**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

908/05

**Director**

Lic. Hernán Wilkinson

hernan.wilkinson@gmail.com

Buenos Aires, 2021

**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,* ninguno se focaliza en la evolución de su aprendizaje. Debido a esto, no hay material en qué 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 extrae una herramienta de validación de pasos de TDD. Para alimentar dicha herramienta se recolectaron datos de alumnos de dos cuatrimestres de *Ingenieria 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 de la aplicación de TDD, se calcularon indicadores como: grado de aplicación de la técnica y tiempo en cada estado de la misma entre otros. Luego, se compararon los resultados obtenidos entre ejercicios y parciales y, posteriormente, entre ambos cuatrimestres analizados.

Finalmente, toda la información será presentada en tablas y gráficos junto a las conclusiones arribadas.

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

**Índice**

[**Introducción**](#_f5lfcgqsk10x) **4**

[**Estado del arte**](#_p1ogrsuptm5u) **5**

[**Detalle del proceso realizado**](#_1zepzjbyxd8i) **6**

[TDDGuru](#_7zoa8y2081il) 6

[Datos recolectados](#_jcjrd3ha96y9) 6

[Consideraciones sobre los datos](#_g3vz59e7f1op) 7

[Organización de los archivos .changes](#_sm4ucaehod5r) 8

[Validación de los archivos](#_fv0nckvgsx30) 8

[Información en TDDGuru a extraer](#_7r6axe3jc2p8) 9

[Proceso de exportación](#_a6eqsdscmw0s) 9

[Procesamiento de datos exportados](#_15es0g7xijs4) 11

[**Indicadores calculados**](#_uaqjcoa2k9j) **13**

[Indicadores calculados a partir de los datos de TDDGuru](#_rwbi5x34nm37) 13

[**Conclusiones**](#_vn6ve9z0rvg7) **26**

[**Trabajo a futuro**](#_ukljfalievvc) **27**

[**Apendices**](#_qtfuec2ndoao) **28**

[Apendice 1 Enunciados de los ejercicios](#_k7qi8rcq4xm3) 28

[Apendice 2 Enunciados de los parciales](#_hvhhhf8zhjh0) 35

[Apendice 3 Repositorio de scripts y datos](#_9zpethkcpjs9) 40

[**Referencias**](#_eg8vu21rywl5) **40**

# Introducción

*Test Driven Development* **[1]** (*TDD*), es una técnica que surgió durante la década del 2000. Principalmente impulsada por Kent Beck en el contexto de la metodología *Extreme Programing* (XP) 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 para que el test pase.
3. Mejorar el diseño del paso anterior preservando que el test pase.

Realizando ciclos de estos pasos el software se construye de manera iterativa e incremental. Las ventajas de desarrollar el software mediante está técnica son las siguientes:

* 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.

Por estos motivos, es importante su enseñanza académica y que la metodología de la misma sea efectiva para la adquisición de la técnica.

Ahora bien, la falta de estudios sobre su aprendizaje no permite partir de alguna base para planificar estrategias de enseñanza de la misma, o sobre cómo se podría evaluar su adquisición.

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 mejorar la enseñanza de la misma.

A fin de obtener dicha evidencia se utilizará la herramienta *TDDGuru* **[2]**, desarrollada en *CuisUniversity* **[3]**, 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 extraerán datos para realizar diversos análisis y buscar conclusiones acerca de la evolución del aprendizaje de TDD.

Se recolectaron datos de dos cuatrimestres 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.*

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 ( el componente *Pandas* de Python, hojas de cálculo, etc ).

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

# Estado del arte

Maria Siniaalto y Pekka Abrahamsson **[REFSINALTO]** recopilaron diversos trabajos en donde se enseñó y evalúó la aplicación de la técnica de TDD tanto en el ámbito académico como también en la industria, dando resultados variados. Por un lado, 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, se decrementó significativamente el porcentaje de los defectos y los equipos que usaban TDD corregian errores más rápidamente. Por otro lado, el tiempo de desarrollo se incrementó entre un 15% y un 35%. Sin embargo, los estudios apuntaban al efecto de TDD y no a la evolución de su aprendizaje.

Hakan Erdogmus **[REFHAKAN]** realizó el experimento de tomar mediciones del código de un desarrollador que comenzaba a aplicar TDD mediante la herramienta HackyStat. 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.

Otro estudio realizado 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 WEB-CAT **[REFEDWARDS]**. La misma pone énfasis en analizar los tests como parte del desarrollo como nuevo enfoque además de dar un feedback más enriquecido en cuanto a cómo se realizaron los tests. Los resultados obtenidos fueron que se mejoró la nota de los alumnos, la técnica fue aceptada ampliamente, los estudiantes se sentían más seguros del código entregado, como también al realizar cambios al mismo y, finalmente, los alumnos prefieren usar TDD aunque la consigna no lo pida. Lo que muestra la importancia del feedback sobre el desarrollo realizado.

Más allá de los trabajos mencionados, no se encontraron trabajos que traten sobre la evolución del aprendizaje de TDD a lo largo de varios ejercicios/desarrollos en un periodo de tiempo.

# Detalle del proceso realizado

## TDDGuru

La herramienta TDDGuru, a partir del log de cambios de CuisUniversity, marca los pasos de TDD que se realizaron correctamente, 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.*

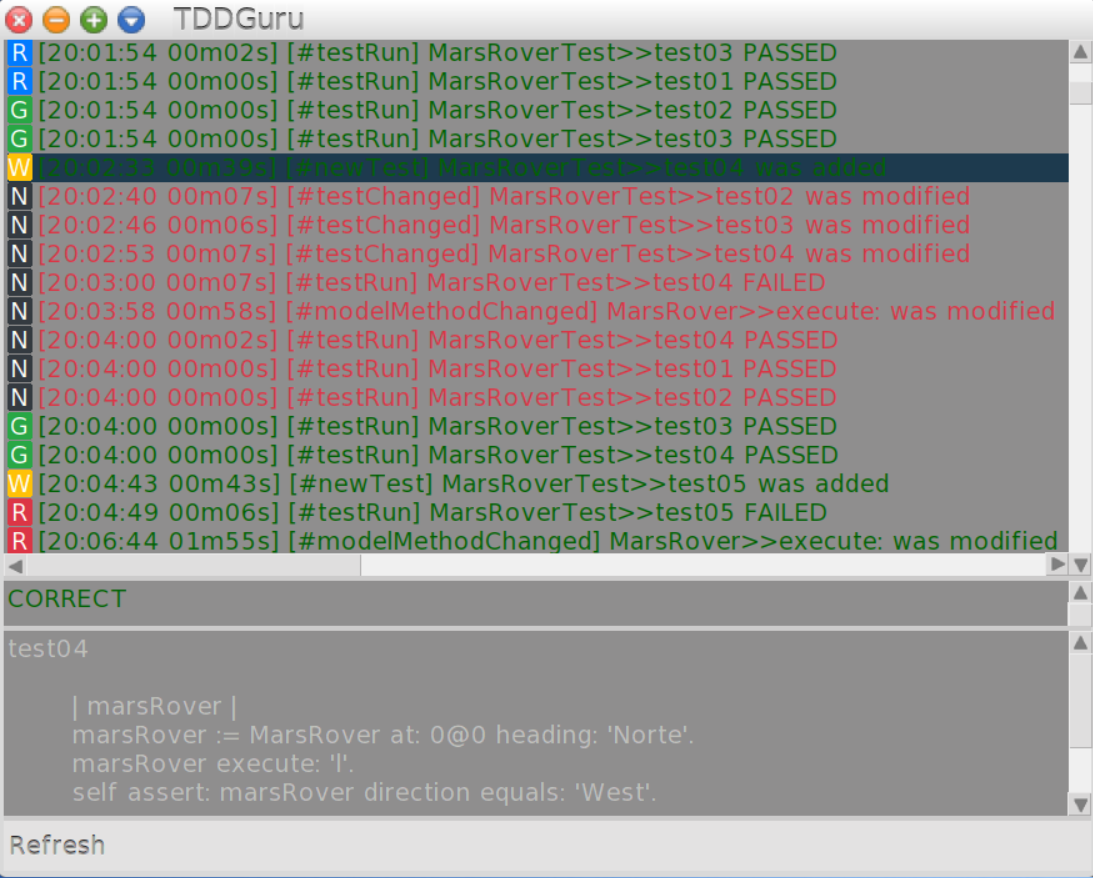


Figura 1. Interfaz de TDDGuru. Se cargó un archivo .changes

TDDGuru opera sobre los archivos .changes de CuisUniversity, por lo que, obteniendo este archivo del ambiente de trabajo del desarrollador es posible importarlo en esta herramienta y obtener la información de aplicación de TDD.

## Datos recolectados

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 (2C 2020) y el primero del 2021 (1C 2021).

Para el caso de los ejercicios en el 2C 2020 son los siguientes:

* *Mars Rover (MR) Objetivo: Reemplazar if por polimorfismo*
* *Terni Lapilli (TEL) Objetivo: Aplicar el patrón de diseño State.*
* *Portfolio 1 (PO1) Objetivo: Descubrir el patrón de diseño composite.*
* *Portfolio 2 (PO2) 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) Objetivo: Desarrollar una aplicación compleja por medio de TDD.*

Los enunciados se pueden consultar en el apéndice **[1]***.* Para los dos primeros, no hay una correlatividad y, para Portfolio 2 se debe tomar como base lo desarrollado en Portfolio 1 o utilizar una solución que provee la cátedra. Lo mismo ocurre con Tus Libros 2 y 3 con respecto a Tus Libros 1. Para el 1C 2021, los ejercicios son similares, con la salvedad de que se reemplaza *Terni Lapilli* por *Mars Rover El Regreso (MRR) Objetivo: Aplicar el patrón de diseño observer.* Estos ejercicios se realizan en grupos de dos personas.

Para los parciales, se realizan de manera individual y 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

## Consideraciones sobre los datos

* No es posible validar que la solución entregada por los grupos haya sido una creación 100% del mismo. 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. Para el caso del parcial se tiene más confianza de que no ocurrió lo anterior.
* 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 parecido al primero, por otra parte 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 a resolver por sobre la aplicación de TDD.

## Organización de los archivos .changes

Dado que los alumnos organizan los archivos de su repositorio a su criterio, fue necesario establecer una nomenclatura para los mismos. 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.

## 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.

Si bien *TDDGuru* permite seleccionar a partir de qué cambio realizar el análisis o hacerlo desde el archivo entero, 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.

## 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.

--------------------------------------------HASTA ACA PARA REVISAR----------------------------------------------

## 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* **[4]**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.

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

La simulación de carga del archivo completo (paso II) se desarrolló como extensión de TDDGuru, se llaman a los mensajes que se enviarian al seleccionar 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.

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.

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.

Este mismo algoritmo se aplica 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 y Cantidad de tests.

## Procesamiento de datos exportados

El proceso anteriormente mencionado genera un archivo .csv para los ejercicios y otro para los parciales, a su vez, esto se aplica a ambos cuatrimestres analizados.

Analizando el tiempo en los estados se detectaron valores que intuitivamente eran muy superiores al tiempo 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* **[5]**para visualizar los tiempos. Dicho método permite visualizar dónde se encuentran la mediana, los outliers, 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.

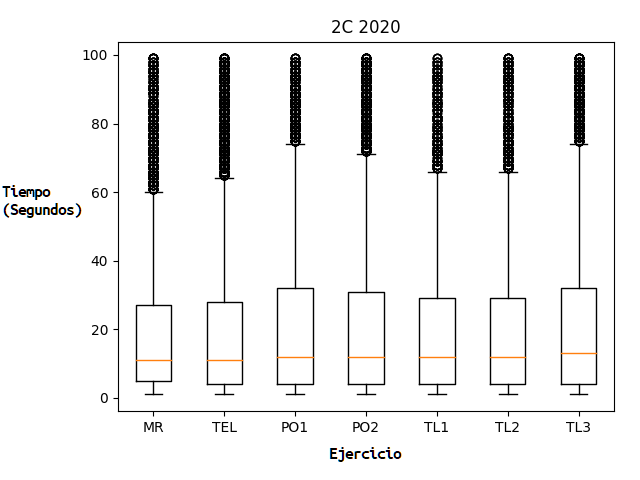


Figura 3. Boxplots de tiempo por ejercicio para el 2C 2020

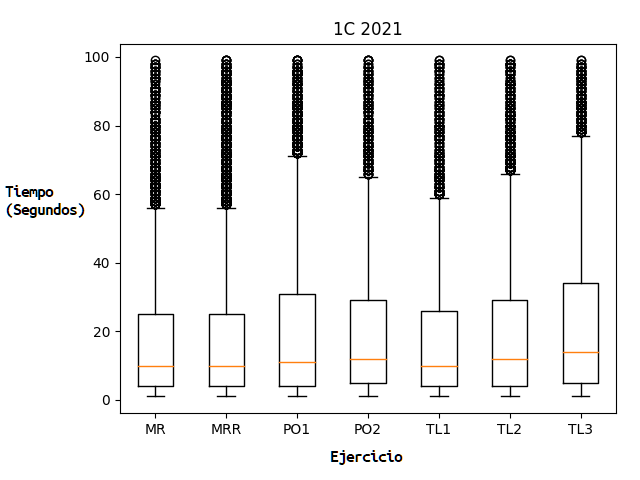


Figura 4. Boxplots de tiempo por ejercicio para el 1C 2021

Como se puede ver en ambas figuras, los outliers comienzan a aparecer alrededor de los 60 segundos. Por este motivo se decidió filtrar los datos que excedan el tiempo anteriormente nombrado.

# Indicadores calculados

## Indicadores calculados a partir de los datos de TDDGuru

El paso siguiente es calcular una serie de indicadores con la información obtenida de TDDGuru para compararlos y sacar conclusiones. Para esto se utilizará la librería *Pandas* de Python, la cual provee métodos versátiles para el manejo de archivos csv.

A continuación se detallan los indicadores para los ejercicios:

*-Tiempo por ejercicio*

Tiempo de todos los grupos por ejercicio. A continuación en la *FIGURAS XY y XY*

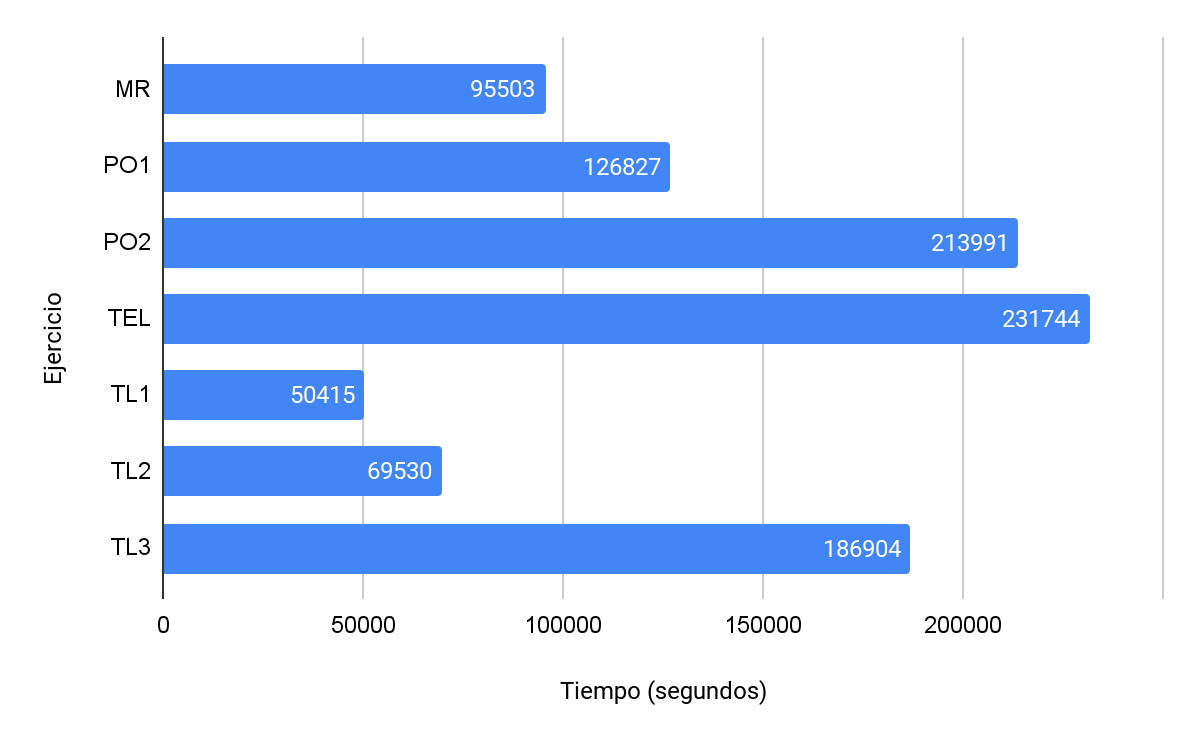


Figura XY. Tiempo por ejercicio para el 2C 2020

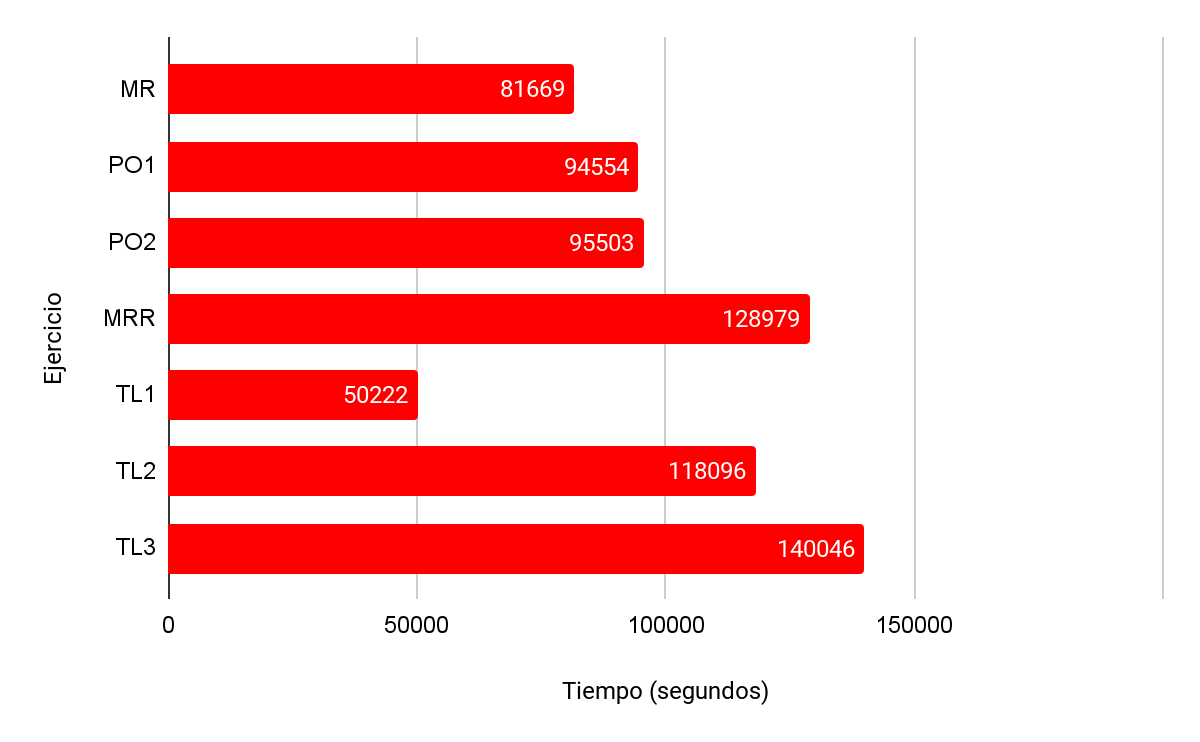


Figura XY. Tiempo por ejercicio para el 1C 2021

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 modelo 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. Una posible explicación es que, como se comentó en las consideraciones de los datos, los grupo pudieron haberse pasado ideas de cómo encarar los ejercicios, reduciendo así los tiempos globales.

*-Tiempo en los estados de TDD*

Se realizó la agrupación por estado de TDD la suma de todos los tiempos de los grupos. A continuación en las *Figuras XY y XY*

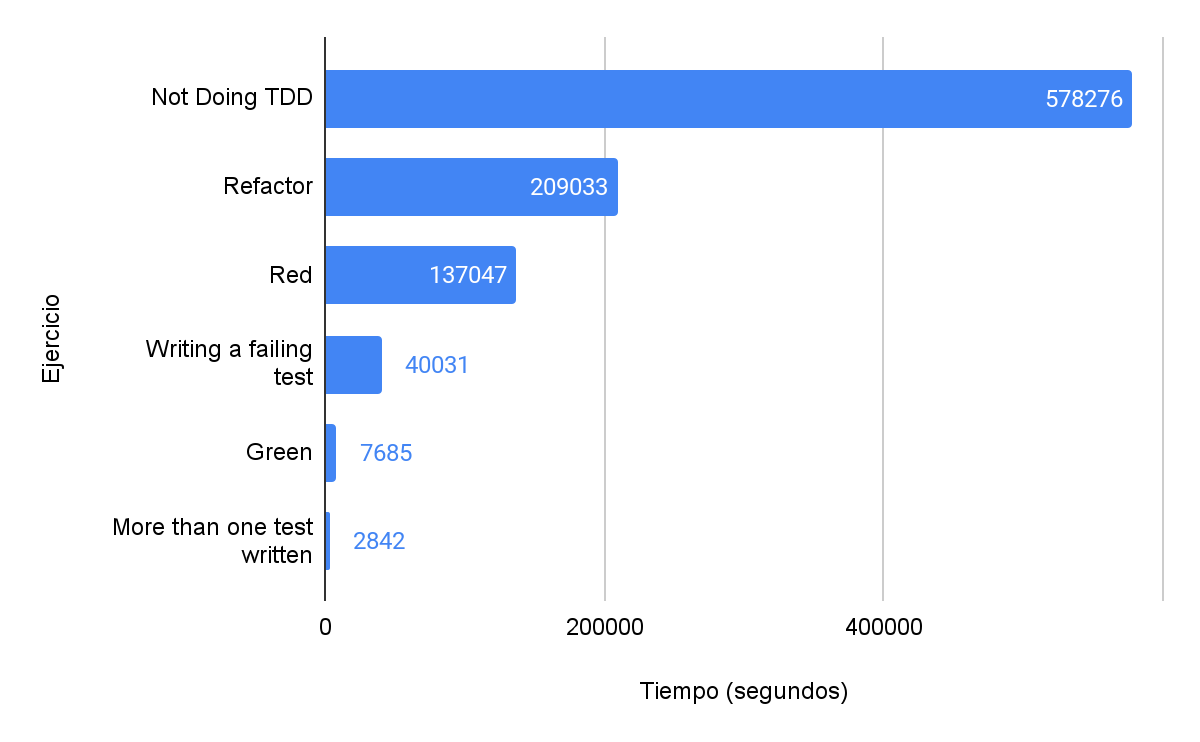


Figura XY. Tiempo en los estados de TDD para el 2C 2020

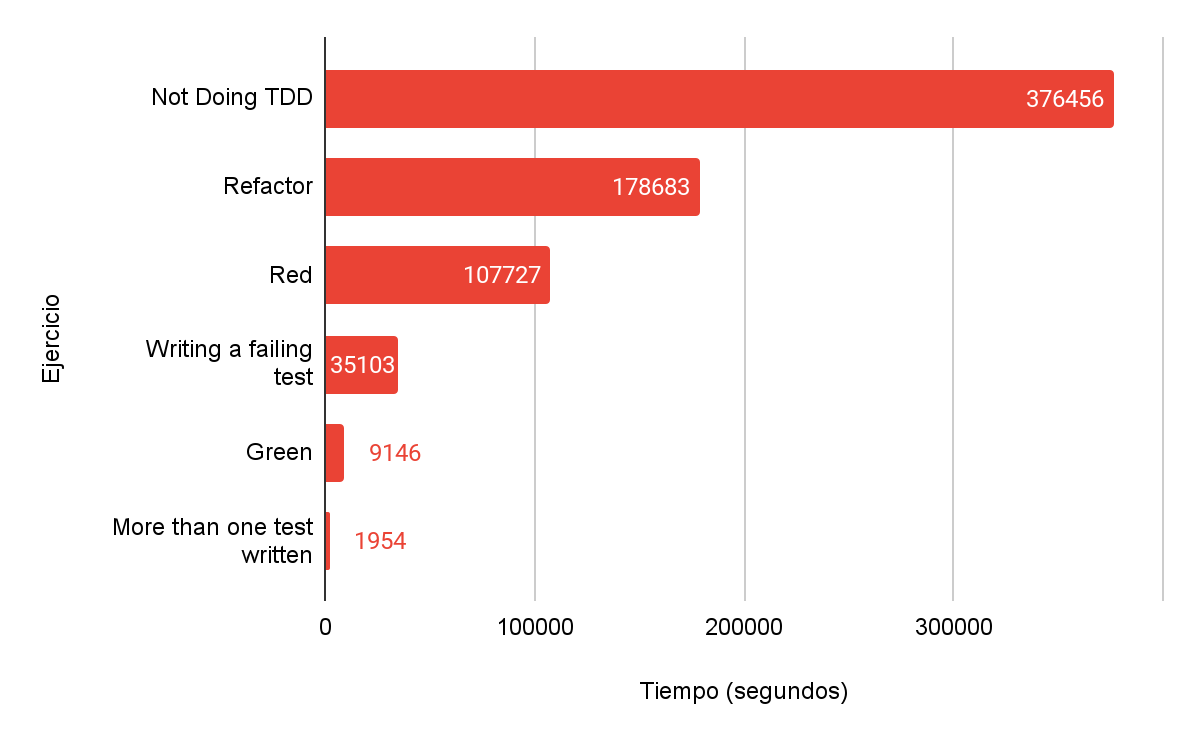


Figura XY. Tiempo en los estados de TDD para el 1C 2021

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 TDD el que presenta mayor tiempo es *Refactor,* lo que muestra que hay un interés en los alumnos en la mejora del modelo que se construye.

El estado que sigue en tiempo al anterior nombrado es *Red,* lo que denota que presenta una mayor dificultad el refactorizado del código a 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.

Para los parciales los indicadores son:

*-Estados de TDD correctos por estudiante*

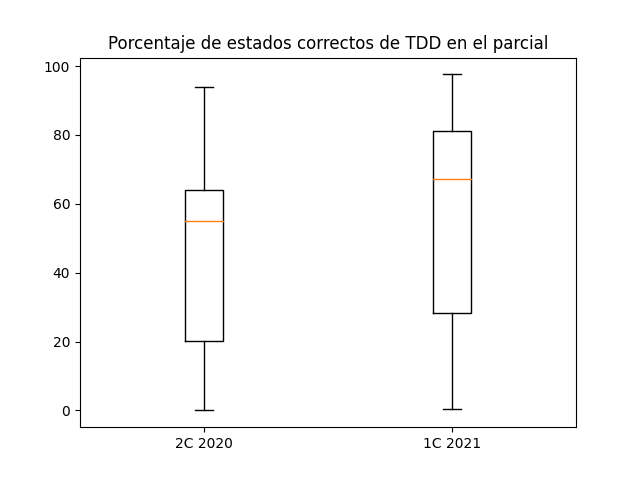
Se elaboró un *boxplot* para cada cuatrimestre analizado para mostrar los porcentajes de estados correctos por estudiante. A continuación en la *Figura XY*

Figura XY. 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 en la práctica de la técnica. Además, la concentración de datos aumentó de 60% de estados correctos a aproximadamente el 70% y 80% en el 1C 2021. Hubo una mejora 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 *Figuras XY.*

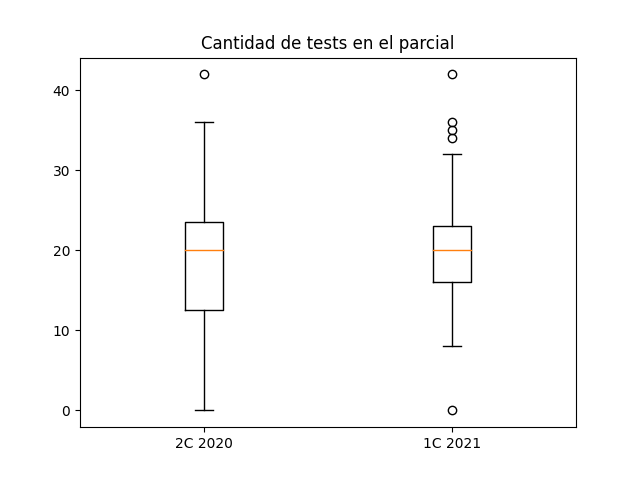


Figura XY. Cantidad de tests realizados por alumno para el 2C 2020

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 mayor 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 XY y XY*

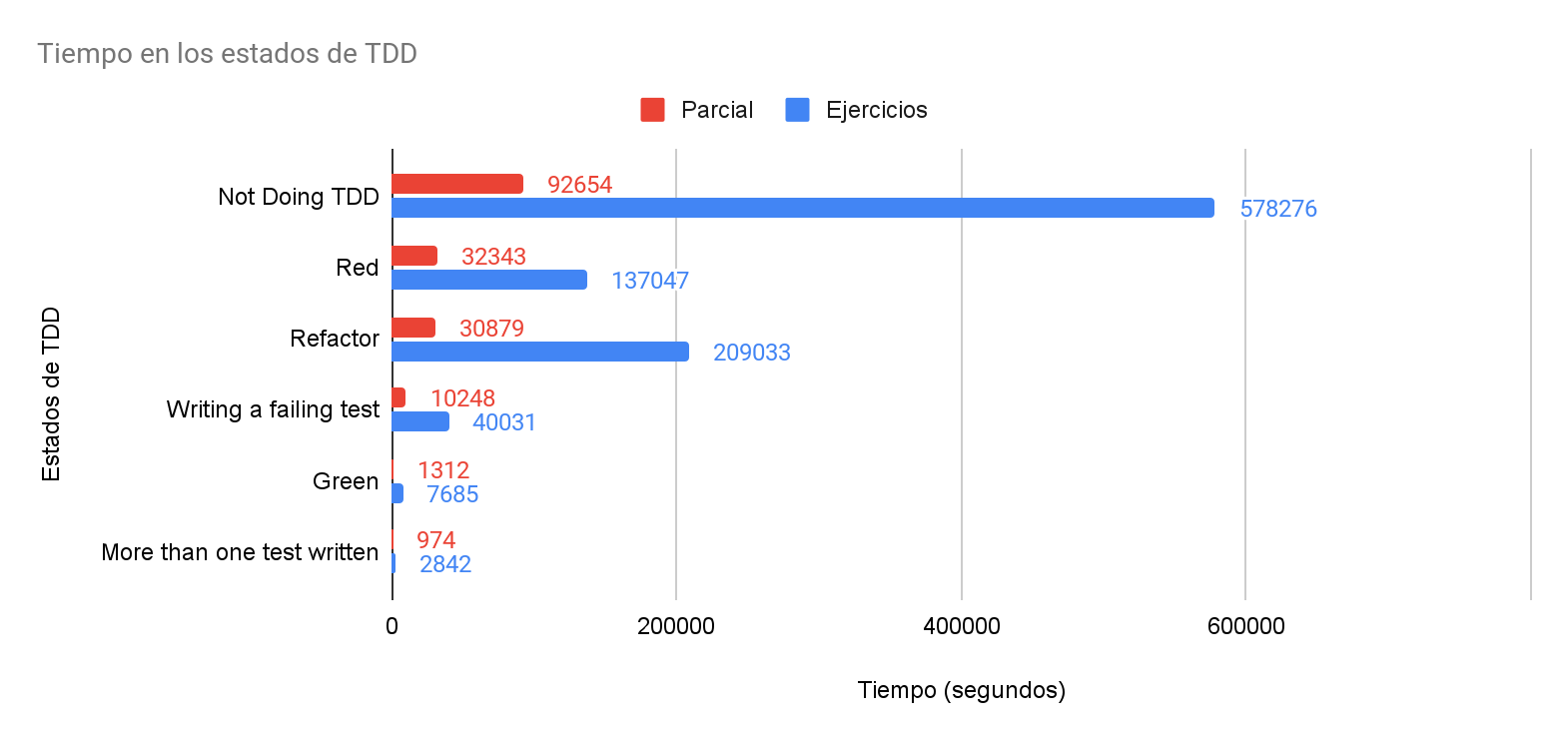


Figura XY. Tiempo en los estados de TDD para todos los alumnos/grupos para el

2C 2020

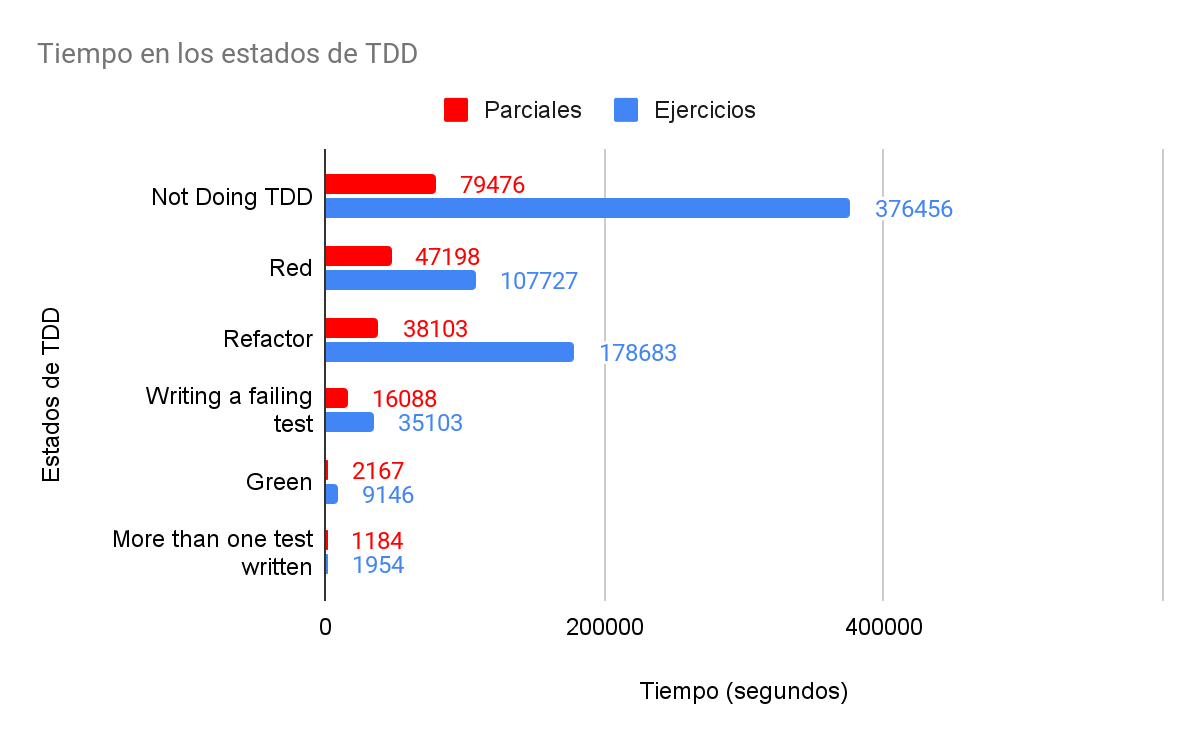
**

Figura XY. 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 nombrada 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 acotado a diferencia de 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.

**Indicadores a 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.

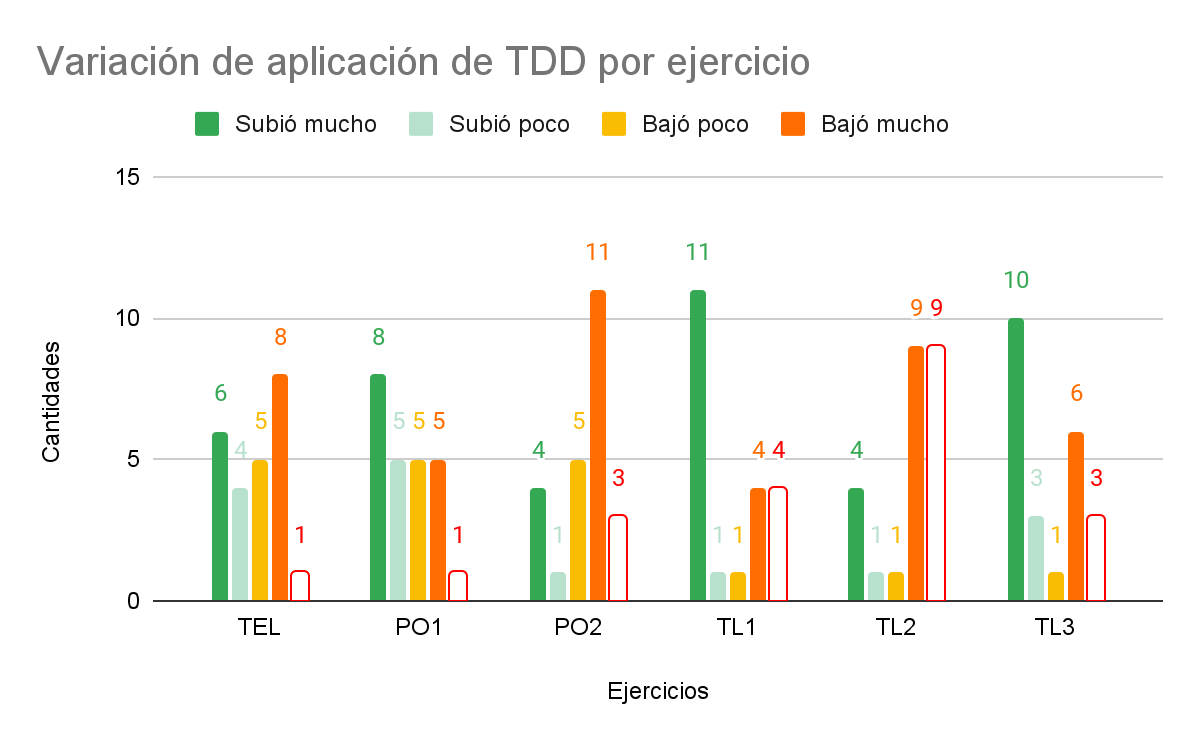
Analizando los archivos .changes se detectó que los que corresponden a valores cercanos a 0 en el porcentaje de estados correctos no aplicaban la técnica o comenzaban aplicándola pero a los pocos estados dejaban de hacerlo. Para los valores cercanos a 18% se notó en los archivos .changes que se aplicaba 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 ***Apendice XY***

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:

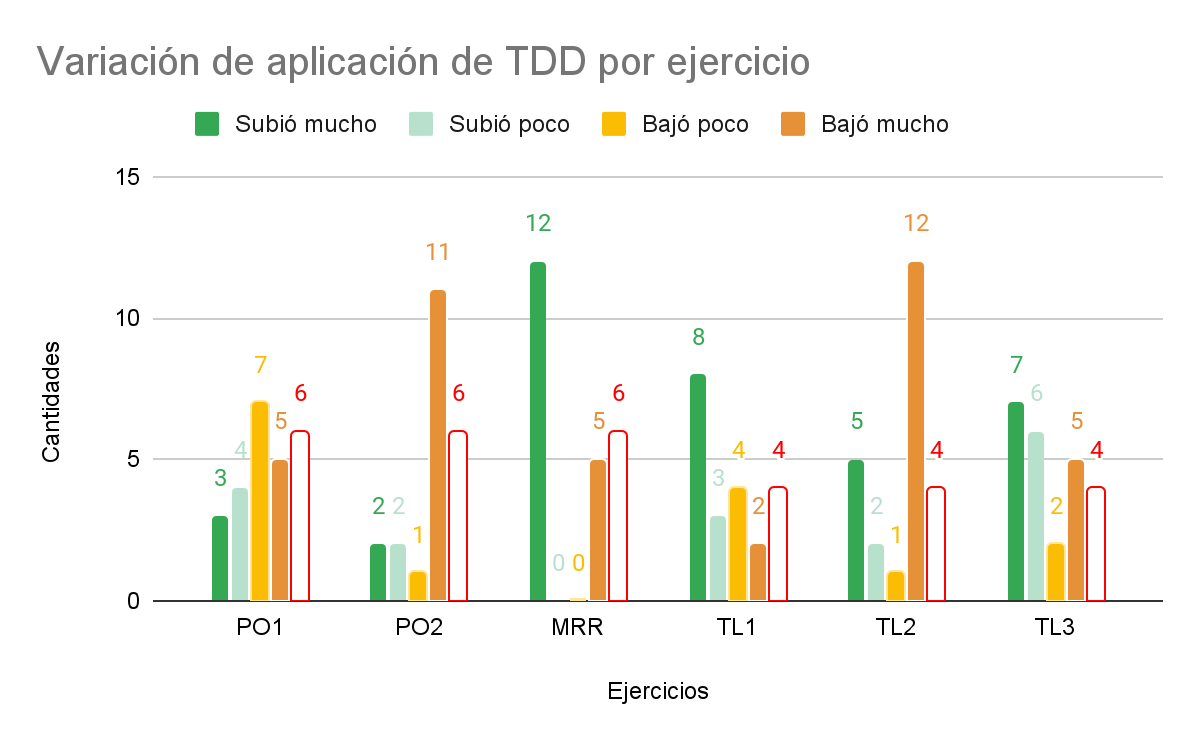


En el caso de que en el ejercicio anterior del grupo no haya habido datos, se decidió no contabilizar la diferencia. De esta forma se cuantificó la cantidad de grupos que subieron o bajaron su aplicación de TDD.

Con esta información se elaboraron los siguientes gráficos para cada cuatrimestre en las **Figuras XY y XY**. 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 XY**. Variación de aplicación de TDD por ejercicio para el 2C 2020.



**Figura XY**.Variación de aplicación de TDD por ejercicio para el 1C 2021.

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 de la **Figura XY (el de los tiempos por ejercicio)** donde en las segundas partes de los ejercicios se incrementa el tiempo y se presume que 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.

*- 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 XY y XY** se visualiza para cada cuatrimestre dicho promedio. Adicionalmente se separaron los datos en *Con Aplicación de TDD* (se descartan porcentajes de estados correctos muy bajos por grupo) y *Todos* (se tienen en cuenta todos los datos).

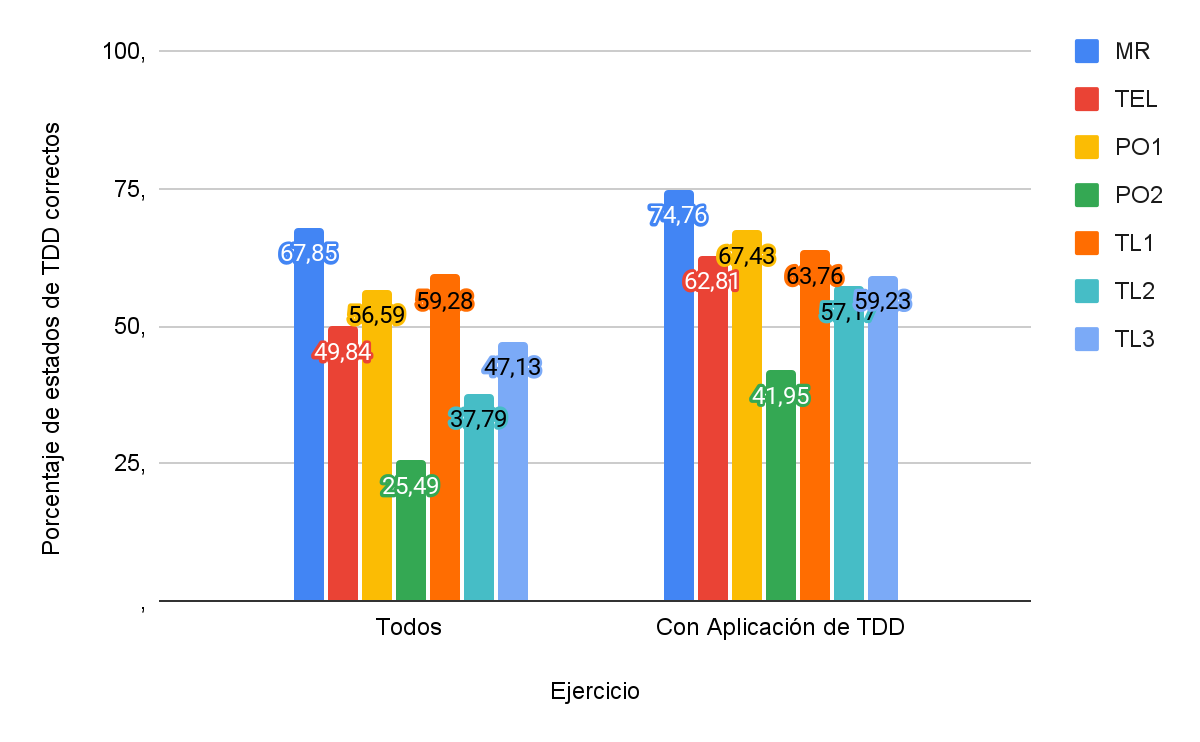


Figura XY. Aplicación de TDD por ejercicio para el 2C 2020

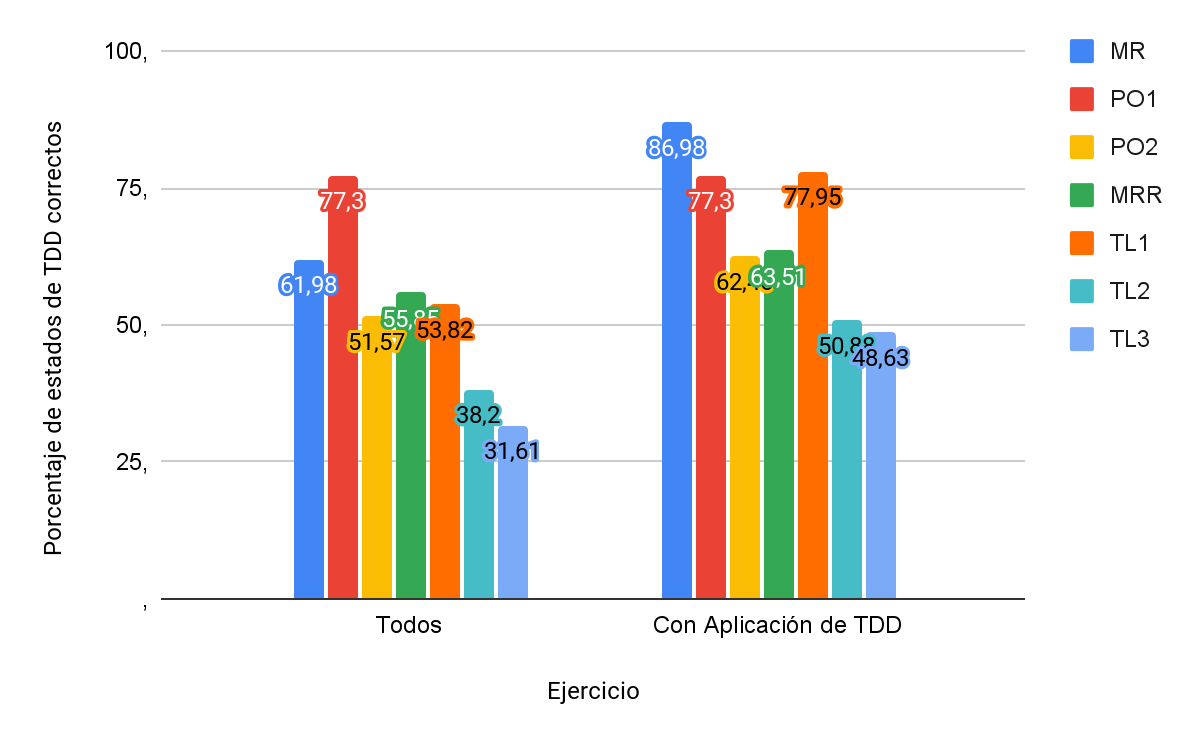


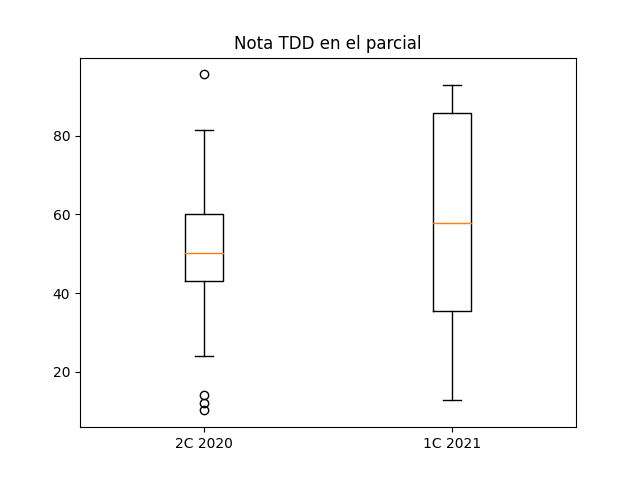
Figura XY. Aplicación de TDD por ejercicio para el 1C 2021.

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 en el paso de TL2 a TL3, una posible explicación es la ya mencionada variabilidad de dinámica de los alumnos en cada cuatrimestre, donde pudo haber alguna colaboración entre los grupos.

*- Nota de TDD en parcial*

Porcentaje de la nota del parcial que corresponde a TDD. Para este indicador se utilizó una planilla de corrección de parciales, la cual consiste en una serie de tópicos a verificar si fueron cumplidos por el alumno en el parcial. Por cada alumno el corrector del examen asigna el porcentaje de cumplimiento de cada tópico 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 los tópicos que corresponden a TDD y por cada alumno se verificó que puntaje de TDD se obtuvo. En el **Apendice XY** se listan los tópicos para los parciales de los cuatrimestres analizados.



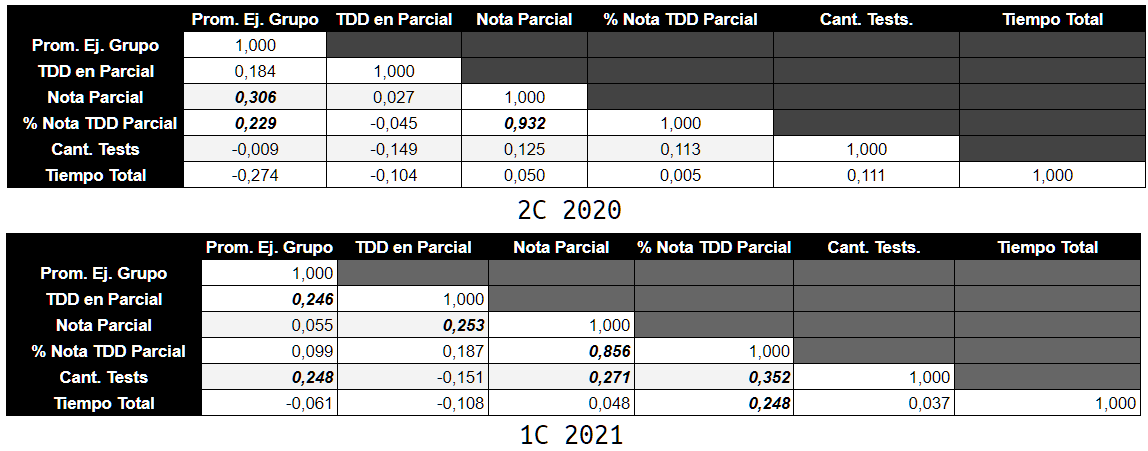
**Figura XY**. Porcentaje de notas de TDD para ambos cuatrimestres.

En la **Figura XY** 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 **[6].** 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 XY** se presentan las matrices de correlación para cada cuatrimestre analizado

**Figura XY**. 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* 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* lo que muestra que la técnica mejora las calificaciones de los alumnos en concordancia con el trabajo de Edwards **[7]**. Del mismo modo, el indicador del *Porcentaje de nota de TDD en el parcial* y el promedio antes mencionado tienen algo de énfasis, 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* 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*.

La correlación entre la *Nota del parcial* y la *Aplicación de TDD en el parcial* es consistente nuevamente con el estudio de Edwards **[7]** 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* lo cual es lógico ya que si se aplicó la técnica correctamente se espera que así sea en el parcial. Dicho promedio también tiene cierta correlación con la cantidad de tests en el parcial, 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 una mejor resultado en el parcial.

# Conclusiones

A lo largo de los ejercicios los alumnos aplican la técnica de TDD como también el contenido de la materia (patrones de diseño, polimorfismo, el lenguaje de programación Smalltalk, etc) y lo mismo aplica para el examen. Bajo este contexto, la medición de la aplicación de TDD viene condicionada por las características del desarrollo en cuestión, es decir, aplicando conceptos y en una evaluación (con límite de tiempo, nervios, etc). Por lo que se dificulta segregar en las mediciones el aprendizaje de TDD en neto. Una posibilidad sería agregar en la materia ejercicios antes, durante y al final de la cursada con el solo objetivo de evaluar TDD bajo algún contenido ya visto, por ejemplo aplicar el mismo patrón de diseño.

Por otro lado, se pudo evidenciar en ambos cuatrimestres que hay una correlación entre la aplicación de TDD y la nota del parcial. Es esperable que los que aprendieron bien la técnica así lo reflejen en su evaluación, pero también hay que tener en cuenta que en la evaluación juegan los temas mencionados anteriormente en su desarrollo además de TDD, lo que denota que sí se pudieron superar el aprendizaje fue firme.

En el indicador del *Tiempo en los estados de TDD* se puede ver en 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.

# Trabajo a futuro

En el presente trabajo no se detectaron muchos patrones relevantes entre los dos cuatrimestres analizados. Sería interesante calcular los mismos indicadores aquí presentados a lo largo de varios cuatrimestres más y compararlos entre sí, junto con los que ya se presentaron, a fin de buscar algún patrón.

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 archivos 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 o si de detectar alguna modificación en el orden por tiempos cotejar algún cambio introducido en dicho cuatrimestre.

# Apendices

## Apendice 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

## Apendice 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.

**Apendice XY Tópicos de TDD para la corrección de los parciales**

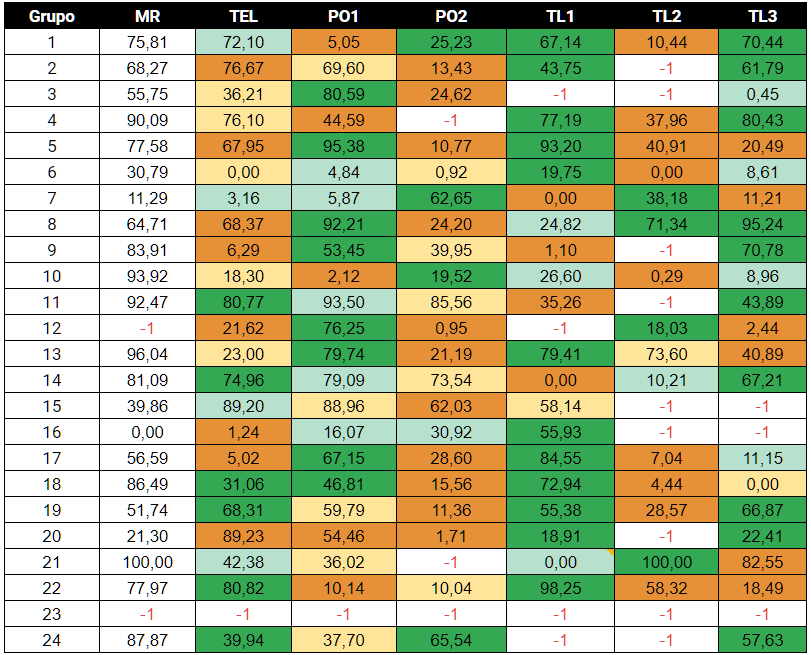
*2C 2020*

* 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)

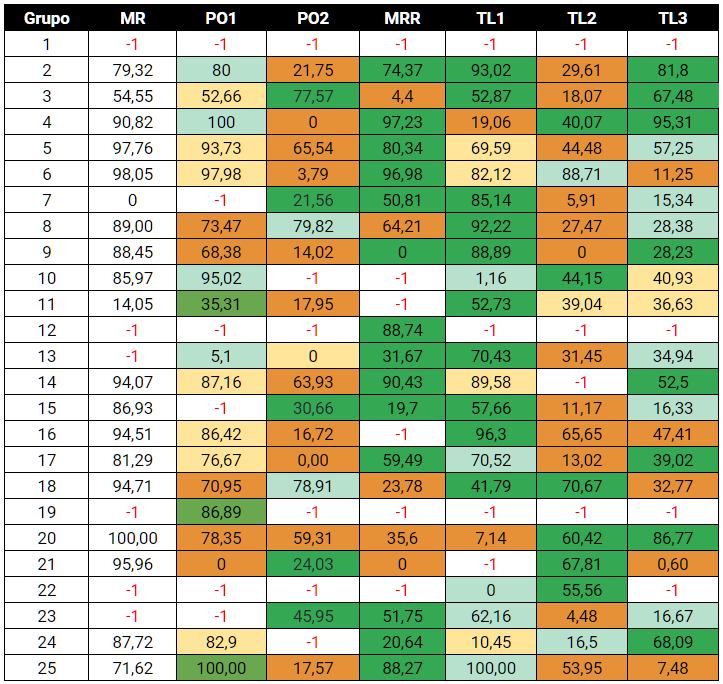
*1C 2021*

* 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)

**Apendice XY** **Tablas de porcentajes de aplicación de TDD por grupo en cada ejercicio**

****

**2C 2020**

****

**1C 2021**

## Apendice 3 **Repositorio de scripts y datos**

<https://github.com/OctavioBit/EvolucionAprendizajeTDD>

# Referencias

**[1]** Beck, K. Test-Driven Development by Example, Addison Wesley - Vaseem, 2003

**[2]** Matias Dinota TDDGuru <https://github.com/mdinota/TDDGuru>

**[3]** CuisUniversity <https://sites.google.com/view/cuis-university/inicio>

**[REFSINALTO]** 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

**[REFHAKAN]** Hakan Erdogmus, The Role of Process Measurement in Test-Driven Development, Conference Paper in Lecture Notes in Computer Science - Agosto 2004

**[4]** Pandas <https://pandas.pydata.org/>

**[5]** <https://en.wikipedia.org/wiki/Box_plot>

**[6]** <https://en.wikipedia.org/wiki/Pearson_correlation_coefficient>

**[7]** 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.