[43]	[13]	[11]	[37]	X	[40]		Controlled experiment	Industria	ISESE'06, ACM
[PS9]	Canfora2006b [26]				[10]	[36]				Experiment	Industria	Chapter. Product-Focused Software Process Improvement. Volume 4034 of the series Lecture Notes in Computer Science pp 383-388, Springer
[PS10]	Cao2008 [27]				[12]					Case study	Industria	Software, IEEE , vol.25.
[PS11]	Causevic2012 [32]									Experiment	Academia	EASE 2012, IET
[PS12]	Causevic2012b [28]									Experiment	Academia	QUATIC 2012, IEEE
[PS13]	Causevic2013 [30]									Industria Empirical Study	Industria	CESI'14, IEEE
[PS14]	Chien2008 [33]				[13]					Experiment	Academia	Proceedings of 2008 IEEE International Symposium on IT in Medicine and Education, IEEE.
[PS15]	Crispin2006 [37]		[65]							Case study	No indica	IEEE Software (Volume:23 , Issue: 6)
[PS16]	Damm2006 [38]	[41]	[62]		[14]					Case study	Industria	Journal of Systems and Software, 2006 - Elsevier
[PS17]	Damm2007 [39]				[15]	[28]				Case study	Industria	EuroSPI 2007, Springer
[PS18]	Desai2008 [41]		[73]							Review	Academia	ACM SIGCSE Bulletin archive Volume 40 Issue 2, June 2008, ACM

Tabla A.1 – continuación

ID	AUTOR	Bi [18]	Mu [134]	Ma [121]	Ca [32]	Ko [104]	Tu [167]	De [41]	Ra [145]	Tipo de estudio (autor)	Academia/ Industria	Revista / conferencia
[PS19]	Desai2009 [42]		[48]	[23]		[34]			[50]	Controlled experiment	Academia	SIGCSE'09, ACM
[PS20]	Dogsa2011 [44]		[59]	[14]					[61]	Case study	Industria	Software Quality Journal, Springer
[PS21]	Domino2003 [46]				[17]					Quasi experiment	Academia	SIGMIS Conference03, ACM
[PS22]	Domino2007 [45]		[77]		[16]					Controlled experiment	Academia	Information Technology and Management, Springer
[PS23]	Edwards2003 [47]	[30]				[32]			[32]	Experiment	Academia	Proceedings of the International Conference on Education and Information Systems Technologies and Application
[PS24]	Edwards2004 [48]					[7]		[5]		Experiment	Academia	ACM SIGCSE Bulletin Homepage Volume 36 Issue 1
[PS25]	Erdogmus2005 [50]	[36]	[6]		[18]	[5]	X	[6]	[5]	Controlled experiment	Academia	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING
[PS26]	Ficco2011 [54]		[70]							Case study	Academia	Journal Advances in Software Engineering, Hindawi Publishing Corp.
[PS27]	Filho2006 [140]				[19]					Case study	Academia	WoSQ'06, ACM.
[PS28]	Flohr2005 [57]		[52]		[20]					Experiment	Academia	PROFES 2005, Springer-Verlag Berlin Heidelberg 2005
[PS29]	Flohr2006 [58]		[56]		[21]	[40]	X		[51]	Controlled experiment	Academia	PROFES 2006, Springer-Verlag Berlin Heidelberg 2006
[PS30]	Fucci2013 [60]									Controlled Experiment (Replication)	Academia	ESEM 2013, ACM/IEEE
[PS31]	Fucci2014 [63]									Controlled experiment	Academia	ESEM'14, ACM
[PS32]	Fucci2014b [61]									Controlled experiment (Replication)	Academia	Article - Empirical Software Engineering - April 2014, Volume 19, Issue 2, pp 277-302
[PS33]	Fucci2015 [64]									Experimento	Academia	EASE '15, ACM
[PS34]	Fucci2015b [62]									Quasi experiment	Industria	Information and Software Technology, Elsevier
[PS35]	George2002 [65]						X		[28]	Master thesis	Academia e Industria	North Carolina State Univ.
[PS36]	George2003 [66]	[24]		[15]		[47]				Experiment	Industria	Proceedings of the 2003 ACM Symposium on Applied Computing
[PS37]	George2004 [67]	[22]	[46]		[22]	[18]				Experiment	Industria	Information and Software Technology, ACM
[PS38]	Geras2004 [69]	[21]	[49]	[16]	[23]	[44]	X		[35]	Experiment	Industria	METRICS'04, IEEE

Tabla A.1 – continuación

ID	AUTOR	Bi [18]	Mu [134]	Ma [121]	Ca [32]	Ko [104]	Tu [167]	De [41]	Ra [145]	Tipo de estudio (autor)	Academia/ Industria	Revista / conferencia
[PS39]	Geras2004Thesis [68]						X			MSc Thesis	Industria	University of Calgary.
[PS40]	Gupta2007 [72]	[23]	[42]	[24]	[24]	[19]	X		[47]	Controlled experiment	Academia	ESEM 2007, IEEE
[PS41]	Hofar2009 [76]				[25]					Quasi Experiment	Academia e Industria	Chapter. Agile Process in Software Engineering and Extreme Programming. Springer.
[PS42]	Huang2006 [77]		[58]							Controlled Experiment	Academia	TAIC PART 2006, IEEE Computer Society.
[PS43]	Huang2007 [79]				[34]					Controlled experiment	Academia	Testing: Academic and Industrial Conference - Practice And Research Techniques, IEEE.
[PS44]	Huang2009 [78]		[5]	[25]	[26]	[20]	X		[52]	Controlled experiment	Academia	Information and Software Technology 51, Elsevier
[PS45]	Janzen2005 [88]							[7]		Survey	Academia e Industria	Cover Feature, IEEE Computer Society, 2005
[PS46]	Janzen2006 [89]	[35]	[54]	[26]	[27]	[6]				Controlled experiment	Academia	CSEET'06, IEEE
[PS47]	Janzen2006b [90]							[8]		Experiment	Academia	SIGCSE'06, ACM
[PS48]	Janzen2006Thesis [87]						X		[42]	PhD Thesis	Academia	Univ. Of Kansas
[PS49]	Janzen2007 [92]	[29]			[29]	[45]				Controlled experiment	Industria	CSEET'07, IEEE
[PS50]	Janzen2007b [91]		[47]					[9]		Field study	Academia e industria	ICSE'07, IEEE
[PS51]	Janzen2008 [86]	[31]	[51]					[10]		Experiment	Academia	SIGCSE'08, ACM
[PS52]	Janzen2008b [85]	[26]	[44]	[20]	[28]	[10]				Case study	Academia e Industria	IEEE Software Vol. 25 issue 2
[PS53]	Jeffries2007 [93]							[11]		Survey	Academia e Industria	Guest Editors Introduction, IEEE Software 2007
[PS54]	Jones2004 [96]							[12]		Survey	Academia	CCSC: Rocky Mountain Conference, Consortium for Computing Sciences in Colleges
[PS55]	Kaufmann2003 [101]						X	[13]	[53]	Experiment	Academia	OOPSLA'03, ACM
[PS56]	Keefe2006 [102]							[14]		Experience report	Academia	Proceedings of the 8th Australasian Conference on Computing Education - Volume 52
[PS57]	Kobayashi2006 [103]				[30]					Case study	Industria	ICSE '06, ACM
[PS58]	Kollanus2008 [106]				[31]					Experiment	Academia	Koli '08 Proceedings of the 8th International Conference on Computing Education Research , ACM

Tabla A.1 – continuación

ID	AUTOR	Bi [18]	Mu [134] [121]	Ma [121] [32]	Ca [104] [41]	Ko [104] [41]	Tu [167] [41]	De [145] [41]	Ra [145] [41]	Tipo de estudio (autor)	Academia/ Industria	Revista / conferencia
[PS59]	Kollanus2008b [105]		[13]							Experiment	Academia	ITiCSE'08, ACM
[PS60]	Laranjeiro2009 [107]		[57]							Case study	Industria	Second International Conference on Dependability, IEEE.
[PS61]	Latorre2014 [109]									Quasi experiment	Industria	IEEE Transactions on Software Engineering
[PS62]	Latorre2014b [110]									Experience Report	Industria	Empirical Software Engineering, Volume 19, Issue 3, Springer
[PS63]	Layman2006 [111]			[32]						Case study	Industria	Journal of Systems Architecture 52, Elsevier
[PS64]	LeJeune2006 [113]			[33]						Case study	Academia	Journal of Computing Sciences in Colleges archive Volume 21, ACM
[PS65]	Lui2004 [114]					[30]			[62]	Case study	Industria	Chapter book, Springer-Verlag Berlin Heidelberg 2004
[PS66]	Madayski2005 [115]		[40]			[23]	X	[16]	[48]	Experiment	Academia	Conf. Softw. Eng. Evol. Emerg. Technol.
[PS67]	Madayski2006 [116]				[35]	[38]	X			Experiment	Academia	PROFES 2006, Springer
[PS68]	Madayski2007 [117]				[36]					Experiment	Academia	PROFES 2007, Springer
[PS69]	Madayski2007b [120]	[43]	[53]	[21]	[39]	[31]	X			Experiment	Industria	EuroSPI 2007, Springer
[PS70]	Madayski2008 [118]				[37]					Experiment	Academia	Journal Software Process: Improvement and Practice, Wiley
[PS71]	Madayski2010 [119]	[42]	[50]	[27]	[38]	[39]				Experiment	Academia	Information and Software Technology 52, Elsevier
[PS72]	Marchenko2009 [122]				[40]					Case study	Industria	Chapter. Agile Processes in Software Engineering and Extreme Programming. Volume 31 of the series Lecture Notes in Business Information Processing, Springer.
[PS73]	Maximilien2003 [124]	[25]	[69]	[17]	[41]	[24]				Case study	Industria	ICSE'03, IEEE
[PS74]	McDaid2008 [125]		[68]							Case study	Academia e Industria	WEUSE IV'08, ACM
[PS75]	Melnik2005 [126]							[17]		Experience report	Academia	ICSE'05, ACM
[PS76]	Misc2006 [128]				[42]					Survey	Academia e Industria	ACM SIGSOFT Software Engineering Notes
[PS77]	Muller2001 [132]							[19]		Case study	Academia	ICSE 2001

Tabla A.1 – continuación

ID	AUTOR	Bi [18]	Mu [134]	Ma [121]	Ca [32]	Ko [104]	Tu [167]	De [41]	Ra [145]	Tipo de estudio (autor)	Academia/ Industria	Revista / conferencia
[PS78]	Muller2002 [130]		[39]		[43]	[22]	X	[18]	[54]	Experiment	Academia	IEEE Proceedings - Software, Volume:149 Issue:5
[PS79]	Muller2007 [131]			[22]	[44]					Quasi experiment	Academia e Industria	Empir Software Eng (2007), Springer
[PS80]	Munir2014 [135]									Experiment	Industria	EASE '14, ACM
[PS81]	Nagappan2008 [136]	[44]	[66]	[18]	[45]	[8]	X		[38]	Case study	Industria	Empirical Software Engineering, Volume 13, Issue 3, Springer
[PS82]	Pancur2003 [139]	[28]	[3]			[21]	X	[20]	[55]	Controlled experiment	Academia	Eurocon 2003, IEEE
[PS83]	Pancur2011 [138]	[10]	[41]	[28]					[49]	Controlled experiment	Academia	Information and Software Technology 53, Elsevier
[PS84]	Rahman2007 [146]					[33]			[56]	Case Study	Academia	37th ASEEE/IEEE Frontiers in Education Conference
[PS85]	Rendell2008 [147]					[41]				Case study	Industria	Agile, 2008. AGILE '08. Conference
[PS86]	Ress2013 [148]									Case study	Industria	J. Technol. Manag. Innov. 2013, Volume 8, Special Issue ALTEC.
[PS87]	Salman2014 [152]									Experiment	Academia e Industria	Masther Thesis, U. de Oulu
[PS88]	Salo2007 [153]				[46]					Case study	Industria	Software Process: Improvement and Practice. Special Issue: Special Issue on Design for Flexibility. Volume 12, Issue 1. Wiley Online Library
[PS89]	Sanchez2007 [154]	[37]	[74]		[47]	[26]				Case study	Industria	Agile Conference (AGILE), 2007, IEEE
[PS90]	Sfetsos2006 [155]				[48]					Survey	Academia e Industria	Empirical Software Engineering, June 2006, Volume 11, Issue 2.
[PS91]	Sfetsos2015 [156]									Case study	Academia	IISA 2015, IEEE
[PS92]	Sherrell2006 [161]				[49]					Case study	Academia	Journal of Computing Sciences in Colleges archive. Volume 22 Issue 2, ACM
[PS93]	Sinialto2007 [162]	[20]	[61]		[50]	[42]				Case study	Industria	ESEM 2007, IEEE
[PS94]	Sinialto2008 [163]		[71]		[51]	[43]	X			Case study	Academia e Industria	Chapter. Balancing Agility and Formalism in Software Engineering CEE-SET 2007, LNCS
[PS95]	Slyngstad2008 [164]	[39]	[63]		[52]	[25]	X		[63]	Case study	Industria	ICSEA 08, IEEE.
[PS96]	Spacco2006 [165]							[21]		Experience report	Academia	OOPSLA '06, ACM

Tabla A.1 – continuación

ID	AUTOR	Bi [18]	Mu [134]	Ma [121]	Ca [32]	Ko [104]	Tu [167]	De [41]	Ra [145]	Tipo de estudio (autor)	Academia/ Industria	Revista / conferencia
[PS97]	Turnu2006 [168]	[34]								Simulation	Academia	Journal of Systems Architecture, Volume 52, Issue 11, Elsevier
[PS98]	Vu2009 [169]	[27]	[45]	[29]	[55]	[46]	X		[57]	Experiment	Academia	2009 Sixth International Conference on Information Technology: New Generations, IEEE
[PS99]	Wasmus2007 [170]				[53]					Case study	Industria	Delft University of Technology. Software Engineering Research Group. Technical Report Series
[PS100]	Wilkerson2012 [171]		[38]	[30]						Quasi experiment	Academia	IEEE Transactions on Software Engineering vol 38 issue 3
[PS101]	Williams2003 [172]		[67]	[19]	[54]	[27]	X		[30]	Case study	Industria	Chapter book, Software Engineering Research, Management and Applications 2009
[PS102]	Xu2009 [174]					[35]			[58]	Case study	Academia	Chapter. Software Engineering Research, Management and Applications 2009, Springer
[PS103]	Yenduri2006 [175]	[45]					X	[22]	[59]	Case study	Academia	In Proceedings Software Engineering Research and Practice
[PS104]	Ynchausti2001 [176]					[29]				Case study	Industria	2nd International Conference on eXtreme Programming and Flexible Processes in Software Engineering, Italy
[PS105]	Zhang2006 [177]					[48]	X		[60]	Case study	Academia	Chapter: Extreme Programming and Agile Processes in Software Engineering Volume 4044
[PS106]	Aniche2015 [2]									Estudio cualitativo	Industria	Journal of the Brazilian Computer Society2015
[PS107]	Cauevic2013 [29]									Experimento	Academia	2013 39th Euromicro Conference Series on Software Engineering and Advanced Applications
[PS108]	Fucci2016 [59]									Experimento	Academia	ESEM '16 Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement , ACM
[PS109]	Hilton2016 [75]									Case study	Academia	Chapter: Agile Processes, in Software Engineering, and Extreme Programming Volume 251 of the series Lecture Notes in Business Information Processing pp 53-65, Springer
[PS110]	Mondal2014 [129]									Case study	Industria	Chapter Emerging Trends in Computing and Communication Volume 298 of the series Lecture Notes in Electrical Engineering pp 225-233
[PS111]	Percival2013 [142]									Survey	Academia	2013 46th Hawaii International Conference on System Sciences

Tabla A.1 – continuación

ID	AUTOR	Bi [18]	Mu [134]	Ma [121]	Ca [32]	Ko [104]	Tu [167]	De [41]	Ra [145]	Tipo de estudio (autor)	Academia/ Industria	Revista / conferencia
[PSI12]	Peters2014 [143]									Case study	Industria	Chapter - Agile Processes in Software Engineering and Extreme Programming, Volume 179 of the series Lecture Notes in Business Information Processing pp 294-301, Springer
[PSI13]	Romano2016 [151]									Estudio cualitativo	Academia e Industria	EASE '16 Proceedings of the 20th International Conference on Evaluation and Assessment in Software Engineering
[PSI14]	Shelton2012 [160]									Case study	Industria	2012 IEEE Fifth International Conference on Software Testing, Verification and Validation

Apéndice B

Especificación de las tareas

B.1. MR especificación de grano grueso

MarsRover API

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

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

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

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

 (0,0)	(1,0)	(2,0)
 (0,1)	(1,1)	(2,1)
 (0,2)	 (1,2)	(2,2)

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 1st f (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).

(0,2)	(1,2)	(2,2)
(0,1)	(1,1)	(2,1)
(0,0)	(1,0)	(2,0)

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 1st f (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).

B.2. MR especificación descompuesta en nanotareas

MarsRover API

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 "b" command.

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 “ffrffrflf” 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 “ffrffrbbblllfrfrbbl” and returns (0,0,N)(2,2)(0,5)(5,0)

B.3. BSK especificación de grano grueso

Bowling Score Keeper

1	4	4	5	6	▲	5	▲	▲	0	1	7	▲	6	▲	▲	2	▲	6
5		14		29		49		60		61		77		97		117		133

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.

B.4. BSK especificación descompuesta en nanotareas

Bowling Score Keeper

The objective is to develop an application that can calculate the score of a single bowling game. 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 todo list. You should be able to incrementally develop a complete solution without an upfront comprehension of all the game's rules. Don't read ahead, and handle the requirements one at a time in the order provided. Solve the problem 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 all 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 requirements.

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.

Requirement: Define 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.

Requirement: Compute the score of an ordinary frame.

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.

Requirement: Define a game, which consists 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 will reuse this game from now on to represent different scenarios, modifying only a few frames each time.

4. Game Score

The score of a bowling game is the sum of the individual scores of its frames.

Requirement: Compute the score of a game.

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.

5. 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.

Requirement: Recognize a strike frame. Compute the score of a strike. Compute the score of a game containing a strike.

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.

6. 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.

Requirement: Recognize a spare frame. Compute the score of a spare. Compute the score of a game containing a spare frame.

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.

7. Strike and Spare

A strike can be followed by a spare. The strike's score is not affected when this happens.

Requirement: Compute the score of a strike when it's followed by a spare. Compute the score of a game with a spare following a strike.

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.

8. 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.

Requirement: Compute the score of a strike that is followed by another strike. Compute the score of a game with two strikes in a row.

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.

9. Multiple Spares

Two spares in a row are possible. The first spare's score is not affected when this happens.

Requirement: Compute the score of a game with two spares in a row.

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.

10. 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.

Requirement: Compute the score of a spare when it's the last frame of a game. Compute the score of a game when its last frame is a spare.

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.

11. 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.

Requirement: Compute the score of a spare when it's the last frame of a game. Compute the score of a game when the last frame is a strike.

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.

12. 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.

Requirement: Compute the score of a game in which the last frame is a spare and the bonus throw is a strike.

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.

13. 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.

Requirement: Check that the score of a perfect game is 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]. It's score is 300.

14. Real Game

Requirement: Check that the score of the game [6, 3] [7, 1] [8, 2] [7, 2] [10, 0] [6, 2] [7, 3] [10, 0] [8, 0] [7, 3] [10] is 135.

B.5. Spreadsheet

Spreadsheet

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

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

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

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

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

B.6. Music Phone

Music Phone Tasks

MusicPhone is an application that runs on a GPS-enabled, MP3-capable mobile phone. MusicPhone will make recommendations for artists the user may like and find upcoming concert events for artists using data gathered from the Last.FM website. The goal of the following tasks is to implement the necessary logic to enable MusicPhone to make recommendations and plan a travel itinerary to concert events.

Task A: Ramp-up

A1. Run the project (Select the project, right click and Run As->Java Application, then select **App - app**), and see three UI windows appear. The Player and GPS UIs are complete. The Recommender window is just a skeleton. You will implement the required functionality in the `Recommender` class.

A2. Run SmokeTest (Select the project, right click and Run As->JUnit Test)) and see the green bar. Check the structure of the sample test in `SmokeTest.java` to get familiar with the application.

A2. Take 5-10 minutes to review the information in the provided documentation.

Task B: Compute distance to a concert

The user will see how far away upcoming concert events are for a particular artist based on the user's current GPS position and the position of the concert's venue. Implement a public method with the signature `public static double computeDistance (GeoPoint, GeoPoint, String)` to calculate the great-circle distance between two geo-coded positions. A geo-coded position is represented by a `GeoPoint` object that specifies a latitude and longitude *in degrees*. The method computes the great-circle distance between two points in either kilometres ("km") or miles ("mi") as specified by the third parameter. Valid latitude values are between -90 and 90 *degrees*; valid longitude values are between -180 and 180 *degrees*. Sometimes a `GeoPoint` has an invalid latitude or longitude. In these cases, the distance returned should be -1.

The formula for the great-circle distance is:

LonD = longitude in degrees;

LatD = latitude in degrees;

LonR = longitude in radians;

LatR = latitude in radians;

$radians = (degrees \times \pi) / 180;$

$$\alpha = \sqrt{\sin^2\left(\frac{\Delta LatR}{2}\right) + \cos(LatR_1) \times \cos(LatR_2) \times \sin^2\left(\frac{\Delta LonR}{2}\right)} \quad (B.1)$$

Great Circle Distance = 2 x arc sin(min(1.0, α)) x Radius; where Radius is the Earth's

radius in kilometers (6371.01) or in miles (3958.76).,

Ejemplos:

Tabla B.1: Composición de la característica Funcionalidad

<i>GeoPoint</i> ₁		<i>GeoPoint</i> ₂		units	Great-circle distance(in units)
<i>LatR</i> ₁	<i>LonR</i> ₁	<i>LatR</i> ₂	<i>LonR</i> ₂		
<i>LatD</i> ₁	<i>LonD</i> ₁	<i>LatD</i> ₂	<i>LonD</i> ₂		
0	0	6.283	6.283	km	0
0	0	0	0		
0	0	1.047	0	km	6671.70
0	0	60	0		
0	0	6.283	6.283	mi	-1
0	0	360	360		
0	0	1.047	0	mi	24145.60
0	0	60	0		
0.630	-1.513	0.592	-2.066	mi	1793.55
36.12	-86.67	33.94	-118.40		

Numbers in roman type are in degrees.

Numbers in italics are in radians.

Task C: Find concerts for an artist

Implement the `getDestinationsForArtist` method of the `Recommender` class. When the user clicks on an artist's name in the list of recommended artists, the UI calls the `getDestinationsForArtist` method to obtain a list of destinations, where each destination contains the info of an upcoming concert together with the distance to the concert's location from the user's current position. The user's current position is provided by the GPS. If the concert's location has an invalid `GeoPoint`, then the distance should be marked as `null`. You'll need to use `IConnector's GetConcertForArtist` method to implement this task.

Task D: Recommend artists

Implement the `getRecommendations` method of the `Recommender` class. `MusicPhone` recommends artists based on the favourite artists of Last.FM users who are the top fans of the currently-playing artist on the user's player. `MusicPhone` recommends the 20 most-frequently-appearing artists among those fans' top artists (as in "people who like this artist also like these other artists..."). `getRecommendations` should return a list of `Recommendation` objects. Each `Recommendation` object consists of an artist's name (`artist`) and the frequency of occurrence (`fanCount`) of that artist. You will need to use `IConnector's getTopFansForArtist` and `getTopArtistsByFan` methods to implement this method.

Example: Suppose the currently playing artist is *Cher*. Suppose *John*, *Sally*, and *Tom* are the only top fans of *Cher*. If *Tom* and *John* both like *U2* as well, then `getRecommendations` should return a list that includes a recommendation containing *U2* with a `FanCount` of 2.

Task E: Compute an itinerary for concerts

E0. Implement the `buildItineraryForArtists` method of the `Recommender` class. In the UI, the user can click on buttons that will add or remove a selected artist to a list of artists who the user would like to see in concert. Whenever this list is changed, the application recalculates an itinerary of upcoming concerts by calling this method. An itinerary is a list of `Destinations`, where each destination contains the concert information and the distance from the previous destination. The first destination's distance is the distance from the user's current position. The itinerary must be chronologically ordered according to the start date of the concerts in it. Satisfy the following constraints in any order, but tackling them *one by one*:

E1. The itinerary starts with the concert having the earliest start date for any of the selected artists.

E2. An artist has at most one concert in the itinerary (some artists may have no concerts, however).

E3. No two concerts in the itinerary may take place on the same day.

E4. If multiple artists have concerts on the same day, the itinerary includes only the concert that is closest in distance to the previous destination.

E5. If the same artist has multiple concerts on the same day, the itinerary includes only the concert that is closest in distance to the previous destination.

B.7. Sudoku

Sudoku Verifier

Sudoku is a game with few simple rules, where the goal is to place nine sets of positive digits (1...9) into the cells of a fixed grid structure. A valid Sudoku solution should conform to the following rules:

- A cell in a Sudoku game can only store positive digits, i.e. 1...9.
- A “sub-grid” is a 3x3 arrangement of cells.
- All digits appear only once in a sub-grid, i.e. they cannot repeat.
- The Sudoku board (or global grid) consists of a 3x3 arrangement of sub-grids, yielding a 9x9 arrangement of cells.
- A digit can appear only once in the rows of the global grid.
- A digit can appear only once in the columns of the global grid.

Your task is to check the validity of a given solution for a Sudoku game:

1. You should read the candidate solution from a string variable, which would have been displayed like below, when printed on the screen:

```
1 2 3 4 5 6 7 8 9
9 1 2 3 4 5 6 7 8
8 9 1 2 3 4 5 6 7
7 8 9 1 2 3 4 5 6
6 7 8 9 1 2 3 4 5
5 6 7 8 9 1 2 3 4
4 5 6 7 8 9 1 2 3
3 4 5 6 7 8 9 1 2
2 3 4 5 6 7 8 9 1
```

2. You shall check whether the provided string follows the correct format (i.e. 9 lines with 9 entries in each line).
3. You shall check the validity of the candidate solution against the rules listed above.
4. You shall throw `?CustomSudokuException?` for all error cases including but not limited to: wrong format for a solution string.
5. Your program shall provide the functionality to return a string message on the validity of the solution:
 - If it is valid, the following message shall be displayed: “This is a valid solution”, followed by the solution itself (as above).
 - If the solution is not valid you shall return a failure message, indicating the reason why it is not valid, e.g. “9 appears more than once in row 5” (Assume that the lower left corner of the grid has the coordinates (1, 1)). Again, you shall display the solution after this error message.

Apéndice C

Protocolo de medición

En el presente Anexo presentamos el protocolo de medición aplicado a la extracción de las métricas de los experimentos: Playtech, Elektrobit, Seminario UPM, FSecure, UPV y ESPEL. En los experimentos anotados se ha operacionalizado la variable respuesta calidad.

Se han tomado en cuenta los siguientes experimentos reportados en la literatura científica: [25,42,48,50,65,69]. Se han identificado los elementos de la operacionalización según nuestro protocolo.

C.1. Medición de experimentos

Tabla C.1: Protocolo de medición del experimento TDD en C++ (Elektrobit) - Operacionalización del constructo de la variable respuesta calidad

Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos MR	Implementación de instrumentos MP	Implementación de instrumentos BSK	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de medición sobre el código fuente	Obtención del resultado
Calidad externa del código fuente.	Ecuación 4.1	Definición del conjunto de casos de prueba para evaluar la calidad del código fuente con respecto a la especificación de cada tarea.	Suite de test cases para Mars Rover	Suite de test cases Music Phone.	Suite de test cases Bowling Score Keeper.	No es necesario verificar con la implementación de referencia, debido a que se basan en los test cases ya verificados de Java.	Se recibe el código fuente y los test cases en una máquina virtual (Ubuntu). En segunda instancia se recibió por separado nuevos test cases que han sido copiados a la máquina virtual.	1. Copiar el código fuente de los sujetos en el repositorio del IDE, dentro de la máquina virtual. 2. Anotar el orden que determina el Test Case Suite para la ejecución de los casos de prueba en el ambiente de pruebas. En base al orden de la ejecución de los test cases, crear el formato para registrar las mediciones. 3. Abrir el proyecto a medir. 4. En este caso no es necesario cargar la suite de casos de prueba porque ya la incluyeron en el proyecto, pero en algún momento fueron cargados. 5. En este caso no fue necesario cargar las librerías la ejecución del suite de casos de prueba. Ya vinieron incluidas en la configuración del IDE pero en algún momento cargaron y configuraron librerías.	1. Registrar individualmente los estados correspondientes PASSED, FAILED, ERR por cada assertion.
Subcaracterística accuracy del modelo de calidad ISO9126-1.			Eclipse, Java, Oracle VM.	Eclipse, Java, Oracle VM.	Eclipse, Java, Oracle VM.			6. Modificar el código de los test cases, en el caso que se interrumpa la ejecución de manera no controlada, se deshabilita la aserción correspondiente. 7. Ejecutar el proyecto. Si contiene errores en la ejecución de los test cases y sus respectivas aserciones, regresar al paso 6. 8. Guardar la salida de la ejecución en un fichero.	2. Se obtiene el total PASSED, FAILED, ERR por cada user story. 3. Se sunariza las divisiones de assertions PASS de los user story tackled para el total de assertions. Se divide para el número de user story tackled y se obtiene el porcentaje donde M: Número de User Story Tackled 4. Las mediciones obtenidas son almacenadas en el fichero metricsExtraction. Los casos de prueba y los ficheros de la medición se encuentran en un Portal Web (Google Sites).

Tabla C.2: Protocolo de medición del experimento TDD en Java (UPM) - Operacionalización del constructo de la variable respuesta calidad

Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos MR	Implementación de instrumentos MP	Implementación de instrumentos BSK	Implementación de instrumentos SU	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
Calidad externa del código fuente.	Ecuación 4.1	Definición conjunta de casos de prueba para evaluar la calidad del código fuente con respecto a la especificación de cada tarea.	Suite de test cases: Mars Rover 1.3,	Suite de test cases Music Phone tests 1.2	Suite de test cases Bowling Score Keeper 1.1	Suite de test cases Sudoku	Se verifican las salidas esperadas de los casos de prueba, en base a una implementación de referencia. Luego se corrigen los casos de prueba con error.	El producto corresponde al código fuente de los sujetos y se encuentran en un repositorio de productos.	1. Copiar el código fuente de los sujetos en el repositorio del IDE. 2. Anotar el orden, que establece Junit, para la ejecución de los casos de prueba en el ambiente de pruebas. En base al orden de la ejecución de los test cases, crear el formato para registrar las mediciones. (Este paso es para la medición sin plugin) 3. Abrir el proyecto a medir. 4. Cargar la suite de casos de prueba 5. Cargar las librerías necesarias para la ejecución del suite de casos de prueba. 6. Modificar el código fuente en los casos que sea necesario: a) Nombres de métodos que tienen que estar con igual nombre que en el instrumento de medida. b) Formato de salidas (return) de funciones importantes que deben estar con el mismo número de parámetros que en el instrumento de medida. c) Modificación o creación de métodos constructores para que puedan conectar con los test cases 7. Ejecutar el proyecto. Si contiene errores regresar al numeral 5 para verificar el cumplimiento de dichos pasos. 8. Guardar la salida de la ejecución en un fichero. En el caso de Sudoku, la medición se realiza con el plugin de forma que se obtiene directamente la medida de Calidad.	1a. En el caso de Sudoku el plugin ofrece directamente la medida de calidad. 1b. Registrar individualmente el estado de cada assertion, con el valor respectivo PASS, FAIL, ERR por cada User Story 2. Se obtiene el total PASS, FAIL, ERR por cada user story. 3. Se suman las divisiones de assertions PASS de los user story tackled para el total de assertions. Se divide para el número de user story tackled y se obtiene el porcentaje 4. Las mediciones obtenidas son almacenadas en el fichero metricsExtraction. Los casos de prueba y los ficheros de la medición se encuentran en un Portal Web (Google Sites).
Subcaracterística accuracy del modelo de calidad ISO9126-1.			Formato Excel para registrar las mediciones. (fichero metrics-Extraction(new-metrics)-.xls)	Formato Excel para registrar las mediciones. (fichero metrics-Extraction(new-metrics)-.xls)	Formato Excel para registrar las mediciones. (fichero metrics-Extraction(new-metrics)-.xls)	Plugging junit.jar				

Tabla C.3: Protocolo de medición del experimento TDD en Java (UPV) - Operacionalización del constructo de la variable respuesta calidad

Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos MR	Implementación de instrumentos BSK	Implementación de instrumentos SU	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de medición de aplicación sobre el código fuente	Obtención del resultado
Calidad externa del código fuente.	Ecuación 4.1	Definición del conjunto de casos de prueba para evaluar la calidad del código fuente con respecto a la especificación de cada tarea.	Suite de test cases: Mars Rover 1.3,	Suite de test cases Bowling Score Keeper 1.1	Suite de test cases Sudoku	Se verifican las salidas esperadas de los casos de prueba, en base a una implementación de referencia. Luego se corrigen los casos de prueba con error.	El producto responde al código fuente de los sujetos y se encuentran en un repositorio de productos.	1. Copiar el código fuente de los sujetos en el repositorio del IDE. 2. Anotar el orden, que establece Junit, para la ejecución de los casos de prueba en el ambiente de pruebas. En base al orden de la ejecución de los test cases, crear el formato para registrar las mediciones. (Este paso es para la medición sin plugin)	1a. En el caso de Sudoku el plugin ofrece directamente la medida de calidad. 1b. Registrar individualmente el estado de cada asertion, con el valor respectivo PASS, FAIL, ERR por cada User Story
Subcaracterística accuracy del modelo de calidad ISO9126-1.		Formato Excel para registrar mediciones. (archivo metrics-Extraction-(new-metrics)-.xls) Java runtime environment 1.7	Formato Excel para registrar mediciones. (archivo metrics-Extraction-(new-metrics)-.xls) Java runtime environment 1.7	Formato Excel para registrar mediciones. (archivo metrics-Extraction-(new-metrics)-.xls) Java runtime environment 1.7	Plugin junit.jar			3. Abrir el proyecto a medir. 4. Cargar la suite de casos de prueba. 5. Cargar las librerías necesarias para la ejecución del suite de casos de prueba. 6. Modificar el código fuente en los casos que sea necesario: a) Nombres de métodos que tienen que estar con igual nombre que en el instrumento de medida. b) Formato de salidas (return) de funciones importantes que deben estar con el mismo número de parámetros que en el instrumento de medida. c) Modificación o creación de métodos constructores para que puedan conectar con los test cases 7. Ejecutar el proyecto. Si contiene errores regresar al numeral 5 para verificar el cumplimiento de dichos pasos. 8. Guardar la salida de la ejecución en un fichero. En el caso de Sudoku la medición se realiza con el plugin de forma que se obtiene directamente la medida de Calidad.	2. Se obtiene el total PASS, FAIL, ERR por cada user story. 3. Se suman las divisiones de assertions PASS de los user story tackled para el total de assertions. Se divide para el número de user story tackled y se obtiene el porcentaje 4. Las mediciones obtenidas son almacenadas en el fichero metricsExtraction. Los casos de prueba y los ficheros de la medición se encuentran en un Portal Web (Google Sites).

Tabla C.4: Protocolo de medición del experimento TDD en Java (ESPEL) - Operacionalización del constructo de la variable respuesta calidad

Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos MR	Implementación de instrumentos MP	Implementación de instrumentos BSK	Implementación de instrumentos SU	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
Calidad externa del código fuente.	Ecuación 4.1	Definición del conjunto de casos de prueba para evaluar la calidad del código fuente con respecto a la especificación de cada tarea.	Suite de test cases: Mars Rover 1.3,	Suite de test cases Music Phone tests 1.2	Suite de test cases Bowling Score Keeper 1.1	Suite de test cases Sudoku	Se verifican las salidas esperadas de los casos de prueba, en base a una implementación de referencia. Luego se corrigen los casos de prueba con error.	El producto corresponde al código fuente de los sujetos y se encuentran en un repositorio de productos.	1. Copiar el código fuente de los sujetos en el repositorio del IDE. 2. Anotar el orden, que establece Junit, para la ejecución de los casos de prueba en el ambiente de pruebas. En base al orden de la ejecución de los test cases, crear el formato para registrar las mediciones. (Este paso es para la medición sin plugin) 3. Abrir el proyecto a medir. 4. Cargar la suite de casos de prueba. 5. Cargar las librerías necesarias para la ejecución del suite de casos de prueba. 6. Modificar el código fuente en los casos que sea necesario: a) Nombres de métodos que tienen que estar con igual nombre que en el instrumento de medida. b) Formato de salidas (return) de funciones importantes que deben estar con el mismo número de parámetros que en el instrumento de medida. c) Modificación o creación de métodos constructores para que puedan conectar con los test cases 7. Ejecutar el proyecto. Si contiene errores regresar al numeral 5 para verificar el cumplimiento de dichos pasos.	1a. En el caso de MR, BSK, Sudoku, el plugin ofrece directamente la medida de calidad. 1b. Registrar individualmente el estado de cada assertion, con el valor respectivo PASS, FAIL, ERR por cada User Story 2. Se obtiene el total PASS, FAIL, ERR por cada user story. 3. Se suman las divisiones de assertions PASS de los user story tackled para el total de assertions. Se divide para el número de user story tackled y se obtiene el porcentaje
Subcaracterística accuracy del modelo de calidad ISO9126-1.			Plugging junit.jar Java runtime environment 1.7	Formato en Excel para registrar las mediciones. (archivo metrics-Extraction-(new-metrics)-.xls) Java runtime environment 1.7	Plugging junit.jar Java runtime environment 1.7	Plugging junit.jar Java runtime environment 1.7				

Tabla C.4 – continuación

Elabora- ción de la va- riable respon- sa	Definición de la métrica	Definición del ins- trumento de medida	Imple- men- tación de instru- mentos MR	Imple- men- tación de instru- mentos MP	Imple- men- tación de instru- mentos BSK	Imple- men- tación de instru- mentos SU	Verificación del instru- mento de medición	Recepción del resul- tado de la tarea experi- mental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
			IDE ECLIPSE	IDE ECLIPSE	IDE ECLIPSE	IDE ECLIPSE			8. Guardar la salida de la ejecución en un fichero. En el caso de MR, BSK, Sudoku, la medición se realiza con el plugin de forma que se obtiene directamente la medida de Calidad	4. Las mediciones obtenidas son almacenadas en el fichero metricsExtraction. Los casos de prueba y los ficheros de la medición se encuentran en un Portal Web (Google Sites).

C.2. Operacionalización de experimentos de la literatura

Tabla C.5: Operacionalización del constructo de la variable respuesta calidad del experimento de Canfora [25]

	Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
DESCRIPCIÓN:	Test cases? accuracy.	Media del número de aserciones por método. (Si hay más aserciones por método, más completo es el caso de prueba para ese método.)	No hay creación del instrumento.	Método de conteo manual. Java, Eclipse, Junit	No hay verificación del instrumento de medida.	El investigador cuenta con el código fuente de 28 sujetos experimentales, profesionales, quienes ejecutan TDD y TAC (Testing after coding). Además se han recibido los formularios que han llenado los sujetos, en donde por cada requerimiento, han listado las aserciones correspondientes que escribieron con el propósito de probar los métodos de las clases Java (código fuente).	En este experimento, la medida la toma el sujeto experimental, al escribir el valor de "Number of Assertions" por método en el formulario correspondiente. El paper no menciona si hay características que deberían tener las aserciones para darse como válidas.	El investigador lee la columna "Number of Assertions" por cada método del respectivo formulario, luego se totaliza por proyecto. Se obtiene promedio y la desviación estándar del número de aserciones por método y por proyecto en el experimento.
QUIEN:							Sujeto experimental	Investigador.
COMO:							En un formulario.	
CUAN-DO:								

Tabla C.6: Operacionalización del constructo de la variable respuesta calidad del experimento de Geras et.al [69]

Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
DES- CRIP- CIÓN Quiénes tienen más FAIL no planeados (Calidad del producto)	1. Número de FAIL ASSERTIONS no planeados en la ejecución del caso de prueba.	El investigador modifica JUnit para que grabe los resultados de la ejecución de los developer test cases.	Developer test cases. Usan una versión modificada de JUnit que graba el nombre del test case, nombre de la condición (assert) y el resultado de la evaluación.	El paper no indica si hubo un procedimiento para verificar el instrumento de evaluación.	En el experimento se contó con 7 sujetos experimentales en cada grupo, quienes realizaron dos ejercicios aplicando Test Last y Test First. Los sujetos han enviado anónimamente el byte code compilado y los log files generados por los test frameworks a través de un Portal Web.	El sujeto experimental es el encargado de correr las pruebas unitarias (developer test cases) y de aceptación (Custommer test cases) sobre su código fuente. La versión modificada de JUnit graba un registro cada vez que se corre un caso de prueba y cada vez que una condición del test (una asersión) es evaluada. El registro tiene los siguientes campos: nombre del test case, nombre de la condición de prueba y el resultado de la evaluación (PASS o FAIL).	7. Número de fallos no planeados por developer test case
	FAIL ASSERTION no planificado por caso de prueba. (Considerando la intención del programador de que FALLE o que PASE).	El investigador ha creado la herramienta VBEExpect: Customer test framework, para los customer test cases. Se deduce que los Customer Test y los Developer Test Cases los definió el desarrollador. Los Developer tests los define el desarrollador	Customer test cases (Tests de Aceptación). Los sujetos usan VBEExpect un framework de pruebas para diseñar, implementar y ejecutar las pruebas de aceptación. IDE (Puede ser cualquiera, no necesariamente Eclipse)				8. Número de fallos no planeados por customer test case

Tabla C.6 – continuación

Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
<p>Esfuerzo (Si TF requiere más esfuerzo que TL)</p> <p>Testing (Si TF requiere más esfuerzo de Testing que TL)</p> <p>Frecuencia de ejecución de los casos de prueba</p>	<p>Esfuerzo por hora ideal de programación (minutos)</p> <p>Densidad de casos de prueba de aceptación (customer)= Casos de prueba de aceptación / KLOC</p> <p>Densidad de casos de prueba de desarrollo (developer)= Casos de prueba de desarrollo / KLOC</p> <p>Densidad de casos de prueba de aceptación (customer)= Casos de prueba de aceptación / Hora ideal de programación</p> <p>Densidad de casos de prueba de desarrollo (developer)= Casos de prueba de desarrollo / hora ideal de programación</p> <p>Frecuencia de evaluación = Evaluaciones Customer Test case por Hora</p> <p>Frecuencia de evaluación = Evaluaciones Developer Test case por Hora</p>	<p>Son dos aplicaciones: La entrada de tiempos de un proyecto (Registro de tiempos de tareas en un proyecto)</p>					<p>Se obtienen las siguientes medidas: 1. Densidad de customer test cases = Total Customer test cases / KLOC</p> <p>2. Densidad de developer test cases = Total Developer test cases / KLOC</p> <p>4. Densidad de customer test cases por hora ideal de programación = Total customer test cases / hora ideal de programación</p> <p>3. Densidad de developer test cases por hora ideal de programación = Total developer test cases / hora ideal de programación</p> <p>6. Número de evaluaciones customer tests por hora</p> <p>5. Número de evaluaciones developer tests por hora</p>
QUIEN:		Investigador	Sujeto experimental			Sujeto experimental	Investigador
COMO:					Se envían a través de un portal Web		
CUANDO:							

Tabla C.7: Operacionalización del constructo de la variable respuesta calidad del experimento de Edwards [48]

	Elaboración de la variable respuesta	Definición de la métrica	Definición del instrumento de medida	Implementación de instrumentos	Verificación del instrumento de medición	Recepción del resultado de la tarea experimental	Procedimiento de aplicación del instrumento de medición sobre el código fuente	Obtención del resultado
DESCRIPCIÓN:	Code Correctness	Web-CAT Score = $(PASS * ACC_TEST + COB_TEST) * 50$	Los estudiantes crean el conjunto de casos de prueba, en el caso de TDD.	Web-based Center for Automated Testing (Web-CAT). Es una herramienta para evaluar el código fuente y los casos de prueba de los estudiantes.	El sujeto ejecuta los test, se compara la implementación de referencia provista por el instructor para confirmar que la salida esperada del test case del estudiante sea correcta para cada caso de prueba.	El código fuente corresponde a 118 sujetos experimentales (estudiantes) en total, divididos en dos grupos, aplicando no TDD y TDD. Correspondientes a la primavera 2001 y 2003 respectivamente. Se analizó únicamente el primer assignment de cada semestre.	El alumno aplica el instrumento de evaluación a su propio código tantas veces como quiera antes del deadline. El instrumento de evaluación retroalimenta al estudiante, informa los detalles de la ejecución de los test case. Cada test case no pasado es identificado con un mensaje descryptivo, el instrumento informa también sobre el resultado de la validez de los test cases y sobre la completitud de los mismos.	Se toma la medida del Web-CAT Score.
		PASS: Porcentaje de número de casos de prueba pasados.	Para el caso de No TDD los casos de prueba fueron generados por el sistema en base a datos de prueba dados por el instructor.		El instrumento de medida evalúa validez y completitud de los casos de prueba. Este proceso de verificación de los test cases se ejecuta cada vez que el estudiante prueba su código.			Web-CAT Score = $(PASS * ACC_TEST + COB_TEST) * 50$
		ACC_TEST: Porcentaje de número de accurate tests. (que las salidas esperadas son válidas)						
		COB_TEST: Porcentaje de cobertura de código de los tests.						

Tabla C.7 – continuación

	Elaboración de la variable res- puesta	Definición de la métrica	Definición del instru- mento de medida	Implementación de instrumentos	Verificación del instrumento de medición	Recepción del resultado de la tarea experi- mental	Procedimiento de apli- cación del instrumen- to de medición sobre el código fuente	Obtención del re- sultado
	Densidad de defectos (Calidad de código)	Defectos/KSLOC	El inves- tigator desarrolló un test suite integral, se agregan a la suite además los test cases no duplica- dos escritos para TDD por los es- tudiantes y por los test cases generados para el caso de No TDD.	Test suite integral.	Se verificó que haya 100 % de cobertura en la implementación referencial.		Para obtener la densidad de defectos, 18 programas fueron seleccionados, 9 de cada grupo. Se eliminaron sus comentarios y líneas en blanco. Se depuraron a mano. Se hicieron los cam- bios necesarios para apli- car el instrumento de eva- luación y alcanzar pasar el 100 % de los test cases.	Se totalizó el núme- ro de líneas añadidas, cambiadas o removi- das, normalizadas por la longitud del progra- ma (normalized by the program length) y se utilizaron para medir los defectos por KS- LOC.
								Se realizó una regre- sión lineal entre el número de defecto- s/KSLOC identifica- dos y el número de test case fallidos en la ejecución del test suite integral en la muestra lo cual pro- dujo una correlación significativa al nivel de 0.05. En base a la correlación obtenida, se estimaron los defec- tos / KSLOC para los programas restantes en los dos grupos de estudiantes.
QUIEN:			Investigador y Sujeto ex- perimental				Sujeto experimental	Investigador.
COMO:					Mediante una im- plementación de referencia.			De manera automáti- ca.
CUAN- DO:					Cada vez que el estudiante prueba su código.			

Tabla C.8: Operacionalización del constructo de la variable respuesta calidad del experimento de Desai et. al [42]

	Elabora- ción de la va- riable res- puesta	Definición de la métrica	Definición del instru- mento de medida	Implemen-tación de instrumentos	Verificación del instrumento de medición	Recepción del resultado de la tarea experi- mental	Procedimiento de apli- cación del instrumen- to de medición sobre el código fuente	Obtención del re- sultado
DES- CRIP- CIÓN:	Calidad del proyecto	Porcentaje de núme- ro de test cases pasa- dos.	El instruc- tor crea la suite de test cases	Junit, Java, Suite de test cases.	Código fuente de 2 grupos de estudiantes, Test First (TF) y Clasical Tests (CT).	No indica si se verificó el instru- mento de medi- ción	Se ejecuta el código del es- tudiante contra la suite de test cases del profesor.	El investigador obtie- ne el porcentaje de test case pasados para TF y para CT.
QUIEN:			Instructor					
COMO:								
CUAN- DO:								

Apéndice D

Código fuente obtenido de los sujetos experimentales

En este anexo en la figura D.1 presentamos un código fuente tomado de un sujeto experimental. El código fuente pertenece al sujeto FSK06 del experimento FSecure, tarea experimental Mars Rover. Este código está completo y no ha sido retocado.

```
package training.tdd.day1;

//import java.lang.reflect.Array;
import java.util.Arrays;

public class MarsRover {

    private int xStart;
    private int yStart;
    private String direction;
    private String [] d = { "North", "East", "South", "West" };
    private String sphereObstacles;

    public MarsRover(int x, int y, String obstacles) {
        /*
         * x and y represent the size of the grid. Obstacles is a
         * String
         * formatted as follows: ?(o1_x,o1_y)(o2_x,o2_y)...
         * (on_x,on_y)? with no
         * white spaces.
         *
         * Example use: MarsRover rover = new MarsRover(100,100,?
         * (5,5)(7,8)?)
         * //A 100x100 grid with two obstacles at coordinates (5,5)
         * and (7,8)
         */
        xStart = x;
        yStart = y;
    }
}
```

```

        direction = d[0];
        sphereObstacles = obstacles;
    }

    public String executeCommand(String command) {

        /*
        * The command string is composed of "f" (forward), "b"
        * (backward), "l" (left) and "r" (right) Example:
        * The rover is on a 100x100 grid at
        * location (0, 0) and facing NORTH. The rover is given the
        * commands "ffrff" and should end up at (2, 2) facing East.
        *
        * The return string is in the format:
        * ?(x,y,facing)(o1_x,o1_y)(o2_x,o2_y)?..(on_x,on_y)? Where
        * x and y are the final coordinates, facing is the
        * current direction the rover is pointing to (N,S,W,E).
        * The return string should also contain a list
        * of coordinates of the encountered obstacles. No white
        * spaces.
        */
        int length = command.length();
        char singleCommand;
        for (int i = 0; i < length; i++) {
            singleCommand = command.charAt(i);

            if (singleCommand == 'f') {
                forward();
            } else if (singleCommand == 'b') {
                backward();
            } else if (singleCommand == 'l' || singleCommand == 'r') {
                changeDirection(singleCommand);
            }
        }

        String resultString = "(" + Integer.toString(getX()) + ", "
        + Integer.toString(getY()) + ")", " + getFacingDirection();
        return resultString;
    }

    private String changeDirection(char singleCommand) {
        int index = Arrays.asList(d).indexOf(direction);
        if (singleCommand == 'r') {
            if (index == 3) {
                direction = d[0];
            } else {
                direction = d[index + 1];
            }
        } else {
            if (index == 0) {

```

```

        direction = d[3];
    } else {
        direction = d[index - 1];
    }
}
return direction;
}

private String getFacingDirection() {
    return direction;
}

private int getX() {
    return xStart;
}

private int getY() {
    return yStart;
}

private void forward() {
    int index = Arrays.asList(d).indexOf(direction);

    switch (index) {
        case 0:
            yStart++;
            break;
        case 1:
            xStart++;
            break;
        case 2:
            yStart--;
            break;
        case 3:
            xStart--;
            break;
    }

    planetIsSphere();
}

private void backward() {
    int index = Arrays.asList(d).indexOf(direction);

    switch (index) {
        case 0:
            yStart--;
            break;
        case 1:
            xStart--;
            break;
    }
}

```

```
    case 2:
        yStart++;
    break;
    case 3:
        xStart++;
    break;
}

    planetIsSphere();
}

private void planetIsSphere(){
    if (xStart >= 100) {
        xStart = xStart - 100;
    }
    if (yStart >= 100) {
        yStart = yStart - 100;
    }
    if (xStart < 0){
        xStart = 100 + xStart;
    }
    if (yStart < 0){
        yStart = 100 + yStart;
    }
}

    private void faceObstacles() {

    }
}
```

Figura D.1: Ejemplo de código fuente de un sujeto

Apéndice E

Casos de prueba

Se tienen dos suites de casos de prueba para cada tarea experimental (MR y BSK). La suite corresponde a la técnica con que fueron elaborados los casos, ad hoc y PCdE. Ad hoc, significa que los casos de prueba fueron realizados sin ninguna técnica. Por otro lado los casos de prueba elaborados con la técnica de PCdE, con el propósito de ser más efectivos a la hora de hacer el testing del código fuente.

E.1. Casos de prueba Ad hoc

E.1.1. Casos de prueba Ad hoc - MR

Tabla E.1: Casos de prueba Ad hoc tarea: Mars Rover

US ID	C.P. ID	Nombre del caso de prueba	Assert ID	X	Y	Facing	Comando	Obstáculos	Resultados esperados
1	1.1	theRoverStaysAtOriginOnceLandedAndExecutedAnEmptyCommand()	1.1.1	0	0	"N"	""	"	"(0,0,N)"
2	2.1	theRoverFacesSouthAfterLandingAndExecutingATurnCommandToTimesInARow	2.1.1	0	0	"N"	"rr"	"	"(0,0,S)"
2	2.1		2.1.2	0	0	"S"	"ll"	"	"(0,0,N)"
2	2.2	theRoverFacesWestAfterLandingAndExecutingALeftCommandThreeTimesInARow	2.2.1	0	0	"N"	"lll"	"	"(0,0,E)"
2	2.3	theRoverMovesForwardOnceLanding	2.3.1	0	0	"N"	"f"	"	"(0,1,N)"
2	2.3		2.3.2	0	1	"N"	"f"	"	"(0,2,N)"
2	2.4	theRoverShouldChangeFacingThenGiveATurnCommandAfterLanding	2.4.1	0	0	"N"	"l"	"	"(0,0,W)"
2	2.4		2.4.2	0	0	"W"	"rr"	"	"(0,0,E)"
2	2.5	theRoverMovesBackwards- OnceLanding	2.5.1	0	0	"N"	"fb"	"	"(0,0,N)"
2	2.6	theRoverIsFacingTheSameDirectionAfterExecutingFourTimesTheSameTurnCommand	2.6.1	0	0	"N"	"llll"	"	"(0,0,N)"
2	2.6		2.6.2	0	0	"N"	"rrrr"	"	"(0,0,N)"
2	2.7	theRoverFacesEastAfterLandingAndExecutingARightCommandThreeTimesInARow	2.7.1	0	0	"N"	"rrr"	"	"(0,0,W)"
3	3.1	theRoverMovesForwardNorthBoundedOnceLanding	3.1.1	0	0	"N"	"ffffff"	"	"(0,10,N)"
3	3.1		3.1.2	0	10	"N"	"f"	"	"(0,11,N)"
3	3.2	theRoverMovesForwardWestBoundedOnceLanding	3.2.1	0	0	"N"	"fffffffrffffff"	"	"(3,10,W)"
3	3.2		3.2.2	3	10	"W"	"fffffffrffffff"	"	"(2,10,W)"
3	3.3	theRoverMovesForwardSouthBoundedOnceLanding	3.3.1	0	0	"N"	"fffffffrffffff"	"	"(0,5,S)"
3	3.3		3.3.2	0	5	"S"	"fffffffrffffff"	"	"(0,4,S)"
3	3.4	theRoverMovesForwardEastBoundedOnceLanding	3.4.1	0	0	"N"	"fffffffrffffff"	"	"(10,10,E)"
3	3.4		3.4.2	10	10	"E"	"fffffffrffffff"	"	"(11,10,E)"
4	4.1	theRoverMovesBackwardWestBoundedOnceLanding	4.1.1	0	0	"N"	"fffffffrffffff"	"	"(3,10,W)"
4	4.2	theRoverMovesBackwardNorthBoundedOnceLanding	4.2.1	0	0	"N"	"fffffffrffffff"	"	"(0,6,N)"
4	4.2		4.2.2	0	6	"N"	"fffffffrffffff"	"	"(0,5,N)"
4	4.3	theRoverMovesBackwardEastBoundedOnceLanding	4.3.1	0	0	"N"	"fffffffrffffff"	"	"(2,10,E)"
4	4.3		4.3.2	2	10	"E"	"fffffffrffffff"	"	"(1,10,E)"

Tabla E.1 – continuación

US ID	C.P. ID	Nombre del caso de prueba	Assert ID	X	Y	Fa- cing	Comando	Obstáculos	Resultados es- pera- dos
4	4.4	theRoverMovesBackwardSouthBoundedOnce- Landing	4.4.1	0	0	"N"	"ffffffflb bbb"	"	"(0,15,S)"
4	4.4		4.4.2	0	15	"S"	"ffffffflb bbb"	"	"(0,16,S)"
5	5.1	moveRightInsideGridOnClearPathNorth- Bounded	5.1.1	0	0	"N"	"ffffr ffff"	"	"(5,5,E)"
5	5.1		5.1.2	5	5	"E"	"b"	"	"(4,5,E)"
5	5.2	moveLeftInsideGridOnClearPathNorth- Bounded	5.2.1	0	0	"N"	"ffffr ffffffflfl"	"	"(9,5,W)"
5	5.2		5.2.2	9	5	"W"	"b"	"	"(10,5,W)"
5	5.3	moveRightInsideGridOnClearPathSouth- Bounded	5.3.1	0	0	"N"	"ffffr ffffffflfl"	"	"(11,4,E)"
5	5.3		5.3.2	11	4	"E"	"b"	"	"(10,4,E)"
5	5.4	moveLeftInsideGridOnClearPathSouth- Bounded	5.4.1	0	0	"N"	"ffffr ffffffflfl"	"	"(9,5,W)"
5	5.4		5.4.2	9	5	"W"	"b"	"	"(10,5,W)"
6	6.1	theRoverShouldSpawnOnTheLeftSideWhen- CrossingRightBackwards	6.1.1	0	0	"N"	r99flb	"	"(0,0,W)"
6	6.2	theRoverShouldSpawnOnTheLeftSideWhen- CrossingRight	6.2.1	0	0	"N"	r99ff	"	"(0,0,E)"
6	6.3	theRoverShouldSpawnOnTheNorthSideWhen- CrossingSouth	6.3.1	0	0	"N"	"llf"	"	"(0,99,S)"
6	6.4	theRoverShouldSpawnOnTheRightSideWhen- CrossingLeftBackward	6.4.1	0	0	"N"	"rb"	"	"(99,0,E)"
6	6.5	theRoverShouldSpawnOnTheRightSideWhen- CrossingLeft	6.5.1	0	0	"N"	"lf"	"	"(99,0,W)"
6	6.6	roverShouldSpawnOnThebottomWhenCro- ssingNorthBackwards	6.6.1	0	0	"N"	99flb	"	"(0,0,S)"
6	6.7	theRoverShouldSpawnOnTheNorthSideWhen- CrossingSouthBackwards	6.7.1	0	0	"N"	"b"	"	"(0,99,N)"
6	6.8	theRoverShouldSpawnOnTheBottomSide- WhenCrossingNorth	6.8.1	0	0	"N"	99ff	"	"(0,0,N)"
7	7.1	theRoverEncountersAnObstacleOnTheCorners- OfThePlanetWhileMovingForward	7.1.1	0	0	"N"	99f	"(0, 99)"	"(0,98,N)(0,99)"
7	7.1		7.1.2	0	0	"N"	r99ff99f	"(99, 99)"	"(99,98,N)(99,99)"
7	7.1		7.1.3	0	0	"N"	fr99frf	"(99, 0)"	"(99,1,S)(99,0)"
7	7.2	theRoverEncountersAnObstacleInTheMiddle- OfThePlanetWhileMovingForward	7.2.1	0	0	"N"	50fr50f	"(50, 50)"	"(49,50,E)(50,50)"
7	7.2		7.2.2	0	0	"N"	99fr50fl150f	"(50, 50)"	"(50,51,S)(50,50)"
7	7.2		7.2.3	0	0	"N"	r99ff50fl50f	"(50, 50)"	"(51,50,W)(50,50)"
7	7.2		7.2.4	0	0	"N"	r50fl50f	"(50, 50)"	"(50,49,N)(50,50)"

Tabla E.1 – continuación

US ID	C.P. ID	Nombre del caso de prueba	Assert ID	X	Y	Fa- cing	Comando	Obstáculos	Resultados espe- rados
7	7.3	theRoverEncounters3ObstaclesInARow	7.3.1	0	0	"N"	"rfffrfffr"	"(1,1)(2,2) (3,1)"	"(2,1,N)(3,1) (2,2)(1,1). ^o "(2,1,E)(3,1)" "(2,1,N)(2,0) (3,1)(2,2). ^o "(2,1,S)(2,0)" "(2,1,N)(1,1) (2,0)(3,1). ^o "(2,1,W)(1,1)"
7	7.3		7.3.2	0	0	"N"	"rfffrfffr"	"(2,0)(2,2) (3,1)"	
7	7.3		7.3.3	0	0	"N"	"fffrfffrfffr"	"(1,1)(2,0) (3,1)"	
7	7.4	theRoverEncounters2ObstaclesInARow	7.4.1	0	0	"N"	"rfffrfffrff"	"(1, 1)(3,1)"	"(2,2,N)(1,1)(3,1). ^o "(1,0,N)(1,1)"
7	7.4		7.4.2	0	0	"N"	"rfffrfffrff"	"(1, 0)(1,2)"	"(2,1,N)(1,0)(1,2). ^o "(0,0,E)(1,0)"
7	7.5	theRoverEncounters2ObstaclesInARowFrom- Landing	7.5.1	0	0	"N"	"frff"	"(0, 1)(1,0)"	"(0,0,N)(0,1)(1,0). ^o "(0,0,N)(0,1)"
7	7.6	theRoverEncountersAnObstacleOnTheEdges- OfThePlanetWhileMovingForward	7.6.1	0	0	"N"	r50ff99f	"(50, 99)"	"(50,98,N)(50,99)"
7	7.6		7.6.2	0	0	"N"	fr50frfl	"(50, 0)"	"(50,1,N)(50,0). ^o "(50,1,S)(50,0)"
8	8.1	theRoverEncountersAnObstacleOnTheEdges- OfThePlanetWhileMovingBackwardRightTop	8.1.1	0	0	"N"	r50ffrr99b	"(50, 99)"	"(50,98,S)(50,99)"
8	8.1		8.1.2	0	0	"N"	fr50ffl	"(50, 0)"	"(50,1,N)(50,0). ^o "(50,1,S)(50,0)"
8	8.2	theRoverEncountersAnObstacleInTheMiddle- OfThePlanetWhileMovingBackward	8.2.1	0	0	"N"	99fr50ff50b	"(50, 50)"	"(50,51,N)(50,50)"
8	8.2		8.2.2	0	0	"N"	r50ffrr50b	"(50, 50)"	"(50,49,S)(50,50)"
8	8.3	theRoverEncountersSevenObstaclesOnATour- AroundThePlanet	8.3.1	0	0	"N"	rr50brblbblbr50br- blbr50brblbblbr50- brblbr50brblbblbr- 50brblbr50brblbblb- r50bl	"(0, 50)(0,99)(50,99)- (99,99)(99,50)- (99,0)(50,0)"	"(0,0,N)(0,50)- (0,99)(50,99)- (99,99)(99,50)- (99,0)(50,0). ^o "(0,49,S)(0,50)"
8	8.3		8.3.2	0	0	"N"	l50blbrbbrbl50blbr- bl50blbrbbrbl50bl- brbl50blbrbbrbl- 50blbrbl50blbrbbr- bl50b	"(50,0)(99,0)- (99,50)(99,99)- (50,99)(0,99)- (0,50)"	"(0,0,N)(50,0)- (99,0)(99,50)- (99,99)(50,99)- (0,99)(0,50). ^o "(49,0,W)(50,0)"
8	8.4	theRoverEncountersAnObstacleOnTheCor- nersOfThePlanetWhileMovingBackwardsLeft	8.4.1	0	0	"N"	ll99b	"(0, 99)"	"(0,98,S)(0,99)"
8	8.4		8.4.2	0	0	"N"	r99ff99fr	"(0, 99)"	"(1,99,E)(0,99)"
9	9.1	theRoverEncountersAnObstacleOnTheEdges- OfThePlanetWhileMovingForwardRight	9.1.1	0	0	"N"	50fr99ff	"(99,50)"	"(98,50,N)(99,50). ^o "(98,50,E)(99,50)"

Tabla E.1 – continuación

US ID	C.P. ID	Nombre del caso de prueba	Assert ID	X	Y	Fa- cing	Comando	Obstáculos	Resultados espera- dos
9	9.1		9.1.2	0	0	"N"	99fr50fl	"(50, 99)"	"(49, 99, N)(50, 99). ^o "
9	9.1		9.1.3	0	0	"N"	r50fl	"(50, 0)"	"(49, 99, E)(50, 99)" "(49, 0, N)(50, 0). ^o " "(49, 0, E)(50, 0)"
9	9.2	theRoverEncountersAnObstacleOnThe- CornersOfThePlanetWhileMovingBackwards- Right	9.2.1	0	0	"N"	99fl99b	"(99, 99)"	"(98, 99, W)(99, 99)"
9	9.2		9.2.2	0	0	"N"	l99b	"(99, 0)"	"(98, 0, W)(99, 0)"
9	9.3	theRoverEncountersAnObstacleOnTheEdges- OfThePlanetWhileMovingBackwardRight	9.3.1	0	0	"N"	50fl99b	"(99, 50)"	"(98, 50, W)(99, 50)"
9	9.3		9.3.2	0	0	"N"	99fl50b	"(50, 99)"	"(49, 99, W)(50, 99)"
9	9.3		9.3.3	0	0	"N"	l50b	"(50, 0)"	"(49, 0, W)(50, 0)"
10	10.1	theRoverEncountersAnObstacleOnTheEdges- OfThePlanetWhileMovingBackwardLeft	10.1.1	0	0	"N"	rfl50frb	"(0, 50)"	"(1, 50, E)(0, 50)"
10	10.1		10.1.2	0	0	"N"	r99fl99fr50b	"(50, 99)"	"(51, 99, E)(50, 99)"
10	10.1		10.1.3	0	0	"N"	fr99frfl50b	"(50, 0)"	"(51, 0, E)(50, 0)"
10	10.2	theRoverEncountersAnObstacleOnTheEdges- OfThePlanetWhileMovingForwardLeft	10.2.1	0	0	"N"	rfl50flfr	"(0, 50)"	"(1, 50, N)(0, 50). ^o " "(1, 50, W)(0, 50)"
10	10.2		10.2.2	0	0	"N"	r99fl99fl50fr	"(50, 99)"	"(51, 99, N)(50, 99). ^o " "(51, 99, W)(50, 99)"
10	10.2		10.2.3	0	0	"N"	fr99flb150fr	"(50, 0)"	"(51, 0, N)(50, 0). ^o " "(51, 0, W)(50, 0)"
10	10.3	theRoverEncountersAnObstacleOnThe- Cor- nersOfThePlanet WhileMovingForwardLeft	10.3.1	0	0	"N"	rfl99flf	"(0, 99)"	"(1, 99, W)(0, 99)"
11	11.1	roverEncountersAnObstacleWhen Tries To- SpawnEastGoingBackward	11.1.1	0	0	"N"	30flflb	"(99, 30)"	"(0, 30, E)(99, 30)"
11	11.2	roverEncountersAnObstacleWhen Tries To- SpawnWestGoingForward	11.2.1	0	0	"N"	r99fl30frf	"(0, 30)"	"(99, 30, E)(0, 30)"
11	11.3	roverEncountersAnObstacleWhen Tries To- SpawnSouthWhileGoingForward	11.3.1	0	0	"N"	99frflf	"(1, 0)"	"(1, 99, N)(1, 0)"
11	11.4	roverEncountersAnObstacleWhen Tries To- SpawnWestGoingBackward	11.4.1	0	0	"N"	r99fl30frflb	"(0, 30)"	"(99, 30, W)(0, 30)"
11	11.5	roverEncountersAnObstacleWhen Tries To- SpawnSouthWhileGoingBackward	11.5.1	0	0	"N"	99frflflb	"(1, 0)"	"(1, 99, S)(1, 0)"
11	11.6	roverEncountersAnObstacleWhen Tries To- SpawnNorthWhileGoingForward	11.6.1	0	0	"N"	llf	"(0, 99)"	"(0, 0, S)(0, 99)"
11	11.7	roverEncountersAnObstacleWhen Tries To- SpawnEastGoingForward	11.7.1	0	0	"N"	30flf	"(99, 30)"	"(0, 30, W)(99, 30)"
11	11.8	roverEncountersAnObstacleWhen Tries To- SpawnNorthWhileGoingBackward	11.8.1	0	0	"N"	b	"(0, 99)"	"(0, 0, N)(0, 99)"

E.1.2. Casos de prueba Ad hoc- BSK

Tabla E.2: Casos de prueba Ad hoc tarea: BSK, Clase Frame

US ID	Test Case ID	Nombre del caso de prueba	Assert ID	T1	T2	Nombre del frame	Resultados esperados
1	1.1	testEmptyFramesCreated	1.1.1	"1"	"1"	NA	Frame no nulo
1	1.2	testFrameWithScoreIsCreated	1.2.1	1	2	NA	Frame no nulo
1	1.3	testFramesCreatedWithCorrectName	1.3.1	"1"	"1"	"Frame"	Nombre de la clase igual a "Frame"
2	2.1	testFrameScore	2.1.1	2	4	NA	Score de frame = 6
2	2.1		2.1.2	0	0	NA	Score de frame = 0
2	2.2	testAllPinsFrameScore	2.2.1	10	0	NA	Score de frame = 10

Tabla E.3: Casos de prueba Ad hoc tarea: BSK, clase Bowling Game

US\$ ID	Test Case ID	Nombre del caso de prueba	Assert ID	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	Bonus	Resultados esperados
3	3.1	testEmptyGameIsCreated	3.1.1	(1,1)	(2,2)	(3,3)	(4,4)	(5,5)	(6,6)	(7,7)	(8,8)	(9,9)	(10,10)	NA	NA	Game no nulo
3	3.2	testGameWithFramesIsCreated	3.2.1	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	NA	NA	Game no nulo
4	4.1	testGetFrame	4.1.1	(1,2)	(5,4)	(4,3)	NA	NA	NA	NA	NA	NA	NA	NA	NA	Fl.tiro1=1
4	4.1		4.1.2	(1,2)	(5,4)	(4,3)	NA	NA	NA	NA	NA	NA	NA	NA	NA	Fl.tiro1=2
4	4.1		4.1.3	(1,2)	(5,4)	(4,3)	NA	NA	NA	NA	NA	NA	NA	NA	NA	F2.tiro1=5
4	4.1		4.1.4	(1,2)	(5,4)	(4,3)	NA	NA	NA	NA	NA	NA	NA	NA	NA	F2.tiro2=4
4	4.1		4.1.5	(1,2)	(5,4)	(4,3)	NA	NA	NA	NA	NA	NA	NA	NA	NA	F3.tiro1=4
4	4.1		4.1.6	(1,2)	(5,4)	(4,3)	NA	NA	NA	NA	NA	NA	NA	NA	NA	F3.tiro2=3
4	4.2	testAddFrame	4.2.1	(1,0)	(1,0)	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	Fl.tiro1=F2.tiro1
4	4.2		4.2.2	(1,0)	(1,0)	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	Fl.tiro2=F2.tiro2
4	4.3	testGameScore	4.3.1	(1,2)	(2,3)	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	Game score=8
4	4.3		4.3.2	(1,2)	(2,3)	(5,4)	NA	NA	NA	NA	NA	NA	NA	NA	NA	Game score=17
5	5.1	testStrikeInTheMiddleForm	5.1.1	(4,5)	(2,4)	(10,0)	(1,8)	(0,5)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=72
5	5.2	testStartWithAStrikeForm	5.2.1	(10,0)	(2,0)	(0,0)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=52
5	5.3	testStrikeInTheMiddleContent	5.3.1	(5,1)	(7,2)	(10,0)	(9,0)	(0,5)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=72
5	5.4	testSingleStrikeZeroContentForm	5.4.1	(10,0)	(0,0)	(0,0)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=48
5	5.5	testStartWithAStrikeContent	5.5.1	(10,0)	(2,5)	(0,0)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=62
6	6.1	testStartWithASpareForm	6.1.1	(5,5)	(2,5)	(1,1)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=59
6	6.2	testSingleSpareZeroBonusForm	6.2.1	(5,5)	(0,0)	(1,1)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=50
6	6.3	testSpareInTheMiddleForm	6.3.1	(4,5)	(2,4)	(5,5)	(1,8)	(0,5)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=64
6	6.4	testSpareInTheMiddleContent	6.4.1	(5,1)	(7,2)	(0,10)	(0,1)	(0,5)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=55
6	6.5	testStartWithASpareContent	6.5.1	(4,6)	(2,5)	(0,0)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=57
6	6.6	testSpareWithFirstThrowZeroForm	6.6.1	(0,10)	(2,4)	(5,5)	(1,8)	(0,5)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=67
7	7.1	testSpareAfterStrikeAtTheBeginningZeroBonusContent	7.1.1	(10,0)	(2,8)	(0,0)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=68
7	7.2	testSpareAfterStrikeInTheMiddleZeroBonusForm	7.2.1	(1,5)	(10,0)	(2,8)	(0,0)	(0,0)	(0,0)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=55
7	7.3	testSpareAfterStrikeAtTheBeginningForm	7.3.1	(10,0)	(2,8)	(1,0)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=70
7	7.4	testDoubleStrikeDoubleSpareForm	7.4.1	(10,0)	(10,0)	(0,10)	(2,8)	(1,3)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=91
7	7.5	testSpareAfterStrikeInTheMiddleContent	7.5.1	(3,5)	(10,0)	(10,0)	(1,9)	(2,5)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=92
7	7.6	testSpareAfterStrikeInTheMiddleForm	7.6.1	(4,5)	(10,0)	(10,0)	(1,9)	(3,7)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=99
7	7.7	testSpareAfterStrikeInTheMiddleZeroBonusForm	7.7.1	(1,5)	(10,0)	(1,9)	(0,0)	(0,0)	(1,4)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=60
7	7.8	testSpareAfterStrikeAtTheBeginningContent	7.8.1	(10,0)	(0,10)	(1,2)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=72
8	8.1	testMultipleStrikeInTheMiddleZeroBonusForm	8.1.1	(1,5)	(1,4)	(10,0)	(10,0)	(10,0)	(0,0)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=90

Tabla E.3 – continuación

U\$ ID	Test Case ID	Nombre del caso de prueba	Asset ID	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	Bonus	Resultados esperados
8	8.2	testStartWithAMultipleStrikesContent	8.2.1	(10,0)	(10,0)	(1,2)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=75
8	8.3	testStartWithAMultipleStrikesForm	8.3.1	(10,0)	(10,0)	(0,0)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=68
8	8.4	testMultipleStrikesInTheMiddleForm	8.4.1	(4,5)	(2,4)	(10,0)	(10,0)	(10,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=106
8	8.5	testMultipleStrikeInTheMiddleContent	8.5.1	(5,1)	(7,2)	(10,0)	(10,0)	(10,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=106
9	9.1	testMultipleSpareInTheMiddleContent	9.1.1	(5,1)	(7,2)	(0,10)	(4,6)	(0,5)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=68
9	9.2	testStartWithMultipleSpareContent	9.2.1	(4,6)	(3,7)	(0,10)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=75
9	9.3	testMultipleSpareInTheMiddle	9.3.1	(1,5)	(1,4)	(4,6)	(4,6)	(0,10)	(1,9)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=76
9	9.4	testMultipleSpareInTheMiddleForm	9.4.1	(5,5)	(1,9)	(1,1)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=62
9	9.5	testStartWithMultipleSpareForm	9.5.1	(5,5)	(1,9)	(1,1)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,1)	NA	NA	Game score=62
10	10.1	testLastSpareForm	10.1.1	(1,5)	(2,5)	(1,1)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,5)	(4,0)	NA	Game score=61 o
10	10.2	testLastSpareZeroBonusContent	10.2.1	(1,5)	(2,5)	(1,1)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(5,5)	(4,0)	NA	Game score=61 o
11	11.1	testLastStrikeContent	11.1.1	(1,5)	(2,5)	(1,1)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(4,6)	NA	(0,0)	Game score=57 o
11	11.2	testLastStrikeZeroBonus	11.2.1	(1,5)	(2,5)	(1,1)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(4,6)	NA	(0,0)	Game score=57
12	12.1	testBonusIsStrikeWhenLastIsStrike_WithoutBonusThrowsLastIsStrikeContent	12.1.1	(1,5)	(2,5)	(1,1)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(10,0)	(10,0)	NA	Game score=57 o
12	12.2	testBonusIsStrikeWhenLastIsStrikeContent	12.2.1	(1,5)	(2,5)	(1,1)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(10,0)	(10,0)	NA	Game score=67 o
12	12.3	testBonusIsStrikeWhenLastIsSpareContent	12.3.1	(1,5)	(2,5)	(1,1)	(4,2)	(8,0)	(2,3)	(1,3)	(1,6)	(2,0)	(10,0)	(10,0)	NA	Game score=67
13	13.1	testBestScoresAllSparesContent	13.1.1	(9,1)	(9,1)	(9,1)	(9,1)	(9,1)	(9,1)	(9,1)	(9,1)	(9,1)	(9,1)	NA	(10,0)	Game score=191 o
13	13.2	testBestScoresAllStrikesContent	13.2.1	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,10)	NA	Game score=300 o
				(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	NA	(10,10)	Game score=300

E.2. Casos de prueba con PCdE

E.2.1. Clases de equivalencia - MR

Tabla E.4: Definición de clases de equivalencia de la tarea Mars Rover

Condición externa	ID Clase	Clases válidas	ID Clase	Clases no válidas
1. Ancho del grid	1	Entero positivo > 0	2	ancho = 0
			3	ancho < 0
			4	Decimal
			5	Carácter
			6	Null
2. Alto del grid	7	Entero positivo > 0	8	Alto=0
			9	Alto <0
			10	Decimal
			11	Carácter
			12	Null
3. Posición inicial Rover coordenada X	13	Entero positivo tal que: $X \geq 0$ y $X < \text{Ancho del grid}$	14	Entero < 0
			15	Entero $\geq$ ancho del grid
			16	Decimal positivo y/o negativo
			17	Carácter
			18	Null
4. Posición inicial Rover coordenada Y	19	Entero positivo tal que: $Y \geq 0$ y $Y < \text{Alto del grid}$	20	Entero < 0
			21	Entero $\geq$ alto del grid
			22	Decimal positivo y/o negativo
			23	Carácter
			24	Null
5. Facing inicial del Rover	25	Carácter N	33	Valor numérico
	26	Carácter S	34	Carácter diferente a válido
	27	Carácter E	35	Más de un carácter válido
	28	Carácter W	36	Null
	29	Carácter n		
	30	Carácter s		
	31	Carácter e		
	32	Carácter w		
6. Array de caracteres de comandos	37	Sin movimiento	49	String de carácter diferente a válido
	38	Carácter f	50	Valor numérico
	39	Carácter b	51	Combinación carácter válido y no válido
	40	Carácter l	52	Null
	41	Carácter r		

Tabla E.4 – continuación

Condición externa	ID Clase	Clases válidas	ID Clase	Clases no válidas
	42	Carácter F		
	43	Carácter B		
	44	Carácter L		
	45	Carácter R		
	46	Combinación aleatoria de caracteres válidos minúscula (fblr)		
	47	Combinación aleatoria de caracteres válidos mayúscula (FBLR)		
	48	Combinación aleatoria de caracteres válidos mayúscula y minúscula (FfBbLlRr)		
7. Wrapping from one edge of the grid to another	53	Sin wrapping		
	54	Dimensión de grid válido, coordenada inicial del rover válido y movimiento válido que ocasione wrapping de derecha a izquierda del grid.		
	55	Dimensión de grid válido, coordenada inicial del rover válido y movimiento válido que ocasione wrapping de izquierda a derecha del grid.		
	56	Dimensión de grid válido, coordenada inicial del rover válido y movimiento válido que ocasione wrapping de arriba a abajo del grid.		
	57	Dimensión de grid válido, coordenada inicial del rover válido y movimiento válido que ocasione wrapping de abajo a arriba del grid.		
	58	Dimensión de grid válido, coordenada inicial del rover válido y movimiento válido que ocasione wrapping combinado abajo-arriba, arriba-abajo, der-izq, izq-der del grid		
8. Concatena obstáculos	59	Sin obstáculos	62	Obstáculos inválidos
	60	Un obstáculo válido	63	Combinación de obstáculos válidos e inválidos
	61	Múltiples obstáculos válidos	64	Obstáculo válido en posición inicial válida del rover
9. Coordenada X de un obstáculo	65	Entero positivo tal que: $X \geq 0$ y $X < \text{ancho del grid}$	66	Entero < 0
			67	Entero $\geq$ alto del grid
			68	Decimal positivo y/o negativo
			69	Carácter
			70	Null

Tabla E.4 – continuación

Condición externa	ID Clase	Clases válidas	ID Clase	Clases no válidas
10. Coordenada Y de un obstáculo	71	Entero positivo tal que: $Y \geq 0$ y $Y < \text{ancho del grid}$	72	Entero < 0
			73	Entero $\geq$ alto del grid
			74	Decimal positivo y/o negativo
			75	Carácter
			76	Null

E.2.2. Casos de prueba - MR

Tabla E.5: Casos de prueba con PCdE de la tarea: MR

US ID	C.P. ID	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	Ancho	Alto	X	Y	Facing inicial	Comando	Obstáculos	Resultado esperado
0	24	2	7	13	19	27	37	53	59	-	-	0	3	0	2	"E"	""	""	Excepción Ancho de grid inválido
0	25	3	7	13	19	25	37	53	59	-	-	-3	3	2	2	"N"	""	""	Excepción Ancho de grid inválido
0	29	1	8	13	19	25	37	53	59	-	-	10	0	7	0	"N"	""	""	Excepción Alto de grid inválido
0	30	1	9	13	19	26	37	53	59	-	-	3	-1	1	1	"S"	""	""	Excepción Alto de grid inválido
1	02	1	7	13	19	25	37	53	59	-	-	2	2	0	0	"N"	""	"(0,0.N)"	
1	03	1	7	13	19	26	37	53	59	-	-	10	10	9	9	"S"	""	"(9,9.S)"	
1	04	1	7	13	19	27	37	53	59	-	-	2	8	1	7	"E"	""	"(1,7.E)"	
1	05	1	7	13	19	28	37	53	59	-	-	10	1	4	0	"W"	""	"(4,0.W)"	
1	06	1	7	13	19	29	37	53	59	-	-	3	5	1	4	"N"	""	"(1,4.N)"	
1	07	1	7	13	19	30	37	53	59	-	-	2	5	0	2	"S"	""	"(0,2.S)"	
1	08	1	7	13	19	31	37	53	59	-	-	2	8	1	7	"E"	""	"(1,7.E)"	
1	09	1	7	13	19	32	37	53	59	-	-	5	5	4	0	"W"	""	"(4,0.W)"	
1	34	1	7	14	19	25	37	53	59	-	-	10	6	-9	5	"N"	""	""	Excepción Posición inicial X fuera del grid
1	35	1	7	15	19	26	37	53	59	-	-	10	10	10	9	"S"	""	""	Excepción Posición inicial X fuera del grid
1	39	1	7	13	20	26	37	53	59	-	-	20	10	5	-5	"S"	""	""	Excepción Posición inicial Y fuera del grid
1	40	1	7	13	21	25	37	53	59	-	-	100	100	1	100	"N"	""	""	Excepción Posición inicial Y fuera del grid
1	45	1	7	13	19	34	37	53	59	-	-	1	2	0	0	"X"	""	""	Excepción Facing inválido.
1	46	1	7	13	19	35	37	53	59	-	-	10	10	9	9	"EW"	""	""	Excepción Facing inválido.
1	47	1	7	13	19	36	37	53	59	-	-	6	3	1	1	null	""	""	Excepción Facing inválido.
2	10	1	7	13	19	25	38	53	59	-	-	2	2	0	0	"N"	"(0,1.N)"	"(0,1.N)"	
2	14	1	7	13	19	26	42	53	59	-	-	10	10	1	1	"S"	"(1,0.S)"	"(1,0.S)"	
3	11	1	7	13	19	26	39	53	59	-	-	10	10	9	7	"S"	"(9,8.S)"	"(9,8.S)"	
3	15	1	7	13	19	27	43	53	59	-	-	6	3	5	2	"E"	"(4,2.E)"	"(4,2.E)"	
4	13	1	7	13	19	28	41	53	59	-	-	10	1	4	0	"W"	"(4,0.N)"	"(4,0.N)"	
4	17	1	7	13	19	25	45	53	59	-	-	1	1	0	0	"N"	"(0,0.E)"	"(0,0.E)"	
5	12	1	7	13	19	27	40	53	59	-	-	2	8	1	7	"E"	"(1,7.N)"	"(1,7.N)"	
5	16	1	7	13	19	28	44	53	59	-	-	3	2	2	1	"W"	"(2,1.S)"	"(2,1.S)"	
6	18	1	7	13	19	25	46	53	59	-	-	3	5	1	1	"N"	"(2,2.E)"	"(2,2.E)"	
6	19	1	7	13	19	26	46	53	59	-	-	2	5	0	2	"S"	"(0,2.W)"	"(0,2.W)"	
6	20	1	7	13	19	27	46	53	59	-	-	2	8	1	7	"E"	"(0,4.S)"	"(0,4.S)"	
6	21	1	7	13	19	28	46	53	59	-	-	5	5	4	0	"W"	"(2,1.W)"	"(2,1.W)"	
6	22	1	7	13	19	25	47	53	59	-	-	3	20	0	6	"N"	"(2,7.S)"	"(2,7.S)"	
6	23	1	7	13	19	28	48	53	59	-	-	7	4	5	2	"W"	"(1,0.E)"	"(1,0.E)"	
6	48	1	7	13	19	25	49	53	59	-	-	12	11	8	8	"N"	"(1,0.E)"	"(1,0.E)"	Excepción movimiento inválido
6	50	1	7	13	19	27	51	53	59	-	-	5	5	3	3	"E"	"(1,0.E)"	"(1,0.E)"	Excepción movimiento inválido
6	51	1	7	13	19	28	52	53	59	-	-	5	10	4	6	"W"	null	"(0,0.S)"	Excepción movimiento inválido
7	54	1	7	13	19	28	46	56	59	-	-	10	2	8	0	"W"	"(0,0.S)"	"(0,0.S)"	
8	55	1	7	13	19	26	46	57	59	-	-	2	2	1	1	"S"	"(0,1.N)"	"(0,1.N)"	
9	52	1	7	13	19	25	46	54	59	-	-	3	5	1	0	"N"	"(0,2.E)"	"(0,2.E)"	
10	53	1	7	13	19	27	46	55	59	-	-	4	3	0	0	"E"	"(3,2.W)"	"(3,2.W)"	

Tabla E.5 – continuación

US ID	C.P. ID	C1	C2	C3	C4	C5	C6	C7	C8	C9	C 10	Ancho	Alto	X	Y	Facing ini- cial	Comando	Obstáculos	Resultados esperados
11	56	1	7	13	19	27	46	58	59	-	-	2	2	0	1	"E"	"blfrfrbblbfrbrb- frbrblbblbrflbrb"	"	"(0,0,W)"
12	57	1	7	13	19	25	46	-	60	65	71	10	10	1	7	"S"	"rfrff"	"(0,9)"	"(0,8,N)(0,9)"
12	58	1	7	13	19	27	46	-	61	65	71	5	5	4	4	"E"	"rfflbbbbb"	"(0,0)(1,1)(2,2) (3,3)(2,0)"	"(2,1,E)(1,1)"
12	59	1	7	13	19	28	46	-	61	65	71	10	10	9	9	"W"	"ffff"	"(8,9)(9,8)"	"(9,9,W)(8,9)"
12	60	1	7	13	19	26	46	-	62	65	-	4	3	1	0	"S"	"bblf"	"O1O2O3"	Excepción: Formato Obstáculo inválido
12	61	1	7	13	19	28	46	-	63	65	71	4	3	1	2	"W"	"blff"	"XYZ(2,1)"	Excepción: Formato Obstáculo inválido
12	62	1	7	13	19	25	46	-	64	65	71	3	5	2	1	"N"	"fffff"	"(2,1)"	Excepción: Rover inicializado sobre obstáculo
12	63	1	7	13	19	26	46	-	62	66	71	2	2	0	0	"S"	"blf"	"(-1,1)"	Excepción: Posición obstáculo inválida
12	64	1	7	13	19	27	46	-	62	67	71	7	2	3	1	"E"	"lflbb"	"(10,0)"	Excepción: Posición obstáculo inválida
12	68	1	7	13	19	28	46	-	62	65	72	10	10	5	5	"W"	"lllbbbbbbr"	"(9,-9)"	Excepción: Posición obstáculo inválida
12	69	1	7	13	19	25	46	-	62	65	73	7	5	2	2	"N"	"rffflbb"	"(1,20)"	Excepción: Posición obstáculo inválida

E.2.3. Clases de equivalencia - BSK

Tabla E.6: Definición de clases de equivalencia de la tarea BSK: Clase Frame

Condición externa	ID Clase	Clases válidas	ID Clase	Clases no válidas
1. Tiro 1 de un frame	1	Entero positivo ≥ 0 y ≤ 10	2	Entero < 0
			3	Entero > 10
			4	Decimal positivo y/o negativo
			5	Carácter
			6	Null
2. Tiro 2 de un frame	7	Entero positivo ≥ 0 y ≤ 10	8	Entero < 0
			9	Entero > 10
			10	Decimal positivo y/o negativo
			11	Carácter
			12	Null
3. Score de un frame ordinario	13	Tiro1 y tiro2 con valores válidos, tal que $((\text{tiro1} < > 10) + \text{tiro2}) \geq 0$ y ≤ 9	14	Tiro1 y tiro2 con valores válidos tales que $((\text{tiro1} < > 10) + \text{tiro2}) > 10$
			15	Tiro1 y/o tiro2 con valores inválidos tales que $((\text{tiro1} < > 10) + \text{tiro2}) \geq 0$ y ≤ 9
			16	Tiro1 y/o tiro2 con valores inválidos tales que $((\text{tiro1} < > 10) + \text{tiro2}) \geq 0$ y ≤ 9
			17	Tiro1 y/o tiro2 con valores inválidos tales que $((\text{tiro1} < > 10) + \text{tiro2}) > 10$
4. Identifica si el frame es spare	18	Tiro1 y tiro2 con valores válidos, tal que $(\text{tiro1} < > 10) + \text{tiro2} = 10$	19	Tiro1 y tiro2 con valores válidos tales que $((\text{tiro1} < > 10) + \text{tiro2}) < > 10$
			20	Tiro1 y/o tiro2 con valores inválidos tales que $((\text{tiro1} < > 10) + \text{tiro2}) = 10$
5. Identifica si el frame es strike	21	Tiro1 y tiro2 con valores válidos, tal que $\text{tiro1} = 10$ y $\text{tiro2} = 0$	22	Tiro1 y tiro2 con valores válidos tales que $(\text{tiro1} < > 10)$ y $(\text{tiro2} < > 0)$
			23	Tiro1 y/o tiro2 con valores inválidos tales que $\text{tiro1} = 10 + \text{tiro2} < > 0$
6. Cantidad de tiros de un frame	24	Frame con dos tiros	25	Frame con más de dos tiros válidos
			26	Frame=null

Tabla E.7: Definición de clases de equivalencia de la tarea BSK: Clase BowlingGame

Condición externa	ID Clase	Clases válidas	ID Clase	Clases no válidas
1. Composición de un single game de frames ordinarios	1	Single game con 10 frames sin bonus	2	Single game con 10 frames ordinarios válidos con bonus
			3	Single game con 10 frames ordinarios inválidos
			4	Single game con 10 frames ordinarios válidos e inválidos
			5	Single game con más de 10 frames ordinarios válidos
			6	Single game con 10 frames ordinarios = null
			7	Single game = null
2. Score de un single game (No spares y no strikes)	8	Score de single game = $\sum 10$ scores de frames ordinarios válidos tales que score del single game ≥ 0 y ≤ 90	9	Score de single game = $\sum 10$ frames ordinarios válidos e inválidos tales que score single game ≥ 0 y ≤ 90

Tabla E.7 – continuación

Condición externa	ID Clase	Clases válidas	ID Clase	Clases no válidas
			10	Score de single game = $\sum$ 10 frames ordinario válidos e inválidos tales que score single game > 90
			11	Score de single game = $\sum$ 10 frames ordinario válidos e inválidos tales que score single game < 0
			12	Score = null
3. Score de un game que incluye spare entre el frame 1 y 8 seguido de un frame ordinario y/o seguido de un frame spare	13	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame spare válido si y solo si Score de frame spare = 10 + tiro1 de frame ordinario siguiente válido tal que score de frame spare es ≥ 10 y < 20)	15	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame spare inválido si y solo si Score de frame spare = 10 + tiro1 de frame ordinario siguiente válido tal que score de frame spare es ≥ 10 y < 20)
	14	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame spare válido si y solo si Score de frame spare = 10 + tiro1 de frame ordinario siguiente válido tal que score de frame spare es ≥ 10 y < 20) + (Score de frame spare 2)	16	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame spare válido cuando Score de frame spare = 10 + frame ordinario siguiente inválido tal que score de frame spare > 20)
			17	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame spare válido cuando Score de frame spare = 10 + frame ordinario siguiente inválido tal que score de frame spare < 10)
			18	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame spare inválido si y solo si Score de frame spare = 10 + tiro1 de frame ordinario siguiente válido tal que score de frame spare ≥ 10 y < 20) + (Score de frame spare 2)
			19	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame spare válido si y solo si Score de frame spare = 10 + tiro1 de frame ordinario siguiente inválido tal que score de frame spare < 10) + (Score de frame spare 2)
			20	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame spare válido si y solo si Score de frame spare = 10 + tiro1 de frame ordinario siguiente inválido tal que score de frame spare > 20) + (Score de frame spare 2)
4. Score de un game que incluye un strike entre el frame 1 y 9 seguido de un frame ordinario	21	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame strike correcto si y solo si Score de frame strike = 10 + score de frame ordinario siguiente válido tal que score de frame Strike es ≥ 10 y < 20)	22	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame strike incorrecto cuando Score de frame Strike = 10 + score de frame siguiente inválido tal que score de frame Strike ≥ 10 y < 20)
			23	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame strike incorrecto cuando Score de frame Strike = 10 + score de frame siguiente inválido tal que score de frame Strike > 20)
			24	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame strike incorrecto cuando Score de frame Spare = 10 + score de frame siguiente inválido tal que score de frame Strike < 10)

Tabla E.7 – continuación

Condición externa	ID Clase	Clases válidas	ID Clase	Clases no válidas
5. Score de un game que incluye un spare seguido de strike y/o strike seguido de spare entre el frame 1 y 8	25	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame spare correcto seguido de strike correcto si y solo si Score de frame spare = 20) + ($\sum$ score de frame Strike válido)	27	Score de bowling game = ($\sum$ Score de frames ordinarios válidos) + (Score de frame spare incorrecto seguido de Strike correcto con score de frame Spare = 20) + (score de frame Strike)
	26	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame strike correcto seguido de spare correcto si y solo si Score de frame strike = 20) + ($\sum$ score de frame Spare válido)	28	Score de bowling game = ($\sum$ Score de frames ordinarios válidos) + (Score de frame strike incorrecto seguido de Spare incorrecto con score de frame Strike = 20) + (score de frame Spare)
6. Score de un game que incluye strikes consecutivos entre el frame 1 y 9	29	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame Strike1 correcto si y solo si Score de frame Strike1 = 20 + (Tiro1 válido de frame posición siguiente + 1) tal que score de frame Strike $\geq$ 20 y $\leq$ 30) + (Score de frame Strike 2 correcto)	30	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame Strike1 incorrecto cuando Score de frame Strike1 = 20 + (Tiro1 de frame posición siguiente + 1) inválido tal que score de frame Strike1 $>$ 30) + (Score de frame Strike 2)
			31	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + (Score de frame Strike1 incorrecto cuando Score de frame Strike1 = 20 + (Tiro1 de frame posición siguiente + 1) inválido tal que score de frame Strike1 $<$ 20) + (Score de frame Strike 2)
7. Estructura de un game con bonus	32	Game con un frameBonus donde solo admite valores en Tiro1 cuando lastFrame es Spare	35	Game sin frameBonus cuando lastFrame es Spare válido
	33	Game con un frameBonus ordinario cuando lastFrame es Strike	36	Game con 1 frameBonus cuando lastFrame es Spare válido con valor de Tiro2
	34	Game con 2 frameBonus cuando lastFrame es Strike y primer frameBonus es Strike, donde solo admite valores en tiro1 de frameBonus2	37	Game sin frameBonus cuando lastFrame es Strike.
			38	Game con 1 frameBonus cuando lastFrame es Strike y frameBonus1 es Strike
			39	Game con 2 frameBonus cuando lastFrame es Strike y frameBonus1 es Strike, que admite valores en Tiro2 de frameBonus2
8. Score de un game con Spare o Strike en lastFrame	40	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Spare válido = 10 + Tiro1 válido de bonusFrame $\geq$ 0 y $<$ 10 y tiro2 = 0 tal que score de LastFrame Spare $\geq$ 10 y $<$ 20)	45	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Spare inválido = 10 + Tiro1 válido de bonusFrame $<$ 10 y tiro2 = 0 tal que score de LastFrame Spare $\geq$ 10 y $<$ 20)
	41	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Spare válido = 10 + Tiro1 válido de bonusFrame = 10 y Tiro2 = 0 tal que score de LastFrame Spare = 20)	46	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Spare = 10 + Tiro1 inválido de bonusFrame $<$ 10 y tiro2 = 0 tal que score de LastFrame Spare $<$ 10)

Tabla E.7 – continuación

Condición externa	ID Clase	Clases válidas	ID Clase	Clases no válidas
	42	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Strike válido = 10 + score de frameBonus ordinario válido tal que score de LastFrame Strike ≥ 10 y < 20)	47	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Spare = 10 + Tiro1 inválido de bonusFrame < 10 y tiro2 = 0 tal que score de LastFrame Spare > 20)
	43	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Strike válido = 10 + (tiro1 + tiro2 de frameBonus Spare válido) tal que score de LastFrame Strike = 20)	48	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Strike = 10 + score de frameBonus ordinario inválido tal que score de LastFrame Strike ≥ 10 y < 20)
	44	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Strike válido = 20 + (tiro1 de frameBonus2 válido) tal que score de LastFrame Strike ≥ 20 y ≤ 30 cuando frameBonus1 es Strike)	49	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Strike = 10 + score de frameBonus ordinario inválido tal que score de LastFrame Strike < 10)
			50	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Strike = 10 + score de frameBonus ordinario inválido tal que score de LastFrame Strike > 20)
			51	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Strike = 10 + (tiro1 + tiro2 de spare frameBonus inválido) tal que score de LastFrame Strike = 20)
			52	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Strike = 20 + (tiro1 de frame Bonus2 inválido) tal que score de LastFrame Strike < 20 cuando frameBonus1 es strike)
			53	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Strike = 20 + (tiro1 de frame Bonus2 inválido) tal que score de LastFrame Strike > 30 cuando frameBonus1 es strike)
9. Best Score	54	Game con 10 frame Strike válidos y 2 frameBonus strike válidos = 20 tal que Best Game score = 300	55	Game con 10 frames no strikes con 2 frameBonus válidos tal que game Score = 300
			56	Game con 10 frames No Strikes sin bonus tal que game Score = 300

Tabla E.7 – continuación

Condición externa	ID Clase	Clases válidas	ID Clase	Clases no válidas
10. Score de un real game	57	Score de bowling game = ($\sum$ score de frames ordinarios válidos) + ($\sum$ Score de frame Strike válidos) + ($\sum$ Score de frame Spare válidos) + (score de lastFrame Spare o Strike válido) tal que score Bowling Game ≥ 0 y ≤ 300	58	Score de bowling game = ($\sum$ score de frames ordinarios válidos o Inválidos) + ($\sum$ Score de frame Strike válidos o inválidos) + ($\sum$ Score de frame Spare válidos o inválidos) + (score de lastFrame Spare o Strike correcto o incorrecto) tal que score Bowling Game ≥ 0 y ≤ 300
			59	Score de bowling game = ($\sum$ score de frames ordinarios válidos o Inválidos) + ($\sum$ Score de frame Strike válidos o inválidos) + ($\sum$ Score de frame Spare válidos o inválidos) + (score de lastFrame Spare o Strike correcto o incorrecto) tal que score Bowling Game < 0
			60	Score de bowling game = ($\sum$ score de frames ordinarios válidos o Inválidos) + ($\sum$ Score de frame Strike válidos o inválidos) + ($\sum$ Score de frame Spare válidos o inválidos) + (score de lastFrame Spare o Strike correcto o incorrecto) tal que score Bowling Game < 0

E.2.4. Casos de prueba - BSK

Tabla E.8: Casos de prueba con PCdE de la tarea: BSK, clase Frame

US ID	C.P. ID	C1	C2	C3	C4	C5	C6	T1	T2	Resultados esperados
1	1	1	7	-	-	-	24	9	0	throw1=9, throw2=0
1	2	2	7	-	-	-	24	-1	7	Excepción: throw1 inválido
1	3	3	7	-	-	-	24	11	0	Excepción: throw1 inválido
1	7	1	8	-	-	-	24	1	-9	Excepción: throw2 inválido
1	8	1	9	-	-	-	24	0	20	Excepción: throw2 inválido
2	12	1	7	13	-	-	24	4	5	Score de un solo frame = 9
2	13	1	7	14	-	-	24	6	5	Excepción: throws inválidos
2	14	2	9	15	-	-	24	-5	12	Excepción: cantidad de pins inválido.
2	15	3	8	16	-	-	24	11	-15	Excepción: cantidad de pins inválido.
2	16	3	9	17	-	-	24	12	11	Excepción: cantidad de pins inválido.
5	20	1	7	-	-	21	24	10	0	strike=true
5	21	1	7	-	-	22	24	7	0	strike=false
5	22	1	7	-	-	23	24	10	1	strike=false
6	17	1	7	-	18	-	24	0	10	spare=true
6	18	1	7	-	19	-	24	9	0	spare=false
6	19	3	8	-	20	-	24	20	-10	spare=false

Tabla E.9: Casos de prueba con PCdE de la tarea: BSK, clase BowlingGame

US ID	Test Case ID	C1	C2	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	Fn	Bonus 1	Bonus 2	BonusResultado esperado
3	25	1		(0,0)	(4,5)	(0,9)	(7,2)	(8,1)	(2,3)	(6,2)	(5,4)	(4,0)	(3,5)	NA	NA	NA	NA	Game size=10 frames
3	26	2		(4,4)	(9,0)	(7,2)	(6,9)	(3,3)	(3,3)	(5,1)	(1,3)	(1,1)	(6,2)	NA	NA	(5)	NA	Estructura de juego inválida
3	27	3		(20,20)	(10,10)	(-2,-2)	(-3,-3)	(-4,-4)	(-5,-5)	(6,6)	(7,7)	(8,8)	(9,9)	NA	NA	NA	NA	Excepción frames inválidos
3	28	4		(0,0)	(0,9)	(-1,11)	(2,7)	(3,6)	(4,5)	(-5,4)	(6,3)	(7,2)	(8,8)	NA	NA	NA	NA	Excepción frames inválidos
3	29	5		(0,0)	(0,9)	(1,8)	(2,7)	(3,6)	(4,5)	(5,4)	(6,3)	(7,2)	(8,1)	(9,0)	(1,1)	NA	NA	Excepción frame inválido
3	30	6		null	null	null	null	null	null	null	null	null	null	NA	NA	NA	NA	Excepción frames inválidos
4	32	1	8	(2,7)	(3,6)	(4,5)	(5,4)	(6,3)	(7,2)	(8,1)	(9,0)	(1,1)	(0,9)	NA	NA	NA	NA	Game = 10 frames y Game score = 83
4	33	1	9	(20,20)	(-10,-10)	(-2,-2)	(-3,-3)	(-4,-4)	(-5,-5)	(6,6)	(7,7)	(8,8)	(9,9)	NA	NA	NA	NA	Excepción frames inválidos
4	34	1	10	(0,0)	(0,9)	(9,0)	(2,7)	(3,6)	(4,5)	(-5,41)	(6,3)	(7,2)	(8,1)	NA	NA	NA	NA	Excepción frame inválido
4	35	1	11	(8,0)	(7,1)	(2,2)	(1,1)	(2,3)	(4,3)	(3,0)	(6,6)	(-4,4)	(-25,-25)	NA	NA	NA	NA	Excepción frames inválidos
5	45	1	21	(3,6)	(4,5)	(10,0)	(2,1)	(6,3)	(5,0)	(1,8)	(1,6)	(0,0)	(2,2)	NA	NA	NA	NA	Game score = 68
5	46	1	22	(2,5)	(10,0)	(-5,7)	(1,8)	(7,2)	(5,4)	(4,4)	(1,3)	(4,0)	(8,1)	NA	NA	NA	NA	Excepción frames inválidos
5	47	1	23	(0,0)	(3,6)	(2,2)	(2,7)	(3,2)	(10,0)	(7,5)	(1,8)	(4,4)	(1,3)	NA	NA	NA	NA	Excepción frame inválido
5	48	1	24	(10,0)	(4,-5)	(0,9)	(2,7)	(8,1)	(2,3)	(6,2)	(5,4)	(4,0)	(3,5)	NA	NA	NA	NA	Excepción frame inválido
6	37	1	13	(3,3)	(0,0)	(0,9)	(7,2)	(9,0)	(5,5)	(6,2)	(1,8)	(2,5)	(1,1)	NA	NA	NA	NA	Game score = 75
6	39	1	15	(6,2)	(1,8)	(2,5)	(0,5)	(-9,19)	(2,5)	(6,3)	(0,0)	(0,9)	(7,2)	NA	NA	NA	NA	Excepción frame inválido
6	40	1	16	(1,1)	(3,6)	(0,10)	(20,0)	(3,3)	(7,0)	(3,6)	(8,1)	(7,1)	(3,2)	NA	NA	NA	NA	Excepción frame inválido
6	41	1	17	(5,3)	(4,4)	(1,1)	(2,6)	(5,1)	(3,4)	(7,2)	(0,8)	(0,10)	(-9,1)	NA	NA	NA	NA	Excepción frame inválido
7	49	1	25	(4,4)	(9,0)	(7,2)	(6,4)	(10,0)	(3,3)	(5,1)	(1,3)	(1,1)	(6,2)	NA	NA	NA	NA	Game score= 88
7	50	1	26	(2,3)	(6,2)	(5,4)	(4,0)	(3,3)	(0,0)	(10,0)	(0,10)	(7,2)	(8,1)	NA	NA	NA	NA	Game score= 87
7	51	1	27	(3,3)	(-15,25)	(10,0)	(1,1)	(6,2)	(1,8)	(2,5)	(7,2)	(0,9)	(2,7)	NA	NA	NA	NA	Excepción frame inválido
7	52	1	28	(5,3)	(4,4)	(1,1)	(2,6)	(10,0)	(12,-2)	(2,5)	(6,2)	(5,4)	(4,0)	NA	NA	NA	NA	Excepción frame inválido
8	53	1	29	(10,0)	(10,0)	(1,3)	(1,1)	(6,2)	(1,8)	(2,5)	(7,2)	(0,9)	(2,7)	NA	NA	NA	NA	Game score= 92
8	54	1	30	(5,3)	(4,4)	(1,1)	(10,0)	(10,0)	(13,-7)	(1,8)	(2,5)	(7,2)	(4,0)	NA	NA	NA	NA	Excepción frame inválido
8	55	1	31	(6,2)	(1,8)	(2,5)	(7,2)	(1,1)	(10,0)	(10,0)	(-5,8)	(6,2)	(7,2)	NA	NA	NA	NA	Excepción frame inválido
9	38	1	14	(6,2)	(1,8)	(2,5)	(5,5)	(9,1)	(0,5)	(6,3)	(0,0)	(0,9)	(7,2)	NA	NA	NA	NA	Game score=85
9	42	1	18	(11,-1)	(3,7)	(3,6)	(7,2)	(3,6)	(4,4)	(5,3)	(3,3)	(4,5)	(8,1)	NA	NA	NA	NA	Excepción frame inválido
9	43	1	19	(0,0)	(3,6)	(2,2)	(2,7)	(6,2)	(1,1)	(6,4)	(-5,15)	(4,0)	(6,1)	NA	NA	NA	NA	Excepción frame inválido
9	44	1	20	(5,4)	(2,4)	(4,3)	(9,0)	(8,2)	(17,-7)	(3,3)	(0,0)	(2,6)	(1,1)	NA	NA	NA	NA	Excepción frame inválido
10	56	32		(6,2)	(1,8)	(2,5)	(6,2)	(1,3)	(1,1)	(2,5)	(7,2)	(0,9)	(7,3)	NA	NA	(8)	NA	Game score=81
10	59	35		(2,5)	(7,2)	(0,9)	(2,6)	(6,2)	(6,2)	(1,8)	(2,5)	(7,2)	(5,5)	NA	NA	null	NA	Excepción Game inválido
10	60	36		(2,5)	(6,2)	(5,4)	(1,1)	(6,2)	(1,8)	(2,5)	(7,2)	(0,9)	(0,10)	NA	NA	(7,2)	NA	Excepción Game inválido
10	64	40		(6,2)	(1,8)	(2,5)	(6,2)	(1,3)	(1,1)	(2,5)	(7,2)	(0,9)	(7,3)	NA	NA	(8,0)	NA	Game score=81

Tabla E.9 – continuación

US ID	Test ID	C1	C2	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	Fn	Bonus 1	Bonus 2	Resultados esperados
10	69	45		(5,3)	(4,4)	(1,1)	(2,6)	(6,2)	(1,8)	(2,5)	(6,2)	(5,4)	(11,-1)	NA	NA	(9,0)	NA	Excepción frame inválido
10	70	46		(7,2)	(0,9)	(2,6)	(6,2)	(6,2)	(1,8)	(2,5)	(1,1)	(6,2)	(2,8)	NA	NA	(-1,0)	NA	Excepción frame inválido
10	71	47		(6,2)	(5,4)	(1,1)	(6,2)	(1,8)	(2,5)	(7,2)	(2,6)	(6,2)	(9,1)	NA	NA	(11,0)	NA	Excepción frame inválido
11	57	33		(2,5)	(7,2)	(0,9)	(4,4)	(1,1)	(2,6)	(2,5)	(6,2)	(5,4)	(10,0)	NA	NA	(5,4)	NA	Estructura válida
11	58	34		(2,5)	(6,2)	(5,4)	(1,1)	(6,2)	(1,8)	(2,5)	(7,2)	(1,8)	(10,0)	NA	NA	(10,0)	(9,0)	Estructura válida
11	61	37		(2,6)	(6,2)	(1,8)	(2,5)	(6,2)	(5,4)	(6,2)	(5,4)	(4,0)	(10,0)	NA	NA	null	NA	Excepción Game inválido
11	63	39		(2,6)	(6,2)	(6,2)	(1,8)	(2,5)	(7,2)	(4,4)	(1,1)	(2,6)	(10,0)	NA	NA	(10,0)	(4,3)	Excepción Game inválido
11	66	42		(2,5)	(7,2)	(0,9)	(4,4)	(1,1)	(2,6)	(2,5)	(6,2)	(5,4)	(10,0)	NA	NA	(5,4)	NA	Game score=86
11	67	43		(5,3)	(4,4)	(1,1)	(2,6)	(6,2)	(1,8)	(2,5)	(6,2)	(5,4)	(10,0)	NA	NA	(0,10)	NA	Game score=87
11	68	44		(2,6)	(6,2)	(1,8)	(2,5)	(6,2)	(5,4)	(6,2)	(5,4)	(4,0)	(10,0)	NA	NA	(10,9)	NA	Game score=99
11	72	48		(7,2)	(0,9)	(5,1)	(1,3)	(1,1)	(5,3)	(4,4)	(1,1)	(2,6)	(10,0)	NA	NA	(-5,7)	NA	Excepción frame inválido
11	73	49		(2,5)	(7,2)	(4,4)	(1,1)	(2,6)	(2,6)	(6,2)	(1,8)	(2,5)	(10,0)	NA	NA	(-3,-3)	NA	Excepción frame inválido
11	74	50		(2,5)	(6,2)	(5,1)	(1,3)	(1,1)	(6,2)	(5,4)	(1,1)	(6,2)	(10,0)	NA	NA	(6,6)	NA	Excepción frame inválido
11	75	51		(1,3)	(1,1)	(2,5)	(7,2)	(0,9)	(7,2)	(0,9)	(2,6)	(6,2)	(10,0)	NA	NA	(-4,14)	NA	Excepción frame inválido
11	76	52		(6,2)	(1,8)	(2,5)	(7,2)	(0,9)	(6,2)	(5,4)	(1,1)	(6,2)	(10,0)	NA	NA	(10,0)	(-1)	Excepción frame inválido
11	77	53		(2,5)	(6,2)	(5,4)	(4,0)	(5,1)	(1,3)	(1,1)	(7,2)	(4,4)	(10,0)	NA	NA	(10,0)	(12)	Excepción frame inválido
12	65	41		(5,3)	(4,4)	(1,1)	(2,6)	(6,2)	(1,8)	(2,5)	(6,2)	(5,4)	(8,2)	NA	NA	(10,0)	NA	Game score = 87
13	78	54		(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	(10,0)	NA	NA	(10,0)	(10)	Game score=300
13	79	55		(10,10)	(10,10)	(15,15)	(15,15)	(15,15)	(15,15)	(15,15)	(15,15)	(25,25)	(10,0)	NA	NA	(10,0)	(10)	Excepción Game inválido
13	80	56		(15,15)	(15,15)	(15,15)	(15,15)	(15,15)	(15,15)	(15,15)	(15,15)	(15,15)	(15,15)	NA	NA	NA	NA	Excepción Game inválido
14	81	57		(7,3)	(2,6)	(10,0)	(5,5)	(10,0)	(10,0)	(1,1)	(2,6)	(6,2)	(7,3)	NA	NA	(9,0)	NA	Game score=130
14	82	58		(10,0)	(3,2)	(-2,-2)	(-3,-3)	(-4,-4)	(-5,5)	(6,6)	(7,7)	(14,-4)	(9,9)	NA	NA	NA	NA	Excepción frames inválidos
14	83	59		(10,0)	(3,2)	(-2,-2)	(-3,-3)	(-4,-4)	(-5,5)	(0,0)	(-1,-3)	(0,1)	(0,0)	NA	NA	NA	NA	Excepción frames inválidos
14	84	60		(2,7)	(3,6)	(4,5)	(5,4)	(6,3)	(7,2)	(8,25)	(9,0)	(100,100)	(0,9)	NA	NA	NA	NA	Excepción frames inválidos

Apéndice F

Código fuente para la medición

F.1. Métodos para medir con PCdE

F.1.1. Método gotoStart()

```
public void gotoStart(int x1, int y1, String facing){
    this.x=x1;
    this.y=y1;
    if (facing=="N" || facing=="n") this.direction=Direction.N;
    if (facing=="S" || facing=="s") this.direction=Direction.S;
    if (facing=="E" || facing=="e") this.direction=Direction.E;
    if (facing=="W" || facing=="w") this.direction=Direction.W;
}
```

Figura F.1: Método *gotoStart()*

F.1.2. Método getSize()

```
public int getSize() {
    return frames.size();
}
```

Figura F.2: Método *getSize()*

F.1.3. Método `public boolean findBonus()`

```
public boolean findBonus(){
    boolean findBonus;
    if(bonus != null){
        findBonus=true;
    }else{
        findBonus=false;
    }
    return findBonus;
}
```

Figura F.3: Método *findBonus()*

F.2. Ejemplo de intervención en la elaboración de la variable que retorna los valores de un método

Este ejemplo pertenece al sujeto FSK04 del experimento FSecure. En la figura F.4 presentamos el método intervenido: *executeCommand* de la tarea MR. En la línea 31 se observa el retorno de un valor nulo en el método *executeCommand*. Sin embargo, luego de un análisis del código se observa que el sujeto si ha desarrollado código fuente para controlar los movimientos del rover como se observa de la línea 8 a la línea 30.

El método *executeCommad* debería devolver un String con el formato “ $(x,y,facing)(x1,y1)(x2,y2)...(xn,yn)$ ” tal como se indicó en las especificaciones, entonces el medidor escribe el siguiente código en la línea 31 (Figura F.4).

Esta intervención en el código permite al método retornar la posición del Rover en el formato sugerido. De esta forma se conectan los casos de prueba y se obtienen las métricas.

```

1  public String executeCommand(String command){
2  char[] command_list = null;
3
4  if(command != null){
5      command_list = splitStringIntoCharacters(command);
6  }
7
8  for( int i = 0; i < command_list.length;i++)
9  {
10     switch(command_list[i]){
11         case 'f':
12             command_forward_x();
13             command_forward_y();
14             break;
15
16         case 'b':
17             command_reverse_x();
18             command_reverse_y();
19             break;
20
21         case 'l':
22             facing_command('l');
23             break;
24
25         case 'r':
26             facing_command('r');
27             break;
28     }
29 }
30 }
31 return null;
32 }

```

Figura F.4: Método *executeCommad()* del sujeto FSK04

```

}
return "("+ Integer.toString(posx).trim()+ "," +
        Integer.toString(posy).trim()+ ","+
        convierteFacing(facing) +")";
}

```

Figura F.5: Intervención en el *return* del método *executeCommad()* del sujeto FSK04

F.3. Ejemplo de intervención en la modificación del formato de salida de un método

En la figura F.6 presentamos el código fuente del sujeto PTE011 relativo a la tarea MR del experimento Playtech, el formato de salida de la sentencia *return* debe ser (x,y)(x1,y1)(x2,y2)...(xn,yn), sin embargo el sujeto ha escrito su código con el siguiente formato ?(x,y)(x1,y1)(x2,y2)...(xn,yn)?

```
String result = "?(" + row + ',' + column + ',' +
    direction + ')';
for (String obstacle : obstacleReport)
    result = result + '(' + obstacle + ')';
return result + '?';
```

Figura F.6: Sentencia *return* del método *executeCommand* del sujeto PTE011

El medidor elimina el carácter: ? para poner el valor de salida en el formato correcto. El nuevo código fuente lo mostramos en la Figura F.7:

```
//String result = "?(" + row + ',' + column + ',' +
//    direction + ')';
String result = "(" + row + ',' + column + ',' +
    direction + ')';
for (String obstacle : obstacleReport)
    result = result + '(' + obstacle + ')';
//return result + '?';
return result;
```

Figura F.7: Intervención en la sentencia *return* del método *executeCommand* del sujeto PTE011

Otro ejemplo, en la tarea MR, en el formato del Facing del Rover (orientación del Rover), de acuerdo al estándar, los valores puede ser: “N”, “S”, “E” y “W”. Sin embargo, el sujeto del ejemplo, ha puesto los valores: “NORTH”, “SOUTH”, “EAST” y “WEST”, como se muestra en el código fuente de la figura F.8. El código del ejemplo pertenece al sujeto FSK07 en la tarea MR del experimento FSecure, se observa como el sujeto usa los valores del Facing incorrectos (líneas 1, 5, 6, 8, 9, 11, 12, 14 y 15).

El medidor realiza la corrección, poniendo los valores correctos para la orientación del Rover, escribe en el código fuente los valores: “N”, “S”, “E” y “W”, tal como mostramos a continuación en la Figura F.9 (líneas 1, 5, 6, 8, 9, 11, 12, 14 y 15):

En la Figura F.10 presentamos otro ejemplo correspondiente al código fuente del sujeto ERHL19, de la tarea MR, del experimento Ericsson. En el Facing, el sujeto usa un número (int), en lugar de usar el estándar String. Esto daría un error al conectar el código a los

casos de prueba. Como mostramos en el siguiente código fuente, la variable que ha usado es *direction = 0* en la línea 1.

El medidor implementa el siguiente código fuente presentado en la Figura F.11 para pasar el valor de la variable numérica *direction* a la variable de texto: *directionresult*.

Tenemos otro ejemplo en la Figura F.12, el caso del sujeto FSOU2802 en la tarea MR del experimento FSecure. Tiene intercambiado el valor de X por Y y viceversa. El código original del sujeto estaba como se presenta en la Figura F.12.

Con la ayuda del compilador se determinó que el sujeto el valor de X lo puso en la variable *location_y* y el valor de Y lo puso en *location_x*. El medidor identifica este error y lo puso correctamente como mostramos en la Figura F.13

Como se observa en el ejemplo de la Figura F.13, el código viejo no es eliminado, sino es comentado. Aplicamos esta práctica en todas las intervenciones, para mantener el código fuente original y no eliminarlo. De esa manera conocemos cuales son los cambios implementados por el medidor.

```
1 private enum Direction { NORTH, EAST, SOUTH, WEST;
2 public static String toString(Direction dir){
3     String result = "";
4     switch (dir) {
5
6         case NORTH:
7             result ="NORTH";
8             break;
9         case EAST:
10            result = "EAST";
11            break;
12        case SOUTH:
13            result = "SOUTH";
14            break;
15        case WEST:
16            result = "WEST";
17            break;
18    }
19    return result;
20 };
```

Figura F.8: Formato incorrecto para la orientación del Rover, variable *dir*

```
1 private enum Direction { N, E, S, W;
2 public static String toString(Direction dir){
3     String result = "";
4     switch (dir) {
5         case N:
6             result ="N";
7             break;
8         case E:
9             result = "E";
10            break;
11        case S:
12            result = "S";
13            break;
14        case W:
15            result = "W";
16            break;
17    }
18    return result;
19 };
```

Figura F.9: Formato corregido para la orientación del Rover, variable *dir*

```

1 direction = 0; // 0=N, 1=S, 2=E, 3=W
2
3 if (command.size() > 0){
4     char buffer[20];
5     sprintf(buffer, "(%u,%u,%u)", pos_x, pos_y, direction);
6     return buffer;
7     //return direction as integer...
8 }
9 else
10 return "";

```

Figura F.10: Uso de formato de tipo entero para la orientación del Rover

```

switch (direction)
{
    case 0:
        direction_result = 'N';
        break;
    case 1:
        direction_result = 'S';
        break;
    case 2:
        direction_result = 'E';
        break;
    case 3:
        direction_result = 'W';
        break;
    default:
        direction_result = 'N';
}

if (command.size() > 0){
    char buffer[20];
    sprintf(buffer, "(%u,%u,%u)", pos_x, pos_y, direction);
    return buffer;
    //return direction as integer...
}
else
return "";

```

Figura F.11: Uso de formato de tipo caracter para la orientación del Rover


```
public String getRoverLocation() {  
    return String.format("(%d,%d,%s)", this.location_x ,  
        this.location_y , this.direction);  
}
```

Figura F.12: MR: Variables de salida X e Y intercambiadas

```
public String getRoverLocation() {  
    //return String.format("(%d,%d,%s)", this.location_x ,  
        //this.location_y , this.direction);  
    return String.format("(%d,%d,%s)", this.location_y ,  
        this.location_x , this.direction);  
}
```

Figura F.13: MR: Variables de salida X e Y corregidas

F.4. Ejemplos de implementación del Bonus en la tarea BSK

En el siguiente código fuente de la Figura F.14 del sujeto PTE06 correspondiente al experimento Playtech, presentamos los métodos *setBonus(int firstThrow, int secondThrow)* y *score()*, relativa a la primera alternativa de implementación válida, en donde se crea el frame bonus y se calcula el score total tomando en cuenta el frame bonus.

En la Figura F.15 presentamos la segunda alternativa de implementación válida, donde calcula el score del juego del bowling en base a los frames en donde se encuentra incluido el bonus como un frame. En la Figura F.15 presentamos el ejemplo del código del sujeto FSOU2804.

Ambas alternativas serían válidas para el manejo del bonus en la tarea BSK.

```
public void setBonus(int firstThrow , int secondThrow) {
    bonus = new Frame(firstThrow , secondThrow);
}

public int score(){
    int gameScore = 0;
    for(int i = 0; i < frames.size(); i++){
        Frame f = frames.get(i);
        int frameScore = f.score();
        if(f.isStrike()){
            if(f.isLastFrame()){
                if (bonus!=null)
                    frameScore+=bonus.score();
            }else{
                Frame nf = frames.get(i+1);
                frameScore += nf.score();
                if(nf.isStrike()){
                    if(nf.isLastFrame()){
                        if (bonus!=null)
                            frameScore+=bonus.getFirst();
                    }else{
                        frameScore+=frames.get(i+2).getFirst();
                    }
                }
            }
        }
        }else if(f.isSpare()){
            if(f.isLastFrame()){
                if (bonus != null)
                    frameScore+=bonus.getFirst();
            }else{
                Frame nf = frames.get(i+1);
                frameScore+= nf.getFirst();
            }
        }
        gameScore+=frameScore;
    }
    return gameScore;
}
```

Figura F.14: Implementación de Bonus en la tarea BSK

```
public void setBonus(int firstThrow , int secondThrow) {  
    //to be implemented  
}  
  
public int score(){  
    int result = 0;  
    for (Frame f : frames) {  
        if (!f.isBonus()) {  
            result += f.score();  
        }  
    }  
    return result;  
}
```

Figura F.15: Alternativa de implementación de Bonus en la tarea BSK

F.5. Ejemplos de control de entradas inválidas en las tareas

Los sujetos pueden implementar el control de entradas inválidas de varias formas según su criterio, a continuación se presentan ejemplos. En la Figura F.16 presentamos un ejemplo de control de entradas inválidas mediante alternativas válidas que analiza el medidor.

El código fuente ha sido desarrollado por el sujeto PTE06, tarea MR, experimento Playtech. El Rover inicia en la posición (5,2,E), línea 5. En la línea 9: hay una entrada inválida “B” (mayúscula) para lo cual tenemos la alternativa de salida válida (5,2,E). Es decir, no hay manejo de excepciones en el código fuente del sujeto para la entrada inválida, pero tenemos una alternativa válida (5,2,E) que daría por pasado el caso de prueba.

En el ejemplo de la Figura F.17, se hace el manejo de excepciones en la tarea BSK mediante la clase *Exception*. El sujeto PTE09 del experimento Playtech, usa la clase *Exception* para controlar las entradas inválidas. Sin embargo no pasa los casos de prueba respectivos. Como se observa en el siguiente código, el sujeto ha comentado el llamado a los métodos de validación (*validate*).

El medidor cambia el nombre de la clase que maneja excepciones, la reemplaza con *customException*, que es la clase apropiada para conectarla a los casos de prueba. El medidor elimina los comentarios de las líneas 2 y 3 para facilitar la prueba del control de excepciones. En la Figura F.18 presentamos el código fuente a continuación:

En otros casos, los sujetos suelen también controlar las entradas inválidas mediante mensajes a la pantalla. Cuando identifican una entrada inválida, en lugar de lanzar una Excepción, presentan un mensaje en pantalla. En esta situación el medidor también interpreta la intención del sujeto para dar por válido o no el caso de prueba respectivo.

```

@Test
public void US03_equivalencePartitioning15() {
try {
    MarsRover mr = new MarsRover(6,3,"");
    mr.goToStart(5, 2, "E");
    //assertEquals("(4,2,E)", mr.executeCommand("b"));
    //TODO Standard_ValidInputs: Comportamiento del sistema
    //ante entradas validas - Aceptando mayusculas y minusculas
    assertEquals("(5,2,E)", mr.executeCommand("B"));
    //TODO Alternating2_InvalidInput: Se usa para las entrada
    //INVALIDAS cuando el sujeto ha definido especificamente
    //que ante una entrada invalida el sistema no harj nada.
} catch (Exception e){
    fail("excepcion inesperada NO controlada por el sujeto");
}

    //TODO Alternating1_InvalidInput: Se usa para las entrada
    //INVALIDAS definidas y controladas por el sujeto
    //especificamente a traves de excepciones
    //mr.executeCommand("B");
/*} catch (RuntimeException e){
    fail("excepcion NO controlada por el sujeto");
} catch (Exception u){
    assertTrue("excepcion definida por el sujeto",true);
}*/
}
}

```

Figura F.16: Control de entradas invdas mediante alternativas vlidas

```

public Frame(int i, int j){// throws Exception {
    /*    validate(i);
    validate(j);*/
    this.first = i;
    this.second = j;
}

private void validate(int i) throws Exception {
    if ( i < 0 || i >10 )
        throw new Exception("Value "+i+" not allowed.");
}

```

Figura F.17: Control de entradas invdas mediante la clase *Exception*

```
public Frame(int i, int j) throws customException {  
    validate(i);  
    validate(j);  
    this.first = i;  
    this.second = j;  
}  
  
private void validate(int i) throws customException {  
    if ( i < 0 || i > 10 )  
        //throw new Exception("Value "+i+" not allowed.");  
        throw new customException();  
}
```

Figura F.18: Control de entradas inválidas mediante clase alternativa *customException*