Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

Tesis de Licenciatura

Análisis de la evolución del aprendizaje y uso de TDD

Autor
Gerardo Fuentes
gera.fuentes.exactas@gmail.com
LU 908/05

Director
Lic. Hernán Wilkinson
hernan.wilkinson@gmail.com

Buenos Aires, 2022

Análisis de la evolución del aprendizaje y uso de TDD

Actualmente si bien hay estudios sobre la aplicación de la técnica de *Test Driven Development* (de ahora en adelante TDD), ninguno se enfoca en la evolución de su aprendizaje. Debido a esto, no hay material empírico sobre el cual basarse para planificar estrategias de enseñanza de la técnica, o su evaluación.

En el presente trabajo se analizó la aplicación de TDD y la progresión de su aprendizaje en el contexto de la cursada de la materia *Ingeniería de Software I* de la Facultad de Ciencias Exactas y Naturales de la UBA.

Para este análisis se utilizó el modelo y la información que genera una herramienta de validación de uso de TDD. Como input de dicha herramienta se recolectaron datos de alumnos de dos cuatrimestres de la materia *Ingeniería de Software I*, puntualmente datos de ejercicios y parciales. Debido a algunas características de la información, debió realizarse una "curaduría" de datos previa a ser procesada por la herramienta de TDD.

Una vez que se obtuvo la información del uso de TDD, se calcularon diversos indicadores con el fin de estudiar la evolución del aprendizaje de la técnica, su aplicación y detectar algún patrón entre los distintos valores calculados. Luego, se compararon los resultados obtenidos entre ejercicios y parciales para posteriormente cotejarlos entre ambos cuatrimestres analizados.

Se realizó también una encuesta a los alumnos de los cuales se extrajeron datos con el fin de validar algunos indicadores obtenidos y obtener información de carácter más personal por parte de los estudiantes.

Se elaboraron tablas y gráficos con toda la información para su mejor comprensión junto con las conclusiones arribadas.

Se pudo observar que, si bien los alumnos aplicaron TDD en los ejercicios, no lograron hacerlo en todo su contexto, se mostrarán los motivos.

Palabras Clave: TDD, TDDGuru, CuisUniversity, Evolución de aprendizaje, Ingeniería de Software

Agradecimientos

A Dina y Adolfo, mis viejos que dieron todo para que nunca me falte nada y tenga las herramientas para que logre lo que me proponga y para que nunca deje de crecer y aprender. Gracias a ustedes llegué a donde estoy y voy a continuar logrando todo lo que me proponga.

A Sole, mi amor que me dió su valiosa compañía, tiempo y ayuda en este camino que recorrimos juntos.

A Hernan, por su compromiso, dedicación y entusiasmo con el que llevamos este trabajo adelante.

A mis familiares que siempre me preguntaban por cómo iba la facu y siempre me escucharon.

A todos en Codes, por acompañarme y brindarme siempre lo que necesité para avanzar con la facu.

A Rito, Kevin, Broly, Zamba y la gran manada. Mis queridos peques que siempre me acompañaron.

A mis compañeros, Christian LT ,Ale, Nacho, Cristian Ferro, Cato, Tincho, Martin y muchos más por compartir tardes de TP's y estudios.

Índice

Introducción

TDD es una técnica de desarrollo de software que surgió durante la década del 2000. Principalmente impulsada por Kent Beck en el contexto de la metodología *Extreme Programing* (XP)**[1][2][3]** y su trabajo en *C3* (Chrysler Comprehensive Compensation System). La técnica propone que el desarrollo de software se realice en iteraciones de tres pasos principales:

1. Escribir un test que falle, debe ser lo más simple posible.
2. Desarrollar lo mínimo necesario, sin abarcar condiciones no testeadas aún, para que el test pase.
3. *Refactorizar* el diseño de lo desarrollado hasta ese momento (Refactorizar significa modificar el diseño manteniendo el resultado de la ejecución).

Realizando ciclos de los pasos anteriores el software se construye de manera iterativa e incremental, dando lugar a las siguientes ventajas en el desarrollo del software:

- El código es inherentemente testeable, dado que este existe si hay un test que lo cubre.
- El código es minimal, dado que en cada ciclo se desarrolla lo mínimo indispensable para que el test pase.
- Hay feedback constante sobre la correctitud funcional.
- Si se aplican heurísticas de diseño y buenos refactorings, el diseño final será simple y minimalista.

Para lograr adquirir estas ventajas en el desarrollo de software, en la formación académica de los alumnos es importante la metodología de enseñanza de la técnica, poder medir cómo TDD fue aplicado y poder evaluar la evolución de su aprendizaje.

Previamente a la formalización de *Test Driven Development* **[4]** (*TDD*) como técnica de desarrollo de software se presentaron varias ideas que comparten el espíritu de la misma.

Alan Perlis **[5]** planteó que un sistema de software se puede diseñar mejor si las pruebas se entrelazan con el diseño del mismo, en lugar de utilizarlas al final de su diseño. Una simulación que cumple con los requerimientos del desarrollo contiene el control que organiza el diseño del sistema, es decir, guía el mismo. Repitiendo sucesivamente este proceso de pruebas y diseño entrelazados el modelo finalmente se convierte en el propio sistema de software. Las pruebas y el reemplazo de simulaciones con módulos que son más profundos y detallados extienden el modelo de simulación controlando el lugar y el orden en que se hacen estas acciones.

Richard W. Hamming **[6]** sugería no escribir una línea de código hasta que se decida cuál va a ser su test de aceptación, es decir, qué tests se van a aplicar al desarrollo para decidir qué se está ejecutando correctamente. Propone que si uno comienza donde todos los problemas ocurren, entonces, se va a comenzar a decidir escribir el programa de la manera más directa posible a la solución, tal que sea

fácilmente testeable. Como consecuencia se va a llegar mucho antes a cumplir el requerimiento buscado.

A lo planteado anteriormente hay excepciones (continua Hamming), por ejemplo, hay problemas sobre los que no se tiene toda la información. Como consecuencia, se comienza a programar tratando de averiguar de qué se trata el problema. Por ejemplo, supongamos un proyecto de IT para el departamento de defensa, puede ser construir un misil guiado. Si esto último es toda la especificación recibida, no se tiene idea de, por ejemplo, los rangos de vuelo, impacto, ni de muchas otras cosas. La idea de programar algo que haga algo no es una buena manera de encarar un problema de final abierto porque uno no sabe que está haciendo. Eso lleva a resultados muy malos.

A veces la programación es un medio para encontrar qué se está tratando de hacer, ahora, eso usualmente lleva a que una vez desarrollado el programa, uno más o menos entendió el problema, pero luego se debe reprogramar todo desde cero. Utilizar la programación para esto último no es lo más recomendable.

Se sugiere pensar antes de programar y particularmente definir cómo se va a decidir si el programa es correcto.

En el presente trabajo se buscará evidencia de la evolución de aprendizaje y uso de TDD, con el objetivo de tener información como referencia histórica que permita a los docentes reflexionar sobre la enseñanza de la misma y por lo tanto mejorarla.

A fin de obtener dicha evidencia se utilizará la herramienta *TDDGuru* **[7]**, desarrollada en *CuisUniversity* **[8]**, la cual permite validar qué pasos de la técnica se aplicaron correctamente y cuáles no. A partir de su modelo y la información que el mismo provee, se extrajeron datos para realizar diversos análisis y buscar conclusiones acerca de la evolución del aprendizaje de TDD.

Se recolectaron datos de dos cuatrimestres, el segundo del 2020 (2C 2020) y el primero del 2021 (1C 2021) de la materia *Ingeniería de Software 1* de la facultad de Ciencias Exactas y Naturales de la UBA, en concreto, datos de ejercicios realizados en grupo y del segundo parcial de la materia (individual). Debido a algunas características de dicha información, que se detallarán más adelante, debió realizarse una “curaduría” de datos previa a ser procesada por *TDDGuru.*

Cabe aclarar que en los cuatrimestres analizados en el presente trabajo la cursada fue realizada de manera remota debido a la pandemia del COVID-19 por lo que, tanto el dictado de las clases como el trabajo de los grupos fue atípico con respecto a otros años.

Una vez que se obtuvo la información de la aplicación de TDD, se desarrolló un proceso de exportación de la misma a fin de ser trabajada por herramientas más acordes a este tipo de tareas (como el componente *Pandas* de Python, hojas de cálculo, etc).

Con toda la información recopilada se realizaron distintos análisis que se presentarán en tablas y gráficos junto a las conclusiones obtenidas.

Además, se realizó una encuesta a los alumnos con el objetivo de validar algunos de los indicadores como también la experiencia del aprendizaje de TDD.

Dentro de los hallazgos del presente trabajo se pudo detectar que la aplicación de los alumnos de la técnica es mejorable en general, dado que de los datos de TDDGuru se encontró que el estado de no aplicación de TDD es el que más tiempo presentaba en los ejercicios. Una de las etapas en donde más se halló

dificultad para aplicar la técnica es en los ejercicios que son continuación de otros, donde los alumnos deben aplicar modificaciones a un modelo previamente desarrollado (por ellos mismos o tomado de la cátedra).

Se buscaron correlaciones entre los distintos datos obtenidos y se detectaron algunas relaciones a discutir más adelante, sin embargo, las mismas no se preservaron entre los cuatrimestres analizados.

Otra información obtenida de la encuesta realizada es que los alumnos consideran que no les costó entender la técnica pero sí aplicarla. Otro dato es que el paso que más difícil les parece aplicar es el de realizar la implementación más simple posible para que el test pase. Por otro lado, los alumnos mayormente utilizarían TDD en el ámbito laboral y académico, pero en este último hay cierto porcentaje que indicó rotundamente que no utilizaría la técnica.

Finalmente, se observaron en diversos indicadores mediciones variadas en cuanto al cumplimiento de TDD, lo que indica que en general los alumnos no son capaces de aplicar la técnica en cualquier contexto.

En el capítulo 1 se discutirá sobre otros trabajos relacionados con la enseñanza de TDD. En el capítulo 2 se detallarán todos los pasos y procedimientos de obtención de la información, como también el refinamiento de los datos disponibles. Luego, en el capítulo 3, se presentará el detalle de los indicadores calculados, es decir, cómo se obtuvieron, un gráfico para presentarlos y una apreciación del mismo. En el capítulo 4 se discutirá sobre un seguimiento a casos puntuales de alumnos con resultados buenos y mejorables en la materia y su relación con el aprendizaje de TDD, se elaboraron indicadores adicionales a los presentados. Luego en el capítulo 5 se presentará la encuesta realizada a los alumnos con las preguntas, las respuestas obtenidas y un análisis de las mismas. Finalmente en los capítulos 6, 7, 8 y 9, las conclusiones, el trabajo a futuro, los apéndices y las referencias respectivamente.

1. Estado del arte

Maria Siniaalto y Pekka Abrahamsson **[9]** recopilaron diversos trabajos en donde se enseñó la técnica de TDD y también se evaluó su aplicación. Tanto en el ámbito académico como también en la industria, dando resultados variados. Los resultados de los trabajos fueron, por un lado, los siguientes:

- La calidad del código producido en la utilización de la técnica es superior al que se produce sin la misma.
- La cobertura de los tests es superior con TDD.
- Bajó significativamente el porcentaje de los defectos.
- Los equipos que usaban TDD corregian errores más rápidamente, en comparación a los que no utilizaban la técnica.

Por otro lado, y en contraposición, el tiempo de desarrollo se incrementó entre un 15% y un 35% en los trabajos analizados.

Hakan Erdogmus **[10]** realizó el experimento de tomar mediciones del código de un desarrollador que comenzaba a aplicar TDD y, mediante la herramienta HackyStat **[11]** tomó datos sobre su codificación y tiempo en las distintas etapas. A través de los datos recolectados elaboró métricas que permitieron visualizar los momentos donde hubo mayor y menor aplicación de la técnica como también su calidad, demostrando así, la importancia de la medición en el proceso de aprendizaje de la misma.

Con el fin de medir el cumplimiento de TDD un trabajo propuso desarrollar la herramienta Butterfly**[12]** la cual detecta las acciones de los programadores (agregar test, editar test, agregar código, etc) y con las mismas mediante heurísticas determina si se está aplicando la técnica. La misma fue testeada con desarrolladores provenientes de la industria de software con un desarrollo puntual. Los datos extraídos de dicha herramienta fueron concordantes con las acciones de los desarrolladores y se espera que a futuro permita caracterizar el aporte de TDD al desarrollo de software.

Otro estudio realizado **[13]** planteó que a un grupo de alumnos se les proponga utilizar TDD en un ejercicio y, para su evaluación, se reemplaza la herramienta de calificación utilizada previamente por otra llamada *WEB-CAT* **[14]**. La misma pone énfasis en analizar los tests como nuevo enfoque del desarrollo, además de dar un feedback más enriquecido en cuanto a cómo se realizaron los tests. Los resultados obtenidos fueron los siguientes:

- Los alumnos mejoraron sus calificaciones con respecto a exámenes anteriores.
- La técnica fue aceptada ampliamente por los estudiantes
- Los estudiantes se sentían más seguros del código entregado, como también al realizar cambios al mismo.

- Los alumnos prefieren usar TDD aunque la consigna no lo pida. Lo que muestra la importancia del feedback sobre el desarrollo realizado.

La técnica de TDD tiene una amplia variedad de temas de investigación. Los trabajos nombrados en el presente capítulo tratan temas como su impacto en el desarrollo de software, herramientas para medir su aplicación y también evaluarlo en el contexto de un exámen. Los resultados de los trabajos refuerzan los beneficios de TDD y su aceptación en general pero, más allá de los trabajos mencionados, no se encontraron otros que traten sobre la evolución del aprendizaje de la técnica.

Como se mencionó anteriormente, poder analizar cómo los alumnos aprenden la técnica orientará a los docentes a planificar estrategias de enseñanza y evaluación.

2. Detalle del proceso realizado

En este capítulo se detallarán los pasos realizados para obtener los datos para calcular los indicadores de la aplicación de TDD, como también, se detallarán los componentes de software utilizados para tal fin.

2.1 CuisUniversity

CuisUniversity es un ambiente de desarrollo de Smalltalk, creado especialmente para la enseñanza de la programación orientada a objetos. Se basa en Cuis, la implementación Argentina de Smalltalk.

2.2 TDDGuru

TDDGuru valida y reporta los pasos de TDD que se realizaron correctamente y marca en donde el desarrollador no siguió bien la técnica. La herramienta corre bajo CuisUniversity y realiza la validación a partir del log de cambios. Adicionalmente muestra, el tiempo en cada etapa , el detalle de qué no se aplicó correctamente en la técnica, entre otros datos como se muestra en la *Figura 1*.

La validación se realiza *postmortem,* es decir, una vez que se finaliza la codificación que se desea analizar, el usuario debe ejecutar el análisis del mismo.

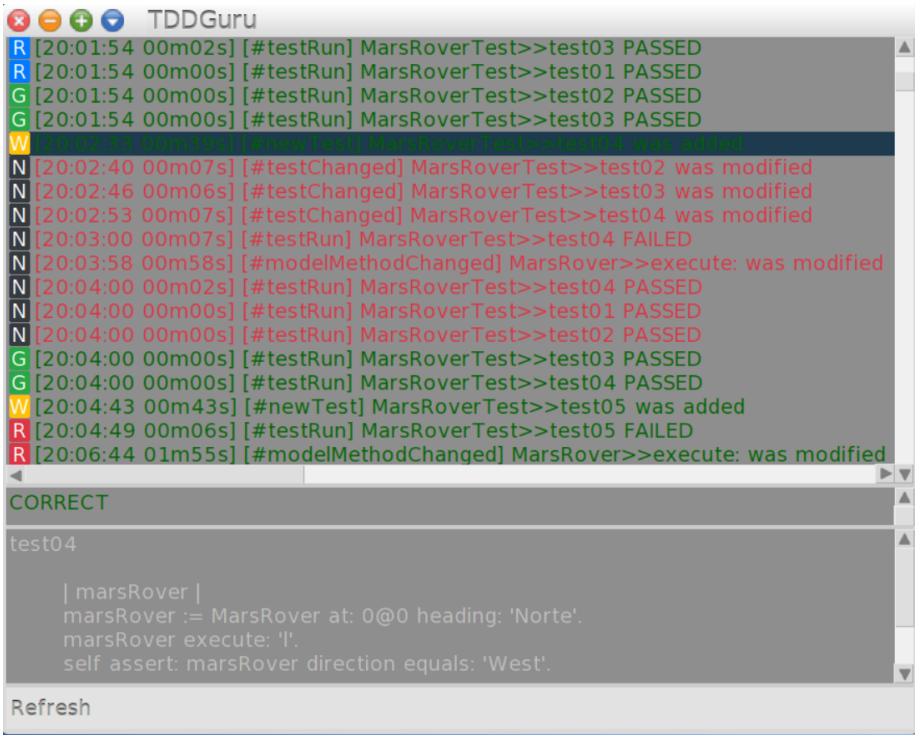


Figura 1. Interfaz de TDDGuru. Se cargó un archivo .changes

El log de cambios de CuisUniversity se almacena en archivos *.changes*, por lo que, obteniendo este archivo del ambiente de trabajo del desarrollador es posible importarlo en TDDGuru y obtener la información de aplicación de TDD.

2.3 Datos recolectados

2.3.1 Ejercicios

El contexto del estudio de TDD del presente trabajo es el de los ejercicios y parciales de la materia *Ingeniería del Software I* de la facultad de Ciencias Exactas y Naturales de la UBA durante el segundo cuatrimestre del 2020 y el primero del 2021.

Para el 2C del 2020, los ejercicios fueron:

- *Mars Rover (MR):* Este ejercicio tiene por objetivo reemplazar if por polimorfismo.
- *Terni Lapilli (TEL):* Este ejercicio tiene por objetivo *a*plicar el patrón de diseño *State*.
- *Portfolio 1 (PO1):* Este ejercicio tiene por objetivo *d*escubrir el patrón de diseño *Composite*.
- *Portfolio 2 (PO2):* Este ejercicio tiene por objetivo descubrir el patrón de diseño *Visitor* y aplicarlo sobre *Composite*.
- *Tus Libros 1, Tus Libros 2 y Tus Libros 3 (TL1, TL2 y TL3 respectivamente)*: El objetivo de estos ejercicios es desarrollar una aplicación compleja por medio de TDD.

Los enunciados se pueden consultar en el apéndice **1**. Para los ejercicios MR y TEL las entregas son autocontenidas y, para Portfolio 2 se puede, o bien tomar como base lo desarrollado en Portfolio 1 ó utilizar una solución que provee la cátedra. Lo mismo ocurre con Tus Libros 2 y 3 con respecto a Tus Libros 1. Para estos últimos casos la continuación implica más complejidad ya que se debe extender/modificar el modelo previamente desarrollado (o tomado de la cátedra).

Para el 1C 2021, los ejercicios son similares, con la salvedad de que se reemplaza Terni Lapilli por Mars Rover El Regreso (MRR) cuyo objetivo es aplicar el patrón de diseño O*bserver*. Estos ejercicios se realizan en grupos de dos personas.

Algunas observaciones en la resolución de los ejercicios a tener en cuenta para el presente análisis:

- En MR, PO1 y TL1 se ayuda mucho en clase.
- Para PO2 no se ayuda tanto como en los casos anteriores.
- En TEL los alumnos no llegaron a desarrollar todo lo solicitado.
- En TL2 se puede utilizar la solución de la cátedra al momento de iniciar el desarrollo.
- TL3 es muy complejo.

2.3.2 Parciales

En el caso de los parciales, se realizan de manera individual. Se recolectó la información del segundo parcial de la materia, dado que es donde se aplica TDD. En el caso del 2C 2020 el objetivo del parcial es mixto/balanceado tanto en diseño como en testing y en el 1C 2021 es aplicar el patrón de diseño *State*.

Para este contexto se utilizaron los archivos .changes de los repositorios de los alumnos y los .changes entregados de los parciales. En la *Figura 2* se muestran los totales de archivos utilizados.

Cuatrimestre	.changes Ejercicios	.changes Parciales	Grupos	Alumnos
2C 2020	170	35	25	47
1C 2021	147	38	25	50

Figura 2. Cantidades de archivos .changes, grupos y alumnos por cuatrimestre

2.4 Consideraciones sobre los datos

- Existe la posibilidad de que los alumnos se hayan copiado de soluciones de ejercicios de otros cuatrimestres (donde se dió la misma consigna) o incluso de otros grupos en las presentes cursadas. Como se verá más adelante en la encuesta realizada, los alumnos reconocieron haber recibido o dado ayuda externa, sin embargo esto se da en un porcentaje relativamente bajo con lo que se asume que no afecta la validez del análisis realizado . Para el caso del parcial se tiene más confianza de que no hubo copia alguna.
- La dinámica de cada cursada es distinta, quizás haya más casos de colaboraciones entre grupos en unas que en otras.
- Dado que Ingeniería del Software I es una materia avanzada en la carrera, puede haber alumnos con más experiencia en programación que otros (debido a experiencias laborales) o incluso que conozcan TDD de antemano, por lo que, no se puede asumir que todos los alumnos comienzan la materia con el mismo nivel.
- El segundo parcial del 2C 2020 fue sobre un tema parecido al primero, sin embargo el segundo parcial del 1C 2021 fue más difícil en comparación con el cuatrimestre anterior.
- Para muchos alumnos es el primer contacto con el lenguaje de programación Smalltalk, que es el que se utiliza en los ejercicios. Es lógico presuponer que los alumnos se enfocarán en aprender la dinámica, sintaxis y forma de encarar los desarrollos en el lenguaje que están aprendiendo, además de las consignas en sí a resolver y los contenidos de la materia por sobre la aplicación de TDD.
- Si bien hay definido un alcance de lo que debe cubrir la solución de los ejercicios y los parciales, el criterio de corrección es subjetivo al corrector. Con lo cual dos soluciones similares corregidas por dos docentes distintos pueden resultar en puntajes diferentes.

2.5 Organización de los archivos .changes

Dado que los alumnos organizan los archivos de su repositorio a su criterio, fué necesario establecer una nomenclatura para los mismos a fin de automatizar su lectura. Para esto, a cada alumno y grupo se los numeró de forma arbitraria y se renombraron, de forma manual, los archivos de los ejercicios con la siguiente estructura:

<R>_<A>_<N>_<E>_<S>.changes donde:

R: Identificador del repositorio
A: Identificador del alumno
N: Número de archivo (en caso de que haya más de uno por alumno y ejercicio)
E: Identificador del ejercicio
S: Estado del archivo (OK o ERROR)

Los identificadores de ejercicio posibles son:
MR: Mars Rover
TEL: Terni Lapilli (*Solo en 2C 2020*)
MRR: Mars Rover Regreso (*Solo en 1C 2021*)
PO1: Portfolio 1
PO2: Portfolio 2
TL1: Tus Libros 1
TL2: Tus Libros 2
TL3: Tus Libros 3

El estado del archivo representa si se puede utilizar (OK) o tiene algún inconveniente que lo deja fuera de consideración para los datos (ERROR).

En el caso de los parciales el renombre de los archivos fue el siguiente:

2P_A_S.changes donde:
A: Identificador del alumno
S: Estado del archivo (OK o ERROR)

Siguiendo el mismo criterio que el de los ejercicios para el identificador del alumno y el estado del archivo. En los datos de la *Figura 2* se tienen en cuenta solo los archivos en estado OK tanto para .changes de ejercicios como para parciales.

2.6 Validación de los archivos

Algunos archivos .changes cargados con TDDGuru arrojaron errores. Por este motivo se marcaron con el estado ERROR y no se tuvieron en cuenta para el análisis. Los mismos no fueron una cantidad significativa por lo que se asume que no afecta la validez del trabajo.

Si bien *TDDGuru* permite seleccionar a partir de qué cambio realizar el análisis o hacerlo desde el archivo completo, se eliminaron manualmente los cambios en cada archivo que no tuvieran relación con el ejercicio o el parcial en cuestión. Se

tomó esta decisión ya que más adelante se desarrolló un proceso que lee todos los archivos .changes en un directorio para extraer la información de TDDGuru y se necesitó que los mismos estén en condiciones de poder leerse de forma completa.

2.7 Información en TDDGuru a extraer

A partir del modelo de TDDGuru los datos extraídos son los de la aplicación de TDD en cada modificación del modelo de la solución que el alumno va construyendo. Por cada paso realizado los datos extraídos son:

- Ejercicio Realizado
- Identificador de Repositorio
- Identificador de Estudiante
- Paso de TDD
- Si el paso fue correcto o no
- Tiempo en el paso de TDD
- Cantidad de Tests realizados.

2.8 Proceso de exportación

Con el objetivo de procesar los datos para ser analizados, se desarrolló la exportación de los mismos hacia un archivo en formato CSV. Esta decisión fue tomada para utilizar la librería *Pandas* **[15]** de Python que permite manipular fácilmente este tipo de archivos y también, para tener unificados los datos de todos los alumnos en cada cuatrimestre. Tanto para los ejercicios como para los parciales.

El proceso fue desarrollado como extensión a los packages de TDDGuru. En el *Algoritmo 1* se describe el procedimiento que se ejecuta en dicho proceso para el caso de los ejercicios.

La simulación de carga del archivo completo (paso II) se desarrolló como extensión de TDDGuru, y envía los mensajes utilizados al cargar el archivo entero en la interfaz de usuario.

El modelo de TDDGuru plantea el concepto de *evento,* el cual representa un cambio de estado de TDD o una transición entre estados de la técnica. A partir de estos (y el nombre del archivo) es de donde se extrae la información para ser exportada. En el caso de que el estado de TDD sea incorrecto, TDDGuru indica cual es el error que se está cometiendo. Para el caso en que se esté refactorizando y se agreguen métodos que no se utilizan, TDDGuru concatena el texto *Probably not doing TDD* en el mensaje de error. En estos casos se tomó la decisión de no contemplar estos eventos dado que, no se tiene certeza de que se esté aplicando o no TDD. De todos modos, dichos casos se daban en una medida muy baja.

Para cada ejercicio se tomó en cuenta su fecha de inicio, a fin de descartar eventos que no tengan relación con el ejercicio en cuestión. Se sabe qué ejercicio se está analizando por la nomenclatura de cada archivo. En cuanto a las fechas, estas fueron provistas por el cronograma de la materia.

```
Input: Directorio conteniendo los archivos .changes a procesar
Output: Archivo .csv con la información extraída de TDDGuru
Procedimiento:
(I) Para cada archivo a en el directorio de input:
(II)     Cargar a en el modelo de TDDGuru, simulando una carga
         de archivo completo
(III)       Se busca el primer evento e del modelo de TDDGuru,
            tal que:
            - No tenga la marca Probably not doing TDD
            - Su fecha sea posterior a la del inicio del
              ejercicio
            - No sea ejecución de test
(IV)             Extraer del evento e los datos de: Cantidad
                 de tests, Estado Correcto o incorrecto,
                 tiempo en el estado
(V)              Extraer del nombre del archivo a los
                 datos de: Ejercicio, Repositorio y
                 Estudiante
(VI)            Grabar los datos extraídos en una nueva línea
                del archivo .csv
```

Algoritmo 1. Proceso de exportación de datos de TDDGuru a un archivo .csv

Otra condición que se verifica es que el evento desde donde se inicia no sea una ejecución de test, ya que, si lo primero que hace un alumno en el ejercicio es ejecutar los tests provistos por la cátedra, esta acción no aporta contenido al análisis de cómo se aplica TDD.

El algoritmo 1 se aplica también para el análisis de los ejercicios del 2C 2020 y del 1C 2021, cambiando las fechas de inicio de los ejercicios. Para el caso de los parciales las fechas antes mencionadas no aplican y los datos que se extraen son:

- Estudiante
- Estado de TDD
- Es correcto o no
- Tiempo en el estado
- Cantidad de tests.

2.9 Procesamiento de datos exportados

El proceso anteriormente mencionado genera un archivo .csv para los ejercicios y otro para los parciales para ambos cuatrimestres analizados.

Estudiando el tiempo en los estados se detectaron valores que, intuitivamente, eran muy superiores al que puede pasar un desarrollador trabajando en el código en cuestión. Por lo que se decidió hacer un análisis del tiempo por estado para cada ejercicio utilizando el método de *boxplot* **[16]** para visualizar los

mismos. Dicho método permite visibilizar dónde se encuentran la mediana, los outliers (valores alejados de la media), la concentración principal de los datos, entre otra información.

En las *Figuras 3* y *4* se presentan los boxplots para los cuatrimestres 2C 2020 y 1C 2021 respectivamente.

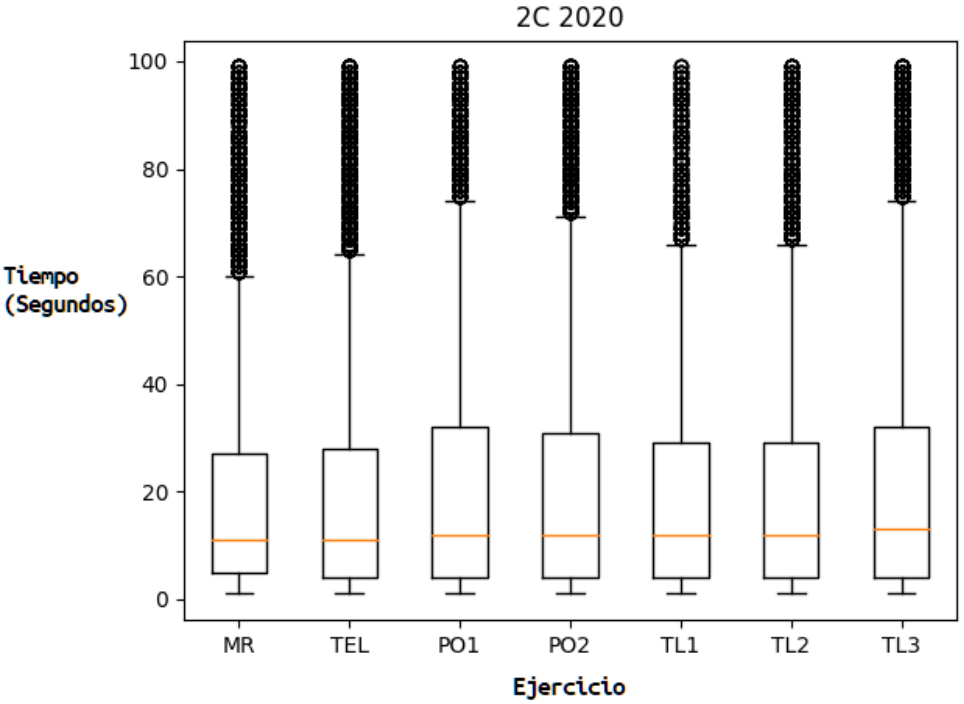


Figura 3. Boxplots de tiempo por ejercicio para el 2C 2020

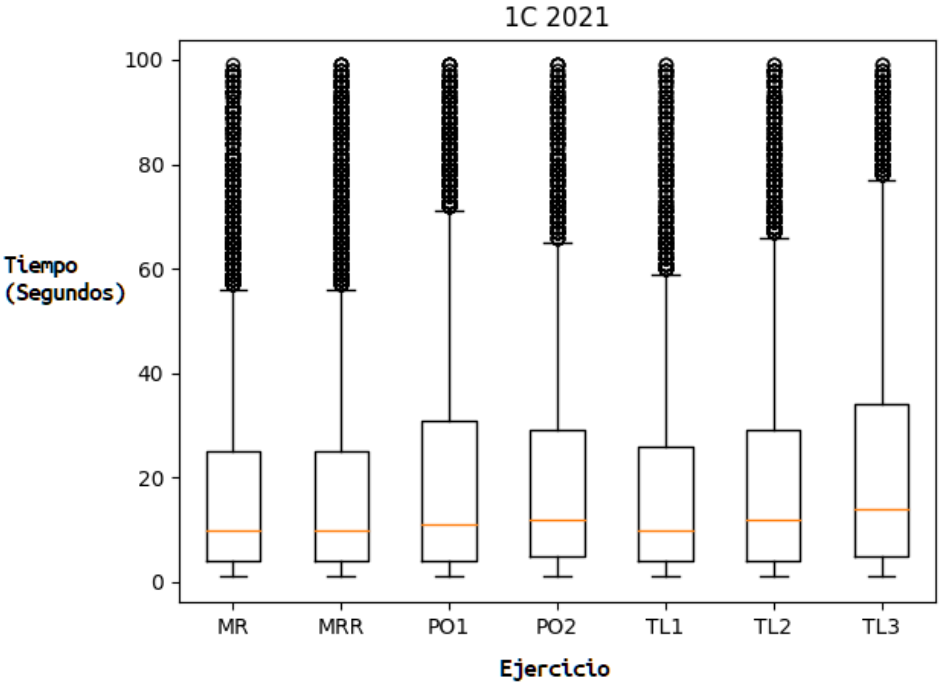


Figura 4. Boxplots de tiempo por ejercicio para el 1C 2021

Como se puede ver en ambas figuras, los outliers comienzan a aparecer alrededor de los 60 segundos. El objetivo de este análisis es obtener un umbral de corte para los tiempos en los eventos. Si el tiempo es muy elevado, puede no ser un evento válido de desarrollo. Por este motivo se decidió filtrar los datos que excedan los 60 segundos.

Gracias a la herramienta TDDGurú fué posible extraer la información de la aplicación de TDD. Hubo bastante trabajo manual para ordenar los archivos a extraer información, lo que puede ser mejorable a futuro mediante alguna lectura automática, pero con el cuidado de no “invadir” de ninguna forma el código ni la forma de trabajar de los alumnos para este fin.

Se presentaron varias consideraciones a tener en cuenta acerca de los datos y los alumnos, sin embargo, se asume que las mismas no afectan la validez del presente trabajo.

Por una cuestión de experiencia en tecnologías, se optó por extraer los datos de CuisUniversity y trabajarlos en herramientas de Python. Sería perfectamente viable encarar todo el trabajo sin recurrir a herramientas externas al ambiente de desarrollo de TDDGurú.

Gracias al proceso descrito en el presente capítulo se obtuvo la información para comenzar a analizarla. A continuación se presentan los indicadores calculados.

3. Indicadores calculados

En este capítulo se presentarán los indicadores calculados con la información ya procesada descrita anteriormente, como también, otros indicadores obtenidos de los anteriores u otras fuentes.

3.1 Indicadores calculados a partir de los datos de TDDGuru

Los siguientes indicadores fueron obtenidos netamente de los datos de TDDGuru.

3.1.1 Ejercicios

Los indicadores que se presentarán a continuación aplican a los ejercicios resueltos en grupo de cada cuatrimestre, los objetivos son buscar las dificultades de resolución de los ejercicios, como así también, las dificultades de aplicación de estados de TDD también a partir del tiempo.

-Tiempo por ejercicio

Tiempo de todos los grupos por ejercicio. Se visualizan en la *Figura 5* para ambos cuatrimestres.

Se puede observar que para los ejercicios que son una continuación de otros (PO1 y PO2 por un lado y TL1, TL2 y TL3 por otro) el tiempo en las partes siguientes se va incrementando. El resultado es lógico ya que se agrega complejidad en el modelo en las siguientes partes en que se trabaja, debiendo ajustar el mismo a la consigna agregada.

Se vé también que para el caso del 2C 2020 hay diferencias de tiempo considerables entre ejercicios del tipo anteriormente planteados con respecto a los del 1C 2021. En este último cuatrimestre, el tema de TDD fue enseñado antes que en el 2C 2020, lo que pudo haber ayudado a reducir los tiempos globales.

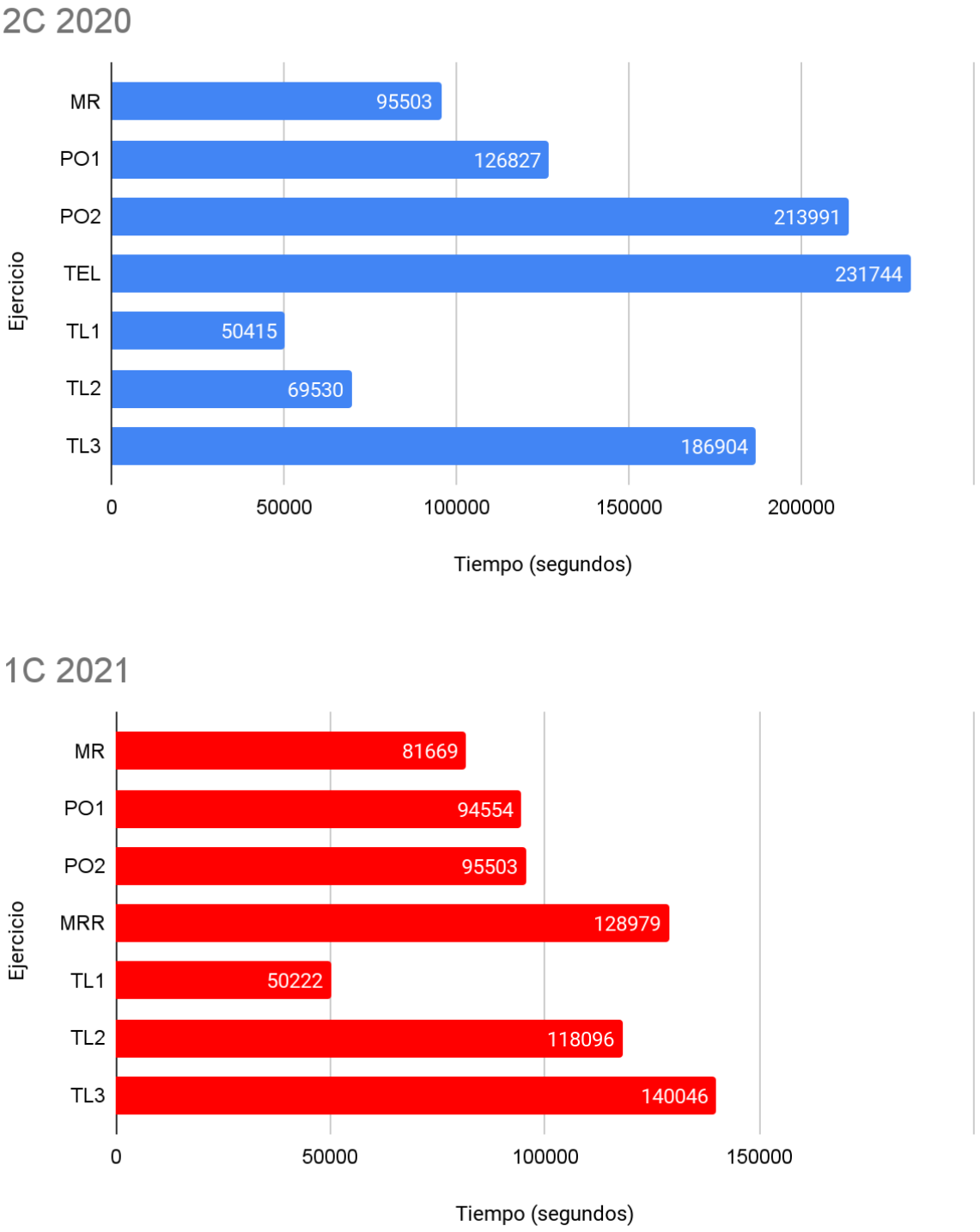


Figura 5. Tiempo por ejercicio para ambos cuatrimestres.

-Tiempo en los estados de TDD

El modelo de TDDGuru del que se extrajeron los datos para analizar maneja los siguientes estados:

- *Not Doing TDD*: No se está aplicando TDD.
- *Refactor*: Se está aplicando refactoring (recordando, modificar el modelo preservando su funcionalidad).
- *Red*: Se ejecutaron test y fallaron.
- *Writing a failing test*: Se está escribiendo un test con el fin de que falle.
- *Green*: Se ejecutaron tests y pasaron correctamente.
- *More than one test written*: Se escribió un test y antes de ejecutarlo se comenzó a escribir otro.

Se realizó la agrupación por estado de TDD de la suma de todos los tiempos de los grupos. A continuación en la *Figura 6*.

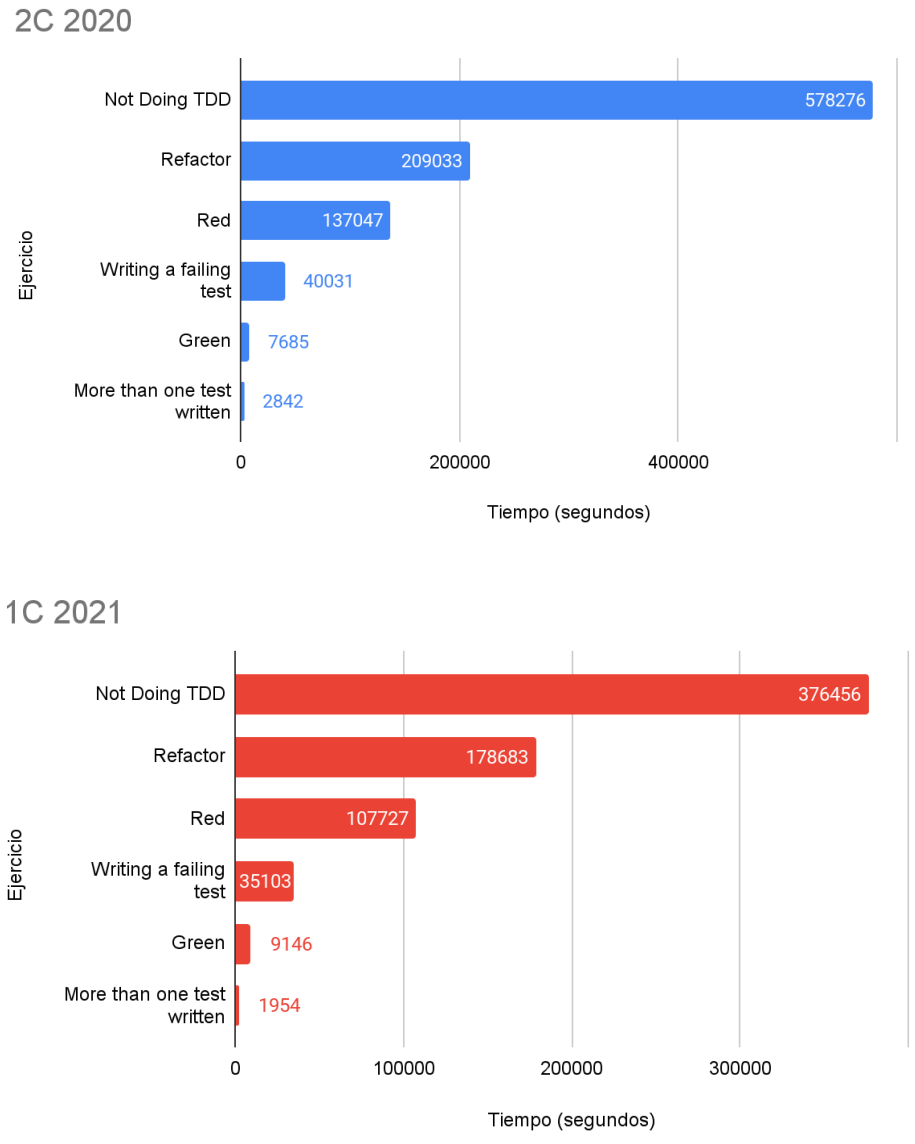


Figura 6. Tiempo en los estados de TDD para ambos cuatrimestres.

Lo primero a observar es que el orden de los estados por el tiempo es el mismo en ambos cuatrimestres, por lo que hay un patrón que se repite.

Se puede apreciar también que el estado de TDD con la mayor cantidad de tiempo en ambos cuatrimestres es *Not Doing TDD* con una diferencia considerable con respecto al tiempo en los demás estados de TDD. Esto último denota que en los cuatrimestres analizados la aplicación de TDD es mejorable.

Dentro de los estados con aplicación de la técnica el que presenta mayor tiempo es *Refactor,* lo que muestra que hay un interés en los alumnos en mejorar el modelo que se construye.

El estado que sigue en tiempo al anterior nombrado es *Red,* lo que denota que, se presenta una mayor dificultad al momento de refactorizar el código que al momento de hacer pasar los tests que se proponen.

Luego sigue el estado *Writing a Failing Test*, el cual tiene una diferencia amplia con respecto al siguiente. Lo que es coherente ya que en este estado se deben plantear los tests a ejecutar y requiere de creatividad a diferencia del estado siguiente en tiempo (*Green*) que es un resultado de la verificación de los tests.

El último estado en tiempo es *More Than One Test Written.* Que indica que dentro de todo, es un error poco frecuente escribir más de un test al aplicar la técnica.

3.1.2 Parciales

A continuación se presentan los indicadores que aplican a los parciales. Los objetivos son ver globalmente la aplicación de TDD por cuatrimestre, verificar la cantidad de tests realizados y detectar los estados de la técnica que más se dificulta aplicar a partir del tiempo en el contexto del parcial.

-Estados de TDD correctos por estudiante

Se elaboró un *boxplot* para cada cuatrimestre analizado para mostrar los porcentajes de estados correctos por estudiante. A continuación en la *Figura 7*.

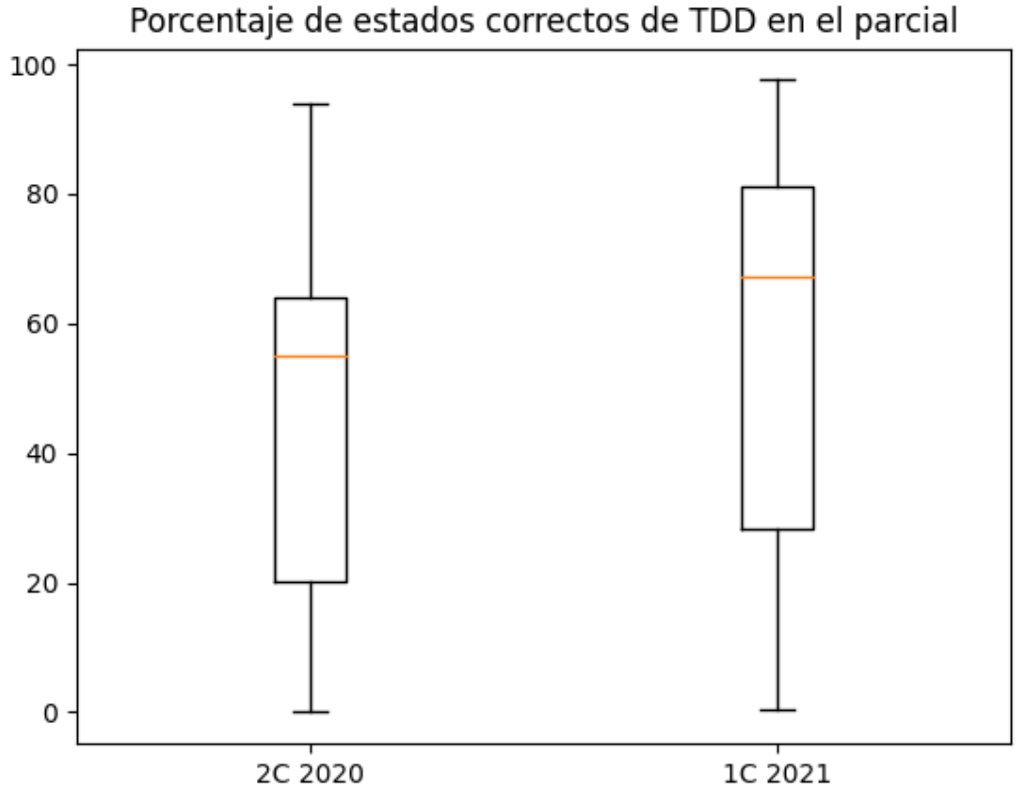


Figura 7. Boxplots para los porcentajes de aplicación de TDD en el parcial para ambos cuatrimestres.

En el 1C 2021 se comenzó antes con TDD que en el 2C 2020 y, como se puede apreciar, la mediana del porcentaje de estados correctos en el parcial aumentó con respecto al 2C 2020, se presume que se debe a que los alumnos tuvieron más tiempo practicando la técnica. Además, la concentración de datos aumentó de 60% de estados correctos a aproximadamente el 70% y 80% en el 1C 2021. Se evidencia una mejora general en la aplicación de TDD.

-Cantidad de tests realizados

Se elaboraron boxplots para cada cuatrimestre analizado. Se presenta la cantidad de tests realizados por estudiante. A continuación en la *Figura 8*.

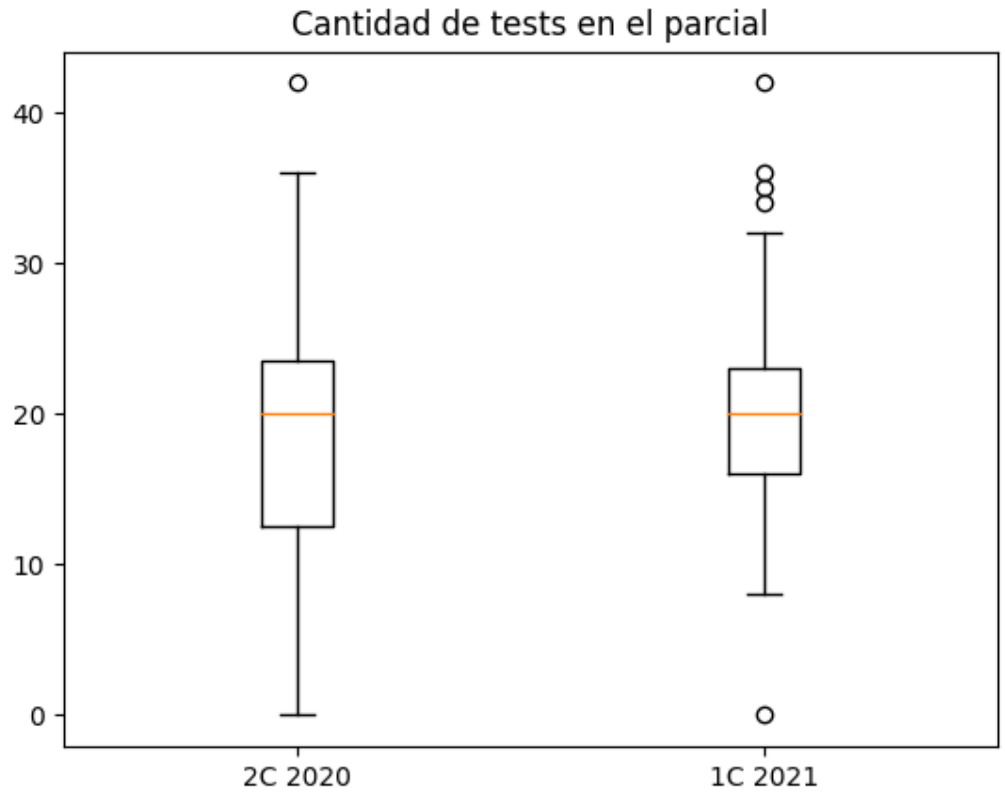


Figura 8. Cantidad de tests realizados por alumno para ambos cuatrimestres

Se puede ver que la mediana es similar en ambos cuatrimestres analizados, con lo que la cantidad de tests no sufrió un cambio importante. Pero, hay menor dispersión en los valores en el 1C 2021. De analizar más cuatrimestres, un punto a revisar sería si esta dispersión aumenta (y modifica la mediana) o disminuye.

-Tiempo en los estados de TDD

Suma de todos los tiempos de los alumnos para cada estado de TDD. Se presenta un gráfico por cada cuatrimestre analizado. En cada gráfico se agregan los datos de los ejercicios por grupo, para compararlos. A continuación en las *Figuras 9* y *10*.

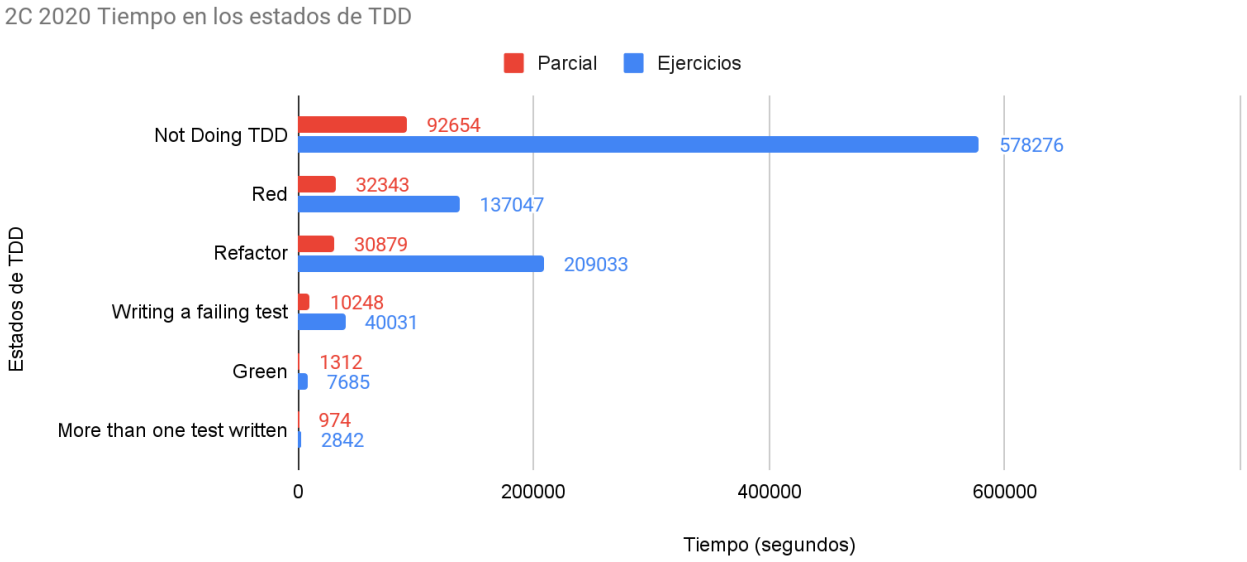


Figura 9. Tiempo en los estados de TDD para todos los alumnos/grupos para el 2C 2020

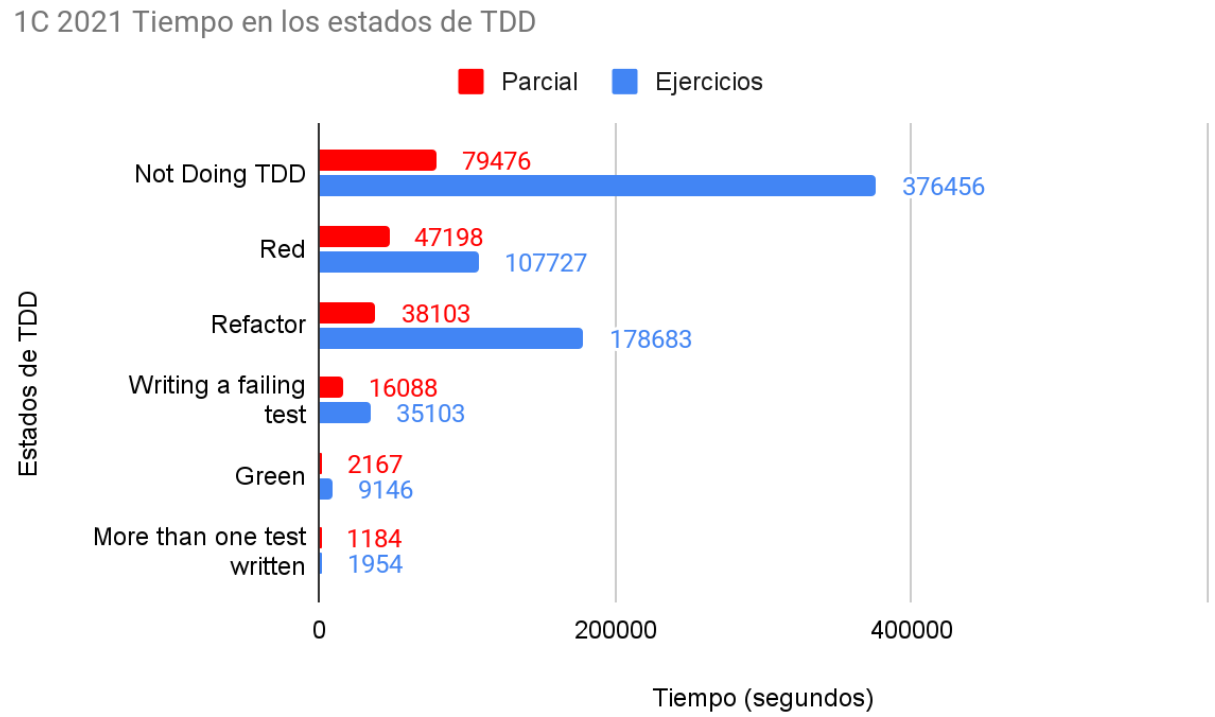


Figura 10. Tiempo en los estados de TDD para todos los alumnos para el 1C 2021

Como se puede verificar el estado que más tiempo tiene es *Not Doing TDD* esto es consistente con la consideración sobre los datos anteriormente nombrados acerca de que los alumnos ponen su enfoque en aprender Smalltalk y cumplir con las consignas de los ejercicios por sobre el aprendizaje de TDD.

Algo destacable es que el orden por tiempo de los estados casi se preserva con respecto al de los ejercicios. La única salvedad son los estados Red y Refactor que se encuentran en orden invertido. Esto denota que en el contexto del parcial los

alumnos se preocupan más por hacer que los tests pasen (estado *Red*) que por mejorar el modelo (*Refactor*) dado que el tiempo en el parcial es más acotado comparado con la resolución de los ejercicios.

Por lo demás el orden por tiempo se mantiene con respecto a los ejercicios por lo que, fuera de lo mencionado anteriormente, la forma de trabajar es similar a la de los ejercicios.

3.2 Indicadores obtenidos a partir de los anteriores u otras fuentes

- Variación de aplicación de TDD por ejercicio

Con el objetivo de analizar la mejora entre cada ejercicio de los grupos se calcularon los porcentajes de aplicación de TDD por cada grupo para cada ejercicio.

Puede ocurrir que no haya datos para algún caso. Esto puede deberse a que no se encontró información en los archivos .changes o este archivo directamente no existía para ese grupo.

Analizando los archivos .changes se detectó que los que corresponden a valores cercanos a 0 en el porcentaje de estados correctos no aplicaban la técnica o comenzaban aplicándola pero, a los pocos estados dejaban de hacerlo. Para los valores cercanos a 18% se notó en los archivos .changes que se aplicaba la técnica (con dificultad). Por este motivo se consideró el valor *18* como umbral para la aplicación o no de la técnica en el análisis. Esta validación se realizó mediante TDDGuru. Las tablas para ambos cuatrimestres se pueden consultar en el *Apéndice 4*.

Con las tablas anteriormente mencionadas se analizaron las diferencias de porcentaje de aplicación de TDD entre un ejercicio y el siguiente. Con estas diferencias se planteó la siguiente escala:

Subió mucho	diferencia >15
Subió poco	diferencia en (0,15]
Bajó poco	diferencia en [-15,0)
Bajó mucho	diferencia < -15
Sin datos	-1

En el caso de que en el ejercicio anterior del grupo no haya habido datos, se decidió no contabilizar la diferencia. No se considera la diferencia igual a 0 ya que indicaría que no hubo cambios y esta situación no ocurrió. De esta forma se cuantificó la cantidad de grupos que subieron o bajaron su aplicación de TDD.

Con esta información se elaboraron los siguientes gráficos para cada cuatrimestre en la *Figura 11*. Se comienza presentando la información a partir del segundo ejercicio de cada cuatrimestre (TEL para el 2C 2020 y PO1 para el 1C 2021) dado que las diferencias se obtienen con respecto al primer ejercicio realizado (MR en ambos cuatrimestres)

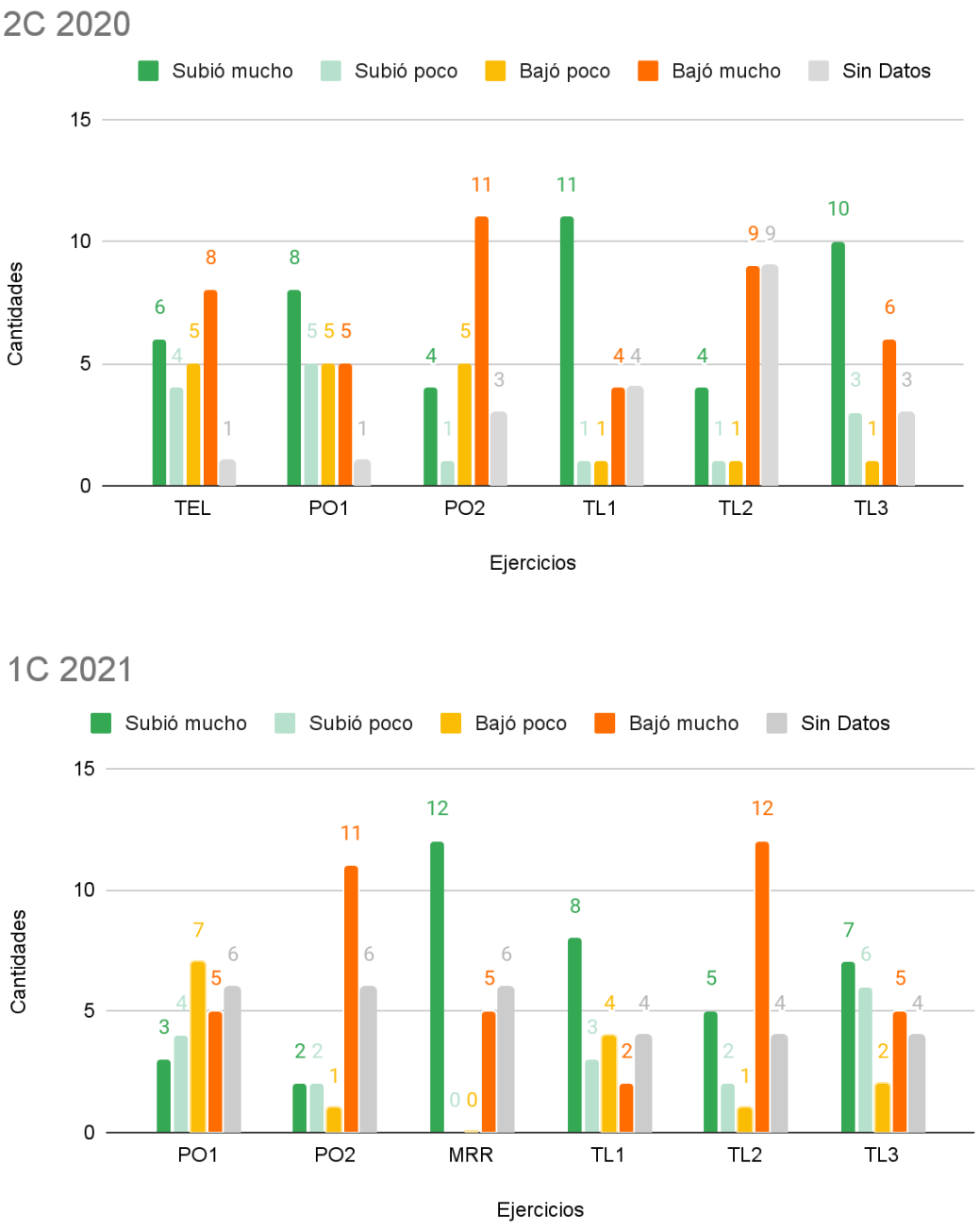


Figura 11.Variación de aplicación de TDD por ejercicio para ambos cuatrimestres.

Se puede observar que en PO2 y TL2 en ambos cuatrimestres el resultado general es que se tiende a bajar la aplicación de TDD. Lo que es consistente con el resultado del indicador de tiempos por ejercicio (Ver *Figura 5*) donde en las segundas partes de los ejercicios se incrementa el tiempo, lo que se presume se debe, a que se agrega complejidad en el modelo, por lo que, los alumnos se concentran en resolver las consignas más que en la aplicación de la técnica. Por otro lado, hay 12 grupos que mejoraron mucho en el ejercicio MRR en comparación con los que bajaron su cumplimiento de TDD, lo que puede indicar que este ejercicio presenta alguna facilidad para aplicar la técnica. En ambos cuatrimestres se ve una

mejora en TL1, en principio se podría pensar que los alumnos ganaron experiencia con los ejercicios anteriores, pero luego en TL2 ocurre la baja de desempeño nombrada previamente. Aún así en TL3 a pesar de ser un ejercicio complejo se vé una relativa mejora en la aplicación de TDD. Lo que muestra que los alumnos en general no pueden aplicar TDD en todo contexto.

- Promedio general de estados de TDD por ejercicio

A fin de medir por ejercicio la cantidad de estados de TDD correctos se realizó un promedio de los mismos para todos los grupos. Por cada grupo se calculó el porcentaje de estados correctos sobre el total de estados por ejercicio y luego se calculó el promedio por cada ejercicio para todos los grupos. En las *Figuras 12* y *13* se visualiza para cada cuatrimestre dicho promedio. Adicionalmente, para los porcentajes cercanos a 0 se detectó que los alumnos ni siquiera habían intentado aplicar la técnica. Esta validación se hizo mediante TDDGurú y por apreciación visual en cuanto a cómo se aplicaban los pasos. Se determinó que si el porcentaje supera el 18% los alumnos intentaban aplicar la técnica. Por lo que, la referencia *Con Aplicación de TDD* indica que se excluyen porcentajes inferiores a 18% y la referencia *Todos* indica que se considera la totalidad de los datos..

Se puede observar que se mantiene la tendencia de “empeorar” al momento de pasar a una segunda parte de un ejercicio para el caso de PO1 a PO2 para ambos cuatrimestres y de TL1 a TL2 y luego TL3 para el 1C 2021. En el caso de TL2 a TL3 en el 2C 2020 se puede ver una mejora. Una posible explicación es la ya mencionada variabilidad de dinámica de los alumnos en cada cuatrimestre, donde pudo haber alguna colaboración entre los grupos.

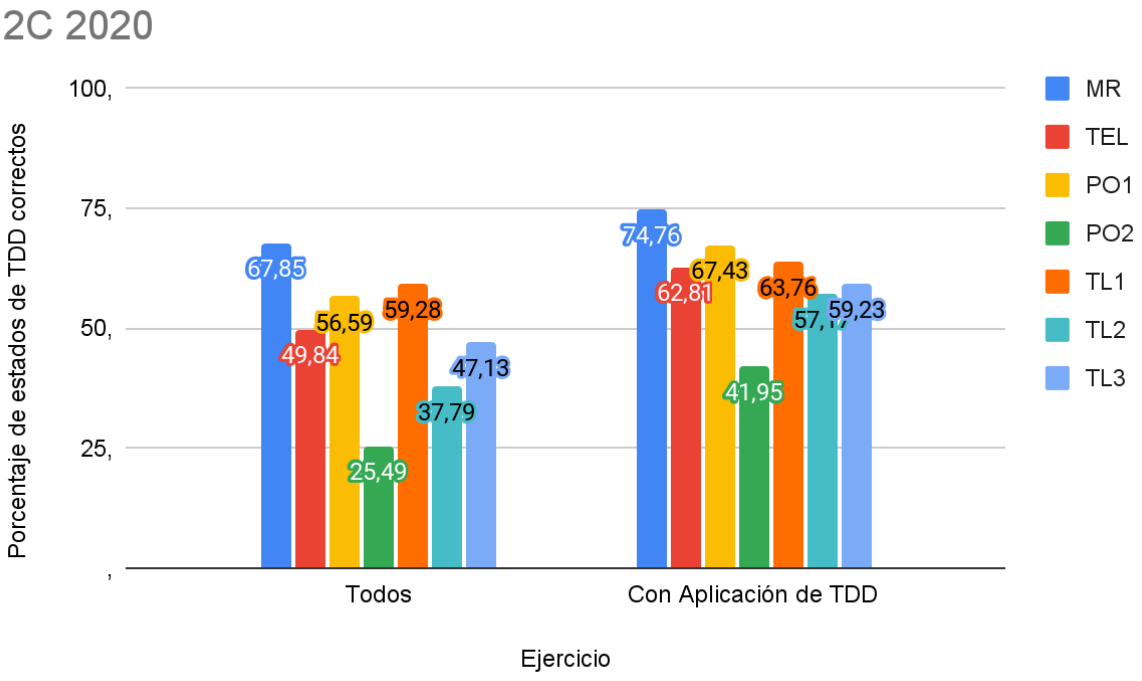


Figura 12. Aplicación de TDD por ejercicio para el 2C 2020

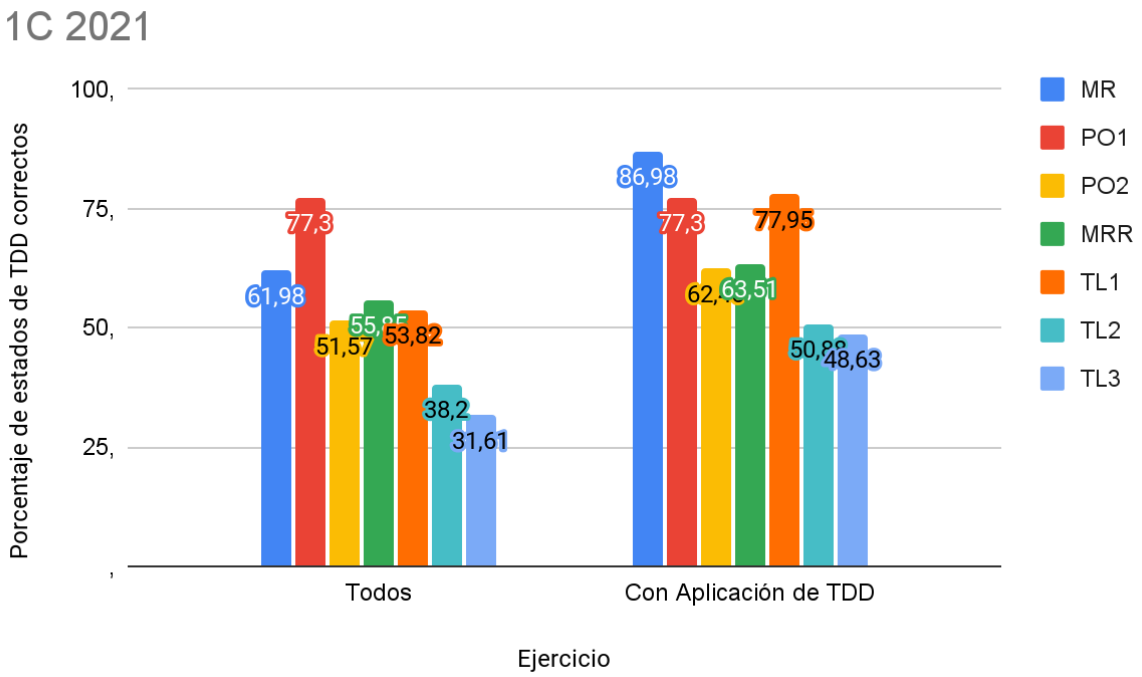


Figura 13. Aplicación de TDD por ejercicio para el 1C 2021.

- Nota de TDD en el parcial

Con el objetivo de comparar el cumplimiento de TDD en ambos cuatrimestres a partir de la nota del parcial, se elaboró un indicador con las notas que corresponden a TDD. Para esto, a cada nota del segundo parcial de los alumnos se les calculó el porcentaje de la misma correspondiente a TDD. Para esta tarea se utilizó una planilla de corrección de parciales, la cual consiste en una serie de condiciones a verificar si fueron cumplidas por el alumno en el parcial y su puntaje asociado. Por cada alumno el corrector del examen asigna el porcentaje de cumplimiento de cada condición y con estos valores (y algunos más que decide la cátedra) se determina la nota del examen.

Para calcular la nota de TDD en el parcial se tomaron las condiciones a verificar en la solución y su puntaje asociado que corresponden a TDD, excluyendo las que corresponden al desarrollo del modelo, y por cada alumno se verificó qué puntaje de TDD se obtuvo. En el *Apéndice 3* se listan las condiciones para los parciales de los cuatrimestres analizados.

Por ejemplo si se tienen las condiciones de corrección:

(TDD) C1, puntaje 0.5
(TDD) C2, puntaje 0.5
(TDD) C3, puntaje 1
(TDD) C4, puntaje 2
(TDD) C5, puntaje 2
(MODELO) C6, puntaje 3
(MODELO) C7, puntaje 1

Si un alumno obtiene una nota de 10 (diez) en el parcial, su nota de TDD será 6, ya que solo se tienen en cuenta las condiciones de TDD. El corrector puede determinar si asigna el total del puntaje de una condición o una parte del mismo, es decir, si por ejemplo la condición dice *Se testea rango dentro de x e y* con un valor de 2 puntos, el docente puede asignar 1 punto si así lo considera.

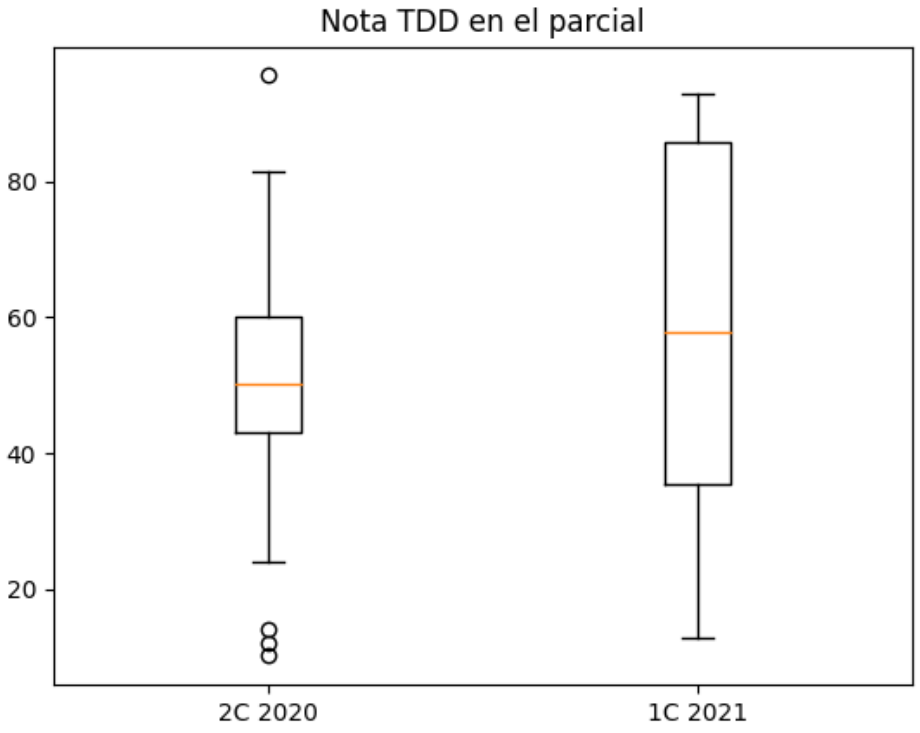


Figura 14. Porcentaje de notas de TDD para ambos cuatrimestres.

En la *Figura 14* se presentan los boxplots de las notas de TDD para ambos cuatrimestres, se puede ver como la media del 1C 2021 es mejor a la del 2C 2020, lo que concuerda con que en el cuatrimestre de la mejora se comenzó antes con la práctica de TDD, y por ende las notas (de TDD) de los alumnos mejoraron.

Correlaciones

Con el objetivo de buscar influencias entre los indicadores previamente obtenidos se realizó una matriz de correlaciones **[17].** En la misma los valores a comparar son:

- Promedio Estados TDD Correctos por Grupo
- Porcentaje de aplicación de TDD en Parcial
- Nota Parcial
- Porcentaje de nota de TDD en parcial
- Cantidad de tests
- Tiempo total por grupo

En la *Figura 15* se presentan las matrices de correlación para cada cuatrimestre analizado. En la misma se puede ver que los valores a comparar se encuentran a lo largo de las filas y columnas, en la fila ***x*** columna ***y*** se halla la correlación del indicador ***x*** con el indicador ***y***. La diagonal de la matriz representa un indicador comparado consigo mismo, por lo que la correlación es 1 (por definición). Dado que se trata de una matriz simétrica, se presentan los valores de la diagonal para abajo.

2C 2020	Prom. Ej. Grupo	TDD en Parcial	Nota Parcial	% Nota TDD Parcial	Cant. Tests.	Tiempo Total
Prom. Ej. Grupo	1,000					
TDD en Parcial	0,184	1,000				
Nota Parcial	***0,306***	0,027	1,000			
% Nota TDD Parcial	***0,229***	-0,045	***0,932***	1,000		
Cant. Tests	-0,009	-0,149	0,125	0,113	1,000	
Tiempo Total	-0,274	-0,104	0,050	0,005	0,111	1,000

1C 2021	Prom. Ej. Grupo	TDD en Parcial	Nota Parcial	% Nota TDD Parcial	Cant. Tests.	Tiempo Total
Prom. Ej. Grupo	1,000					
TDD en Parcial	***0,246***	1,000				
Nota Parcial	0,055	***0,253***	1,000			
% Nota TDD Parcial	0,099	0,187	***0,856***	1,000		
Cant. Tests	***0,248***	-0,151	***0,271***	***0,352***	1,000	
Tiempo Total	-0,069	-0,108	0,058	***0,272***	0,037	1,000

Figura 15. Matrices de correlaciones para ambos cuatrimestres.

Más allá de la alta correlación entre la *"Nota del parcial"* y el *"Porcentaje de nota de TDD en el parcial"* en ambos cuatrimestres (0.932 y 0.856) esperable dado a que el segundo es parte del primero, los demás coeficientes de correlación obtenidos no son muy elevados.

Dentro de los coeficientes con algo de énfasis para el 2C 2020 se encuentran la *"Nota del parcial"* y el *"Promedio de estados correctos de TDD por grupo (Prom Ej. Grupo)"* (0.306) lo que muestra que la técnica mejora las calificaciones de los alumnos en concordancia con el trabajo de Edwards **[13]**. Del mismo modo, el indicador del *"Porcentaje de nota de TDD en el parcial"* y el promedio antes mencionado tienen algo de correlación (0.229), lo que es consistente con el par anterior. Sin embargo, esto mismo no ocurre en el 1C 2021. Una explicación puede ser la mayor dificultad del parcial de este cuatrimestre en comparación con el anterior.

Para el 1C 2021, las correlaciones que se distinguen son "*Cantidad de tests" y "Nota TDD en Parcial"* (0.352) lo cual es razonable ya que si bien cantidad no es calidad, a mayor cantidad de tests se logró una mejor cobertura y nota de TDD. Ocurre lo mismo con la "*Nota del parcial" y "Cantidad de tests" (0.271)*.

La correlación entre la "*Nota del parcial"* y la "*Aplicación de TDD en el parcial"* (0.253) es consistente nuevamente con el estudio de Edwards **[13]** esta vez en la aplicación de la técnica durante el parcial.

El "*Promedio de estados correctos de TDD"* tiene algo de correlación con la "*Aplicación de TDD en el parcial"* (0.246) lo cual es lógico ya que si se aplicó la técnica correctamente se espera que así sea en el examen. Dicho promedio también tiene cierta correlación con la "*Cantidad de tests en el parcial"* (0.248), lo que también es razonable dado que si se aplicó correctamente la técnica se espera que aumente la cantidad de tests realizados.

Por último el *Tiempo total* y la *Nota de TDD en el parcial* tienen un poco de correlación. Esto denota que una mayor de dedicación a los ejercicios ayuda a un mejor resultado en el parcial.

En los indicadores presentados en este capítulo se pudieron ver algunas cosas para reflexionar sobre la enseñanza de TDD como que el estado más detectado es *Not Doing TDD,* lo que indica que se puede continuar mejorando la aplicación de la técnica.

Dentro de los pasos de TDD en los ejercicios el paso que más tiempo tiene es el de *Refactor* por encima de *Red,* lo que indica que a los alumnos les cuesta más mejorar el modelo por encima de hacer que el test pase, pero por otro lado es algo positivo ya que le ponen énfasis en mejorar el código.

Un patrón detectado a tener en cuenta, es la dificultad de los alumnos para aplicar TDD en un ejercicio que parte de una solución inicial (las segundas partes de los mismos).

Por otro lado, como algo positivo que mejoró la aplicación de TDD en general es haber comenzado antes en el cuatrimestre a ver el tema, esto se evidencia en las mediciones de notas y aplicación de TDD en los ejercicios tomadas en el 1C 2021 en comparación con el 2C 2020.

Otro punto positivo detectado en ambos cuatrimestres es la correlación entre la aplicación de TDD (tanto en ejercicios como en el parcial) y la nota del examen. Lo que muestra cómo la técnica ayuda en el desempeño académico.

Otro patrón detectado es el orden de los *tiempos en los estados de TDD* en los ejercicios, en el indicador con el mismo nombre. En un trabajo a futuro se podría intentar ver si este orden se mantiene y si esto provoca alguna diferencia con las demás mediciones tomadas. Idealmente se buscaría ver que el estado *Not Doing TDD* quede como el menos frecuente.

Un dato interesante obtenido en los indicadores de tiempo en los estados de TDD en los ejercicios y el parcial es el orden de *Red* y *Refactor,* en el parcial se prioriza que los test pasen (mayor preponderancia de *Red*) y en los ejercicios se busca mejorar el modelo (mayoría de *Refactor*). Esto es un indicio de cómo el tiempo (acotado en el examen y mayor en los ejercicios) afecta la forma de trabajar.

4. Seguimiento de alumnos

Con el fin de analizar la evolución del aprendizaje de TDD se obtuvieron indicadores adicionales a los ya presentados anteriormente, sobre una selección de alumnos y grupos. Se seleccionaron alumnos bajo el criterio de haber obtenido notas similares en el parcial dentro del grupo. Dado que, se espera hallar datos relevantes en cuanto al aprendizaje de TDD para alumnos de desempeños semejantes. Se seleccionaron 6 alumnos y 3 grupos en total para ambos cuatrimestres analizados. A los alumnos seleccionados, recordar que se los referencia con un número para preservar el anonimato.

Como se puede ver en la *Figura 16* se seleccionaron alumnos donde la nota del parcial para un mismo grupo sea lo más similar posible.

Para los grupos 22 y 6 la aplicación de TDD en el parcial mejoró con respecto a la de los ejercicios (columnas TDD en Parcial y Prom. Ej. Grupo respectivamente). Recordando, la columna Prom. Ej. Grupo es el promedio de porcentajes de estados correctos de TDD en todos los ejercicios del grupo y la columna *TDD en parcial* es el porcentaje de estados correctos de TDD en el parcial. Sin embargo, en los grupos 1, 21 (1C 2021) y 24 se puede visualizar como un alumno mejoró y el otro desmejoró en la aplicación de TDD de los ejercicios al parcial. A diferencia del grupo 21 (2C 2020) donde la aplicación de TDD en el trabajo en grupo y en el parcial son bastante similares para ambos integrantes. Esto da indicio de que haber ejercitado la técnica mediante *pair programing* no siempre lleva al mismo desempeño en desarrollos individuales.

2C 2020

Alumno	Grupo	Prom. Ej. Grupo	TDD en Parcial	Nota Parcial
1	1	46,60	26,03	7,96
2	1	46,60	78,89	6,125
46	6	9,27	65,12	5,93
9	6	9,27	44,44	4,62
37	21	60,16	60,29	9,03
38	21	60,16	70,74	7,61

1C 2021

Alumno	Grupo	Prom. Ej. Grupo	TDD en Parcial	Nota Parcial
41	21	31,40	13,03	7,5
42	21	31,40	60,49	7,4
43	22	27,78	75,92	6,3
44	22	27,78	60,91	6,125
47	24	47,72	39,90	7,3
48	24	47,72	90,98	6,5

Figura 16. Detalle de alumnos seleccionados de los cuatrimestres analizados

A partir de los datos seleccionados, se elaboraron indicadores que aplican solo a los alumnos y grupos anteriormente presentados. A continuación se presentan dichos indicadores.

- Detección de ciclos de TDD por ejercicio y tiempo de los mismos (por grupo)

Con el fin de medir la aplicación de TDD a lo largo de los ejercicios se buscaron los ciclos de la técnica. En este indicador se detectaron los ciclos de TDD por ejercicio. Para esto se desarrolló un programa que recorre los archivos .csv exportados previamente por el procedimiento detallado en la sección **2.8**. Para detectar los ciclos se utiliza como marca el estado *Writing a failing test,* es decir, se recorre el archivo por grupo y ejercicio y se comienza a contar un ciclo al detectar el estado anteriormente mencionado. Luego, al detectar otro del mismo tipo, se comienza a contar un nuevo ciclo. Adicionalmente en el archivo .csv se cuenta con el dato del tiempo, el cual se suma por ciclo.

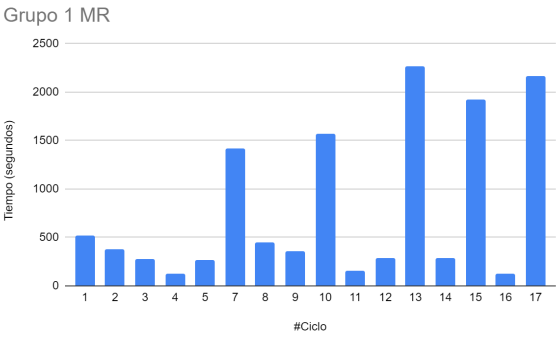
En un escenario ideal de la aplicación de TDD, se debería ver una alta cantidad de ciclos y un tiempo relativamente bajo en cada uno de ellos. Como se puede apreciar en los gráficos hay una disparidad entre las cantidades de ciclos y los tiempos en los distintos ejercicios.

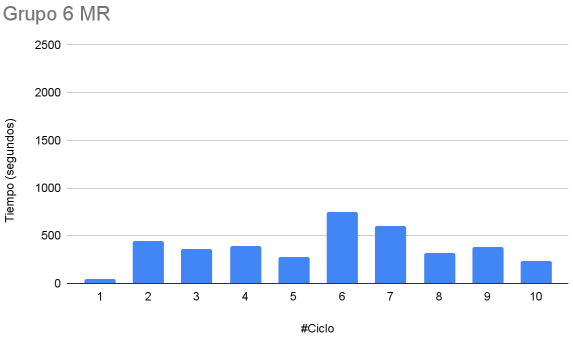
En la *Figura 17* se presentan los gráficos de ciclos de TDD y el tiempo en cada uno. Se muestran algunos de los ejercicios de los grupos ya que no en todos se pudieron detectar ciclos de TDD.

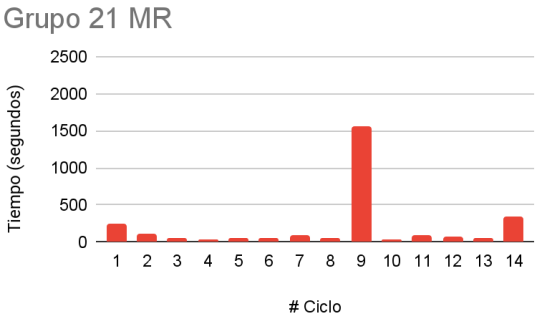
Para el caso de MR, que es el primer ejercicio donde se aplica TDD, se ven varios ciclos con una tendencia a ser de corto tiempo a excepción de algunos que presentan uno mayor, esto último se presume se debe al refactoring realizado. Este ejercicio, recordando, está bastante guiado por los docentes en su solución, con lo que se vé una buena aplicación de la técnica.

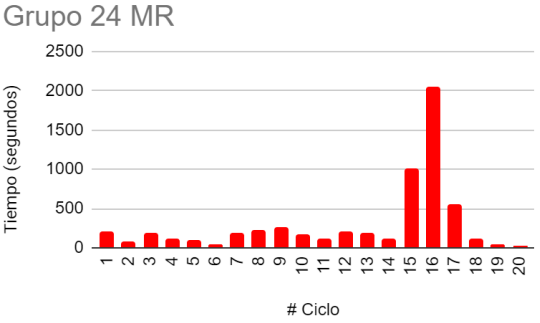
En el 2C 2020 en el ejercicio TEL también se ven varios ciclos con un tiempo por ciclo que tiende a ser relativamente bajo. Este es el siguiente ejercicio a MR, con lo cual la complejidad no llegó a subir mucho, por eso se mantiene una buena aplicación de TDD.

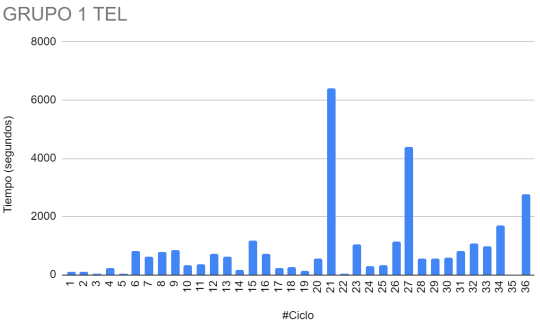
En los siguientes ejercicios analizados, hay variaciones en cuanto a la cantidad de ciclos y sus tiempos. Para las muestras de los ejercicios PO2 y TL3 hay muchas de estas variaciones. Hay bastante menos guía de parte de los docentes en PO2 y hay mucho refactoring para hacer, dado que es una continuación de un ejercicio anterior, por lo que la aplicación de la técnica se vió afectada. Esto se evidencia en los pocos ciclos detectados y el alto tiempo en los mismos, particularmente en el primer y último ciclo detectado. TL3 es un ejercicio bastante complejo, para el grupo 21 del 2C 2020 se vé una aplicación relativamente buena de TDD, mientras que para el grupo 1 del mismo cuatrimestre, se visualiza un patrón similar a PO2.

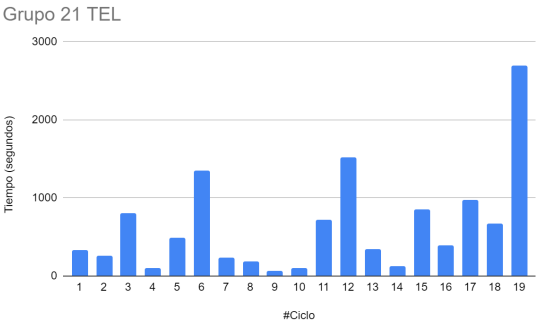

Grupo 1 MR
Tiempo (segundos)
2500
2000
1500
1000
500
0
1 2 3 4 5 7 8 9 10 11 12 13 14 15 16 17
#Ciclo

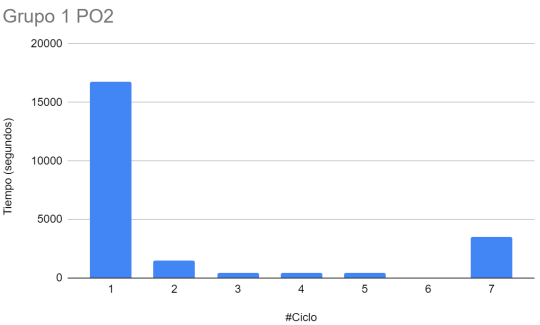

Grupo 6 MR
Tiempo (segundos)
2500
2000
1500
1000
500
0
1 2 3 4 5 6 7 8 9 10
#Ciclo

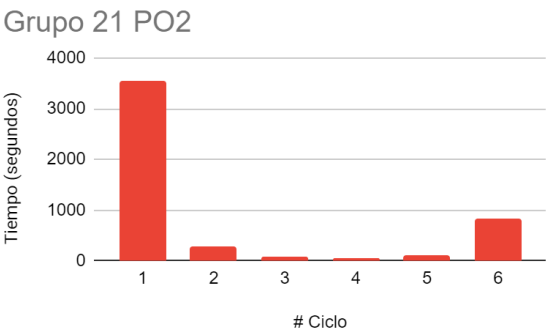

Grupo 21 MR
Tiempo (segundos)
2500
2000
1500
1000
500
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14
Ciclo


Grupo 24 MR
Tiempo (segundos)
2500
2000
1500
1000
500
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
Ciclo


GRUPO 1 TEL
Tiempo (segundos)
8000
6000
4000
2000
0
#Ciclo


Grupo 21 TEL
Tiempo (segundos)
3000
2000
1000
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
#Ciclo


Grupo 1 PO2
Tiempo (segundos)
20000
15000
10000
5000
0
1 2 3 4 5 6 7
#Ciclo


Grupo 21 PO2
Tiempo (segundos)
4000
3000
2000
1000
0
1 2 3 4 5 6
Ciclo

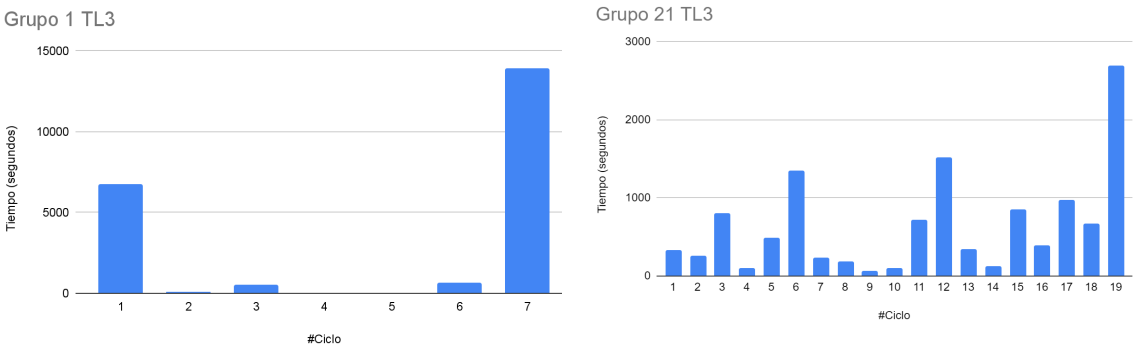


Figura 17. Ciclos y tiempos de los mismos para los grupos seleccionados. En azul el 2C 2020 y en rojo el 1C 2021.

Por las variaciones en los desarrollos según diversos factores (dificultad del ejercicio, ayuda docente, etc.), se desprende que, al momento de cambiar el ejercicio, y con ello el contexto de desarrollo, a los alumnos se les dificulta aplicar la técnica de forma óptima.

-Conteo de líneas de código por ejercicio vs. líneas de código de la solución de la cátedra (por grupo)

Con el objetivo de comparar el código entregado por los alumnos con la solución de la cátedra, se realizó un conteo de las líneas de código de los ejercicios entregados y el de la cátedra para luego compararlos. Para este indicador se tomaron los archivos de las soluciones de los ejercicios de los alumnos y se contaron las líneas mediante un programa desarrollado para este trabajo. En dicho conteo se separaron las líneas correspondientes a test y al desarrollo del modelo de la solución. El mismo criterio se aplicó a la solución de la cátedra.

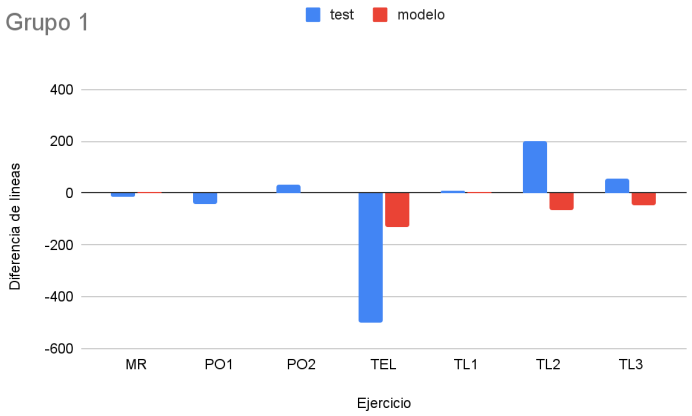
En la *Figura 18* se presentan los gráficos con las diferencias entre las cantidades de líneas de test y modelo de los grupos seleccionados y la solución de la cátedra. Una diferencia positiva (columna del gráfico hacia arriba) indica que el código del grupo tiene más líneas que el de la solución de la cátedra. En caso contrario será una diferencia negativa (columna del gráfico hacia abajo).

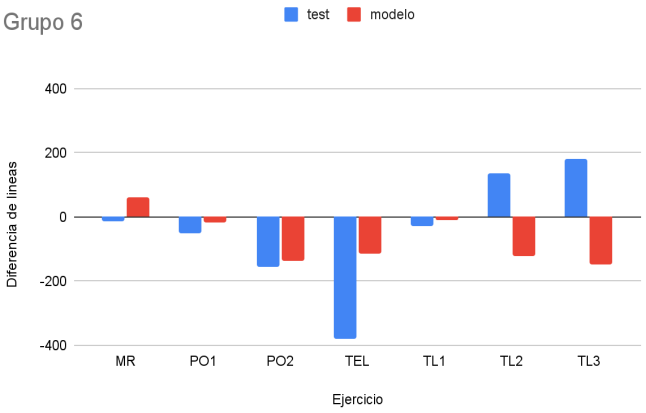
En un escenario ideal, las diferencias de líneas de código deberían estar cercanas a 0.

En el 2C 2020, se ven las bajas diferencias en los ejercicios MR, PO1 y TL1 para los grupos 1 y 6, y también se vé para PO2 en el caso del grupo 1. Recordando, en estos ejercicios hubo bastante ayuda de los docentes con lo que es esperable que las diferencias de código tiendan a ser menores que en los demás. Para el caso de TEL, los alumnos no llegaron a completar lo pedido, por lo que hay una diferencia notable en ambos grupos nombrados. Hay unas diferencias un poco más notorias en TL2 y TL3 que son los ejercicios más complejos de la cursada.

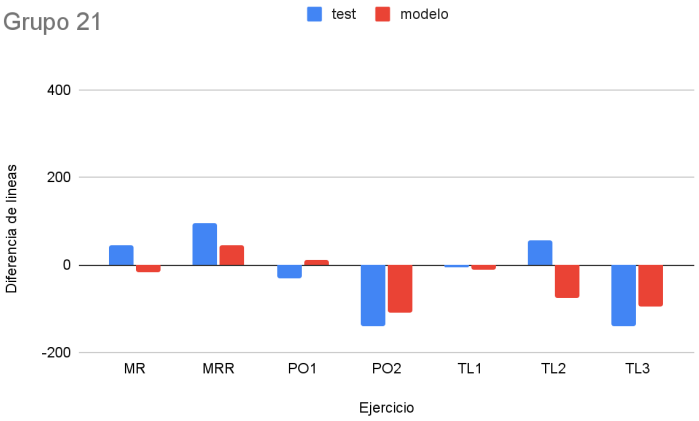
Para el 1C 2021, se llega a visualizar una relativa baja diferencia de líneas en el ejercicio PO1 para los 3 grupos y en menor medida en MR. Por otro lado, en los demás ejercicios MRR, PO2, TL2 y TL3 se da menos guía en las soluciones. TL3 es particularmente el ejercicio más complejo. Por estos motivos, se pueden visualizar en los gráficos mayores diferencias para estos ejercicios.

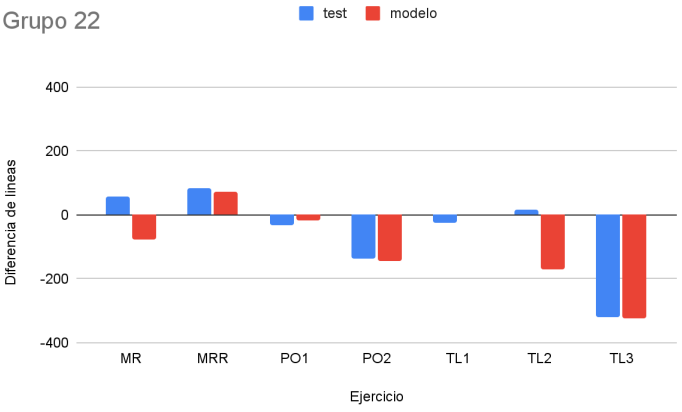
2C 2020





1C 2021





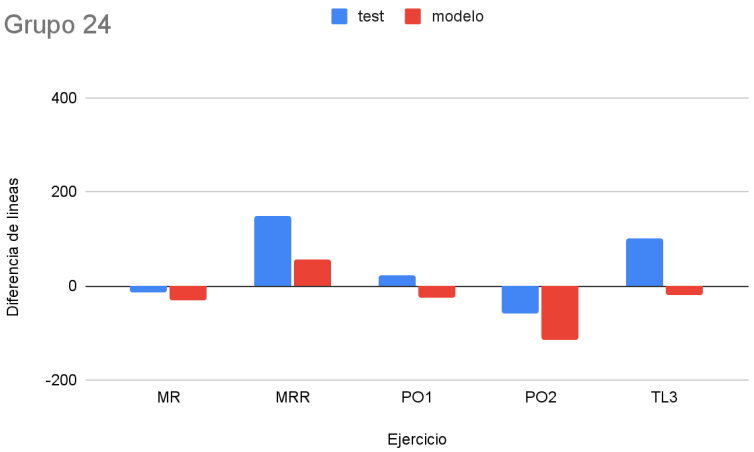


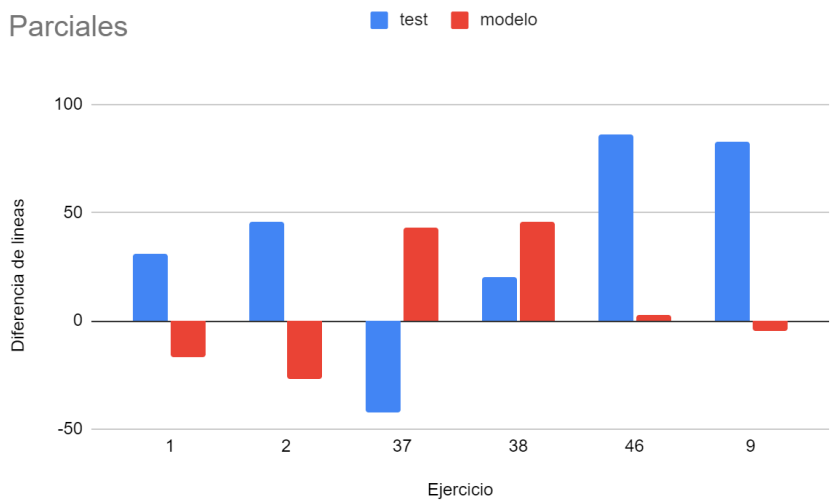
Figura 18. Diferencias de líneas de código de los ejercicios entre las soluciones de los grupos seleccionados para ambos cuatrimestres y la cátedra.

Las diferencias de líneas de código halladas dan un indicio de las dificultades que tuvieron los alumnos en la resolución de los ejercicios. Esto da una idea del contexto de desarrollo en el que los alumnos aprendieron/aplicaron TDD.

- *Conteo de líneas de código de la solución del parcial vs. líneas de código de la solución de la cátedra (por alumno)*

Este indicador es similar al anterior, con la diferencia de que se aplicó a las soluciones del parcial. El objetivo es verificar las dificultades de los alumnos en el examen a partir de las diferencias de líneas de código con las de la solución de la cátedra. En contraposición al indicador anterior, este se aplicó a los alumnos dado que el parcial es individual. A continuación en la *Figura 19* se visualizan las diferencias.

2C 2020



1C 2021

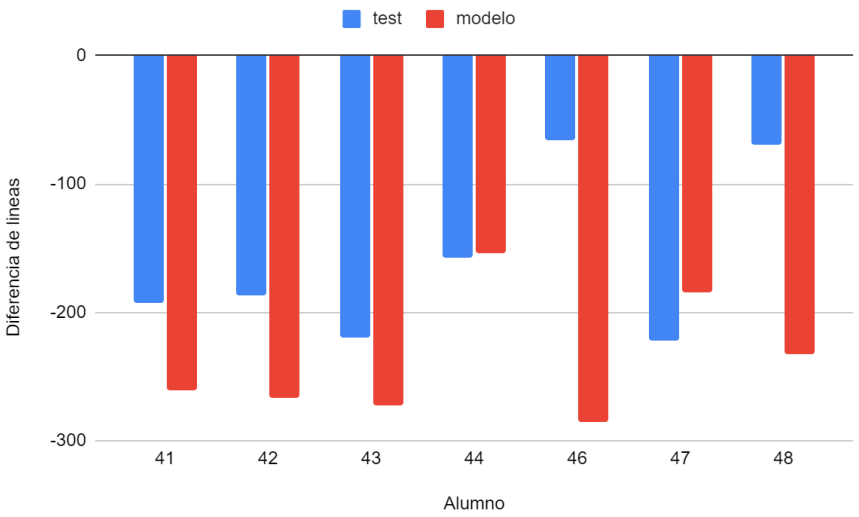


Figura 19. Diferencias de líneas de código entre las soluciones de los grupos seleccionados y la cátedra para el parcial para ambos cuatrimestres.

En el 2C 2020 se pueden ver resultados variados para los alumnos, a diferencia de los resultados para el 1C 2021, recordando, el examen de este último cuatrimestre nombrado fué más difícil que el del 2C 2020. Esto se puede ver en que la cantidad de líneas de todos los alumnos siempre son menores que las de la solución de la cátedra, lo cual muestra que para este examen el contexto para aplicar TDD fue mucho más complicado.

-Detalle de variación de estados correctos de TDD por ejercicio

Con el fin de hacer un seguimiento más detallado de la aplicación de TDD en los grupos seleccionados a lo largo de los ejercicios, se obtuvo el promedio de estados correctos de TDD por ejercicio. En la *Figura 20* se presentan los promedios nombrados por ejercicio. Los mismos se encuentran ordenados en el orden en que se resolvieron.

Para el 2C 2020 no se pudieron extraer datos de TEL, PO2 y TL2 para el grupo 6, ni para PO2 del grupo 21, no se visualizarán estos datos en el gráfico. En cuanto a los hallazgos de los datos, la aplicación de TDD del grupo 6 es en general bastante baja y sus integrantes sacaron las menores notas en el examen, con lo cual la técnica no influyó en su manera de aprender. Para los grupos 1 y 21 el ejercicio MR tiene una buena aplicación de TDD, donde hubo bastante guía de los docentes. Para los demás ejercicios no se detecta una tendencia determinada, hay ejercicios con buena y mala aplicación para ambos grupos.

En el 1C 2021 para el grupo 22, no se pudieron obtener datos relevantes para el presente indicador, por lo que solo se presentarán los promedios relevados para los grupos 21 y 24. Tampoco se obtuvieron datos para PO1, MRR, TL1 y TL3 para el grupo 21 ni para el ejercicio PO2 para el grupo 24, por este motivo no figuran en el gráfico. Para los demás, se puede ver que nuevamente hay un alto porcentaje de estados correctos para el ejercicio MR en ambos grupos. Nuevamente, esto se presupone que se debe a que hubo una mayor ayuda de la cátedra para este ejercicio (de igual modo que en PO1). Ahora bien, para los demás ejercicios, se vé que hay una variación considerable de los porcentajes entre ellos, coincidiendo con el cuatrimestre anterior, no siguiendo un patrón específico. Lo que evidencia nuevamente que a los alumnos en general se les dificulta aplicar la técnica en cualquier contexto.

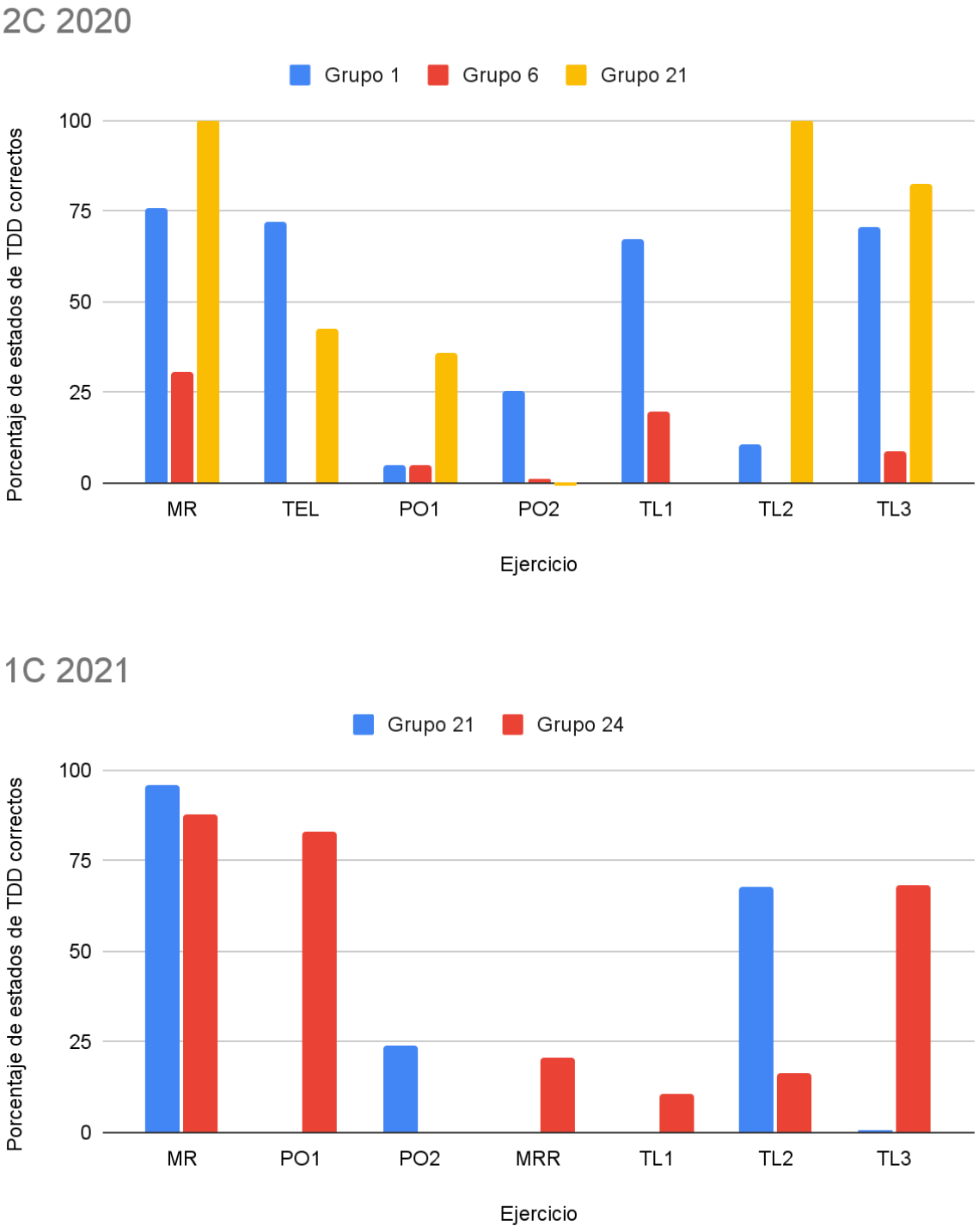


Figura 20. Promedio de estados correctos de TDD por ejercicio para los grupos 21 y 24.

En ambos cuatrimestres se pudo ver que en MR que es un ejercicio con bastante guía de los docentes la técnica fue bien aplicada, pero luego para los siguientes los resultados fueron bastante variados. Algo que no se pudo hallar en ambos cuatrimestres es una mejoría a lo largo del tiempo en la aplicación de la técnica, aunque esto se comentó anteriormente que puede no encontrarse.

Idealmente en este indicador se esperaba detectar un aumento en los porcentajes de aplicación de TDD a medida que avanzan los ejercicios. Esto sería lo ideal en términos de mejora de aprendizaje de la técnica, sin embargo como se vió anteriormente, esto no ocurrió.

Si bien el criterio para el análisis detallado fue seleccionar alumnos con notas similares en el parcial, se pudo verificar que el desempeño de los mismos fue distinto en cuanto al aprendizaje de TDD, con lo cual el *pair programing* (en el contexto particular de la pandemia) puede no tener una alta influencia en la adquisición de la técnica.

En los indicadores presentados en esta sección se pudo apreciar como la aplicación de la técnica es bastante buena cuando hay guía de los docentes, lo cual es esperable. Sin embargo, los alumnos tienden a tener dificultades para aplicarla en ejercicios menos guiados y con una dificultad mayor.

El no poder detectar ciclos de TDD también es una señal de la dificultad de aplicación de la técnica, además de lo visualizado en los ciclos presentados donde no se detecta un patrón de mejora a lo largo de los ejercicios.

Se esperaba que el aprendizaje de TDD fuese mejorando de ejercicio a ejercicio para alumnos con un buen rendimiento en el examen y de notas similares. Sin embargo, cada ejercicio tiene una dificultad diferente y en el examen juegan otros factores, como tiempo y resolución individual. Por lo tanto, los alumnos en general no pueden aplicar TDD en cualquier ámbito, debido a que están aprendiendo la técnica.

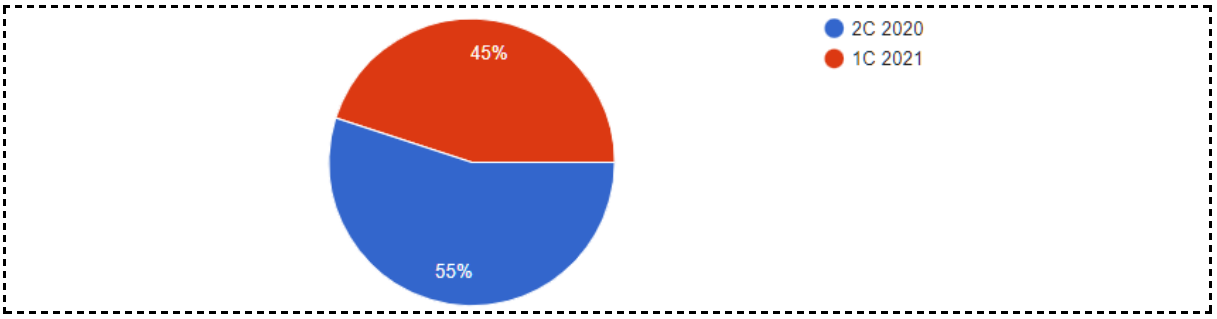
5. Encuesta de experiencia con TDD

Se realizó una encuesta a los alumnos de los cuatrimestres analizados a fin de validar los resultados de los indicadores obtenidos en el presente trabajo y complementar la información obtenida.

De los 97 alumnos de ambos cuatrimestres se obtuvieron 60 respuestas, con lo cual, se puede asumir una representatividad considerable de las respuestas obtenidas.

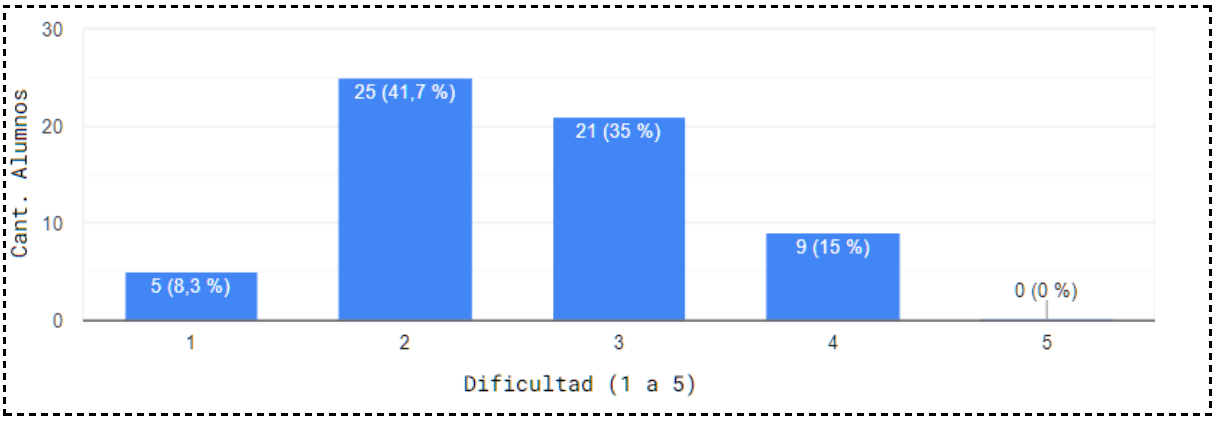
A continuación se presentan las preguntas realizadas junto a las respuestas obtenidas.

Pregunta 1: ¿Cuándo cursaste?



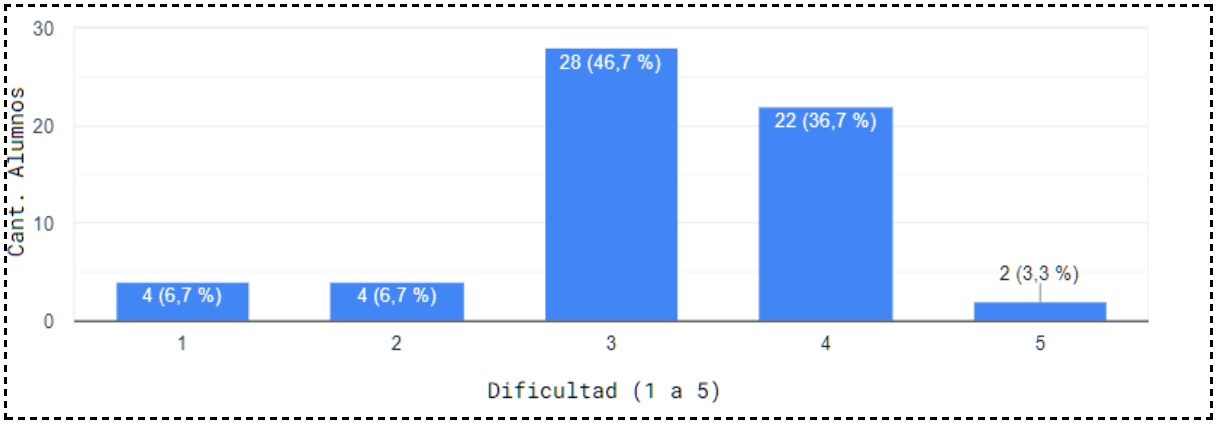
La encuesta fue respondida por una cantidad balanceada de alumnos de cada cuatrimestre por lo que hay una buena representatividad de los datos.

Pregunta 2: ¿Cuánto te costó aprender la técnica?



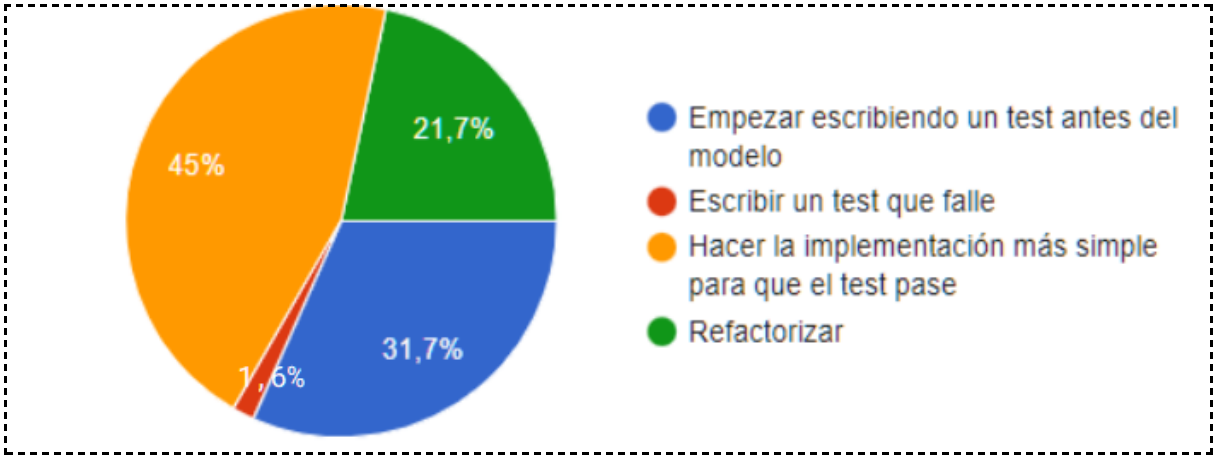
En esta pregunta los alumnos manifestaron que no tuvieron grandes dificultades para aprender la técnica, ahora bien, comprender los pasos y los ejemplos dados en clase es una parte del aprendizaje, otra es su aplicación. Como se verá más adelante la aplicación de TDD si fue de una dificultad mayor según la percepción de los alumnos.

Pregunta 3: ¿Cuánto te costó aplicar la técnica?



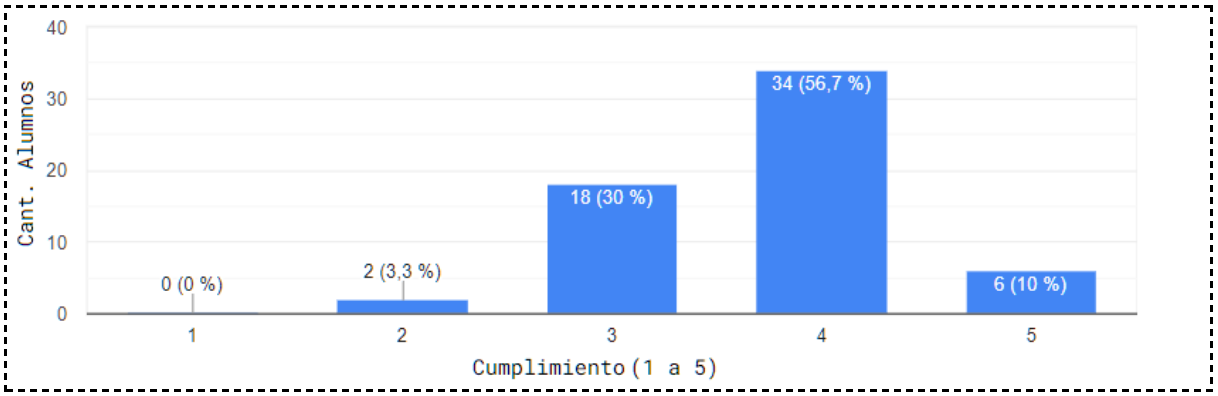
Como se puede ver, si bien la apreciación de los alumnos es que la dificultad de aprender la técnica es entre media a baja, la dificultad de aplicarla es entre media y alta. Lo cual es concordante con la alta cantidad de tiempo de los estados *Not Doing TDD*.

Pregunta 4: ¿Qué paso te parece más difícil de aplicar?



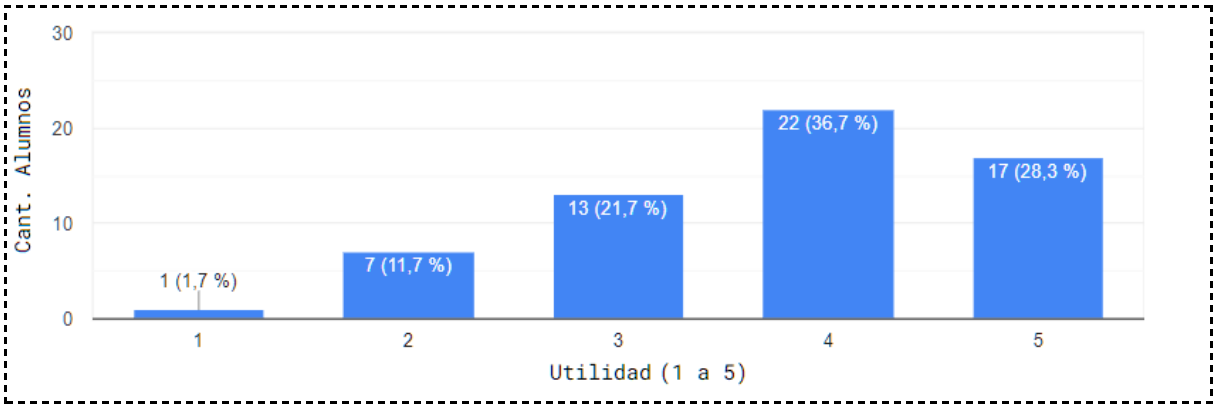
El paso que más cuesta aplicar es *Hacer la implementación más simple para que el test pase*. Este es un punto crítico ya que se debe dar la solución de la funcionalidad que el test en cuestión debe satisfacer. Le sigue *empezar escribiendo un test antes del modelo*, lo que se supone se debe a que lo más usual en el desarrollo con el que vienen trabajando los alumnos es la metodología *test-last* y TDD propone justamente lo contrario. Continúa en dificultad el paso de *Refactorizar* donde se requiere pensar cómo optimizar el código, lo cual puede no ser sencillo. En última instancia, con menos de 2%, el paso de *escribir un test que falle,* lo cual es algo positivo ya que muestra que el primer paso de TDD es el que menos dificultades presenta y es el que ayuda a dar un punto de inicio al desarrollo con relativa facilidad.

Pregunta 5: ¿Qué tan bien sentís que cumpliste/seguiste los pasos de la técnica?



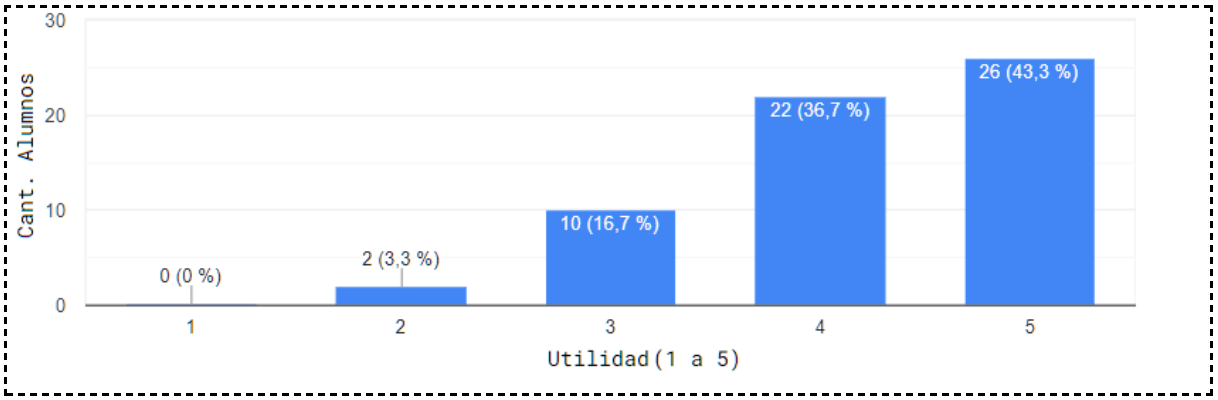
Hay una alta confianza de los alumnos en cuanto a la aplicación de la técnica, lo cual es algo discordante con la elevada cantidad de estados *Not Doing TDD*.

Pregunta 6: ¿Qué tan útil te parece que fué TDD en cuanto a encontrar errores/situaciones que no tuviste en cuenta?



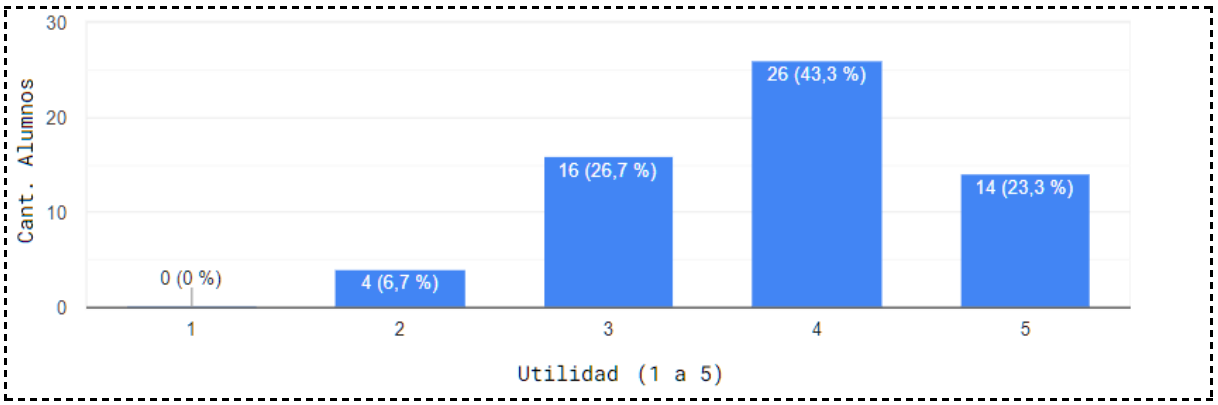
Se puede ver que los alumnos encontraron en TDD una manera de detectar errores o situaciones no tenidas en cuenta en el desarrollo, lo cual es un punto positivo adicional para la enseñanza de la técnica.

Pregunta 7: ¿Qué tan útil fue TDD en el momento de realizar cambios en el código y preservar el modelo funcionando?



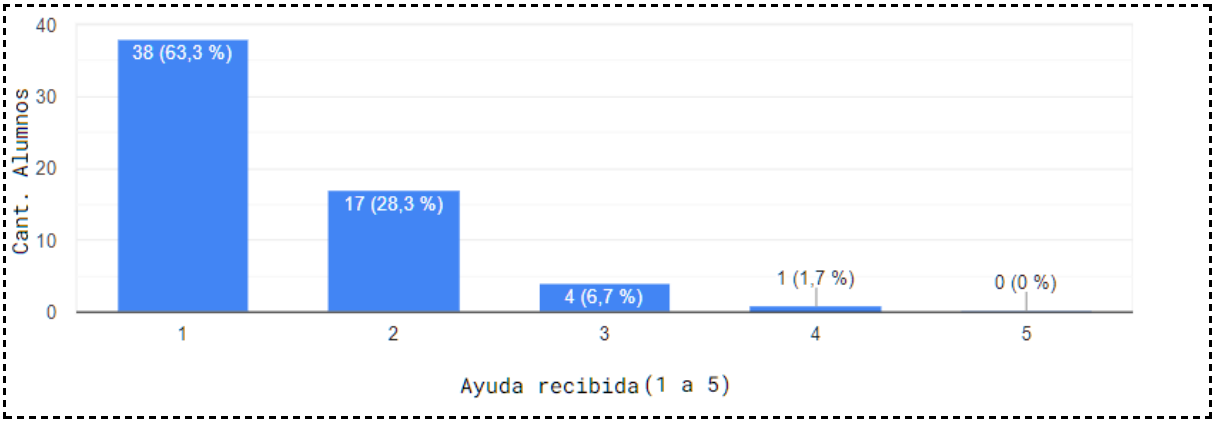
Nuevamente se evidencia otro punto favorable para TDD que es la preservación del modelo funcionando ante ajustes realizados al código. Es también positivo que los alumnos lo reconozcan.

Pregunta 8: ¿Qué tan útil fue TDD en general?



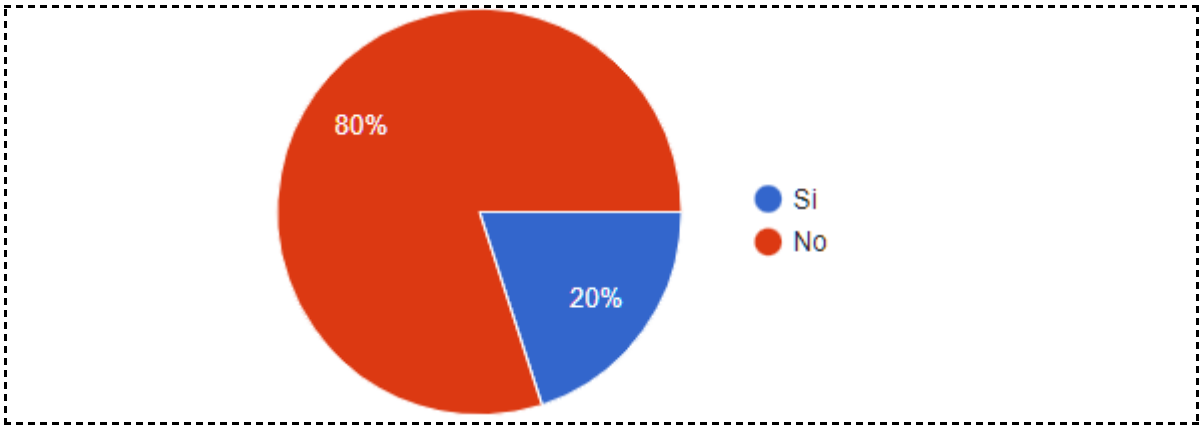
En cuanto a la apreciación de la utilidad de TDD para detectar errores, hacer cambios y en general, se evidencia que es de media a alta, con lo que la técnica es muy bien aceptada por los alumnos.

Pregunta 9: ¿Recibiste ayuda de otros grupos en la resolución de los ejercicios?



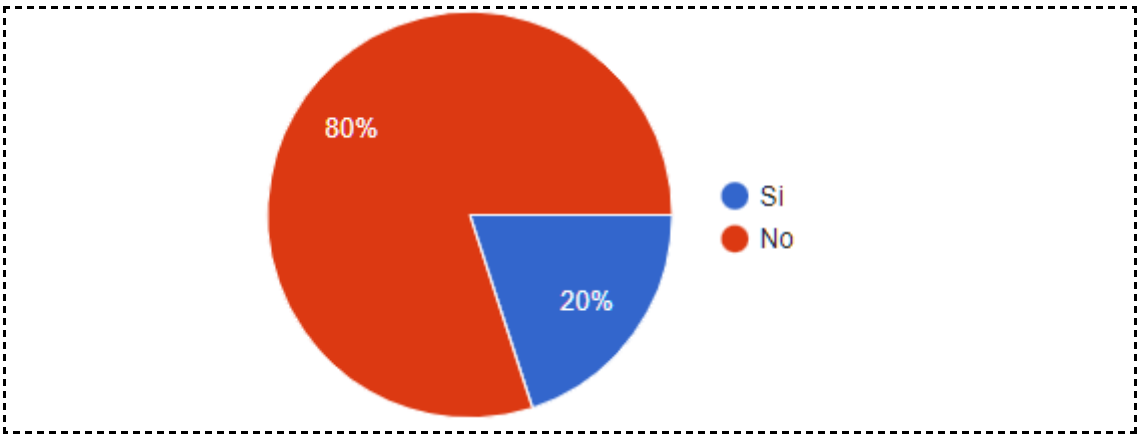
Se puede ver que los alumnos recibieron de poca a ninguna ayuda de otros grupos en la resolución de los ejercicios. El motivo de esta pregunta es que a los docentes siempre les queda la duda de que los alumnos pudieron haberse "copiado" o tomado soluciones de otros cuatrimestres ya que en el esquema de trabajo de los ejercicios esto no se puede verificar.

Pregunta 10: ¿Compartiste tu/tus soluciones de los ejercicios con otros grupos?



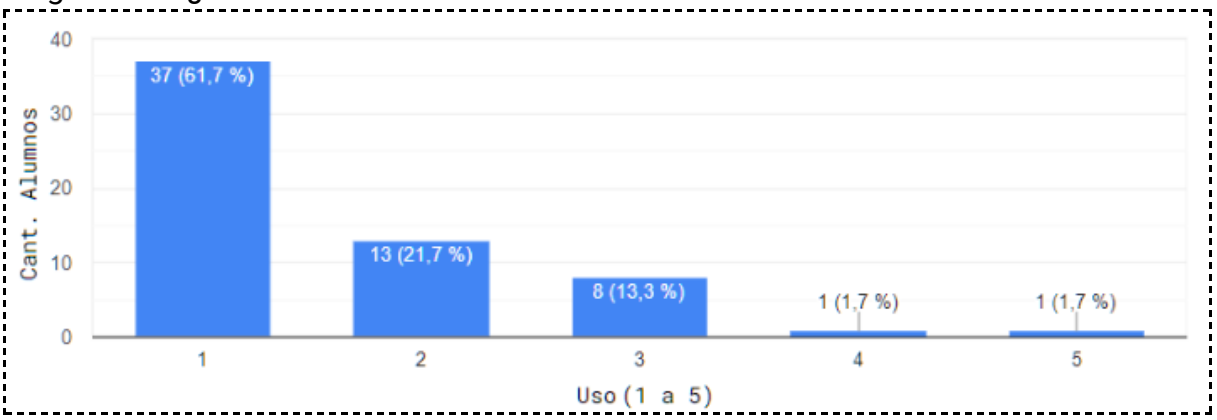
Si bien una parte menor de los alumnos admite haber compartido sus soluciones con otros grupos, la mayoría no lo hizo. Es concordante con la pregunta anterior.

Pregunta 11: ¿Comparaste tu/tus solución/es con la de otros cuatrimestres?



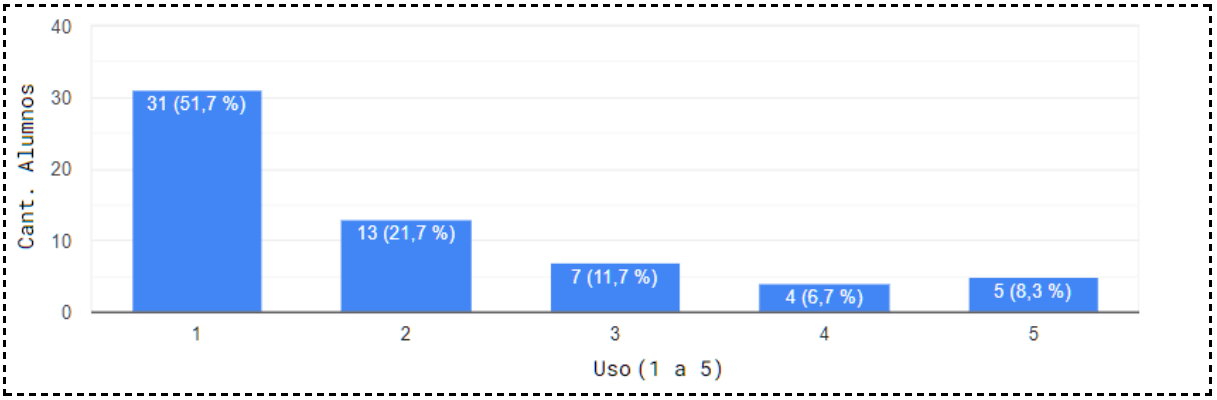
Si bien hay ciertos porcentajes que indican que el trabajo de los grupos no fue totalmente individual dichos valores son relativamente bajos, con lo que se puede asumir que no se ve afectada la validez de los resultados obtenidos en el presente trabajo.

Pregunta 12: ¿Usaste TDD Gurú?



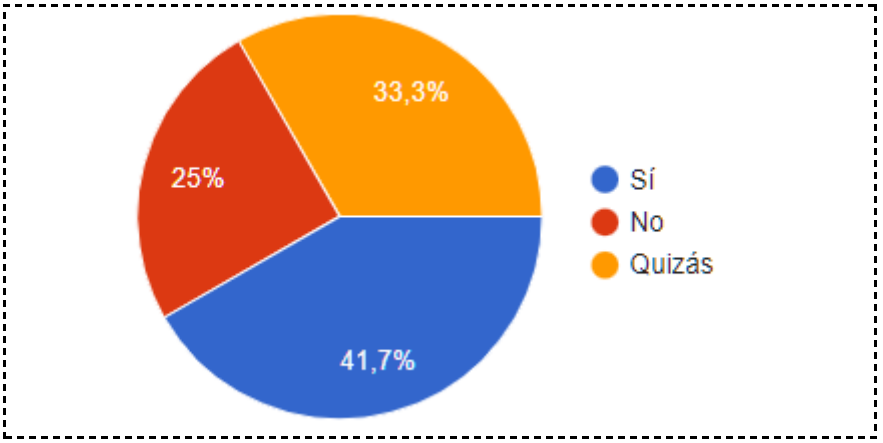
El uso de TDDGurú fue bastante bajo. Una medida a tomar sería buscar la manera de que los alumnos utilicen la herramienta para aprovechar la visibilidad sobre la aplicación de TDD que aporta. A futuro si se logra mejorar el uso de TDDGurú, sería interesante ver si impacta en las preguntas de aplicación y aprendizaje de la técnica de la encuesta.

Pregunta 13: ¿Usaste Coverage/Cobertura de código?

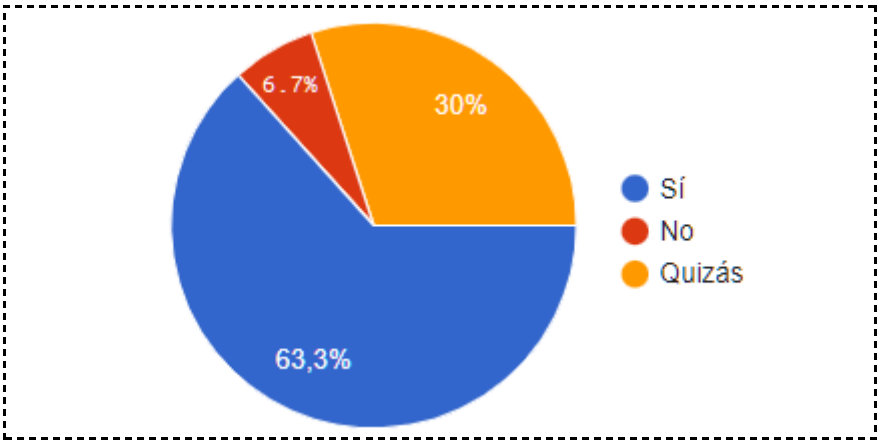


El uso de las herramientas TDD Gurú y Coverage es bastante bajo, lo cual implica que se desaprovechan oportunidades importantes de validar la buena aplicación de TDD mientras se aprende. Una sugerencia es que en la siguiente clase a la presentación de la técnica se muestren con ejemplos el uso de estas herramientas.

Pregunta 14: ¿Usarias TDD para otras materias de la facultad?



Pregunta 15: ¿Usarias TDD para desarrollos laborales?



Hay un porcentaje alto de uso o posible uso de la técnica, tanto en desarrollos laborales como en otras materias de la facultad. Esto último es consistente con el resultado anterior de la buena aceptación de TDD. Sin embargo, hay un porcentaje de gente que no utilizaría la técnica en el contexto de otras materias de la carrera por sobre desarrollos de índole laboral. Este último hallazgo denota que en otras materias de la carrera los alumnos ven alguna incomodidad para aplicar la técnica, lo que puede ser oportuno analizar más en profundidad si por ejemplo, se quiere llevar TDD más allá de la materia *Ingeniería de Software I* o como mínimo incorporar tests automáticos.

6. Conclusiones

A lo largo del presente trabajo se pudo obtener información, en base a los datos analizados, de cómo los alumnos aprenden TDD, características de la aplicación de la técnica y percepciones de los alumnos en cuanto a su aprendizaje.

La información obtenida evidenció los marcos donde los alumnos tienen mayores dificultades. Uno de los mismos son los ejercicios continuación de otros (PO2 y TL2 por ej.). En más de un indicador se pudo ver como el cumplimiento de TDD se deterioraba con respecto al desempeño anteriormente aplicado en la técnica. Si bien este tipo de ejercicios entrenan el refactoring además de, la práctica de extender/modificar el modelo y preservarlo funcionando, trae complicaciones a efectos de aplicar TDD.

Otra dificultad detectada, según la percepción de los alumnos es el paso de TDD de realizar la implementación más simple para que el test pase. Debido a que la forma más común de encarar un desarrollo con la que venían los alumnos es *test-last,* pudo serles complicado enfocar dicho paso de la técnica, ya que, en un test como lo venían trabajando en otros enfoques, se verifica el modelo completo al final. Un punto a reflexionar en la docencia es si se podrían agregar ejemplos de implementaciones lo más simple posible para que un test pase para distintas situaciones ó, ante distintas implementaciones poder debatir cuál sería la de menor complejidad que haga pasar un determinado test. La dificultad que le sigue en cuanto a la aplicación de pasos es la de escribir un test antes que el modelo, lo cual también es razonable por el enfoque *test-last* nombrado.

En cuanto a la aplicación de TDD detectada, algo llamativo es que el estado *Not Doing TDD* fue el más hallado en los indicadores, además de que en el código de los alumnos no siempre se pudieron detectar ciclos de la técnica. Es más, se mencionó que hubo grupos que ni siquiera habían intentado aplicarla. Claramente, los alumnos pueden mejorar en el cumplimiento de TDD durante el cuatrimestre. Una forma puede ser que los docentes incentiven a los alumnos a utilizar TDDGurú. En la encuesta los alumnos admiten no haber utilizado la herramienta nombrada y esto es un desaprovechamiento de los beneficios de la misma para validar el aprendizaje de TDD. Un posible motivo, aunque discutible, es que los alumnos no están acostumbrados a reflexionar sobre lo que hacen, por eso no utilizan TDDGuru ni Coverage, ya que, su objetivo no es mejorar en TDD sino cumplir con lo pedido en las consignas. Otro incentivo para los alumnos a aprender y utilizar la técnica son los resultados del indicador de correlaciones, ya que se pudo ver que la aplicación de TDD estaba correlacionada con la nota del examen, es decir que a mejor cumplimiento de la técnica, mejores resultados académicos se obtendrán.

Por el lado del aprendizaje de los alumnos, en el capítulo de seguimiento se pudo ver cómo los alumnos seleccionados de un mismo grupo no siempre tenían el mismo desempeño en TDD y en el examen. Quizás hay detalles adicionales externos a los datos analizados pero, por lo visto, el aprendizaje en *pair programing* no

siempre puede influenciar el trabajo personal. Se aclara nuevamente que los cuatrimestres analizados se trabajó de manera remota a causa de la pandemia del COVID-19.

Otro punto visto en el aprendizaje fue el del indicador del *Tiempo en los estados de TDD*. Se pudo ver que en ambos cuatrimestres se preserva el orden (en cuanto al tiempo) de los estados para los ejercicios y parciales, exceptuando el caso de los estados *Red* que tiene más tiempo en los ejercicios y *Refactoring* en los parciales. Esto se deduce que se debe a que el estudiante en el contexto de un parcial prioriza que los tests pasen (estado *Red*) por sobre la mejora del mismo (estado *Refactoring*) debido al tiempo acotado de la evaluación. En este punto se vé la influencia del tiempo en la forma de trabajar.

Como puntos positivos en el cumplimiento de la técnica, se pudo ver que en ejercicios con mayor guía/ayuda de los docentes la aplicación de TDD fue mejor, lo cual no es algo llamativo, pero sí se pudo ver una mejora global en el cumplimiento de la técnica en el 1C 2021, donde la presentación de TDD se dió antes que en el 2C 2020. Presentar antes el tema trajo mejores resultados en el mismo, esto es algo a considerar mantener en futuros programas de la materia.

Según la percepción de los alumnos TDD tiene una alta utilidad en general y puntualmente para modificar el modelo y preservarlo funcionando. Sumado a detectar errores o situaciones no previstas. Es decir, hay una alta aceptación de la técnica por parte de los alumnos lo cual es algo a tener en cuenta para llevar su aplicación más allá de la materia Ingeniería de Software I, o también, incluir en las primeras materias de la carrera temas como la utilización de tests automáticos. Sin embargo, en otras preguntas de la encuesta los alumnos consideran que utilizarían TDD laboralmente, pero hay una parte que indica que no aplicaría TDD en otras materias de la facultad. Lo que es un punto a reflexionar, ya que, siendo una carrera con una carga considerable en desarrollo en lenguajes de programación, sería bueno aprovechar los beneficios de TDD en más materias.

El planteamiento inicial del trabajo era ver que a medida que los alumnos ejercitan la técnica, las mediciones determinaran que se iba mejorando, ya sea aumentando la cantidad de estados correctos de TDD, reduciendo los tiempos, aumentando los ciclos correctos de la técnica, etc. Sin embargo, a lo largo de varios indicadores se pudo ver que a los alumnos en general se les dificulta aplicar TDD en cualquier contexto. Ya que, la técnica está en proceso de aprendizaje y la misma es aplicada en los ejercicios, con dificultad variable, y el parcial, con el contexto de la limitación de tiempo para su resolución. No se logró ver en los indicadores un patrón de mejora en la aplicación de TDD a lo largo del tiempo. Los alumnos comienzan aprendiendo TDD como una “receta” sin reflexionar en sus pasos o el porqué de los mismos, puede ser que solo un cuatrimestre no alcance para madurar la técnica y poder aplicarla en toda situación.

7. Trabajo a futuro

En el presente trabajo si bien se detectaron algunos patrones entre los cuatrimestres analizados, quedaron planteadas algunas preguntas en cuanto a por qué se dieron algunos comportamientos. Sería interesante calcular los mismos indicadores aquí presentados a lo largo de varios cuatrimestres más y compararlos junto con los que ya se presentaron, a fin de buscar algún patrón más.

Los cuatrimestres analizados corresponden a años de pandemia por el COVID-19 donde la dinámica del trabajo en grupo y de las clases fué no presencial. Esto debe tenerse en cuenta al momento de analizar más cuatrimestres.

Otra posibilidad es aplicar la medición más allá de la materia Ingeniería de Software I. Podría ser con alumnos que ya conozcan Smalltalk y se les proponga resolver ejercicios sobre conceptos ya vistos utilizando TDD a lo largo de un período de tiempo. Mediante TDDGuru y el cálculo de los indicadores presentados en este trabajo se podrían tomar mediciones más limpias en cuanto a TDD para analizar la evolución del aprendizaje de la técnica.

Para el presente trabajo se realizó la extracción de datos de TDDGuru en CuisUniversity hacía un archivo .csv, para luego ser procesado por scripts en Python y finalmente confeccionar los gráficos en Google Sheet. Sería práctico poder automatizar todo ya sea desde Python o desde CuisUniversity para generar más rápidamente los datos necesarios para comenzar nuevos análisis.

En el análisis de tiempo de los estados de TDD se pudo ver que en ambos cuatrimestres analizados el orden es el mismo. En un análisis a futuro con más cuatrimestres se podría verificar si esto se mantiene ó si se detecta alguna modificación en el orden por tiempos, cotejar algún cambio introducido en dicho cuatrimestre.

8. Apéndices

Apéndice 1 Enunciados de los ejercicios

Mars Rover

Somos parte del equipo que desarrolla los equipos de exploración remota de Marte en la NASA y tenemos que desarrollar el sistema que controle el Mars Rover.

Para eso, se asume que la superficie de Marte es un plano y que se usan puntos para posicionar al Mars Rover en dicho plano, más un punto cardinal que indica hacia donde apunta.

Debido a que Marte está muy lejos, siempre se le envían al Mars Rover un conjunto de comandos empaquetados en un String, donde cada caracter es un comando.

Tener en cuenta que la comunicación puede tener problemas y pueden llegar comandos erróneos en cuyo caso se espera que no se sigan procesando los comandos restantes.

Tener en cuenta que:

- El Mars Rover siempre empieza en un punto inicial (x,y) y apuntando a un punto cardinal (N,S,E,O)
- El rover recibe una secuencia de caracteres que representan comandos sobre cómo moverse
- Los comandos pueden ser:
    - f = mover hacia adelante un punto (forward)
    - b = mover hacia atrás un punto (backwards)
    - l = rotar 90 grados a la izquierda
    - r = rotar 90 grados a la derecha

Resolverlo por medio de TDD.

Terni Lapilli (Solo 2C 2020)

Implementar el juego TerniLapilli (similar al TaTeTi) utilizando TDD. Sugerimos que los jugadores se llamen X y O.

Recordar que suponiendo que ninguno de los jugadores haya logrado formar una línea con sus tres fichas en el momento de colocar la tercera de sus fichas, el juego continua mediante el movimiento de las mismas. Cada jugador, respetando su turno, debe mover una de sus fichas en dirección a una casilla libre que se encuentre a "distancia uno".

El objeto que representa el juego podría responder los siguientes mensajes:

- putXAt: aPosition
- putOAt: aPosition
- isPlayingX
- isPlayingO
- isOver
- isTied
- hasPlayerXWon

- hasPlayerOWon

(Estos mensajes son sugeridos, no obligatorios.)

La solución no debe tener if salvo para los casos donde no tiene sentido evitarlo (por ejemplo, cuando se verifica que aPosition es válida para el tablero).
Recordar que la solución debe seguir las heurísticas de diseño que vimos, entre ellas la de no romper el encapsulamiento.
También, que empieza jugando X.
Como siempre, recomendamos primero hacer que la solución funcione implementando todos los tests (make it run),
y luego a partir de sacar if y código repetido (y sin romper encapsulamiento) deducir el patrón (make it right).

Portfolio 1

Un banco quiere empezar a ofrecer servicios financieros a sus clientes. Dentro de estos servicios está la posibilidad de poder ver agrupaciones de cuentas. Esas agrupaciones de cuentas se denominan Portfolios.

Un portfolio es una agrupación de cuentas o portfolios, y se puede hacer con ellos lo mismo que con una cuenta salvo registrar transacciones.
Por ejemplo si un portfolio es la agrupación de cuenta1 (con balance de $100) y cuenta2 (con balance de $200), se espera que el balance del portfolio sea $300.
También se espera poder preguntarle a un portfolio si alguna de sus cuentas registró una transacción por medio del mensaje #hasRegistered: y poder conocer todas las transacciones de las cuentas que forman parte del portfolio por medio del mensaje #transactions.
Por último, se espera que portfolios puedan formar parte de portfolios no únicamente cuentas, pero hay que asegurarse que estas no se repitan porque sino habría duplicación de información.
Respecto de la estructura del portfolio, tener en cuenta que:
1) Los portfolios se pueden modificar agregándoles cuentas o portfolios
2) Una cuenta o portfolio puede estar en más de un portfolio
3) Cómo es una estructura de árbol en donde no puede haber nodos repetidos, hay que estar seguros que cuando se modifique un portfolio (agregar una cuenta o portfolio) siga siendo un árbol sin nodos repetidos
4) Por ahora no se puede sacar cuentas o portfolios de un portfolio

Resolver este ejercicio por medio de TDD.

Portfolio 2

Portfolio Bancario 2: Transferencias y Reportes Ahora que el banco posee la funcionalidad de tener portfolios, quiere empezar a ofrecer más opciones a sus clientes. Para ello decidió agregar un nuevo tipo de transacción: la transferencia entre cuentas. Además, sumará la posibilidad de sacar reportes sobre las operaciones realizadas.

Una **transferencia** es una transacción que tiene ”dos patas”. La pata de la extracción, de donde se saca la plata, y la pata del depósito a donde se deposita la plata. Una transferencia se realiza entonces entre dos cuentas y por un valor. Algo importante a tener en cuenta es que se tiene que poder navegar desde la pata de la extracción a la transferencia y desde esta última a la pata del depósito y viceversa, aunque evitando tener redundancia de información. Es decir, el valor de la transacción debe estar en un solo lugar y no repetido en cada pata.
Respecto de los **reportes**, se deben poder ejecutar sobre una ReceptiveAccount o un Portfolio de manera indistinta. Se espera tener dos reportes inicialmente:

1. El resumen de cuentas (Account Summary).
2. El neto de transferencias (Transfer Net).

El **resumen de cuenta** debe generar una línea por cada transacción realizada en una cuenta con el siguiente formato:

```
Depósito por 100.
Extracción por 50.
Salida por transferencia de 20.
Entrada por transferencia de 30.
Balance = 60
```

Este sería el resumen de cuenta esperado de una cuenta a la cual se le realizó un depósito por 100, una extracción por 50, se le sacó 20 por una transferencia y recibió 30 por otra transferencia.

El **reporte de neto de transferencias** debe devolver el resultado de sumar todos los depósitos por transferencias y restarle todas las extracciones por transferencias. Para el ejemplo anterior, el neto de transferencias sería 10.

El banco prevé agregar muchos reportes nuevos en un tiempo inmediato, por lo tanto el modelo final para sacar reportes debe cumplir con los siguientes requerimientos de extensibilidad:

1. Al crear nuevos reportes no se tiene que volver a modificar la jerarquía de cuentas.
2. Al crear nuevos reportes no se tiene que volver a modificar la jerarquía de transacciones.
3. Crear nuevos reportes debe implicar crear clases nuevas únicamente y no modificar ninguna existente.

Resolver este ejercicio por medio de TDD.

Bonus track

El CEO del banco nos premiará si logramos tener 2 nuevos reportes de resumen de cuenta para portfolios. El primero deberá mostrar la estructura de árbol completa del portfolio. El segundo, un reporte de cuenta especial que muestre las transacciones indentadas de acuerdo a la profundidad de cada cuenta del portfolio.
Dado un portfolio como se muestra a continuación:

```
johnsAccount := ReceptiveAccount named: ’Cuenta de Juan’.
angiesAccount := ReceptiveAccount named: ’Cuenta de Angeles’.
childrenPort- folio := Portfolio named: ’Portfolio de hijos’
with: johnsAccount with: angiesAccount.
myAccount := ReceptiveAccount named: ’Cuenta mia’.
```

```
familyPortfolio := Portfolio named: 'Portfolio de la familia'
with: myAccount with: childrenPortfolio.
```
El reporte de la estructura de árbol de dicho portfolio debería ser:
```
Portfolio de la familia
  Cuenta Mia
  Portfolio de hijos
    Cuenta de Juan
    Cuenta de Angeles
```

Se espera que el resumen de cuenta especial se muestre de la siguiente manera:

```
Portfolio de la familia Cuenta Mia
  Depósito por xxx
  Extracción por yyy
  Balance = bbb
Portfolio de hijos
  Cuenta de Juan
    Depósito por zzz
    Extracción por nnn
    Balance = bbb
  Cuenta de Angeles
    Salida por extracción de qqq
    Balance = bbb
    Balance = bbb
    Balance = bbb
```

Al igual que los reportes anteriores, la diseño final para resolverlos debe permitir agregar reportes sobre portfolios sin tener que modificar nada.

Mars Rover El Regreso (Solo 1C 2021)

Para hacer un seguimiento del MarsRover, nos han pedido la siguiente funcionalidad
1) Tener un log donde se vayan agregando los cambios de la posición y los cambios de hacia dónde apunta el MarsRover. Se tiene que poder seguir sólo los cambios de posición o sólo los cambios de dirección o ambos simultáneamente.
2) Tener una ventana donde se muestre la posición actual y hacia donde apunta el MarsRover actualmente. De la misma forma que con el log, se debe poder ver sólo la posición actual o sólo los cambios de dirección o ambos simultáneamente.

La diferencia entre 1) y 2) es que 1) es un log donde van quedando todos los cambios y 2) sólo muestra la situación actual en unos text fields de una ventana.
Es necesario que la solución soporte que solo se esté usando el log, o solo la ventana o ambos al mismo tiempo.

Adicionalmente, la solución debe ser extensible. O sea, se debe poder agregar otras maneras de hacer el seguimiento del MarsRover como mandar mensajes a un micro servicio, grabarlo en una base de datos, etc. sin que esto implique una modificación en este.

También la solución debe soportar que se puedan seguir los cambios de algún otro colaborador que el MarsRover pueda tener. Por ejemplo, si se le agrega un colaborador para temperatura, se debe fácilmente y con la menor cantidad de cambios posibles al MarsRover, poder seguir las modificaciones de temperatura.

Construir esta funcionalidad en base a la solución de la cátedra del MarsRover.
ACLARACIÓN: La solución provista no tiene buenos nombres de tests y tienen también código repetido. No es necesario arreglarlo.

Tus Libros

La editorial "TusLibros" desea actualizar su oferta de ventas de libros permitiendo a sus clientes la posibilidad de comprar de manera on-line. El sistema que tienen actualmente funciona de manera batch, de tal manera que reciben archivos con los pedidos y devuelven archivos con los resultados de las compras. Su plan de actualización consta de dos etapas:

1) Desarrollar un sistema que reciba pedidos via una API Rest y dialogue de manera on-line con el Merchant Processor (validador de tarjetas)

2) Reemplazar el sistema actual haciendo que el nuevo sistema lea y genere los archivos que actualmente se usan como interfaz batch.

Como el Merchant Processor cobra por transacción, se debe validar la mayor cantidad de información posible antes de enviarla al mismo.

El Merchant processor ofrece un ambiente de desarrollo, pero también cobra por prueba, por lo que se debe evitar al máximo hacer pruebas con el mismo al menos que sea necesario (o sea, solo usar el ambiente de desarrollo del merchant processor luego de la integración).

El sistema debe guardar todas las compras realizadas para luego poder reponer stock, sacar estadísticas de ventas, etc.

El Merchant Processor provee una interface Rest para realizar los débitos a las tarjetas. La misma se ejecuta haciendo post a https://merchant.com/debit (para producción) y https://merchanttest.com/debit (para desarrollo) y recibe los siguientes parámetros:

1) creditCardNumber: Número de tarjeta de crédito

2) creditCardExpiration: Fecha de expiración con 2 dígitos para el mes y 4 para el año

3) creditCardOwner: Nombre del dueño de la tarjeta. Máximo de 30 caracteres

4) transactionAmout: Cantidad de la transacción, con un máximo de 15 dígitos para la parte entera y dos dígitos para la decimal (siempre se debe generar la parte decimal) utilizando el punto como separador.

Por ejemplo:
https://merchant.com/debit?creditCardNumber=5400000000000001&creditCardExpiration=072011&creditCar dOwner=PEPE%20SANCHEZ&transactionAmount=123.50

Estos datos de ejemplo son los utilizados como datos válidos por el ambiente de desarrollo del Merchant Processor.

Si algún parámetro no tiene el formato correcto devuelve como código de HTTP 400 (Bad request), en caso contrario devuelve como resultado HTTP 200 con el siguiente contenido:

1) En caso exitoso: 0|OK

2) En caso de no poder realizar el débito: 1|DESCRIPCION_DE_ERROR En caso de que la transacción no se pudo realizar se desea pasar la descripción del error al usuario de sistema.

Lamentablemente el up-time del Merchant Processor no es muy bueno, por lo que si el mismo se encuentra caído cuando se desea validar una transacción, se debe generar el pedido en un archivo de input cuyo ID de cliente del nombre será TUSLIBROS (ver más abajo especificación de este archivo)

La interfaz Rest que ofrecerá el sistema debe permitir crear un carrito (el cual será válido durante 30 minutos luego de la última vez que se realizó alguna operación con él), agregar un libro con su cantidad al carrito ya creado, consultar el contenido de un carrito, hacer el check out y listar las compras de un cliente. Las interfaces son:

1) Recurso: /createCart

Parámetros: clientId: ID del cliente que está creando el carrito

password: Password del cliente que valida que puede operar con TusLibros.com

Output: En caso de éxito: 0|ID_DEL_CARRITO

En caso de error: 1|DESCRIPCION_DE_ERROR

2) Recurso: /addToCart

Parámetros: cartId: Id del carrito creado con /createCart

bookIsbn: ISBN del libro que se desea agregar. Debe ser un ISBN de la editorial

bookQuantity: Cantidad de libros que se desean agregar. Debe ser >= 1.

Output: En caso de éxito: 0|OK

En caso de error: 1|DESCRIPCION_DE_ERROR

3) Recurso: /listCart

Parámetros: cartId: Id del carrito creado con /createCart

Output: En caso de éxito: 0|ISBN_1|QUANTITY_1|ISBN_2|QUANTITY_2|....|ISBN_N|QUANTITY_N

En caso de error: 1|DESCRIPCION_DE_ERROR

4) Recurso: /checkOutCart

Parámetros: cartId: Id del carrito creado con /createCart

ccn: Número de tarjeta de credito

cced: Fecha de expiración con 2 digitos para el mes y 4 para el año

cco: Nombre del dueño de la tarjeta.

Output: En caso de éxito: 0|TRANSACTION_ID

En caso de error: 1|DESCRIPCION_DE_ERROR

5) Recurso: /listPurchases

Parámetros: clientId: ID del cliente que quiere ver que compras hizo

password: Password del cliente que valida que puede operar con TusLibros.com

Output: En caso de éxito: 0|ISBN_1|QUANTITY_1|....|ISBN_N|QUANTITY_N|TOTAL_AMOUNT

En caso de error: 1|DESCRIPCION_DE_ERROR

Si el request realizado no cumple con las reglas sintácticas, se debe devolver como HTTP status el código 400 (Bad request). Si cumple con la sintaxis se debe

devolver como HTTP status el código 200 (OK). Los archivos utilizados como interface batch son:

De entrada: CLIENTE_INPUT_AAAA_MM_DD.csv

con el siguiente formato: TipoDeRegistro,RestoDelRegistro donde:

1) Para el tipo de registro 1: Agregar un libro al carrito, en formato ISBN,QUANTITY (Crear el carrito si el mismo no existe)

2) Para el tipo de registro 2: Realizar el checkout del carrito con el siguiente formato: numero_de_tarjeta,fecha_de_expiración,nombre_del_dueño. De Salida: CLIENTE_OUTPUT_AAAA_MM_DD.csv donde cada registro tiene el siguiente formato:

1) Resultado de Transacción: 0 para éxito, 1 en caso de error

2) En caso de éxito, id_de_transacción, total_de_transacción En caso de error: descripción_del_error Ejemplo: Archivo TEMATIKA_INPUT_2010_02_01.csv 1,0321146530,2 1,1933988274,1 2,5400000000000001,072011,PEPE SANCHEZ 1,1933988274,3 2,5400000000000002,132012,KENT BECK Archivo TEMATIKA_OUTPUT_2010_02_01.csv 1,10533,60.53 2,INVALID EXPIRATION DATE

Apéndice 2 Enunciados de los parciales

2C 2020

Adventure Games II
ISEngine

Continuando con el desarrollo de nuestro engine de aventuras gráficas, tenemos nuevos requerimientos provenientes del juego que otro equipo está desarrollando. Nuestra tarea es modelar todos los cambios propuestos a continuación, extendiendo el modelo presentado junto al enunciado. Como siempre, y para seguir manteniendo los estándares de calidad conseguidos, todo cambio a implementar debe realizarse mediante TDD y siguiendo las heurísticas de diseño vistas durante toda la cursada de la materia.

Room

En un juego de aventura gráfica el PlayableCharacter debe moverse en una Room (habitación). La room posee NxM celdas, con N y M enteros mayores o iguales a 1. En las celdas de un room pueden encontrarse StageObjects. Estos pueden ser PortableObject que pueden ser recogidos por el PlayableCharacter (quien los guarda en su backpack dejando de estar en el room), y NoPortableObjects que no se pueden recoger, pero sí su contenido como se explica más adelante. El PlayableCharacter se mueve como el MarsRover pero solo dentro de la room y en posiciones no ocupadas. Por suerte pudimos portar lo que ya teníamos hecho y funciona perfectamente, pero hay que adaptarlo para que funcione dentro la room. La ubicación y dirección inicial no son relevantes aunque deben ser válidas para el room en el que se crea. Es importante notar que una posición solo puede estar ocupada por un elemento (por ej. el PlayableCharacter no puede “pisar” un StageObject y un PortableObject no puede pisar un NoPortableObject) Importante: Se toma como coordenada (0, 0) la celda que está abajo a la izquierda.

Rooms y StageObjects

En cada posición nos podemos encontrar con StageObjects portables (por ej. una llave) y no portables (por ej. una mesa o una puerta) como indicamos previamente. Una posición sólo puede ser ocupada por un StageObject aunque hay que tener en cuenta que los objetos no portables pueden contener otros portables y no portables. Por ej. un armario puede contener un cajón (ambos NoPortableObject) y dentro de este último se encuentran una llave y un papel.

- Ejemplos válidos
    - en la posición (1, 3) hay una key (PortableObject)
    - en la posición (2, 3) hay un cajón (NoPortableObject) con una herramienta (PortableObject)
    - en la posición (3, 1) hay un armario (NoPortableObject) con una bufanda (PortableObject), una libreta (PortableObject) y dos cajones (NoPortableObject), cada cajón tiene una key (PortableObject)

- Ejemplos inválidos
    - En la posición (2, 3) hay una bufanda (PortableObject) y una key (PortaleObject)
    - En la posición (2, 3) hay una bufanda (PortableObject) y un armario (NoPortaleObject).

Este caso es inválido porque el armario no está conteniendo a la bufanda.

Toma de StageObjects

Como mencionamos anteriormente, el PlayableCharacter puede agarrar PortableObjects que se encuentren en el room. Para esto deben estar en una posición contigua de hacia dónde apunta el PlayableCharacter. Luego de agarrar el objeto (acción take), éste pasa a ser parte de los elementos en el backpack y deja de estar en el room. Por ejemplo, si el PlayableCharacter está en la posición (0, 0) apuntando al Norte y en la posición (0, 1) hay una bufanda, el PlayableCharater debe agarrar la bufanda si le envían el mensaje take. No puede agarrar una llave que se encuentra en la posición (1, 0), es decir a su derecha (al Este), ya que está apuntando al Norte. El PlayableCharacter también puede agarrar (acción take) los PortableObjects contenidos en un NoPortableObject. Por ejemplo si hay un armario (NoPortableObject) con una bufanda (PortableObject), una libreta (PortableObject) y dos cajones (NoPortableObject) y cada cajón tiene una key (PortableObject), cuando el PlayableCharacter tome el contenido del armario se agregará a su backpack una bufanda, una libreta y dos llaves, siempre y cuando el backpack tenga lugar. Siempre deben quedar en el backpack todos los elementos que se pudieron agregar. Tener en cuenta que los PortableObjects que se pudieron poner en el backpack deben sacarse del NoPortableObject que lo contenía, no así los NoPortableObjects, ellos deben quedar en el mismo lugar. Por ahora no nos vamos a preocupar si un PortableObject está en más de un NoPortableObject. Vamos a asumir que siempre se crean correctamente los NoPortableObjects.

1C 2021

ISW1 abrió una nueva rama de negocios con el objetivo de hacer juegos de mesa, ¡y qué mejor para empezar que el Truco!

El desarrollo está pensado por iteraciones porque el Truco es un juego complicado. En la primera iteración solo hay que implementar cómo se juega una ronda, pero sin el canto de envido y de truco. Por suerte ya hay desarrollado todo un modelo de cartas españolas personalizadas al Truco. Las cartas españolas tienen 4 palos: espada, oro, basto y copa. Los números de las cartas van del 1 al 12. Al estar especializadas en el truco, no se pueden crear cartas que no se usen en el juego como las de número 8, 9 y los comodines. La clase CartaDeTruco modela las cartas españolas para jugar al Truco. CartaDeTruco permite crear cartas fácilmente como:

- CartaDeTruco anchoDeEspadas. (Esta facilidad solo existe para ciertas cartas bien conocidas)
- CartaDeTruco espadaCon: 1.
- CartaDeTruco oroCon: 6, etc.

Las cartas de truco saben compararse por igualdad y responder #mataA: que devuelve verdadero si la carta receptora del mensaje le gana a la colaboradora al truco, y #empardaCon: que devuelve verdadero cuando la carta receptora del

mensaje es parda (empata) con la colaboradora. Para esta primera iteración, se debe desarrollar lo correspondiente a una Ronda de Truco, teniendo en cuenta lo siguiente:
1. La ronda es solo para dos jugadores, la mano (el que comienza el primer enfrentamiento) y el pie (el que juega segundo en el primer enfrentamiento).
2. La cantidad de cartas con las que empieza cada jugador son 3. Se juega con un solo mazo de cartas.
3. Cada ronda está compuesta de enfrentamientos sucesivos, que como mucho serán 3.
4. En cada enfrentamiento cada jugador juega una carta siguiendo este orden: Primero quién debe empezar el enfrentamiento y luego el otro jugador.
5. La manera de decidir quién debe empezar un enfrentamiento es:
    a. Si es el primer enfrentamiento, debe jugar primero la mano.
    b. Si no, debe jugar primero el que ganó el enfrentamiento anterior
    c. Si el enfrentamiento anterior fue pardo (empate), juega primero el que jugó primero el enfrentamiento anterior al pardo. Si este también fue pardo, juega la mano
6. El enfrentamiento es ganado por un jugador cuando su carta “mata” a la del otro jugador.
7. El enfrentamiento es emparedado cuando las cartas jugadas por ambos jugadores son pardas (el mismo valor).
8. Gana la ronda quién haya ganado dos enfrentamientos.
9. En caso de que el primer enfrentamiento haya sido pardo, quien gane el segundo enfrentamiento gana la ronda. (Solo se pide soportar primer enfrentamiento pardo)
10. No se puede seguir jugando una ronda si ya hay algún ganador En esta primera iteración no se pide soportar la funcionalidad del envido, ni de la flor ni la de cantar truco. Solo se debe asegurar que no haya errores al construir el juego, que el orden en el que juega cada jugador es correcto y saber quién ganó la ronda. Tampoco se pide soportar los casos en donde el segundo y tercer enfrentamiento son pardos. Solo se pide soportar el caso donde el primer enfrentamiento es pardo. Nuestra tarea es modelar lo pedido para la primera iteración mediante TDD y siguiendo las heurísticas de diseño vistas durante toda la cursada de la materia.

ACLARACIONES:

- No se pide hacer ningún tipo de inteligencia artificial que juegue.
- Solo es necesario desarrollar un modelo que permita controlar que se siguen las reglas del juego para la funcionalidad pedida.
- No es necesario desarrollar cómo se reparten las cartas. Los tests deben controlar las cartas repartidas y el flujo del juego.

Apéndice 3 Tópicos de TDD para la corrección de los parciales

2C 2020

- ☐ Tests de que height ser >= 1 y entero (puntaje 0.3)
- ☐ Tests de que width ser >= 1 y entero (puntaje 0.3)
- ☐ Tests de que la coord. x de posición está entre 0 y width-1 y es entera (puntaje 0.3)
- ☐ Tests de que la coord. y de posición está entre 0 y height-1 y es entera (puntaje 0.3)

- ☐ No se puede poner un elemento en una posición ocupada. Se testea directa o indirectamente / verificar que el objeto no sea puesto (puntaje 0.4)
- ☐ Si room conoce directamente al personaje por medio de un colaborador o no lo agrega a la room, se testea que no se puede poner un elemento en la posición del jugador. Se testea directa o indirectamente. verificar que el elemento no sea puesto. Si NO se pueden agregar StageObjects al Room luego de crearse y se verifica que el personaje se mueve correctamente entre los mismos, la nota es 1 también. (puntaje 0.4)
- ☐ Tests de que no puede empezar en una posición inválida (fuera de room y/o de coordenadas no enteras). Verifica que realmente no se puso en la room (puntaje 0.5)
- ☐ Tests de que no puede moverse para adelante fuera del tablero ni a posición ocupada. Verifica que no se movió en el tablero (puntaje 0.6)
- ☐ Tests de que no puede moverse para atrás fuera del tablero ni a posición ocupada. Verifica que no se movió en el tablero (puntaje 0.6)
- ☐ Verifica que cuando se mueve el jugador, se modifica la posición en la room (puntaje 0.6)
- ☐ Test de que no puede tomar de una posición vacía. Verifica que la backpack siga vacía o no tenga el elemento (puntaje 0.3)
- ☐ Test de que no se puede tomar fuera de la Room. Verifica que la backpack siga vacía o no tenga el elemento (puntaje 0.2)
- ☐ Test de que puede tomar PortableObject. Verifica que el mismo queda en el backpack y es sacado del room (puntaje 0.5)
- ☐ Test de que puede tomar PortableObjects de un NonPortableObject. Verifica que quedan en el backpack, que no están en el NonPortableObject y que el NonPortableObject quedó en la room (puntaje 0.6)
- ☐ Test de que puede tomar PortableObjects de NonPortableObjects de manera recursiva. Mismas verificaciones punto anterior. (puntaje 0.6)
- ☐ Test de que se toman objetos hasta que el backpack se llena. Verifica que estén los objetos correctos en el backpack y que los otros quedaron (puntaje 0.4)

1C 2021

- ☐ Tests de cartas de mano son 3 cartas y no repetidas (puntaje 0.5)
- ☐ Tests de cartas de pie son 3 cartas y no repetidas (puntaje 0.5)
- ☐ Test de cartas no repetidas entre mano y pie (puntaje 0.5)
- ☐ Test de pie no puede jugar cuando juega mano(puntaje 0.75)
- ☐ Test de mano no puede jugar cuando juega pie (puntaje 0.75)
- ☐ Test de mano gana cuando gana dos enfrentamientos (puntaje 1)
- ☐ Test de pie gana cuando gana dos enfrentamientos (puntaje 1)
- ☐ Tests de mano/pie ganan cuando primer enfrentamiento pardo (puntaje 1)
- ☐ Test de mano/pie no pueden jugar cartas que no tienen o que jugaron (puntaje 0.5)
- ☐ Test de no se puede jugar cuando hay un ganador (puntaje 0.5)

Apéndice 4 Tablas de porcentajes de aplicación de TDD por grupo en cada ejercicio

Grupo	MR	TEL	PO1	PO2	TL1	TL2	TL3
1	75,81	72,10	5,05	25,23	67,14	10,44	70,44
2	68,27	76,67	69,60	13,43	43,75	-1	61,79
3	55,75	36,21	80,59	24,62	-1	-1	0,45
4	90,09	76,10	44,59	-1	77,19	37,96	80,43
5	77,58	67,95	95,38	10,77	93,20	40,91	20,49
6	30,79	0,00	4,84	0,92	19,75	0,00	8,61
7	11,29	3,16	5,87	62,65	0,00	38,18	11,21
8	64,71	68,37	92,21	24,20	24,82	71,34	95,24
9	83,91	6,29	53,45	39,95	1,10	-1	70,78
10	93,92	18,30	2,12	19,52	26,60	0,29	8,96
11	92,47	80,77	93,50	85,56	35,26	-1	43,89
12	-1	21,62	76,25	0,95	-1	18,03	2,44
13	96,04	23,00	79,74	21,19	79,41	73,60	40,89
14	81,09	74,96	79,09	73,54	0,00	10,21	67,21
15	39,86	89,20	88,96	62,03	58,14	-1	-1
16	0,00	1,24	16,07	30,92	55,93	-1	-1
17	56,59	5,02	67,15	28,60	84,55	7,04	11,15
18	86,49	31,06	46,81	15,56	72,94	4,44	0,00
19	51,74	68,31	59,79	11,36	55,38	28,57	66,87
20	21,30	89,23	54,46	1,71	18,91	-1	22,41
21	100,00	42,38	36,02	-1	0,00	100,00	82,55
22	77,97	80,82	10,14	10,04	98,25	58,32	18,49
23	-1	-1	-1	-1	-1	-1	-1
24	87,87	39,94	37,70	65,54	-1	-1	57,63

2C 2020

Grupo	MR	PO1	PO2	MRR	TL1	TL2	TL3
1	-1	-1	-1	-1	-1	-1	-1
2	79,32	80	21,75	74,37	93,02	29,61	81,8
3	54,55	52,66	77,57	4,4	52,87	18,07	67,48
4	90,82	100	0	97,23	19,06	40,07	95,31
5	97,76	93,73	65,54	80,34	69,59	44,48	57,25
6	98,05	97,98	3,79	96,98	82,12	88,71	11,25
7	0	-1	21,56	50,81	85,14	5,91	15,34
8	89,00	73,47	79,82	64,21	92,22	27,47	28,38
9	88,45	68,38	14,02	0	88,89	0	28,23
10	85,97	95,02	-1	-1	1,16	44,15	40,93
11	14,05	35,31	17,95	-1	52,73	39,04	36,63
12	-1	-1	-1	88,74	-1	-1	-1
13	-1	5,1	0	31,67	70,43	31,45	34,94
14	94,07	87,16	63,93	90,43	89,58	-1	52,5
15	86,93	-1	30,66	19,7	57,66	11,17	16,33
16	94,51	86,42	16,72	-1	96,3	65,65	47,41
17	81,29	76,67	0,00	59,49	70,52	13,02	39,02
18	94,71	70,95	78,91	23,78	41,79	70,67	32,77
19	-1	86,89	-1	-1	-1	-1	-1
20	100,00	78,35	59,31	35,6	7,14	60,42	86,77
21	95,96	0	24,03	0	-1	67,81	0,60
22	-1	-1	-1	-1	0	55,56	-1
23	-1	-1	45,95	51,75	62,16	4,48	16,67
24	87,72	82,9	-1	20,64	10,45	16,5	68,09
25	71,62	100,00	17,57	88,27	100,00	53,95	7,48

1C 2021

Apéndice 5 Repositorio de scripts y datos

https://github.com/OctavioBit/EvolucionAprendizajeTDD

9. Referencias

[1] Beck, K. Extreme Programming Explained: Embrace Change. Addison-Wesley, 1999
[2] Extreme Programming Installed, Mike Hendrickson, Ann Anderson, Chet Hendrickson Addison-Wesley Professional, 2001
[3] Planning Extreme Programming Kent Beck, Mike Hendrickson, Martin Fowler, Addison-Wesley Professional, 2001
[4] Beck, K. Test-Driven Development by Example, Addison Wesley - Vaseem, 2003
[5] Alan Perlis,Software Engineering Conference, Nato, 1968
[6] Richard W. Hamming, History of Computers - Software Conference, 4 de Abril 1995
[7] Matias Dinota TDDGuru https://github.com/mdinota/TDDGuru
[8] CuisUniversity https://sites.google.com/view/cuis-university/inicio
[9] Maria Siniaalto, Pekka Abrahamsson. A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage. VTT Technical Research Centre of Finland. 2007
[10] Hakan Erdogmus, The Role of Process Measurement in Test-Driven Development, Conference Paper in Lecture Notes in Computer Science - Agosto 2004
[11] "Hackystat," http://csdl.ics.hawaii.edu/Tools/Hackystat/
[12] A Tool to Measure TDD Compliance: A Case Study with Professionals, Altieres de Matos, Reginaldo Ré, Marco Aurélio Graciotto Silva Graduate Program in Informatics (PPGI), Federal University of Technology – Paraná (UTFPR),Cornélio Procopio, Parana, Brazil
[13] S. H. Edwards, "Using test-driven development in the classroom: Providing students with concrete feedback on performance," presented at Proceedings of the International Conference on Education and Information Systems: Technologies and Applications (EISTA'03), Agosto 2003.
[14] https://web-cat.org/
[15] Pandas https://pandas.pydata.org/
[16] https://en.wikipedia.org/wiki/Box_plot
[17] https://en.wikipedia.org/wiki/Pearson_correlation_coefficient