Capítulo 6: Anexos y apéndices

6.1. ANEXO A: Lineamientos para el sistema de información

a. Tema planteado:

Sistema de Información para una empresa comercializadora del rubro ferretero. Deberá asumir que la empresa pertenece al grupo de grandes corporaciones e inicia operaciones en el Perú.

b. Objetivo del trabajo:

Ejercitar a los alumnos en la administración de proyectos empleando diferentes herramientas que proporciona la Ingeniera de Software.

c. Descripción:

Se desarrollará un software cliente-servidor que ofrezca apoyo a una empresa comercial del sector ferretero en lo siguiente: venta de artículos y servicios (considerar cotizaciones, boletas y facturas), venta al por mayor y menor (en tienda y puesto en obra), compras (cotizaciones y órdenes de compras), manejo de almacenes e inventarios (considerar devoluciones y roturas dentro del almacén), toma de inventarios (apoyo para realizar toma de inventarios físicos anuales, cíclicos, etc.), recursos humanos (asignación de turnos de trabajo y gestión de empleados).

d. Módulos básicos:

- Seguridad
- Recursos Humanos
- Almacenes
- Compras
- Ventas
- Reportes

e. Metodología del trabajo:

- Se conformarán grupos de proyecto que competirán por la construcción del software que cumpla los lineamientos establecidos y obtenga la mejor calidad en el producto desarrollado.
- Cada “empresa” contará con el apoyo de un asesor.
- Se definirán roles dentro de cada grupo.
- Se tendrá especial énfasis en el que los alumnos apliquen las buenas prácticas de Ingeniería de Software y las recomendaciones basadas en el PMBOK para la gestión de proyecto.

f. Ambiente de desarrollo:

- SO Cliente: Microsoft Windows.
- Entorno de Programación: Visual Studio 2010 – C Sharp (C#).
- Se utilizará la librería NUnit para el diseño de los casos de prueba automatizados y TDD.
- Manejador de Base de Datos: MS SQL Server 2008.

g. Consideración general:

La propuesta del tema se ha delimitado en muchos aspectos a fin de poder establecer un conjunto de requerimientos posibles de desarrollar en el periodo lectivo actual. Las limitaciones establecidas se hacen con el fin de simplificar algunos de los procesos, el proceso real es más complejo y demanda mayores niveles de validación.

6.2. ANEXO B: Encuesta de percepción/intención de uso para TDD y Cascada

a. Objetivo:

Aplicar la misma encuesta a los sujetos tanto para TDD como para Cascada, a fin de medir la percepción/intención en función de la Facilidad de Uso Percibida (PEOU), Utilidad Percibida (PU) e Intención de Uso (ITU). Por consiguiente cada sujeto debe responder la encuesta 2 veces.

b. Cuestionario:

Para cada uno de los siguientes pares de frases, por favor marque una cruz sobre el círculo que más se aproxime a su opinión. Recuerde que no hay respuestas correctas a estas preguntas, brinde su opinión sincera basada en su experiencia.

Por favor lea cada pregunta cuidadosamente antes de marcar su respuesta:

P1	El proceso a seguir para implementar la técnica es complejo y difícil de seguir.	O O O O O	El proceso a seguir para implementar la técnica es simple y fácil de utilizar.
P2	Creo que esta técnica permitiría obtener una solución óptima en los sistemas que se utilice.	O O O O O	Creo que esta técnica NO permitiría obtener una solución óptima en los sistemas que se utilice.
P3	Encuentro la técnica difícil de utilizar.	O O O O O	Encuentro la técnica fácil de utilizar.
P4	Encuentro que las reglas para la implementación de la	O O O O O	Encuentro que las reglas para la implementación de la

	técnica son claras y fáciles de entender.		técnica son confusas y difíciles de entender.
P5	Encuentro que la técnica es útil.	O O O O O	Encuentro que la técnica NO es útil.
P6	Encuentro la técnica difícil de aprender.	O O O O O	Encuentro la técnica fácil de aprender.
P7	Utilizaré esta técnica si tengo que obtener una solución óptima en sistemas del rubro ferretero.	O O O O O	NO utilizaré esta técnica si tengo que obtener una solución óptima en sistemas del rubro ferretero.
P8	Pienso que esta técnica NO mejoraría la calidad del software (medida en función de la cantidad de errores detectados durante el proceso de implementación del software).	O O O O O	Pienso que esta técnica mejoraría la calidad del software (medida en función de la cantidad de errores detectados durante el proceso de implementación del software).
P9	Encontré difícil de aplicar la técnica en un sistema del rubro ferretero.	O O O O O	Encontré fácil de aplicar la técnica en un sistema del rubro ferretero.
P10	De manera general, pienso que esta técnica NO proporcionó una manera eficaz de realizar el sistema de rubro ferretero.	O O O O O	De manera general, pienso que esta técnica proporcionó una manera eficaz de realizar el sistema de rubro ferretero.
P11	Pienso que sería fácil para mí ser	O O O O O	Pienso que sería difícil para mí ser

	experto en el uso de esta técnica.		experto en el uso de esta técnica.
P12	Tengo la intención de utilizar esta técnica en el futuro.	O O O O O	NO tengo la intención de utilizar esta técnica en el futuro.

c. Agrupación de preguntas por variable

#	Preguntas del cuestionario		Variable
P1	El proceso a seguir para implementar la técnica es complejo y difícil de seguir.	El proceso a seguir para implementar la técnica es simple y fácil de utilizar.	PEOU
P3	Encuentro la técnica difícil de utilizar.	Encuentro la técnica fácil de utilizar.	PEOU
P4	Encuentro que las reglas para la implementación de la técnica son claras y fáciles de entender.	Encuentro que las reglas para la implementación de la técnica son confusas y difíciles de entender.	PEOU
P6	Encuentro la técnica difícil de aprender.	Encuentro la técnica fácil de aprender.	PEOU
P9	Encontré difícil de aplicar la técnica en un sistema del rubro ferretero.	Encontré fácil de aplicar la técnica en un sistema del rubro ferretero.	PEOU
P2	Creo que esta técnica permitiría obtener una solución óptima en los sistemas que se utilice.	Creo que esta técnica NO permitiría obtener una solución óptima en los sistemas que se utilice.	PU
P5	Encuentro que la técnica es útil.	Encuentro que la técnica NO es útil.	PU
P8	Pienso que esta técnica NO mejoraría la calidad del software (medida en función de la cantidad de	Pienso que esta técnica mejoraría la calidad del software (medida en función de la cantidad de errores	PU

	errores detectados durante el proceso de implementación del software).	detectados durante el proceso de implementación del software).	
P10	De manera general, pienso que esta técnica NO proporcionó una manera eficaz de realizar el sistema de rubro ferretero.	De manera general, pienso que esta técnica proporcionó una manera eficaz de realizar el sistema de rubro ferretero.	PU
P7	Utilizaré esta técnica si tengo que obtener una solución óptima en sistemas del rubro ferretero.	NO utilizaré esta técnica si tengo que obtener una solución óptima en sistemas del rubro ferretero.	ITU
P11	Pienso que sería fácil para mí ser experto en el uso de esta técnica.	Pienso que sería difícil para mí ser experto en el uso de esta técnica.	ITU
P12	Tengo la intención de utilizar esta técnica en el futuro.	NO tengo la intención de utilizar esta técnica en el futuro.	ITU

Tabla 6.1: Preguntas agrupadas para la medición de cada variable

6.3. ANEXO C: Revisión del estado del arte

a. Introducción

Para la realización del presente trabajo, se utilizó como método de investigación una revisión sistemática de los estudios empíricos relacionados con la técnica *Test Driven Development (TDD).*

Para la revisión respectiva se tuvieron en cuenta los siguientes criterios:

- **Criterio de Inclusión:** Serán considerados aquellos estudios empíricos encontrados que se enfoquen en la aplicación de la técnica TDD y en el enfoque Cascada.

- **Criterio de Exclusión:** Serán desconsiderados aquellos estudios que no se enfoquen en la utilización de la técnica TDD.

En base a los criterios establecidos se pudo determinar si el estudio se incluía o no en la revisión para el presente trabajo de investigación.

Las preguntas planteadas fueron las siguientes:

1. ¿Cuál es la relación que existe entre las metodologías ágiles y TDD?
2. ¿Cuál es la relación que existe entre TDD y Clean Code?
3. ¿Qué trabajos o estudios se han realizado aplicando Test Driven Development y Cascada?

Estas preguntas sirvieron de guía para el diseño del proceso de revisión de los diferentes trabajos de investigación realizados en el ámbito de la presente investigación. Se utilizaron las siguientes bases de datos:

- ACM Digital Library
- Science Direct
- Scopus

Las cadenas de búsqueda ingresadas en cada una de las bases de datos seleccionadas se detallan a continuación:

Pregunta 1: ¿Cuál es la relación que existe entre las metodologías ágiles y TDD?	
Base de datos	**Cadena de búsqueda**
ACM Digital Library	(Abstract: TDD and Abstract:"agile methodologies")
Science Direct	ABSTRACT (TDD) AND ABSTRACT ("agile methodologies")
Scopus	ABS (TDD AND "agile methodologies")

Tabla 6.2: Cadenas de búsqueda relacionadas a metodologías ágiles y TDD

Pregunta 2: ¿Cuál es la relación que existe entre TDD y Clean Code?	
Base de datos	**Cadena de búsqueda**
ACM Digital Library	(Abstract: TDD and Abstract:"clean code")
Science Direct	ABSTRACT (TDD) AND ABSTRACT ("clean code")
Scopus	ABS (TDD AND "clean code")

Tabla 6.3: Cadenas de búsqueda relacionadas a TDD y Clean Code

Pregunta 3: ¿Qué trabajos o estudios se han realizado aplicando Test Driven Development y Cascada?	
Base de datos	**Cadena de búsqueda**

ACM Digital Library	(Abstract:"test driven development") and (Abstract:waterfall or Abstract:cascade)
Science Direct	ABSTRACT(("test driven development") AND ("waterfall" OR "cascade"))
Scopus	(ABS ("test driven development") AND ABS (waterfall OR cascade))

Tabla 6.4: Cadenas de búsqueda relacionadas a TDD y Cascada

Es preciso mencionar que las cadenas de búsqueda fueron definidas en base a las palabras clave identificadas dentro de las preguntas planteadas.

La búsqueda de estas cadenas se realizó a nivel de resúmenes para garantizar que los estudios encontrados presenten como enfoque central de la investigación el tema de interés y así poder obtener los diversos estudios realizados y bajo diferentes contextos.

En base a los resultados obtenidos y eliminación de aquellos estudios empíricos que no cumplían con el criterio de inclusión, se obtuvo un total de 12 estudios, los cuales brindan un enfoque global de lo realizado en la actualidad con respecto al tema del presente trabajo de investigación.

b. Estudios encontrados

En base a la revisión sistemática realizada, en la tabla mostrada a continuación se listan los estudios empíricos encontrados que sirvieron de base para el presente trabajo de investigación. El detalle del estudio, por autor y abstract, se muestra a continuación en la Tabla 6.5:

#	Estudio
1	**"Reducing Test Cost and Improving Documentation In TDD (Test Driven Development)"** [Karamat, 2006] **Autor:** Taha Karamat; Atif Neuman Jamil. **Abstract:** In the fast pace business world of today where competition

	and technology are at their zenith, software development companies need to improve their quality standards in addition to cost reduction in operations. To achieve these challenging objectives various developments are on the verge. In recent past agile methodologies have emerged as one of the most efficient implementations in the world of software development arena. Especially eXtreme Programming (XP) which is integrated by Test first approach recent research proves the emergence of Test Driven Development (TDD) from this concept which is based on formalizing the requirement as a test and secondly to write such a code that can pass the test. This attempt of our research provides a mechanism to reduce the cost of testing, mainly due to troublesome test which fails again and again. We used TDD, analyzed the problem and proposed a workable solution. Test Driven Development is a technique which encourages less documentation resulting in a lot of difficulties for developers in contrast to traditional methods. In order to reduce the burden on developers we have proposed some steps in documentation.
2	**"Bug localization in test-driven development"** [Ficco, 2011] **Autor:** Massimo Ficco; Roberto Pietrantuono; Stefano Russo. **Abstract:** Software development teams that use agile methodologies are increasingly adopting the test-driven development practice (TDD). TDD allows producing software by iterative and incremental work cycle, and with a strict control over the process, favouring an early detection of bugs. However, when applied to large and complex systems, TDD benefits are not so obvious; manually locating and fixing bugs introduced during the iterative development steps is a nontrivial task. In such systems, the propagation chains following the bugs activation can be unacceptably long and intricate, and the size of the code to be analyzed is often too large. In this paper, a bug localization technique specifically tailored to TDD is presented. The technique is embedded in the TDD cycle, and it aims to improve developers' ability to locate bugs as soon as possible. It is

	implemented in a tool and experimentally evaluated on newly developed Java programs.
3	**"Discipline and practices of TDD: (test driven development)"** [Fraser, 2003] **Autor:** Steven Fraser; Dave Astels; Kent Beck; Barry Boehm; John McGregor; James Newkirk; Charlie Poole. **Abstract:** This panel brings together practitioners with experience in Agile and XP methodologies to discuss the approaches and benefits of applying Test Driven Development (TDD). The goal of TDD is clean code that works. The mantra of TDD is: write a test; make it run; and make it right. Open questions to be addressed by the panel include: How are TDD approaches to be applied to databases, GUIs, and distributed systems? What are the quantitative benchmarks that can demonstrate the value of TDD, and what are the best approaches to solve the ubiquitous issue of scalability?.
4	**"To test before or to test after---an experimental investigation of the impact of test driven development"** [Bhadauria, 2009] **Autor:** Vikram S. Bhadauria; Radha K. Mahapatra. **Abstract:** Test Driven Development (TDD) requires the developer to create the test suite before designing and writing the application program. Unlike traditional software development practices, in TDD test development precedes application development. Such a practice also redefines the role of the developer. Lately, TDD is growing in popularity as a part of Agile methodologies. There is a critical need for rigorous empirical research to understand the role and impact of TDD as a software development practice. The goal of this dissertation research is to fill this gap. A laboratory experiment was conducted to understand the influence of TDD on the outcomes of the software development process. Software quality, learning and task satisfaction were examined as outcome variables. In the experiment, groups that used the traditional method of software development were compared with those that used TDD. Individual programmers were also compared with paired programmers, when both used TDD.

5	**"A structured experiment of test-driven development"** [George, 2003] **Autor:** Boby George; Laurie Williams. **Abstract:** Test Driven Development (TDD) is a software development practice in which unit test cases are incrementally written prior to code implementation. We ran a set of structured experiments with 24 professional pair programmers. One group developed a small Java program using TDD while the other (control group), used a waterfall-like approach. Experimental results, subject to external validity concerns, tend to indicate that TDD programmers produce higher quality code because they passed 18% more functional black-box test cases. However, the TDD programmers took 16% more time. Statistical analysis of the results showed that a moderate statistical correlation existed between time spent and the resulting quality. Lastly, the programmers in the control group often did not write the required automated test cases after completing their code. Hence it could be perceived that waterfall-like approaches do not encourage adequate testing. This intuitive observation supports the perception that TDD has the potential for increasing the level of unit testing in the software industry.
6	**"Comparison between test driven development and waterfall development in a small-scale project"** [Zhang, 2006] **Autor:** Lei Zhang, Shunsuke Akifuji, Katsumi Kawai, Tsuyoshi Morioka **Abstract:** In order to popularize the Test Driven Development (TDD) practice in Chinese offshore companies, an experimental research was firstly conducted to compare TDD with the traditional waterfall development in a small-scale project. Although the project scale was small and all the subjects were students, this experiment was designed very strictly to guarantee the reliable evaluation of the efficacy of TDD. Furthermore, it is also the first time to evaluate the maintainability and the flexibility of TDD by experiment.
7	**"Agile business continuity planning using business process modeling notation"** [Anne, 2013]

	Autor: Kirk M. Anne / Fred Grossman **Abstract:** Many current business continuity plans focus on the technology of an organization, not on the processes. This research investigates how using agile methodology and business process modeling can change the efficiency and effectiveness of business continuity planning. Recent surveys of business continuity planners indicate that testing business continuity plans exposes problems and provides an opportunity to correct and improve them. However, in practice, plans are rarely tested after their creation and happen only after they are completed. Current business continuity planning practice is similar to the traditional waterfall method of software design and levies a significant drain on an organization's resources to develop plans. This dissertation explores the use of agile techniques, like designing plans using "test first" design, focusing on the highest value processes first, and developing quality plans with the minimum of effort and error. This dissertation presents an agile methodology that harnesses "test first" development of plans using business process modeling notation (BPMN). It focuses on the continuity of processes, not the causes of a disruption. This methodology allows for a more organic approach to business continuity where IT staff and business process owners collaborate closely and leverage proven agile and test-driven development techniques. The agile techniques allow for a lightweight method of planning that reduces unnecessary work. By using BPMN, a machine executable notation, plans are easier to test by computer. Since BPMN is also graphical, plans developed in BPMN are less ambiguous, less inconsistent, and easier to communicate with others within and outside an organization.
8	**"Effectiveness of test-driven development as an SDLC model: A case study of an elevator controller design"** [Mondal, 2014] **Autor:** Mondal, S. , Das, P.P. **Abstract:** Test-driven development (TDD) is a new software development model where codes are written to meet the tests as specified from the specs. It is an agile method and claims to be more

	effective and efficient than the traditional waterfall (and other derivative) SDLC models. In this paper we use the development of an elevator controller as a target system and compare TDD against waterfall through independent development. Using three progressive "versions" of elevator system, we show the advantages of TDD over Waterfall.
9	**“Test-Driven lecturing”** [Itkonen, 2012] **Autor:** Itkonen, J. **Abstract:** One can easily compare the current style of teaching and lecturing to the so called waterfall model of software development. We first design the course, then execute it, and in the end we make tests to see, if everything went well. As with waterfall model, the assessment comes too late, if anything fails. Therefore, we need a lecturing model which entwines assessment into course execution. Test-driven development (TDD) is a model of software development in which the computer program is designed by writing first small tests that assure the meeting of the requirements. In test-driven lecturing, students first take tests to show how well they master the topics to be discussed. In this way both the students and lecturers are assured that everything is going fine, and if not, the problems are found early, so one can react to them.
10	**“Software evolution in agile development: A case study?”** [Sindhgatta, 2010] **Autor:** Sindhgatta, R. , Narendra, N.C. , Sengupta, B. **Abstract:** The agile development method (ADM) is characterized by continuous feedback and change, and a software system developed using ADM evolves continuously through short iterations. Empirical studies on evolution of software following agile development method have been sparse. Most studies on software evolution have been performed on systems built using traditional (waterfall) development methods or using the open source development approach. This paper summarizes our study on the evolution of an enterprise software system following ADM. We evaluated key characteristics of evolution in the light of Lehman's laws of software evolution dealing with continuous change

	and growth, self-regulation and conservation, increasing complexity and declining quality. Our study indicates that most laws of evolution are followed by the system. We also present our observations on agile practices such as collective code ownership, test driven development and collaboration when the team is distributed.
11	**"Agilists and the art of integrated assessment tool development"** [Knapen, 2010] **Autor:** Knapen, R. , Verweij, P., Janssen, S. **Abstract:** Software engineering for Integrated Assessment needs to address the fact that systems and models developed are increasingly used in participatory settings. Applying old principles of inside-out design and development using waterfall model based processes is no longer sufficient. New software engineering insights based on interaction design and iterative, agile processes for the development help in building systems from a more outside-in perspective. Based on two case studies from large European projects on integrated assessment this switch from applying old to applying new principles will be described, and its effects discussed. Elements were taken from interaction design: personas, story boards, mock-ups, focus groups and focus tasks. These were mixed with an iterative and incremental development process, using agile elements such as: daily stand-ups, user stories, planning games, Test Driven Development and continuous integration. The resulting process was used for the development of the two systems (SIAT and SEAMLESS-IF) for integrated assessment, giving them a more user-oriented focus than what would have resulted from following the old principles.
12	**"Escape the waterfall: Agile for aerospace"** [VanderLeest, 2009] **Autor:** VanderLeest, S.H.a, Buter, A.b **Abstract:** Agile is an umbrella software methodology that incorporates many of the best practices of the last couple of decades. In this paper, we will examine some of those key techniques for possible application in the aerospace domain, starting with a brief literature review to identify the key Agile publications and the germane DO-178B work. Virtually all

	Agile practices can be mapped to a DO-178B software development process. We provide a detailed analysis of the key practices, with a preliminary assessment of the ease of implementation for each. An analysis of a number of the difficulties involving transitioning from a traditional waterfall software development process to Agile practices will show that, though difficult, a transition is possible. The transition to Agile development does not require sudden, sweeping change, but instead can be accomplished through incorporating Agile methods into an existing process. We will document successful integration of test-driven development, pair programming, refactoring, an iterative approach, and other Agile methods into a traditional DO-178B software development process. We conclude with a call for a collaborative effort to further explore Agile as an answer to the urgent need for new approaches to complex systems that have become increasingly difficult to verify and validate.

Tabla 6.5: Relación de estudios empíricos revisados

c. Conclusión

Todos los estudios citados sirvieron de base para lograr definir el tema de estudio que guía el presente trabajo de investigación.

Con el panorama general de los trabajos realizados en la actualidad, se determinaron las actividades, acciones y nuevas aportaciones a brindar con la ejecución del experimento seleccionado dentro del ámbito de la Ingeniería del Software.

d. Referencias bibliográficas

[Karamat, 2006] Taha Karamat; Atif Neuman Jamil; “Reducing Test Cost and Improving Documentation In TDD (Test Driven Development)”, International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2006.

[Ficco, 2011] Massimo Ficco; Roberto Pietrantuono; Stefano Russo; “Bug Localization in Test-Driven Development”, 2011.

[Fraser, 2003] Steven Fraser; Dave Astels; Kent Beck; Barry Boehm; John McGregor; James Newkirk; Charlie Poole; “Discipline and Practices of TDD (Test Driven Development)”, 2003.

[George, 2003] Boby George; Laurie Williams; “A structured experiment of test- driven development”, Information and Software Technology 46, 2003.

[Bhadauria, 2009] Vikram Bhadauria; “To test before or to test after – An Experimental Investigation of the impact of Test Driven Development”, The University of Texas at Arlington, 2009.

[Zhang, 2006] Lei Zhang; Shunsuke Akifuji; Katsumi Kawai; Tsuyoshi Morioka; “Comparison between test driven development and waterfall development in a small-scale project”, 2006.

[Anne, 2013] Kirk M. Anne; Fred Grossman; “Agile business continuity planning using business process modeling notation”, 2013.

[Mondal, 2014] Mondal, S.; Das, P.P.; “Effectiveness of test-driven development as an SDLC model: A case study of an elevator controller design”, 2014.

[Itkonen, 2012] Itkonen, J.; “Test-Driven lecturing”, 2012.

[Sindhgatta, 2010] Sindhgatta, R.; Narendra, N.C.; Sengupta, B.;“Software evolution in agile development: A case study?”, 2010.

[Knapen, 2010] Knapen, R.; Verweij, P.; Janssen, S.; “Agilists and the art of integrated assessment tool development”, 2010.

[VanderLeest, 2009] VanderLeest, S.H.; Buter, A; “Escape the waterfall: Agile for aerospace”, 2009.

6.4. APÉNDICE A: Datos experimentales

Semanas	Grupo1_conTDD	Grupo2_conTDD	Grupo3_sinTDD	Grupo4_sinTDD	PromedioConTDD	PromedioSinTDD	Diferencia
1	25	30	25	26	27.50	25.50	2.00
2	18	22	18	17	20.00	17.50	2.50
3	16	17	16	14	16.50	15.00	1.50
4	22	21	12	10	21.50	11.00	10.50
5	16	17	10	7	16.50	8.50	8.00
6	0	3	1	1	1.50	1.00	.50

Cuadro 6.1: Cantidad de errores detectados (Cascada con TDD y sin TDD)

Sujeto	P1	P3	P4	P6	P9	P2	P5	P8	P10	P7	P11	P12
1	4	5	2	5	5	2	4	4	3	4	2	4
2	3	3	3	3	3	1	3	4	4	2	2	1
3	5	5	2	5	4	1	2	5	4	1	1	1
4	4	4	2	4	2	2	2	5	3	3	2	2
5	4	3	3	3	2	2	3	4	3	2	2	2
6	4	3	2	4	3	2	3	3	3	3	2	3
7	3	4	2	4	3	2	2	4	4	2	2	2
8	4	4	3	4	4	2	2	4	4	2	2	2
9	4	5	3	5	5	2	4	5	4	4	4	4
10	3	3	4	3	3	3	3	3	3	4	3	4
11	4	5	2	4	5	1	2	4	4	2	1	1
12	5	5	3	4	4	2	2	4	4	1	2	2
13	5	4	2	4	2	1	2	4	4	2	1	1
14	4	4	2	4	3	2	2	4	4	2	2	2
15	4	3	4	3	3	2	2	4	4	4	4	1
16	5	4	2	4	4	3	2	4	3	2	3	2
17	4	4	3	4	3	3	3	4	4	2	3	3
18	3	3	3	3	3	4	4	4	3	4	3	4
19	5	5	4	4	5	2	2	5	5	2	4	1
20	4	5	2	4	4	2	2	4	3	2	3	2
21	3	4	5	3	3	3	4	4	3	3	4	4
22	4	5	3	5	5	2	3	5	4	4	4	4

Cuadro 6.2: Puntajes obtenidos al aplicar encuesta sobre TDD

Sujeto	P1	P3	P4	P6	P9	P2	P5	P8	P10	P7	P11	P12
1	4	5	1	5	4	2	2	3	5	4	2	4
2	5	4	1	4	4	3	1	3	3	2	2	1
3	4	3	2	5	4	2	2	3	4	2	2	2
4	4	3	2	3	4	3	3	3	4	3	3	5
5	2	1	5	3	2	3	1	2	2	4	3	4
6	2	4	2	4	4	2	1	4	3	1	2	2
7	2	4	3	3	3	4	2	2	3	3	3	4
8	3	2	3	3	3	2	2	3	4	2	2	2
9	4	3	2	3	3	2	2	3	3	2	3	1
10	2	2	2	3	3	1	2	4	4	2	2	2
11	4	3	3	3	3	3	2	3	3	3	2	2
12	4	2	4	2	3	3	4	2	2	4	4	4
13	4	4	2	4	4	2	1	4	3	3	3	2
14	3	4	2	4	4	2	2	4	4	2	2	2
15	4	3	3	4	3	3	2	3	2	4	3	2
16	4	4	4	4	4	2	4	3	4	4	3	4
17	4	3	4	3	4	4	3	4	3	3	4	4
18	4	4	4	3	4	3	4	3	3	2	3	3
19	4	4	4	3	3	3	4	3	4	2	2	3
20	3	3	3	2	2	3	3	2	4	4	3	3
21	4	4	4	4	4	3	3	4	3	3	3	4
22	4	4	4	4	4	4	4	3	4	4	3	4

Cuadro 6.3: Puntajes obtenidos al aplicar encuesta sobre Cascada

Sujeto	PEOU_TDD	PEOU_CASCADA	PU_TDD	PU_CASCADA	ITU_TDD	ITU_CASCADA
1	4.20	3.80	3.25	3.00	3.33	3.33
2	3.00	3.60	3.00	2.50	1.67	1.67
3	4.20	3.60	3.00	2.75	1.00	2.00
4	3.20	3.20	3.00	3.25	2.33	3.67
5	3.00	2.60	3.00	2.00	2.00	3.67
6	3.20	3.20	2.75	2.50	2.67	1.67
7	3.20	3.00	3.00	2.75	2.00	3.33
8	3.80	2.80	3.00	2.75	2.00	2.00
9	4.40	3.00	3.75	2.50	4.00	2.00
10	3.20	2.40	3.00	2.75	3.67	2.00
11	4.00	3.20	2.75	2.75	1.33	2.33
12	4.20	3.00	3.00	2.75	1.67	4.00
13	3.40	3.60	2.75	2.50	1.33	2.67
14	3.40	3.40	3.00	3.00	2.00	2.00
15	3.40	3.40	3.00	2.50	3.00	3.00
16	3.80	4.00	3.00	3.25	2.33	3.67
17	3.60	3.60	3.50	3.50	2.67	3.67
18	3.00	3.80	3.75	3.25	3.67	2.67
19	4.60	3.60	3.50	3.50	2.33	2.33
20	3.80	2.60	2.75	3.00	2.33	3.33
21	3.60	4.00	3.50	3.25	3.67	3.33
22	4.40	4.00	3.50	3.75	4.00	3.67

Cuadro 6.4: Promedios obtenidos en TDD y Cascada para cada variable

Sujeto	PEOU_TDD	PEOU_CASCADA	DiferenciaPEOU
1	4.20	3.80	.40
2	3.00	3.60	-.60
3	4.20	3.60	.60
4	3.20	3.20	.00
5	3.00	2.60	.40
6	3.20	3.20	.00
7	3.20	3.00	.20
8	3.80	2.80	1.00
9	4.40	3.00	1.40
10	3.20	2.40	.80
11	4.00	3.20	.80
12	4.20	3.00	1.20
13	3.40	3.60	-.20
14	3.40	3.40	.00
15	3.40	3.40	.00
16	3.80	4.00	-.20
17	3.60	3.60	.00
18	3.00	3.80	-.80
19	4.60	3.60	1.00
20	3.80	2.60	1.20
21	3.60	4.00	-.40
22	4.40	4.00	.40

Cuadro 6.5: Diferencia para la variable PEOU

Sujeto	PU_TDD	PU_CASCADA	DiferenciaPU
1	3.25	3.00	.25
2	3.00	2.50	.50
3	3.00	2.75	.25
4	3.00	3.25	-.25
5	3.00	2.00	1.00
6	2.75	2.50	.25
7	3.00	2.75	.25
8	3.00	2.75	.25
9	3.75	2.50	1.25
10	3.00	2.75	.25
11	2.75	2.75	.00
12	3.00	2.75	.25
13	2.75	2.50	.25
14	3.00	3.00	.00
15	3.00	2.50	.50
16	3.00	3.25	-.25
17	3.50	3.50	.00
18	3.75	3.25	.50
19	3.50	3.50	.00
20	2.75	3.00	-.25
21	3.50	3.25	.25
22	3.50	3.75	-.25

Cuadro 6.6: Diferencia para la variable PU

Sujeto	ITU_TDD	ITU_CASCADA	DiferencialTU
1	3.33	3.33	.00
2	1.67	1.67	.00
3	1.00	2.00	-1.00
4	2.33	3.67	-1.34
5	2.00	3.67	-1.67
6	2.67	1.67	1.00
7	2.00	3.33	-1.33
8	2.00	2.00	.00
9	4.00	2.00	2.00
10	3.67	2.00	1.67
11	1.33	2.33	-1.00
12	1.67	4.00	-2.33
13	1.33	2.67	-1.34
14	2.00	2.00	.00
15	3.00	3.00	.00
16	2.33	3.67	-1.34
17	2.67	3.67	-1.00
18	3.67	2.67	1.00
19	2.33	2.33	.00
20	2.33	3.33	-1.00
21	3.67	3.33	.34
22	4.00	3.67	.33

Cuadro 6.7: Diferencia para la variable ITU

6.5. APÉNDICE B: Muestras de los Test Cases creados al aplicar TDD

a. Grupo 1: Construper-Soft

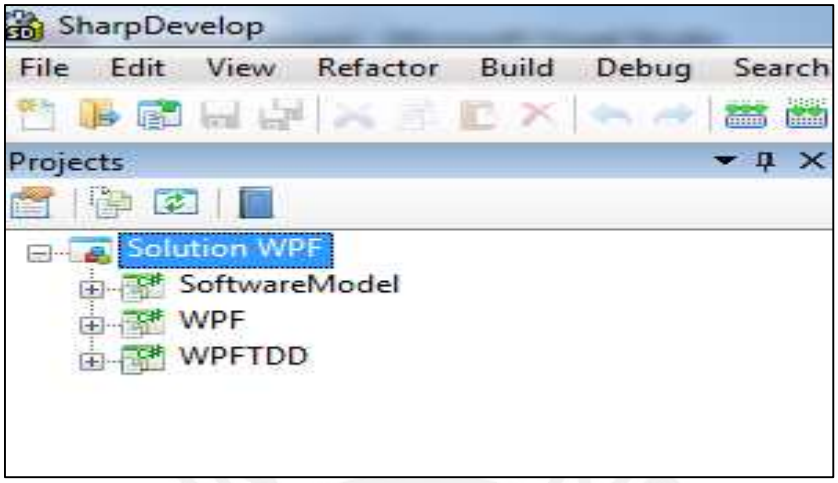

Imagen 6.1: Solución WPF de Construper Soft

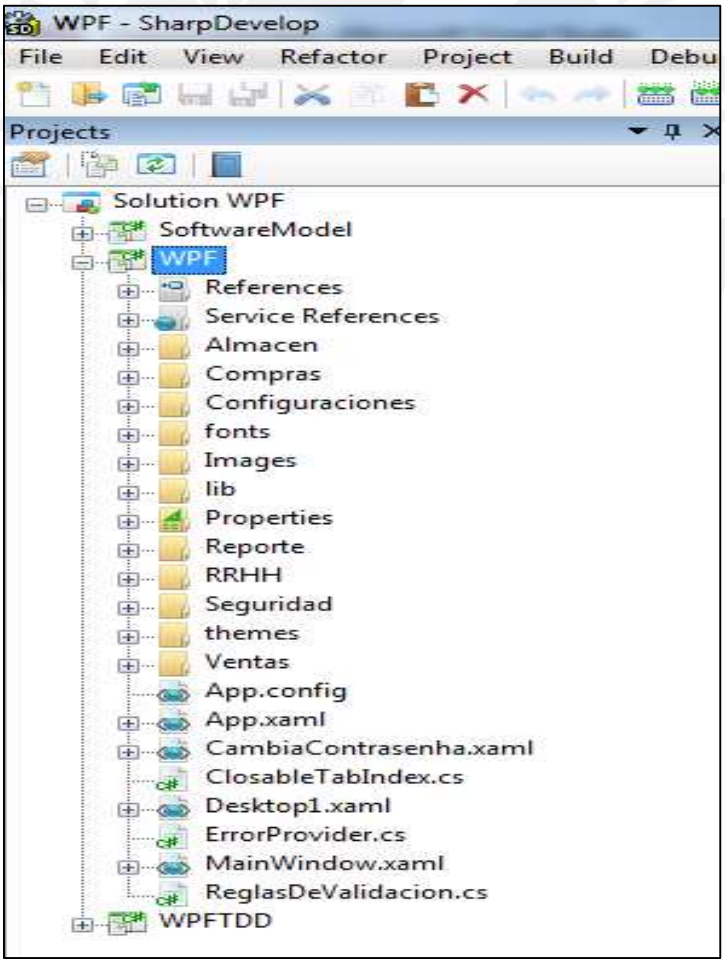

Imagen 6.2: Solución WPF – Implementación

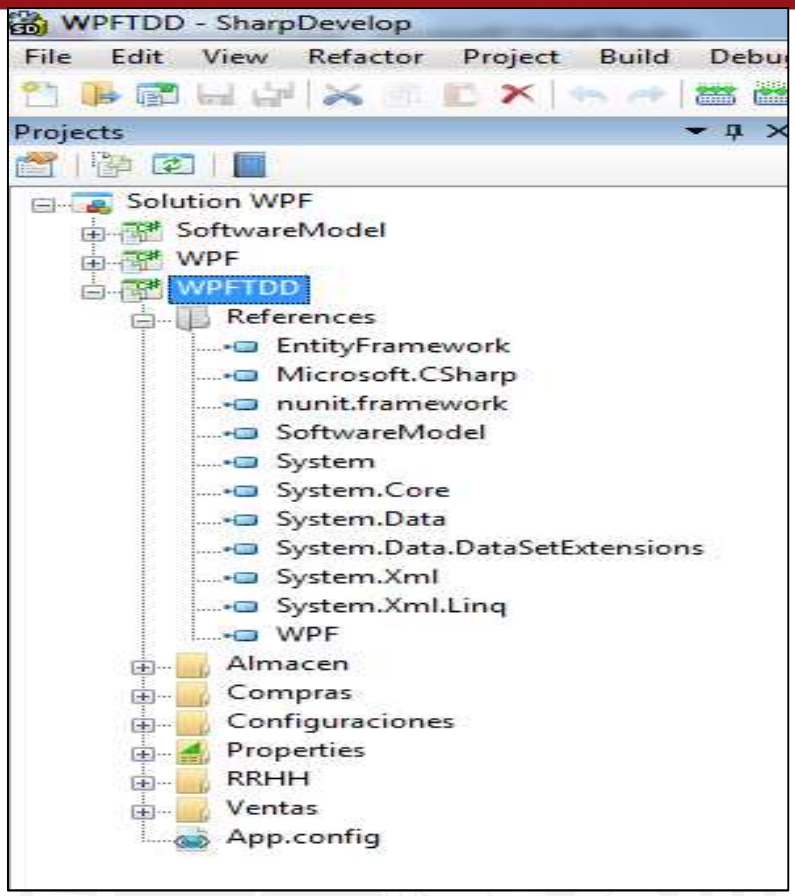

Imagen 6.3: Solución WPF – Implementación TDD

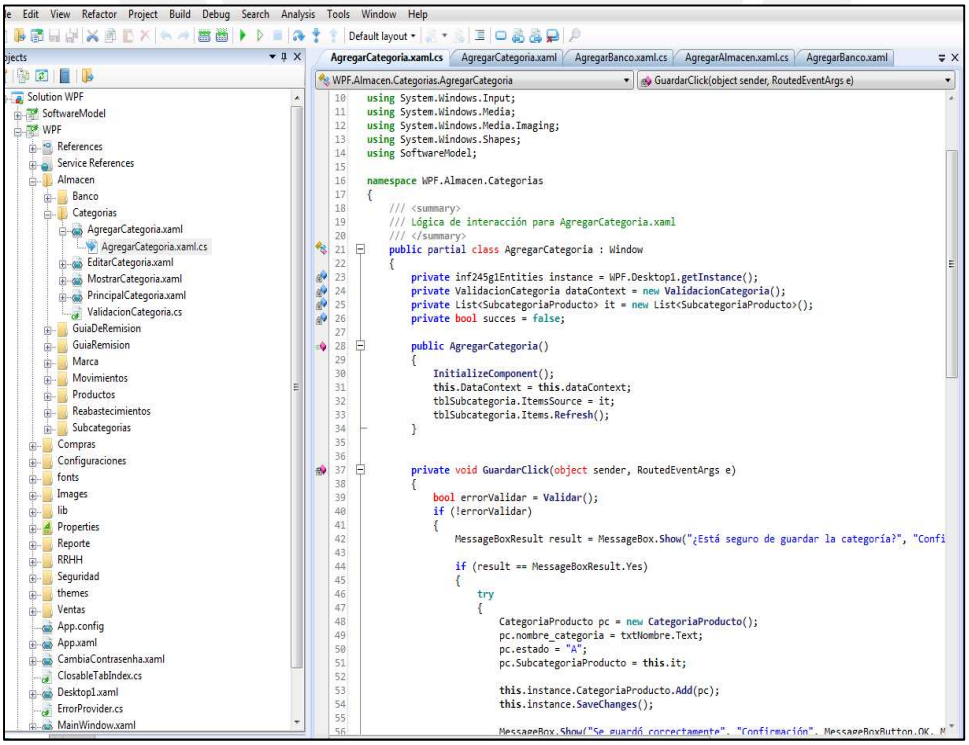

Imagen 6.4: Solución WPF – Ejemplo de codificación

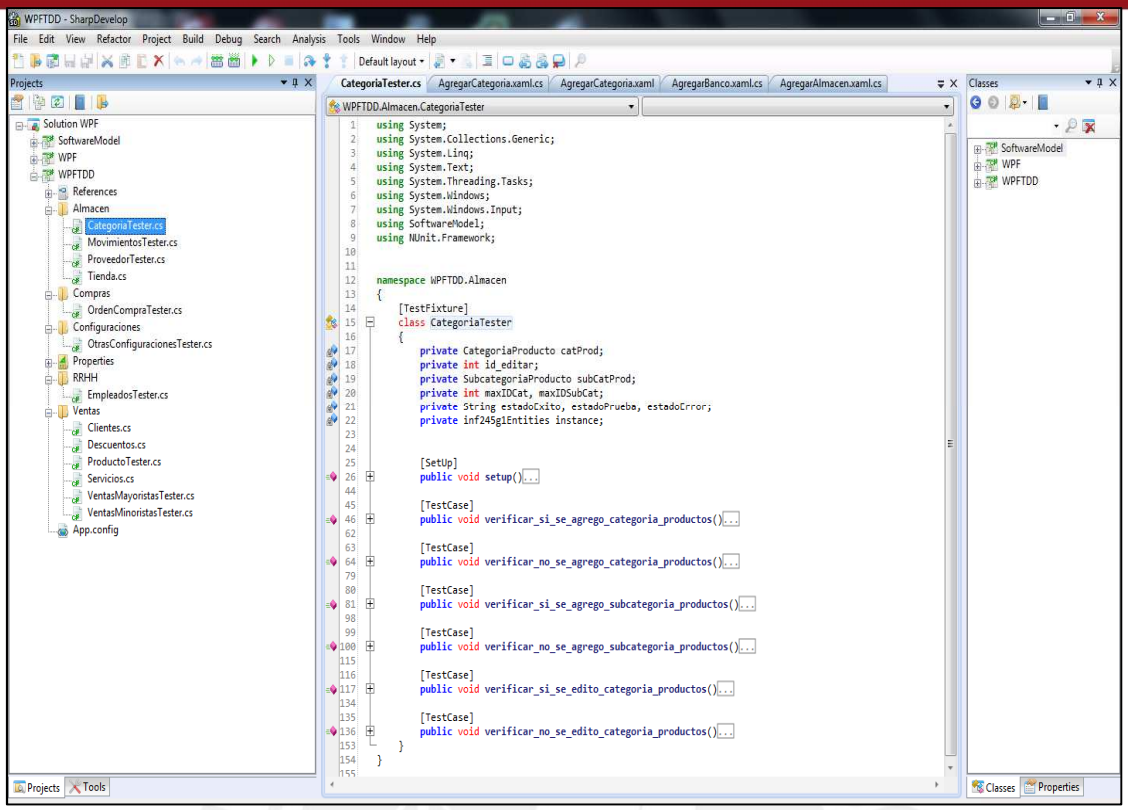

Imagen 6.5: Solución WPF – Ejemplo de TDD Almacén: Categoría

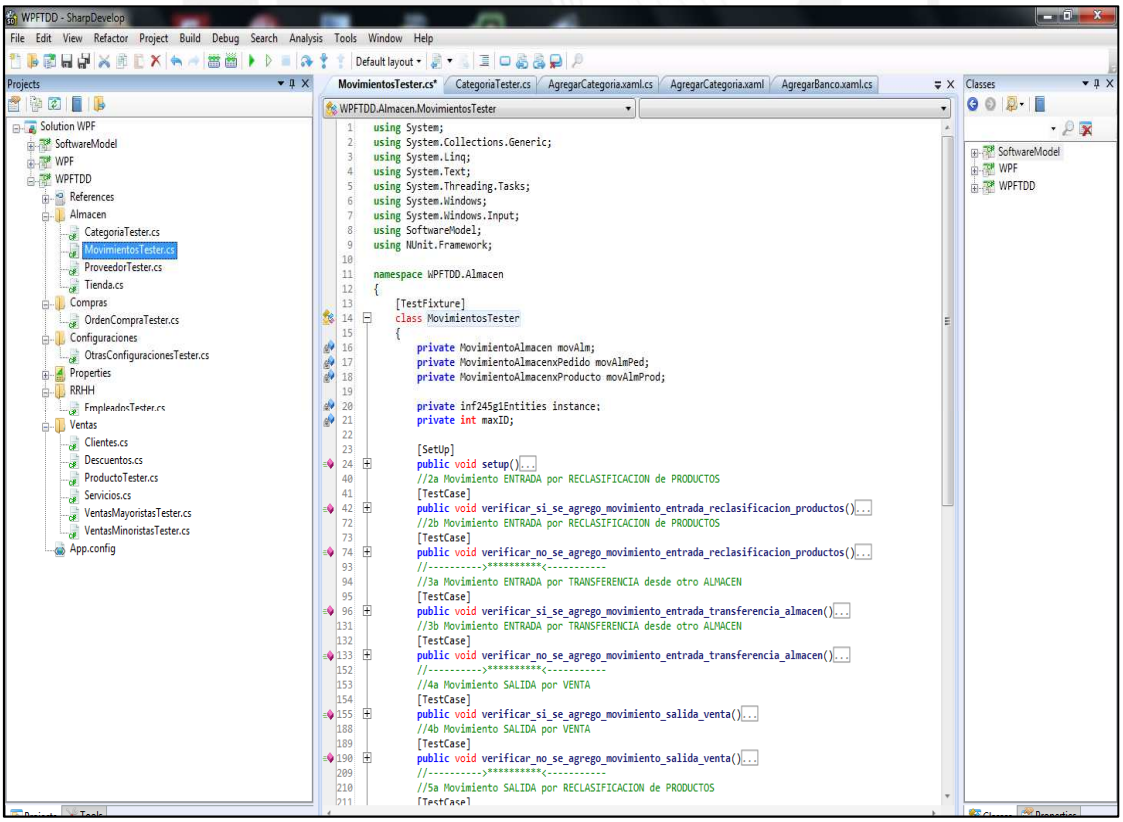

Imagen 6.6: Solución WPF – Ejemplo de TDD Almacén: Movimientos

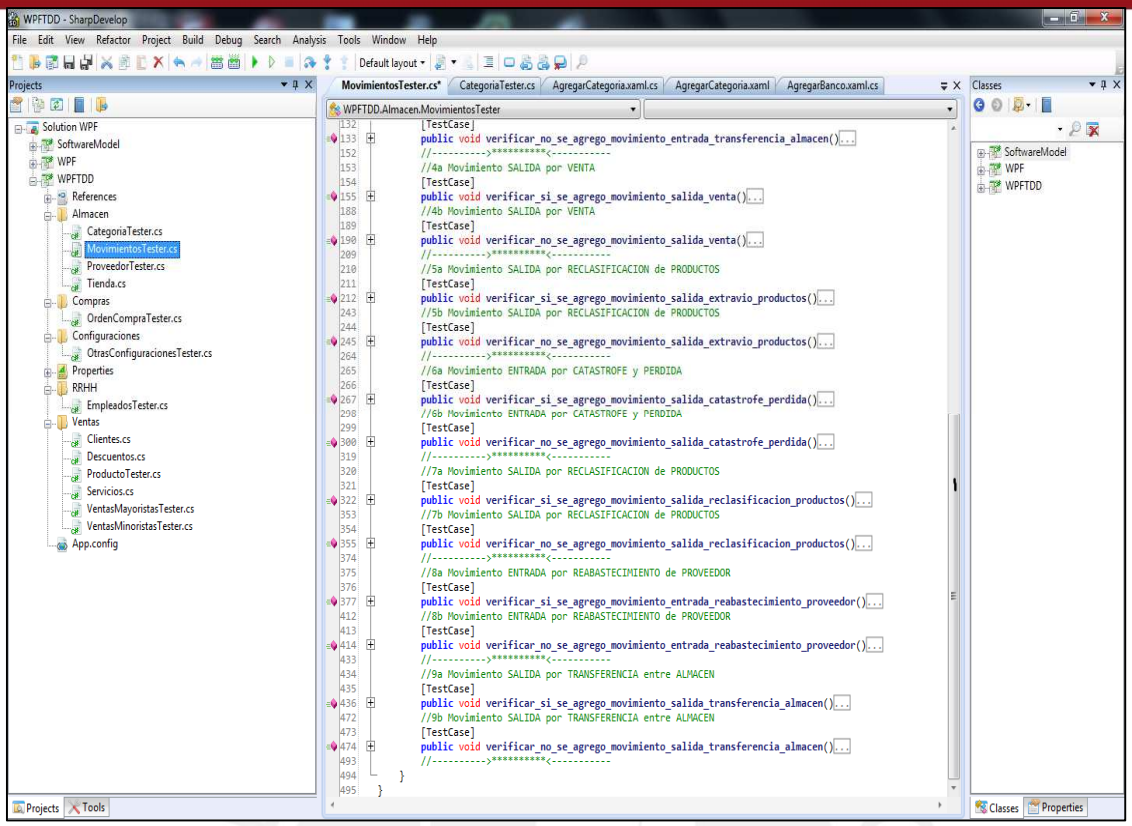

Imagen 6.7: Solución WPF – Ejemplo de TDD Almacén: Movimientos (cont.)

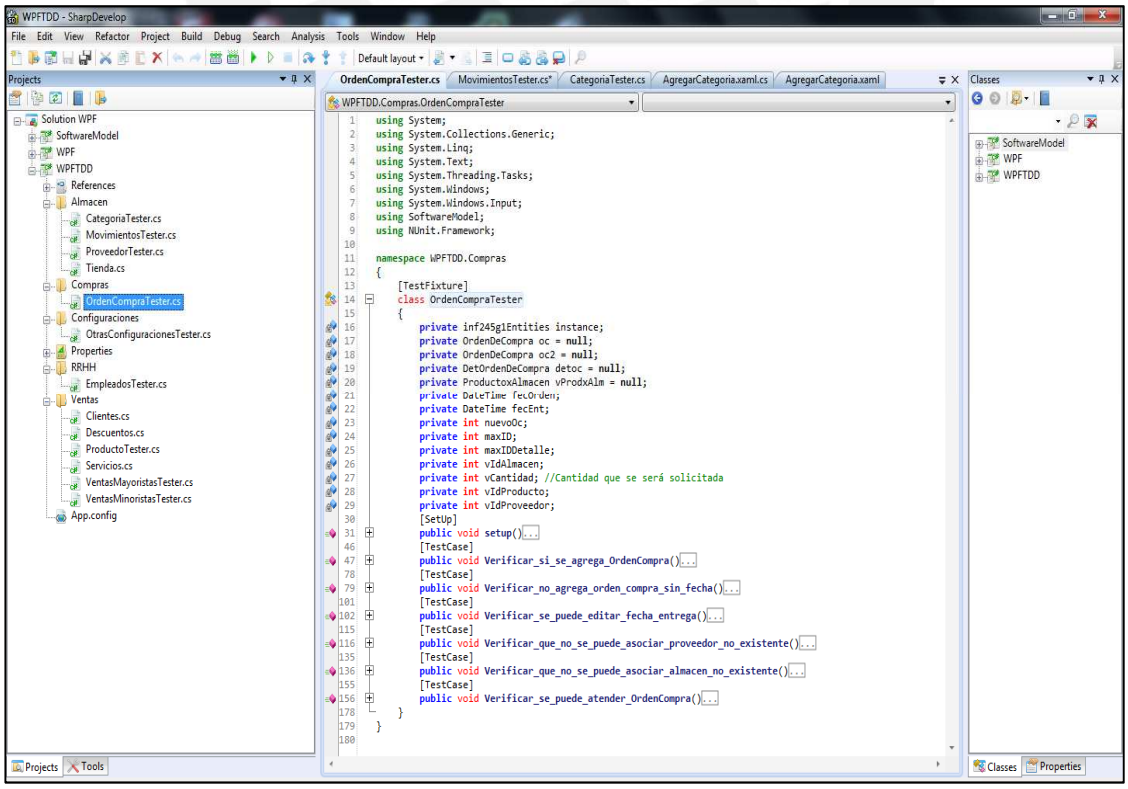

Imagen 6.8: Solución WPF – Ejemplo de TDD Compras: Órdenes

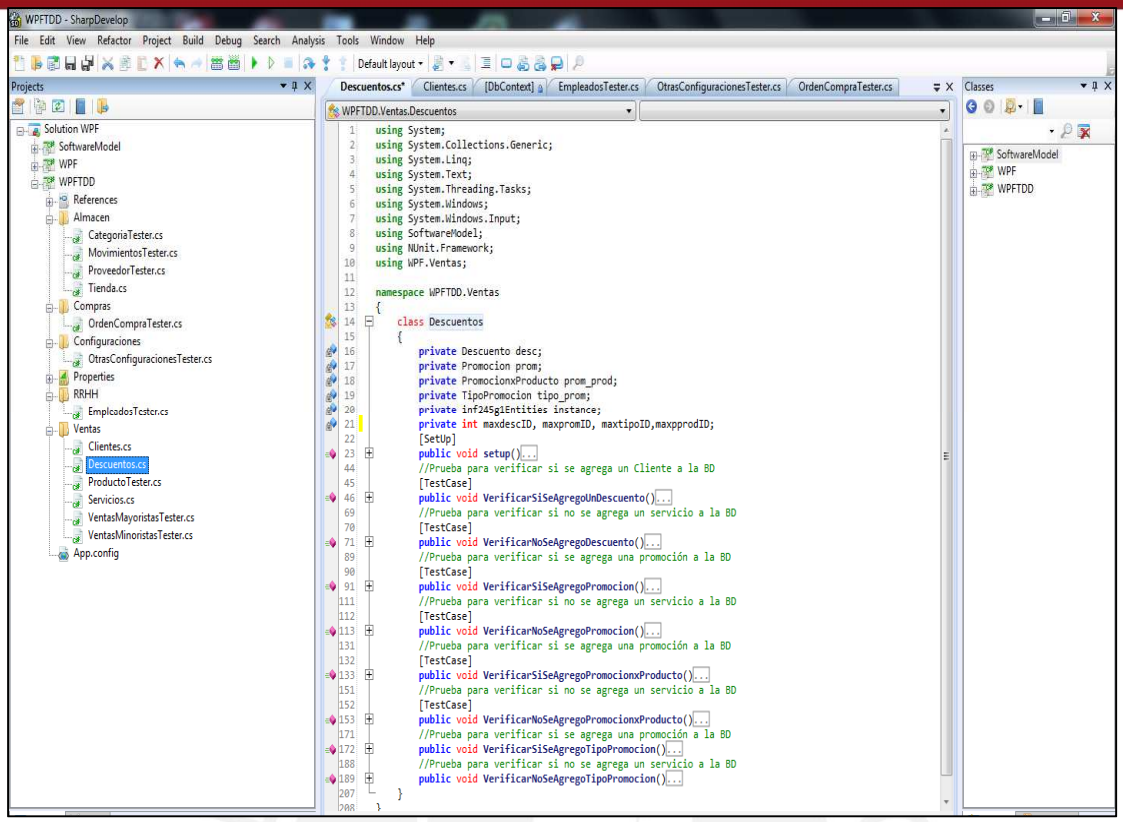

Imagen 6.9: Solución WPF – Ejemplo de TDD Ventas: Descuentos

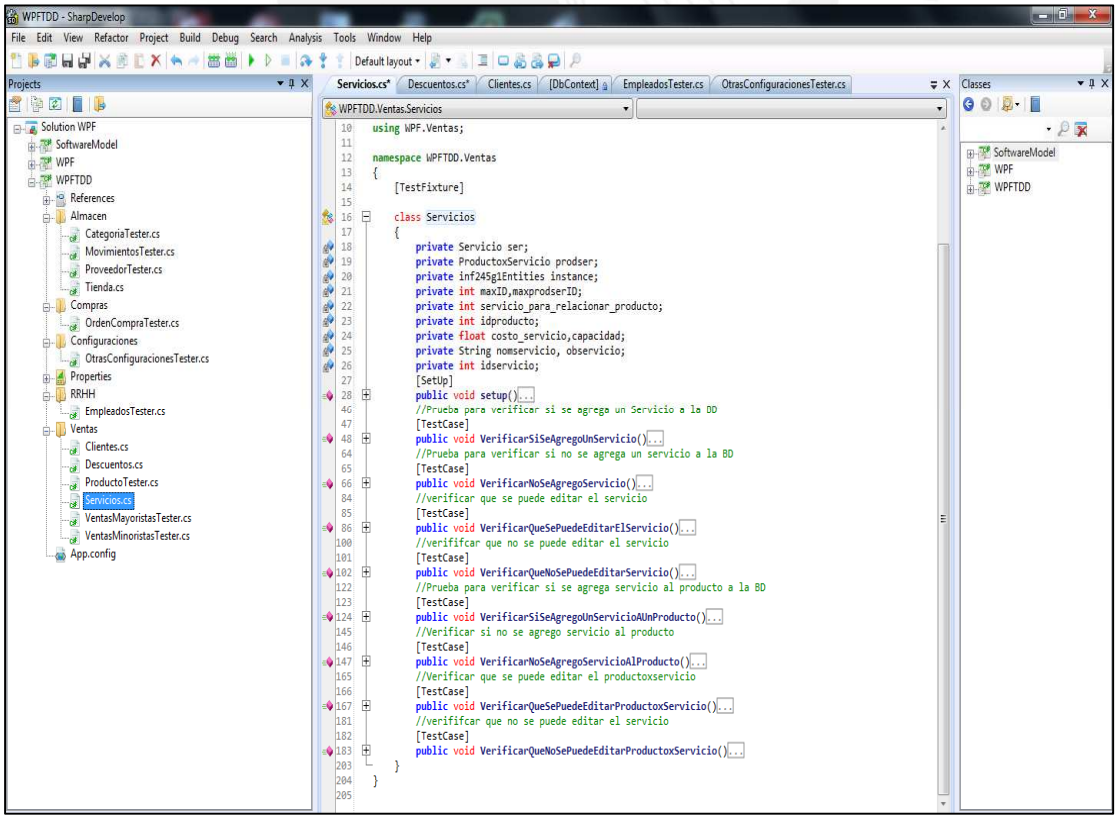

Imagen 6.10: Solución WPF – Ejemplo de TDD Ventas: Servicios

b. Grupo 2: Ferreting Soft

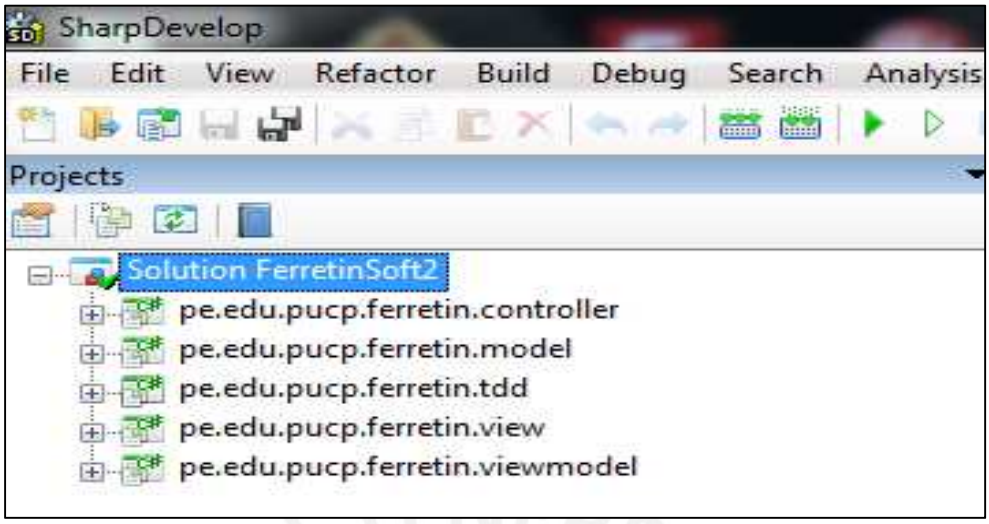

Imagen 6.11: Solución FerretinSoft2 de Ferreting Soft

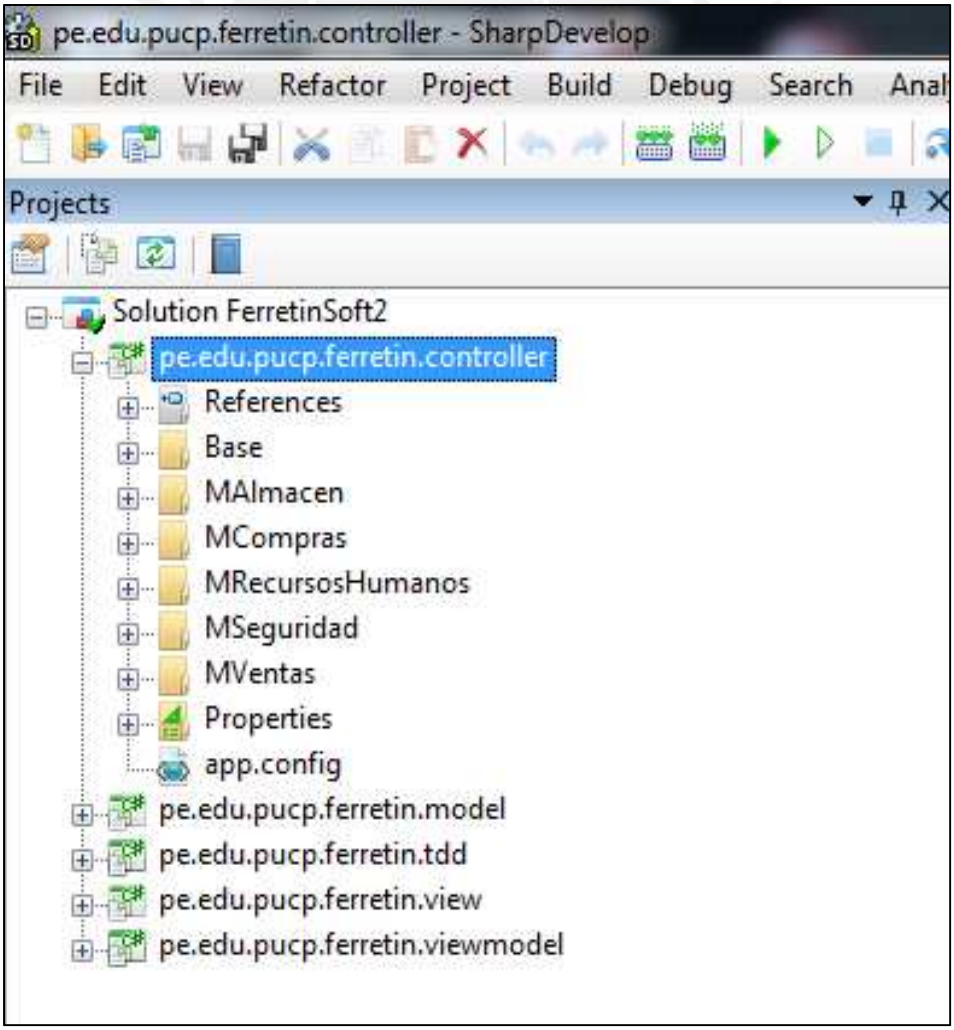

Imagen 6.12: Solución FerretinSoft2 – Implementación

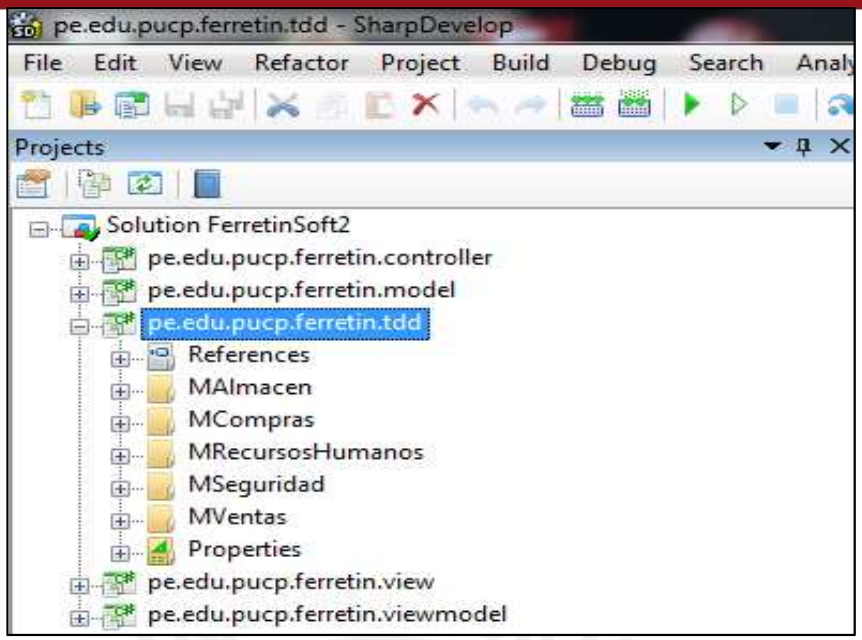

Imagen 6.13: Solución FerretinSoft2 – Implementación TDD

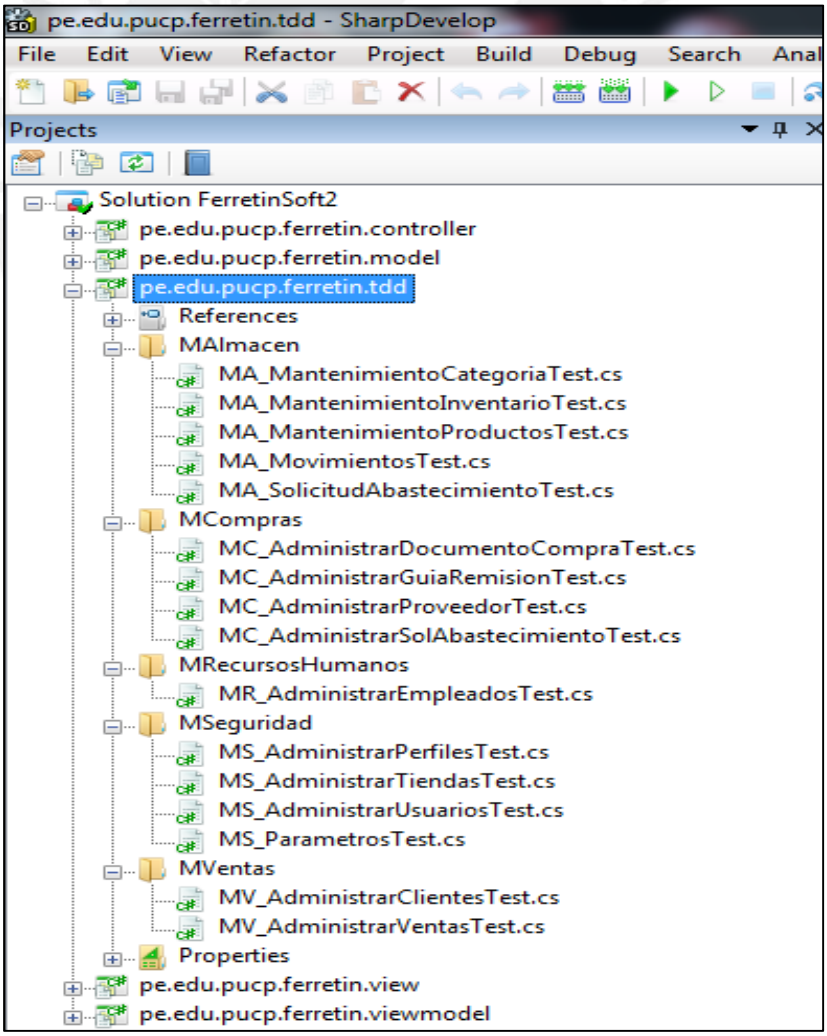

Imagen 6.14: Solución FerretinSoft2 – Ejemplo de codificación

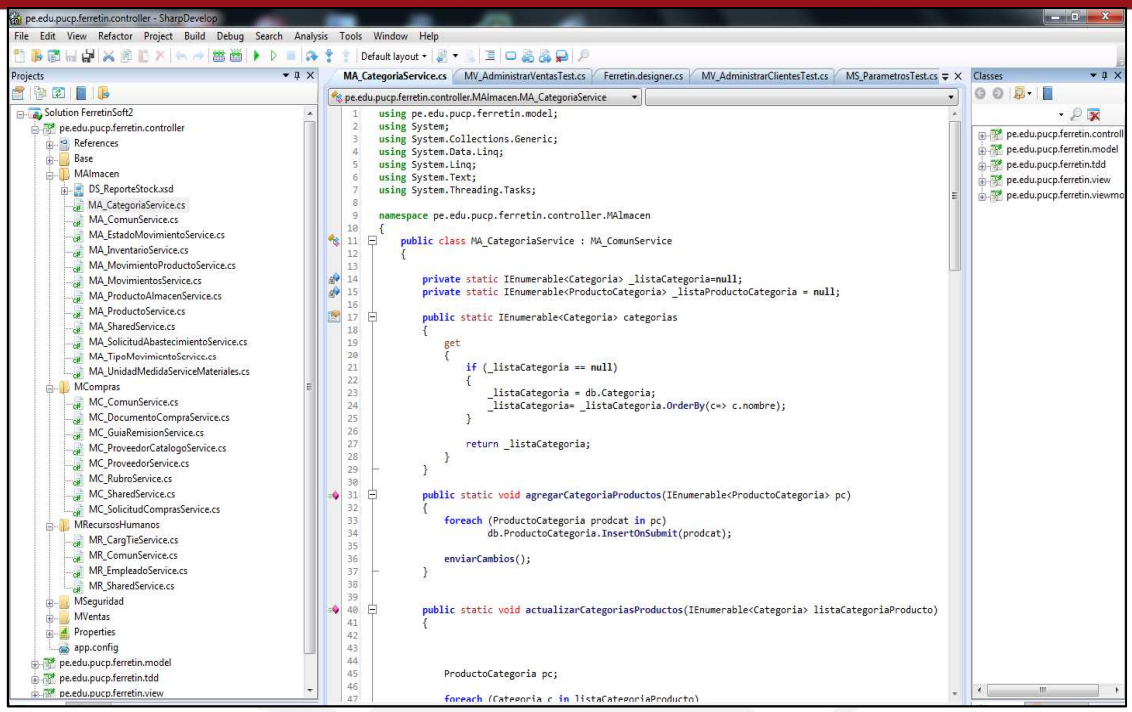

Imagen 6.15: Solución FerretinSoft2 – Ejemplo de TDD Almacén: Categoría

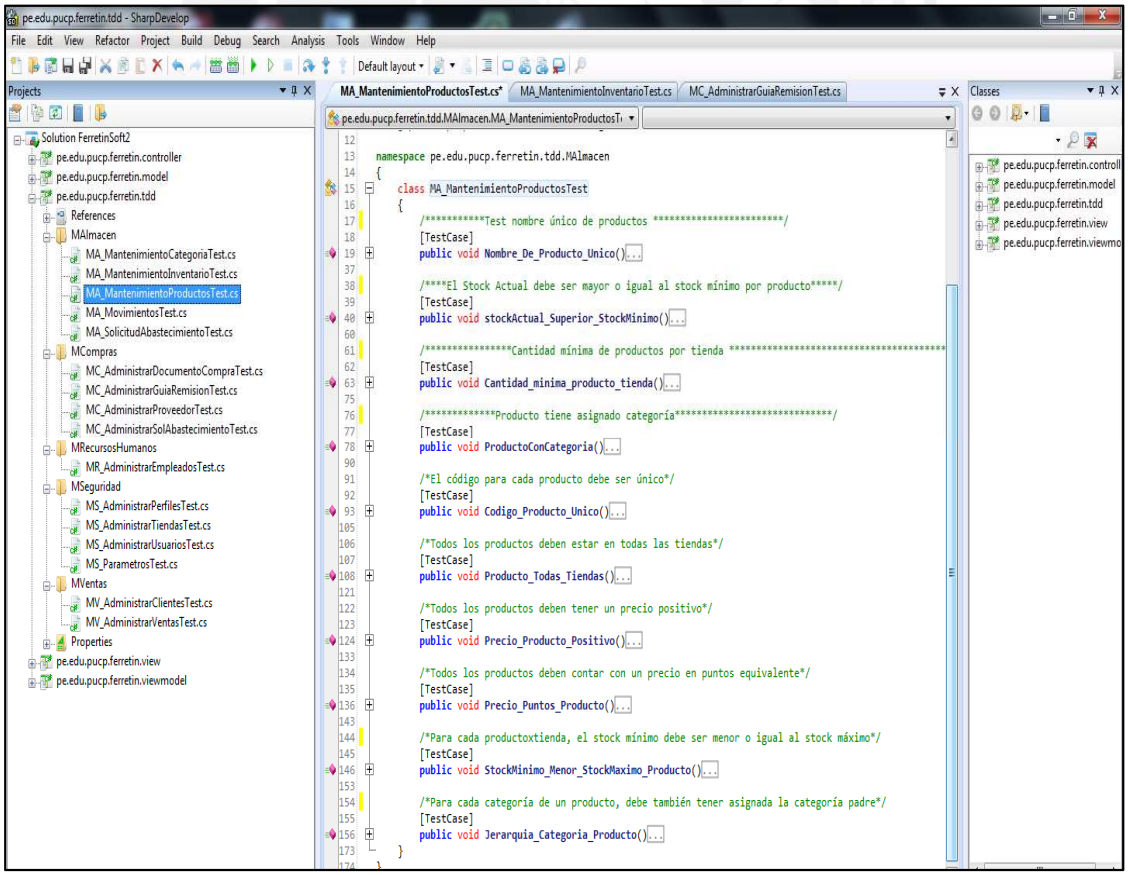

Imagen 6.16: Solución FerretinSoft2 – Ejemplo de TDD Almacén: Productos

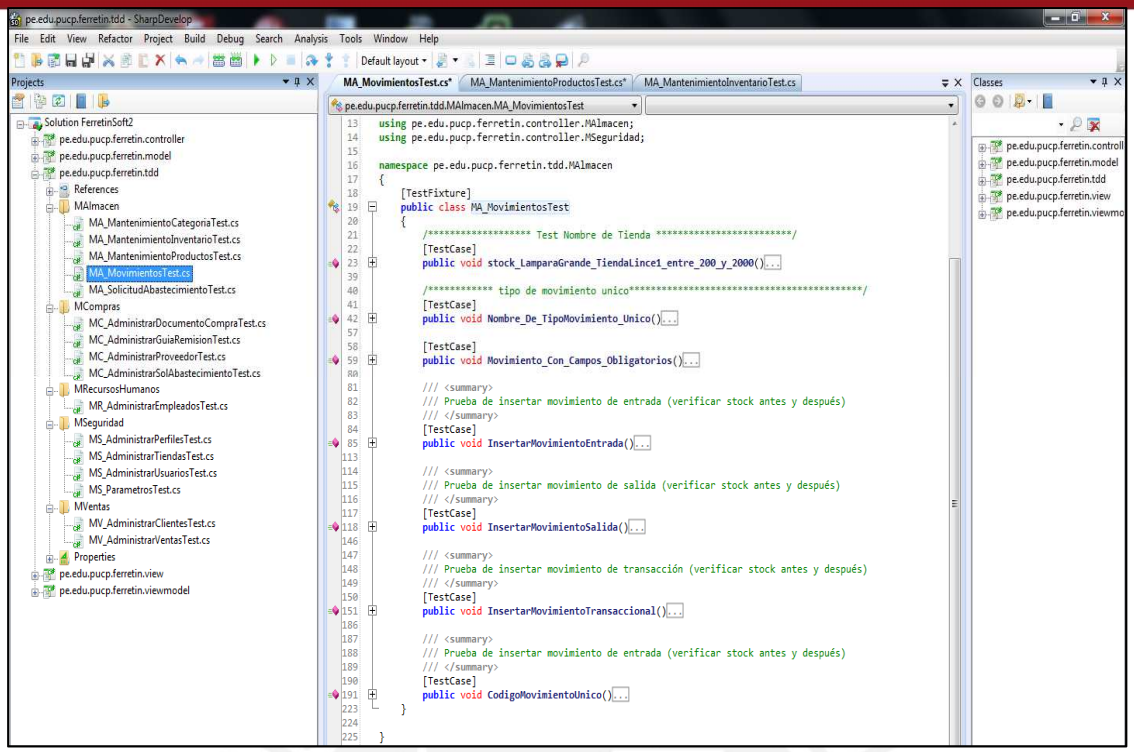

Imagen 6.17: Solución FerretinSoft2 – Ejemplo de TDD Almacén: Movimientos

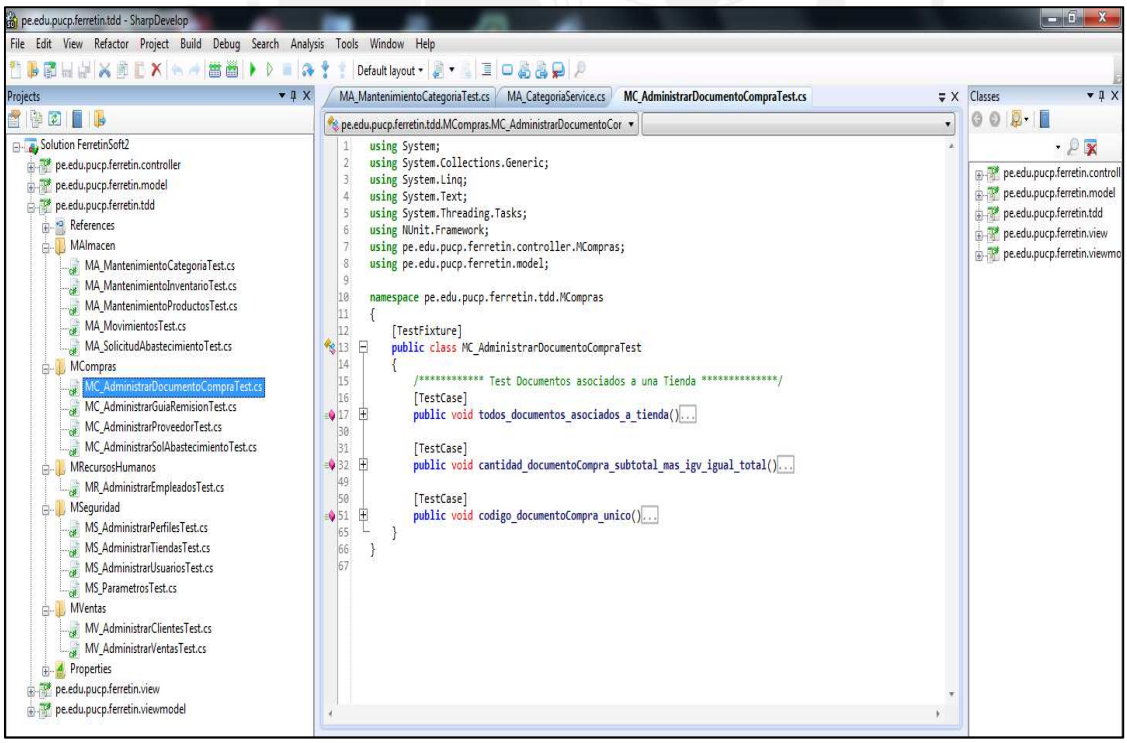

Imagen 6.18: Solución FerretinSoft2 – Ejemplo de TDD Compras: AdministrarDoc.

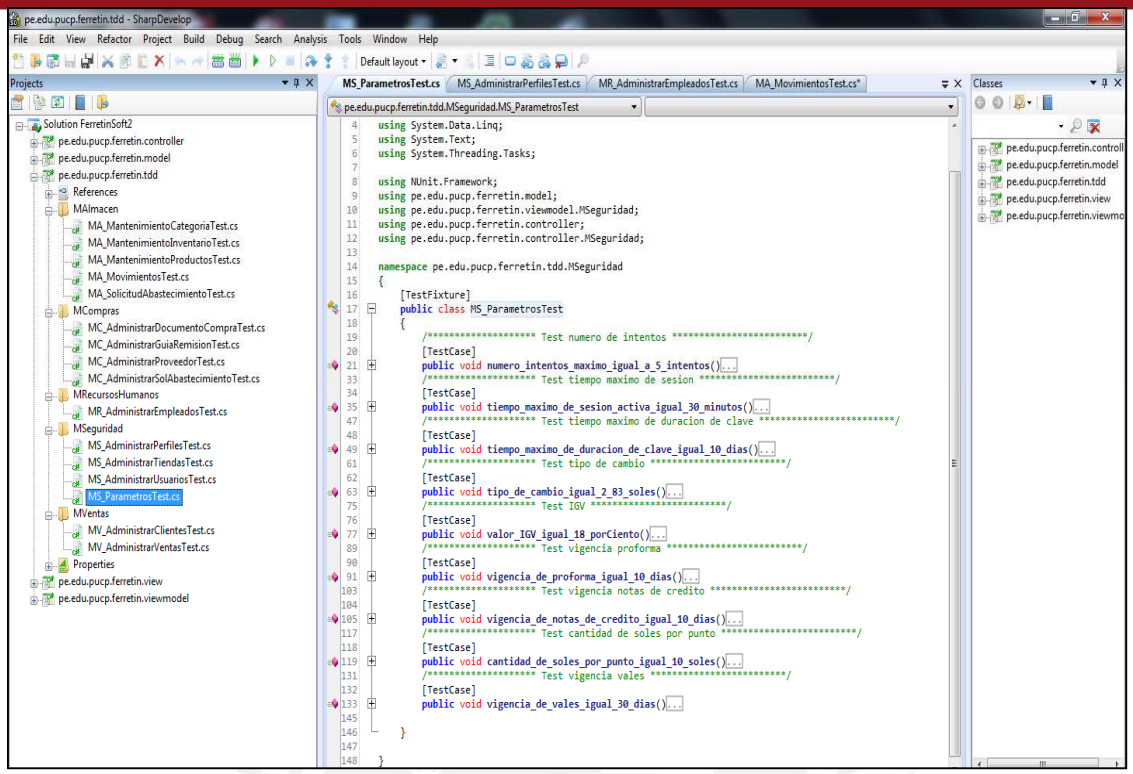

Imagen 6.19: Solución FerretinSoft2 – Ejemplo de TDD Seguridad: Parámetros

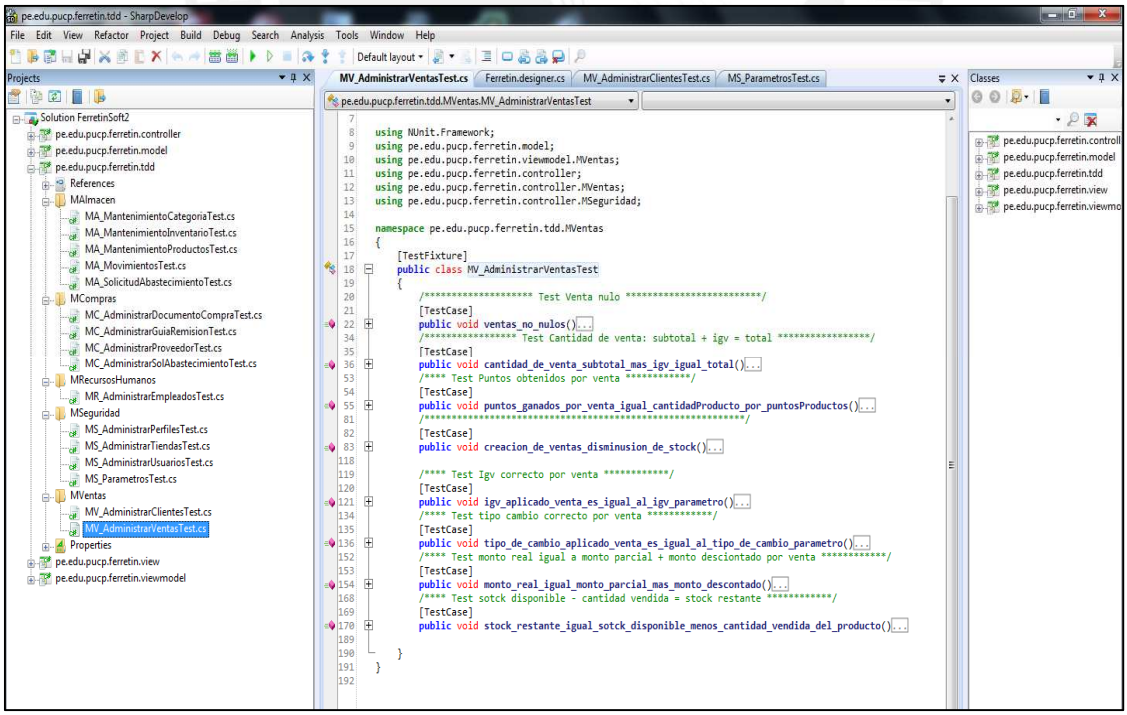

Imagen 6.20: Solución FerretinSoft2 – Ejemplo de TDD Ventas: AdministrarVentas