****

**Universidad EAFIT**

**Escuela de Ingenierías**

**Departamento de Informática y Sistemas**

**Especialización en Desarrollo de Software**

**Verificación y Validación**

**TRABAJO PRÁCTICO – VERIFICACIÓN**

**Presentado a**

*Maritza Ospina Rendón*

**Presentado por**

*Hans Linderman Parra Padilla*

*Juan David Cuartas León*

Medellín, Febrero 20 de 2012

**Tabla de Contenido**

[1. Introducción 3](#_Toc349159617)

[2. Plan de calidad 4](#_Toc349159618)

[3. Verificación de requisitos 5](#_Toc349159619)

[3.1 Lista de chequeo 5](#_Toc349159620)

[3.2 No conformidades 6](#_Toc349159621)

[4. Verificación de casos de uso 8](#_Toc349159622)

[4.1 Lista de chequeo 8](#_Toc349159623)

[4.2 No conformidades 9](#_Toc349159624)

[5. Verificación del diseño detallado 12](#_Toc349159625)

[5.1 Buenas prácticas (Diagramas de Clase) 12](#_Toc349159626)

[5.1.1 Principio Open-Close 12](#_Toc349159627)

[5.1.2 Principio de inversión de dependencia 12](#_Toc349159628)

[5.1.3 Principio de segregación de interfaces 12](#_Toc349159629)

[5.1.4 Principio de responsabilidad individual 13](#_Toc349159630)

[5.1.6 Diagramas 13](#_Toc349159631)

[5.1.7 Nombramiento 14](#_Toc349159632)

[5.2 Lista de chequeo 15](#_Toc349159633)

[5.3 No conformidades 16](#_Toc349159634)

[7. Conclusiones 17](#_Toc349159635)

[8. Referencias 18](#_Toc349159636)

# 1. Introducción

En este trabajo práctico, realizaremos una verificación inicial de los artefactos generados como parte del proyecto integrador BVC Móvil (Bolsa de Valores de Colombia en dispositivos móviles) de la Especialización en Desarrollo de Software.

Para esto se seleccionaron los siguientes artefactos: Requisitos, Casos de Uso y Diagramas de Clase. La verificación de los Requisitos y Casos de Uso se realizó sobre total de ítems elicitados, mientras que para el Diagrama de Clases, se seleccionó el diagrama del paquete principal del sistema: Portafolios.

Esperamos mediante esta práctica, hacer una detección temprana de los defectos que pudieran contener los artefactos generados, y así contribuir a asegurar la calidad del software y por ende, garantizar el éxito del proyecto.

# 2. Plan de calidad

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **REQUISITOS** | | | | | | | | |
| **Artefacto** | **Observaciones** | **C/NC** | **Fechas** | | | **Responsable artefacto** | **Inspección realizada por** | |
| **Planeada según cronograma** | **Entrega de artefacto** | **Ejecución** |
| Documento de requisitos | 34 ítems inspeccionados, fallan 12, cumple el 64.7% | NC | 8-Ago-2012 | 15-Ago-2012 | 16-Feb-2013 | Hans Parra | Juan Cuartas | |
|  |  |  |  |  |  |  |  | |
| **ANÁLISIS** | | | | | | | | |
| **Artefacto** | **Observaciones** | **C/NC** | **Fechas** | | | **Responsable artefacto** | | **Inspección realizada por** |
| **Planeada según cronograma** | **Entrega de artefacto** | **Ejecución** |
| Casos de uso | 24 ítems inspeccionados, fallan 8, cumple el 66.6% | NC | 3-Sept-2012 | 6-Sept-2012 | 17-Feb-2013 | Juan Cuartas | | Hans Parra |
|  |  |  |  |  |  |  | |  |
| **DISEÑO** | | | | | | | | |
| **Artefacto** | **Observaciones** | **C/NC** | **Fechas** | | | **Responsable artefacto** | | **Inspección realizada por** |
| **Planeada según cronograma** | **Entrega de artefacto** | **Ejecución** |
| Diagrama de clases | 35 ítems inspeccionados, fallan 5, cumple el 85,7% | NC | 1-Oct-2012 | 8-Oct-2012 | 18-Feb-2013 | Hans Parra Juan Cuartas | | Hans Parra Juan Cuartas |

# 3. Verificación de requisitos

La siguiente verificación se realizó sobre todos los requisitos del documento “BVCMóvil - Requisitos.html”.

## 3.1 Lista de chequeo

|  |  |  |  |
| --- | --- | --- | --- |
| **Propiedad** | **Descripción** | **C/NC** | **ID\_NC** |
| No ambigüedad | ¿Cada requisito funcional expresa una y solo una función que debe cumplir el sistema? | NC | 1.1 |
| No ambigüedad | ¿El requisito se interpreta igual al ser leído por más de una persona? | C |  |
| No ambigüedad | ¿Está la funcionalidad claramente definida? | C | 1.2 |
| No ambigüedad | ¿La redacción del requisito es simple y clara? | NC | 1.3 |
| Verificable | ¿Cada requisito puede ser medible o verificable? | C |  |
| Verificable | ¿Existen soportes que ayuden a comprender el requisito? | NC | 1.4 |
| Verificable | ¿Cada requisito tiene definida la fuente de información y fecha de especificación? | C |  |
| Completitud | ¿Se encuentran especificados los requisitos funcionales? | C |  |
| Completitud | ¿Se encuentran especificados los requisitos no funcionales? | C |  |
| Completitud | ¿Se encuentran especificadas las reglas de negocio? | NC | 1.5 |
| Completitud | ¿Se encuentran especificados los requisitos de información? | NC | 1.6 |
| Completitud | ¿No faltan requisitos? ¿Todo lo que se supone hace el sistema está incluido en la especificación? | C |  |
| Completitud | ¿Se encuentran identificadas las funcionalidades que involucran el requisito? | C |  |
| Completitud | ¿Están definidas las interfaces gráficas requeridas? | C |  |
| Completitud | ¿Están definidos los límites operativos de las funcionalidades? | C |  |
| Exactitud | ¿Todo requisito contribuye a satisfacer una necesidad real del usuario final del sistema? | C |  |
| Exactitud | ¿Están los acrónimos bien definidos? | C |  |
| Exactitud | ¿El documento se encuentra libre de errores ortográficos y gramaticales? | NC | 1.7 |
| Realizable | ¿El requisito es implementable? | C |  |
| Realizable | ¿Están los requisitos dentro del ámbito del proyecto? | C |  |
| Realizable | ¿Hay identificación de riesgos en los requisitos? | NC | 1.8 |
| Importancia | ¿Es necesario el requisito para el objetivo de la aplicación? | C |  |
| Importancia | ¿Es clara la importancia del requisito dentro de la organización? | C |  |
| Importancia | ¿La omisión del requisito provocaría una deficiencia en el sistema? | NC | 1.9 |
| Importancia | ¿Los requisitos cubren todas las necesidades del cliente, según el listado de necesidades explicado por el cliente? | C |  |
| Importancia | ¿Se encuentra incluida la prioridad de implementación de cada requisito? | C |  |
| Importancia | ¿Se estableció un orden jerárquico en la definición de los requisitos? | NC | 1.10 |
| Consistente | ¿No se encuentra el requisito duplicado o en conflicto con otro requisito? | C |  |
| Consistente | ¿El documento está bien organizado? | C |  |
| Consistente | ¿El esquema general es consistente con los requisitos de alto nivel? | C |  |
| Consistente | ¿Están las referencias o relaciones con otros requisitos bien definidas? | NC | 1.11 |
| Comprensible | ¿La audiencia a la que va dirigido el documento puede comprender los requisitos? | C |  |
| Trazable | ¿Se pueda rastrear hacia atrás y hacia adelante? | C |  |
| Independiente diseño | ¿No se especifica una determinada descomposición del sistema ni ningún aspecto de su posible implementación? | NC | 1.12 |

## 3.2 No conformidades

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID\_NC** | **Descripción** | **Clasificación** | **Fecha**  **Solución** | **Descripción**  **Solución** |
| 1.1 | El requisito **FRQ-0003 Administración de capital**, involucra 2 funcionalidades (depósitos y retiros), se debería pensar en separar estas dos operaciones. | Correctiva |  |  |
| 1.2 | * En el requisito **FRQ-0011 Administración de maestros**, no queda bien claro cuáles son los maestros que permitirá gestionar el sistema. * En el requisito **FRQ-0003 Administración de capital**, no queda bien claro que implica administrar el capital de cada portafolio. | Correctiva |  |  |
| 1.3 | Revisar la redacción del requisito **FRQ-0007 Administración de dividendos**, pues no es simple de entender. | Mejora |  |  |
| 1.4 | Se debería tener una guía del mercado de valores en Colombia, como soporte adicional a la documentación. | Mejora |  |  |
| 1.5 | No se definieron las reglas de negocio. | Correctiva |  |  |
| 1.6 | No se definieron los requisitos de información. | Correctiva |  |  |
| 1.7 | Revisar ortografía como tildes y redacción del documento, hay algunas frases complejas, buscar más simplicidad. | Mejora |  |  |
| 1.8 | Para el requisito **FRQ-0010 Actualización del precio de las acciones**, verificar que implicaciones legales tiene obtener la información del precio de las acciones desde la BVC, podría ser un riesgo para el proyecto que legalmente esta información no se pudiera utilizar para fines como los del aplicativo. | Preventiva |  |  |
| 1.9 | El requisito **FRQ-0009 Sincronización de información** fue clasificado como Funcional, pero debió haber sido clasificado como No Funcional, ya que el sistema podría seguir trabajando de forma local sin afectar su funcionamiento. | Correctiva |  |  |
| 1.10 | El caso de uso **FRQ-0010 Actualización del precio de las acciones**, debería aparecer de segundo en la lista de requisitos, ya que de esta acción depende que se realicen las operaciones de compra-venta de acciones en los dispositivos móviles. | Correctiva |  |  |
| 1.11 | Aunque se ha establecido trazas hacia los objetivos, no se especifica relaciones o dependencias entre requisitos, por ejemplo las transacciones compra y venta de acciones dependen del requisito **FRQ-0010 Actualización del precio de las acciones**. | Correctiva |  |  |
| 1.12 | El requisito **FRQ-0011 Administración de maestros**, hace alusión a que la interfaz de administración debe ser Web, esto es una especificación más de implementación. | Correctiva |  |  |

# 4. Verificación de casos de uso

La siguiente verificación se realizó sobre todos los casos de uso del documento “BVCMóvil - Requisitos.html”.

## 4.1 Lista de chequeo

|  |  |  |  |
| --- | --- | --- | --- |
| **Propiedad** | **Descripción** | **C/NC** | **ID\_NC** |
| Identificador | ¿Cada caso de uso tiene un identificador único y sigue el estándar establecido? | C |  |
| Nombre | ¿Está lleno el campo Nombre? Revise la estructura del nombre que escogió para cada caso de uso. Los nombres de los casos de uso deben ser escogidos del dominio del problema y ser verbos en forma imperativa. Deben ser formados por verbo + objeto + complemento (Ej. Inscribir usuario en el sistema). | NC | 2.1 |
| Autores | ¿Cada caso de uso tiene definido el autor y la fecha de especificación? | C |  |
| Fuentes | ¿Cada caso de uso tiene definida la fuente de información? | C |  |
| Dependencias | ¿Está lleno el campo Dependencias? Si no hay dependencias, se debe indicar con Ninguno. | C |  |
| Descripción | ¿Está lleno el campo Descripción? La descripción debe ser clara y sin ambigüedades. Las salidas del caso de uso deben ser claras y tangibles. Estas deben describir lo que el actor obtiene después de que el caso de uso termina. | NC | 2.2 |
| Actores | ¿Se han identificado todos los actores que intervienen en el caso de uso? | C |  |
| Actores | ¿Para cada caso de uso hay implicado al menos un actor? ¿Los actores indicados corresponden a los definidos? | C |  |
| Actores | ¿Los actores se definen como roles y no como personas, cargos o usuarios? |  |  |
| Precondición | ¿Está lleno el campo Precondición? Las precondiciones deben ser claras con respecto a la información usada en el caso de uso. Estas deben establecer claramente las condiciones que deben cumplirse antes de que el caso de uso se ejecute. | NC | 2.3 |
| Secuencia normal | ¿Está lleno el campo Secuencia normal? Todos los eventos en cada paso deben estar numerados. En cada paso debe estar claro quién hace que (el actor o el sistema). El curso de los eventos debe corresponder a la descripción del caso de uso. | NC | 2.4 |
| Secuencia normal | ¿Todo caso de uso es iniciado por un actor? | NC | 2.5 |
| Secuencia normal | ¿Sólo existe un flujo básico? | C |  |
| Secuencia normal | ¿Todos los casos de uso comienzan y terminan? | C |  |
| Secuencia normal | ¿Si un caso de uso es muy complejo éste está explicado a través de un diagrama de actividades? | NC | 2.6 |
| Secuencia normal | ¿No se tomaron decisiones de diseño en la definición de los casos de uso? |  |  |
| Puntos de Extensión | ¿Está lleno el campo Puntos de Extensión? Los puntos de extensión deben ser identificados con respecto al curso natural de eventos. Estos deben tener su link respectivo para referenciar a los casos de uso que extiende. ¿El link al caso de uso sirve y referencia el caso de uso que es? Si no hay puntos de extensión, se debe indicar con las siglas N/A. | C |  |
| Puntos de Extensión | ¿No existen pasos repetidos entre un caso de uso y otro caso de uso extendido? | C |  |
| Postcondición | ¿Está lleno el campo Postcondición? Las postcondiciones deben ser claras con respecto a la información usada en el caso de uso. Estas deben establecer claramente las condiciones que deben cumplirse después de que el caso de uso se ejecute. | NC | 2.7 |
| Excepciones | ¿Están documentadas todas las excepciones conocidas de los casos de uso? ¿Están documentados los flujos alternos de cada caso de uso? | C |  |
| Importancia | ¿Está lleno el campo Importancia? ¿Si está lleno, sigue el estándar definido (Solo puede llenarse con Vital, Importante o Quedaría bien) | C |  |
| Urgencia | ¿Está lleno el campo Urgencia? ¿Si está lleno, sigue el estándar definido (Solo puede llenarse con Inmediatamente, Hay presión o Puede esperar)? | C |  |
| Include | ¿Los "include" son mencionados dentro del flujo del caso de uso donde son incluidos? | C |  |
| Extend | ¿Los "extend" son mostrados en el flujo básico como posibilidades del sistema y se visualizan en el diagrama? | NC | 2.8 |

## 4.2 No conformidades

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID\_NC** | **Descripción** | **Clasificación** | **Fecha**  **Solución** | **Descripción**  **Solución** |
| 2.1 | El caso de uso **UC-0012 Sincronización de información**, no está escrito en infinitivo. | Mejora |  |  |
| 2.2 | Algunas descripciones están muy básicas, expresando casi lo mismo que el nombre del caso de uso, o algunas se limitan a decir que el caso de uso se ejecuta cuando el usuario hace clic en cierto botón. Se debe dar un poco más de detalle. Adicionalmente, en ninguna descripción se menciona la salida del caso de uso, aunque si se está empleando el uso de las postcondiciones. | Mejora |  |  |
| 2.3 | No se especificó la precondición para los siguientes casos de uso:   * **UC-0001 Administrar mercados** * **UC-0002 Administrar acciones** * **UC-0003 Administrar comisionistas** | Correctiva |  |  |
| 2.4 | El flujo de los siguientes casos de uso es demasiado simple y no explica con suficiente claridad el comportamiento del sistema:   * **UC-0016 Actualizar valores de mercado** * **UC-0012 Sincronización de información** | Correctiva |  |  |
| 2.5 | Los siguientes casos de uso aparecen iniciados por el sistema, deberían ser iniciados por el actor del caso de uso:   * **UC-0011 Generar balance** * **UC-0007 Colocar orden de compra** * **UC-0014 Colocar orden de venta** * **UC-0009 Enrepar acciones** * **UC-0015 Actualizar saldo portafolio** * **UC-0016 Actualizar valores de mercado** | Correctiva |  |  |
| 2.6 | Los siguientes casos de uso son muy complejos y deberían definir un diagrama de secuencia para mejor compresión:   * **UC-0007 Colocar orden de compra** * **UC-0014 Colocar orden de venta** | Mejora |  |  |
| 2.7 | Algunas postcondiciones no están bien explicadas:   * **UC-0005 Depositar efectivo**: Debería indicar que el saldo del portafolio aumenta en la cantidad depositada. * **UC-0006 Retirar efectivo**: Debería indicar que el saldo del portafolio disminuye en la cantidad retirada. * **UC-0009 Enrepar acciones**: La Postcondición no indica nada útil. * **UC-0010 Agregar dividendos**: No se entiende bien la postcondición, es similar al nombre del caso de uso. | Correctiva |  |  |
| 2.8 | Los casos de uso **UC-0017 Procesar orden de compra** y **UC-0018 Procesar orden de venta**, no hacen mención a la extensión del caso de uso “Actualizar Saldo Portafolio”, tal como aparece reflejado en los diagramas de casos de uso. | Correctiva |  |  |

# 5. Verificación del diseño detallado

La siguiente verificación se realizó sobre el diagrama de clases del paquete Portafolio “BVCMóvil - Diagrama de clases módulo portafolio.jpg”, el cual define todas las clases involucradas con el proceso de gestión de portafolios de los inversionistas en el aplicativo de la Bolsa de Valores de Colombia.

## 5.1 Buenas prácticas (Diagramas de Clase)

### 5.1.1 Principio Open-Close

Las entidades de software como las clases, módulos y funciones deben estar abiertas a ser extendidas y cerradas a ser modificadas. Esto garantiza que el software sigue teniendo el mismo comportamiento cuando se implementan cambios. Se debe considerar al escribir una clase que cuando se necesite extender su comportamiento no se modifique, sino que se extienda. Esto se puede asegurar mediante el uso de clases abstractas y clases concretas para la implementación del comportamiento. Algunos casos particulares de este principio son el patrón Template y el patrón Strategy.

### 5.1.2 Principio de inversión de dependencia

* Los módulos de alto nivel no deben depender de módulos de bajo nivel, ambos deben depender de abstracciones.
* Las abstracciones no deben depender de los detalles. Los detalles deben depender de las abstracciones.

Este principio indica que debemos desacoplar los módulos de alto nivel de los módulos de bajo nivel introduciendo una capa de abstracción entre las clases de alto nivel y las de bajo nivel. Esto permite invertir la dependencia: En vez de escribir nuestras abstracciones basadas en detalles, escribimos nuestros detalles basados en abstracciones.

La inversión de dependencia se refiere a la forma en que las dependencias son realizadas. Normalmente cuando un módulo de software (clase o framework) necesita algún otro módulo, este lo inicializa y crea una referencia directa a este, lo cual provocará 2 módulos acoplados. Para desacoplarlos, el primer módulo deberá proporcionar un gancho (una propiedad o un parámetro) y un módulo externo que controla las dependencias inyectará la referencia en el otro módulo.

Aplicando este principio, los módulos pueden ser cambiados por otros módulos, simplemente cambiando el módulo de dependencia. Patrones como el Factory, o el Abstract Factory pueden ser utilizados como frameworks de dependencia.

### 5.1.3 Principio de segregación de interfaces

Este principio indica que los clientes no deben ser forzados a depender de interfaces que no utilizan. Cuando escribimos nuestras interfaces, debemos tener cuidado en agregar únicamente los métodos que deberían estar en ésta, de lo contrario, las clases que implementan la interface se verían obligadas a implementar todos los métodos que define, así no los necesite.

En conclusión, las interfaces que contienen métodos que no son específicos a estas se denominan interfaces contaminadas y deben ser evitadas.

### 5.1.4 Principio de responsabilidad individual

Una clase debe tener una única razón para cambiar. Este principio establece que si tenemos dos razones para cambiar una clase, se debería dividir la funcionalidad en dos clases. Cada clase se ocupará de una única responsabilidad, y en el futuro, si se necesita realizar un cambio, se realizará en la clase responsable. Cuando se debe realizar un cambio en una clase que tiene más responsabilidades, el cambio puede afectar la funcionalidad de otras clases.

1. Menos es más: Diagramas muy largos que contienen muchos elementos proveen menos información que diagramas pequeños pero enfocados. Cuando se lee un diagrama muy largo, la audiencia no sabrá en donde enfocarse, y debido a que hay muchos elementos a los que prestar atención, la audiencia intentará pasar a otro diagrama. Algunos sostienen que al imprimir el diagrama en una hoja A4, este debería poder ser legible.

### 5.1.6 Diagramas

#### 5.1.6.1 Menos es más

#### Diagramas muy largos que contienen muchos elementos proveen menos información que diagramas pequeños pero enfocados. Cuando se lee un diagrama muy largo, la audiencia no sabrá en donde enfocarse, y debido a que hay muchos elementos a los que prestar atención, la audiencia intentará pasar a otro diagrama. Algunos sostienen que al imprimir el diagrama en una hoja A4, este debería poder ser legible.

#### 5.1.6.2 No cruces

Esta es una regla bastante común y bien conocida. Trate de evitar que cualquier par de líneas en el diagrama se crucen, esto permitirá que el diagrama sea más leíble y entendible. Sin embargo, si no es posible evitar el cruce de líneas es indicación de lo siguiente:

* Hay muchos elementos en el diagrama, intente aplicar la regla "Menos es más"
* Hay algo malo en el flujo del modelo, por alguna razón los modelos bien diseñados no tienen problemas de líneas cruzadas.

#### 5.1.6.3 Ortogonalidad

Parce algo tonto, pero las líneas de los diagramas se deberían dibujar únicamente horizontal o verticalmente. Cuando una línea se quiebra, debería quedar en ángulo recto. Nunca dibuje líneas ortogonales. Esto hace lucir mejor los diagramas instantáneamente. Excepciones a esta regla, son los conectores a notas (links) y las relaciones entre los casos de uso.

#### 5.1.6.4 Padres arriba

Cuando se dibujen jerarquías de generalización o realización en un diagrama, siempre asegúrese de que los elementos padre estén más arriba que los elementos hijos de modo que las flechas apunten hacia arriba. Esta regla es muy intuitiva para la mayoría de diseñadores UML. Los diagramas que no siguen esta regla son más difíciles de leer y por alguna razón, toma un esfuerzo mental mayor el entendimiento del diagrama. En caso que se tengan múltiples elementos descendiendo del mismo padre, es recomendable mostrar la jerarquía en un estilo de árbol vertical.

#### 5.1.6.5 Orden

Si se quiere mostrar el mensaje de que el análisis es bueno a través del diagrama diseñado y que resolverá los problemas de los stackeholders, se deberá presentar un diagrama limpio y ordenado. Algunos tips para ordenar:

* Alinear los elementos por uno de sus lados, o por sus centros
* Haga los elementos del mismo tamaño cuando sea posible

### 5.1.7 Nombramiento

* Los nombres de las clases o interfaces deben describir lo que sus instancias realizan. Un buen nombre de clase generalmente es un sustantivo en singular, a veces acompañado de un adjetivo, mientras que los adjetivos son más apropiados para nombrar interfaces.
* Los nombres de las clases abstractas y las interfaces no deben contener referencias a una implementación en particular, aunque se debe considerar esto en cada contexto. Por ejemplo una interface XmlParser podría parecer muy particular para el tratamiento de archivos Xml, pero a su vez podría ser general para un conjunto de funcionalidades heredadas como HtmlParser, YamlParser, etc.
* Las clases o interfaces deben estar contenidas en el paquete o espacio de nombres correcto. Esto se puede validar a partir del número de dependencias que introduce cada artefacto.
* Las convenciones de nombramiento y mejores prácticas también deben ser válidos para nombres de métodos, parámetros, variables locales, comentarios, etc.
* El nombre de las clases e interfaces deben ser consistentes con el resto del modelo de objetos, se debe identificar un nombre válido inmediatamente es creada la clase en el modelo.
* Cuando se especifican patrones de diseño, el nombramiento de las clases e interfaces debe estar influenciado por la jerga del patrón, con el fin de que el nombre comunique acerca de la estructura del código.

## 5.2 Lista de chequeo

|  |  |  |  |
| --- | --- | --- | --- |
| **Propiedad** | **Descripción** | **C/NC** | **ID\_NC** |
| Open-Close | ¿Se definieron clases abstractas para entidades con comportamiento similar? | C |  |
| Inversión de dependencia | ¿Se evitó el acoplamiento entre módulos de alto nivel con módulos de bajo nivel? | NA |  |
| Segregación de interface | ¿Cada interface define únicamente los métodos que debería contener? | NA |  |
| Responsabilidad individual | ¿Cada clase tiene una única responsabilidad? | C |  |
| Diagrama | ¿Los diagramas no son muy largos y son fáciles de entender? | C |  |
| Diagrama | ¿Cada diagrama está enfocado en un aspecto particular del software? | NC | 3.1 |
| Diagrama | ¿Al imprimir el diagrama en una hoja A4, sigue siendo legible a simple vista? | C |  |
| Diagrama | ¿Se evitaron los conectores cruzados en los diagramas? | C |  |
| Diagrama | ¿Se evitaron las líneas ortogonales en los diagramas? | NC | 3.2 |
| Diagrama | ¿Las jerarquías se diseñaron top-down (padres arriba, hijos abajo)? | C |  |
| Diagrama | ¿Para múltiples elementos que heredan de una misma clase se aplicó estilo de árbol vertical? | C |  |
| Diagrama | ¿Todos los diagramas están ordenados? | C |  |
| Diagrama | ¿Se aplicó alineación entre los elementos del diagrama? | C |  |
| Diagrama | ¿Se dibujaron los elementos de los diagramas del mismo tamaño cuando es posible? | C |  |
| Nombramiento | ¿Los nombres de las clases o interfaces son descriptivos? | C |  |
| Nombramiento | ¿Los nombres de las clases o interfaces están definidos en singular? | C |  |
| Nombramiento | ¿Los nombres de las clases o interfaces son consistentes con el modelo de negocio? | C |  |
| Nombramiento | ¿Se evitó el uso de referencias de implementación en los nombres de las clases abstractas o interfaces? | C |  |
| Nombramiento | ¿Cada clase o interface está contenida en el paquete correcto? | NC | 3.3 |
| Nombramiento | ¿Los paquetes tienen bajo acoplamiento? | NC | 3.4 |
| Nombramiento | ¿Los nombres de atributos o métodos siguen las convenciones de nombramiento? | C |  |
| Nombramiento | ¿Los nombres de las clases contenidas en los patrones de diseño siguen la jerga del patrón? | NA |  |
| Completitud | ¿Las clases más importantes están descritas y justificadas? | NC | 3.5 |
| Completitud | ¿Implementa el diseño todos los requisitos explícitos e implícitos del sistema? | C |  |
| Completitud | ¿Los principales objetivos del sistema están plasmados en el diseño? | C |  |
| Atributos | ¿Las clases tienen definidos los atributos? | C |  |
| Atributos | ¿Los atributos tienen nombres significativos? | C |  |
| Atributos | ¿Los atributos tienen definido el tipo de dato? | C |  |
| Atributos | ¿Los atributos tienen definido el tipo de visibilidad? | C |  |
| Relaciones | ¿Las relaciones entre las clases son lógicas y correctas? | C |  |
| Relaciones | ¿Las relaciones entre las clases tienen definida la multiplicidad? | C |  |
| Correcto | ¿El diseño es correcto y no ambiguo? | C |  |
| Modular | ¿El diseño es modular e incorpora reusabilidad de componentes? | C |  |
| Independencia | ¿Es el diseño de alto nivel independiente de la máquina y el lenguaje que se utilizará para ponerlo en práctica? | C |  |
| Herramientas | ¿Se utilizaron herramientas de modelado UML? | C |  |

## 5.3 No conformidades

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID\_NC** | **Descripción** | **Clasificación** | **Fecha**  **Solución** | **Descripción**  **Solución** |
| 3.1 | Se podría pensar en dividir el diagrama del paquete Portafolio en dos diagramas: Uno para la parte de transacciones, y el otro para la parte de portafolios. |  |  |  |
| 3.2 | Arreglar los conectores de herencia que están ortogonales para que aparezcan verticales. |  |  |  |
| 3.3 | Tal como se mencionó en el punto 3.1, se podría pensar en dividir el paquete Portafolio en dos paquetes: Uno para las transacciones, y otro para los portafolios. |  |  |  |
| 3.4 | Parece haber mucho acoplamiento entre el paquete Portafolio y el paquete Usuario, pues hay varias clases del paquete Portafolio dependiendo del paquete Usuario, revisar como se podría disminuir este acoplamiento. |  |  |  |
| 3.5 | Aunque el diagrama es claro, hace falta una explicación y/o justificación de cada clase. |  |  |  |

# 7. Conclusiones

Como se pudo evidenciar en las no conformidades, es necesario hacer ajustes en los artefactos generados, principalmente en el documento de requisitos, y en los casos de uso. Es importante corregir estos defectos, ya que podrían acarrear problemas en las etapas subsecuentes del ciclo del proyecto.

La detección temprana de errores en los requisitos significa un ahorro en costos en etapas posteriores del proceso, ya que a medida que avanzan las fases del ciclo de desarrollo, corregir un error resulta mucho más costoso.

# 8. Referencias

1. Material del curso: Calidad en requisitos, Técnicas de Verificación, Revisiones Pares
2. Material del curso: Inspecciones en Casos de uso, Inspecciones en Diseño, Historias de Usuario.
3. David Choque & Carla Choque, "Checklist, Referencia a la norma: Estándar IEEE - 730", <http://es.scribd.com/doc/55686224/Lista-de-Chequeo-Requerimientos>.
4. Univalle, "Lista de chequeo para la validación de requerimientos", <http://eisc.univalle.edu.co/cursos/web/material/750092M/80/8-Matriz_ValidacionRequerimientos.pdf>.
5. Erick Escorcia, Rafael Rueda, "Lista de Chequeo de casos de Uso", <http://qualdev.uniandes.edu.co/wikiDev/doku.php?id=development:resources:checklists:inspectionreq:checklistusecases>.
6. Gabriel Villalobos Molina, "Diseño sistema informático de las tareas de desarrollo y aseguramiento de la calidad en proyectos informáticos para la empresa Absolutec S.A.", <http://www.uci.ac.cr/Biblioteca/Tesis/PFGMAP697.pdf>.
7. Giornio Sironi, "The class design checklist", <http://www.giorgiosironi.com/2010/04/class-design-checklist.html>.
8. Robert Gowland, "High Level Design Checklist", <http://www.gowland.ca/high-level-design-checklist>.
9. Object Oriented Design, "Designs Principes", <http://www.oodesign.com/design-principles.html>.
10. Geert Bellekens, "UML Best Practice: 5 rules for better UML diagrams", <http://geertbellekens.wordpress.com/2012/02/21/uml-best-practice-5-rules-for-better-uml-diagrams>.