PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

"INTEGRACIÓN DE ORIENTACIÓN A ASPECTOS EN PROYECTO DESARROLLADO EN ORIENTACIÓN A OBJETOS"

CARMEN GLORIA MOLINA VILLALOBOS

TESIS DE GRADO
MAGÍSTER EN INGENIERÍA INFORMÁTICA

Diciembre 2007

Pontificia Universidad Católica de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Informática

"INTEGRACIÓN DE ORIENTACIÓN A ASPECTOS EN PROYECTO DESARROLLADO EN ORIENTACIÓN A OBJETOS"

CARMEN GLORIA MOLINA VILLALOBOS

Profesor Guía: **SILVANA RONCAGLIOLO DE LA HORRA**

Programa: **Magíster en Ingeniería Informática**

Diciembre 2007

Integración de Orientación a Aspectos en proyecto desarrollado en Orientación a Objetos

Carmen Gloria Molina, Escuela de *Ingeniera Informática, PUCV*

***Resumen*— El presente trabajo es una investigación de los beneficios obtenidos al incorporar Orientación a Aspectos en un proyecto desarrollado en Orientación a Objetos. En un módulo de un proyecto en particular se utilizó en las etapas de análisis, diseño y codificación Orientación a Objetos sin incluir orientación a aspectos; y posteriormente en el mismo módulo se desarrollaron las mismas etapas pero incluyendo Orientación a Aspectos. Para la medición del beneficio de la incorporación de Orientación a Aspectos se realizaron encuestas a profesionales de informática con preguntas fijas que permitiesen dar una medición del beneficio. Junto a estas encuestas se utilizó además fórmulas que permitieron medir los beneficios en el diseño y codificación al incorporar Orientación a Aspectos.**

***Palabras Claves*—Orientación a Objetos, Orientación a Aspectos, Asuntos Transversales, Puntos de Corte, Separación de Asuntos**

I. Introducción

ESTE documento contiene una investigación de los beneficios obtenidos al incorporar orientación a aspectos a un proyecto con orientación a objeto.

Actualmente se ha incorporado a los proyectos la orientación a objeto lo que permite mejorar la construcción de los distintos sistemas. Pero la orientación a objeto no tiene un nivel de abstracción total lo que lleva a que exista código duplicado y mezclado dentro de los sistemas. Esto ocurre debido a que un problema o un requerimiento involucra varios aspectos (ejemplo: sincronización, coordinación, etc.), es por esto que se recomienda identificar y separar estos aspectos o asuntos, a esto se le denomina *separación de asuntos*.

Las principales características que aporta la orientación a aspectos son: flexibilidad, adaptabilidad y reusabilidad de los elementos que componen el sistema. La orientación a aspectos guía la forma de expresar, aislar y reusar estos asuntos en el desarrollo de los sistemas.

Los estudios realizados a nivel mundial de orientación a aspectos se iniciaron con la forma de implementación y codificación de aspectos, uno de los lenguajes más utilizados es AspectJ, el cual permite incorporar aspectos en un código de Java. En los últimos años se ha incluido a las investigaciones la incorporación de orientación de aspectos en las etapas de análisis y diseño de un proyecto.

Al utilizar orientación a aspectos se reduce el tiempo de correcciones por cambios inesperados en código que corresponde a un aspecto, ya que se cuenta con una identificación temprana de los aspectos.

La hipótesis de esta investigación dice que si en un proyecto informático con orientación a objeto se le incluye e integra orientación a aspectos en sus etapas de análisis, diseño y construcción, se obtiene como resultado beneficios en el producto de software final del proyecto.

II. Objetivos

A. *Objetivo principal y objetivos específicos*

Realizar un estudio de los beneficios obtenidos al integrar la Orientación a Aspectos con Orientación a Objetos.

Para realizar este estudio se propone ejecutar las etapas de análisis, diseño y construcción de un módulo en un proyecto de software seleccionado para esta investigación; comparar los casos de uso en los que no se incluyó Orientación a Aspectos con los casos de uso en que si fueron incluidos. Con lo anterior verificar los beneficios obtenidos de esta forma realzar la importancia de incorporar en el área informática el uso de Orientación a Aspectos

Los objetivos específicos de este trabajo son los que se nombran a continuación:

- Comprobar si ingenieros con conocimientos en Orientación a Objetos perciben diferencias y beneficios al incorporar Orientación a Aspectos en las etapas de análisis y diseño de un proyecto de software.

- Estudiar, seleccionar y analizar resultados de mediciones que permitan cuantificar los beneficios en un proyecto de software integrando Orientación a Objetos con Orientación a Aspectos.

- Promover el uso de Orientación a Aspectos en todas las etapas de los proyectos informáticos.

B. *Descripción de proyecto a utilizar para medición*

El proyecto escogido para realizar la medición de incorporación de orientación a aspecto corresponde a un proyecto para el Servicio Nacional de Aduanas, enfocándose en análisis y diseño.

Este proyecto corresponde al control de las Declaraciones de Importación y Pago Simultáneo de Viajeros (DIPS Viajeros). Estas operaciones corresponden al registro de ingreso de mercancías al país traídas por los viajeros que llegan al país. Este registro debe ser rápido y ágil debido a que es un procedimiento sencillo y deben satisfacer tanto las necesidades del usuario como del Servicio Nacional de Aduanas. Fueron llevadas a cabo las etapas de análisis y diseño siguiendo los pasos del proceso de desarrollo de software denominado Proceso Unificado (UP), este proceso es iterativo e incremental y esta conformado por las siguientes etapas: Inicio, Elaboración, Construcción y Transición.

Dos aspectos que son claros son auditoria, que corresponde al registros de eventos ocurridos durante la ejecución del sistema, y autenticación, que maneja los permisos de acceso al sistema, ambos aspectos presentan dificultad al tratar de modelarlos en orientación a objetos.

Para este estudio y medición se escogió el módulo de administración de número de folio, el que corresponde a la asignación de este número, el cual es el identificador de cada declaración, y se asigna cierto rango de números a cada Aduana a lo largo del país.

III. Estado del Arte

A fines de años sesenta los proyectos informáticos eran programados siguiendo la metodología y programación estructurada, la que correspondían a programas con código secuencial, selectiva e iterativa, siendo de fácil entendimiento pero que entre sus desventajas destaca el generar bloques de códigos extensos que dificultan su mantenimiento. Para solucionar esta desventaja se dividió el programa en unidades pequeñas llamadas funciones, cada una con un propósito y tarea concreta. Esto no bastó para solucionar los problemas más complejos manteniéndose así complejos bloques de código difíciles de mantener. Es por esto que surgió la programación Orientada a Objetos.

En la Orientación a Objetos es el lenguaje el que se adapta al problema y no el problema al lenguaje. Se enfatiza en los datos y no en los algoritmos como en la programación estructurada. En orientación a objeto el problema se distribuye en objetos, en cambio en la programación estructurada se descompone en funciones.

A pesar de las ventajas obtenidas con Orientación a Objetos se descubrió que aún existía código mezclado y duplicado dentro de los sistemas. Es por esto que el grupo Demeter [1] estudió la programación adaptativa, la cual es una instancia temprana a la que después se llamaría Orientación a Aspectos. La programación adaptativa se basaba en la Ley de Remeter: "Solo conversa con tus amigos inmediatos", la cual aplicada a los objetos, se tiene que la operación de un objeto sólo tendría que utilizar:

- Las operaciones propias del objeto
- Los objetos que tengan asociados o sean atributos del objeto
- Los objetos que recibe como parámetro la operación
- Los objetos que cree la operación

Siguiendo estos principios se logra la encapsulación y se baja el nivel de acoplamiento.

Continuaron con estos estudios el grupo conformado por Cristina Lopes, Karl J., Lieberherr [2] y Gregor Kiczales quienes introdujeron el término orientación a aspectos.

Uno de los objetivos de la programación orientada a aspectos es el de separar conceptos lo que hace que cada cosa quede en su sitio, es decir, que cada decisión se tome en un lugar concreto. Las ventajas de la Orientación a Aspectos son:

- Código menos enmarañado, más natural y más reducido
- Más facilidad para depurar y hacer modificaciones en el código
- En un conjunto grande de modificaciones se tiene un impacto mínimo en el resto del código
- Código más reusable, que se puede acoplar y desacoplar cuando sea necesario.

Las fases de un proyecto utilizadas en orientación a aspectos son las siguientes:

- Captura de requerimientos (identificación y descripción de casos de uso)
- Análisis (identificación de clases, capa de aplicación, capa de dominio, capa de infraestructura, identificación de rebanadas de casos de uso que contienen extensiones de las clases para un cierto aspecto)
- Diseño (distribución de nodos en paquetes, ver detalles de implementación de clases, dividir rebanadas de casos de uso incluyendo aspectos y las extensiones de clases)
- Implementación (generar código de clases, codificar rebanadas de casos de uso en un lenguaje orientado a aspectos)
- Pruebas (pruebas para casos de uso, pruebas para rebanadas)

Para las fases de análisis y diseño en orientación a aspectos existen investigaciones [3 y 4] que buscan generar un UML extendido que incluya aspectos. Se busca traspasar un modelo Orientado a Objetos a uno que integre Orientación a Objetos y Orientación a Aspectos.

A. *Captura de Requerimientos*

En la fase de captura de requerimientos se siguieron los pasos propuestos por Betina Haak [5], para la captura de requerimientos que corresponden a aspectos. Los pasos a seguir para la captura de requerimientos de aspectos son:

1.- Analizar lista de casos de uso del proyecto

2.- Elección de aspectos candidatos

3.-Identificar relación entre casos de uso y aspectos candidatos
4.- Modelar en UML

B. *Identificación de relación entre aspectos del proyecto*

Después de identificar los requerimientos de tipo aspecto se elabora las descripciones e interacciones de estos aspectos con el sistema, siguiendo la propuesta de Frans Sasen [6], la cual describe e identifica los aspectos del sistema, como se ve en la Figura 1.

Nombre	Registrar eventos
Aspecto Involucrado	Auditoria
Tipo	Conflicto
Ejemplo	Cuando se ingresa nuevo rango de números de folio se debe registrar datos de evento
Explicación predicados	
Tiempo de Respuesta	No bloquear acción de siguientes eventos registro debe ser transparente para usuario

Fig. 1. Descripción e interacción de un aspecto.

C. *Análisis*

En la etapa de análisis para el modelamiento de orientación a aspectos se utilizó UML Extendido propuesto por J. Suzuki y Yamamoto [7], en donde se crea un nuevo metamodelo UML llamado "aspecto" y se les define como una clase especial, como se aprecia en la Figura 2.

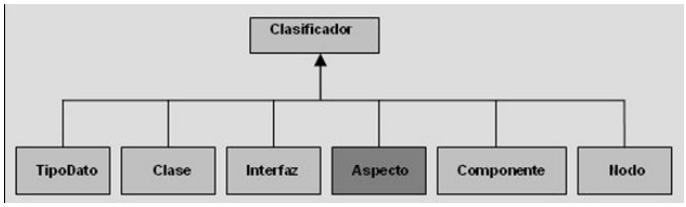


Fig. 2. Clase especial "aspecto".

Dentro del análisis se puede ver que muchas de las clases se encuentran afectadas por un aspecto, es por esto que dentro de la clase aspecto se deben especificar las clases a las cuales un determinado aspecto afecta.

D. *Diseño y Diagrama de packages*

Se modelaron los diagramas de clase e interacción de aspectos siguiendo la propuesta de Mark Bash y Arturo Sanchez [8]. En esta propuesta se sugiere la notación para diagramas de packages, donde cada aspecto es encapsulado en su propio package, y todas las funcionalidades del aspecto son modeladas en el package.

El diagrama de clase muestra cual componente de clase es cruzado por el aspecto, se dibuja una relación de join point y el componente de la clase aspecto. La Figura 3 muestra el package correspondiente al aspecto Auditoria.

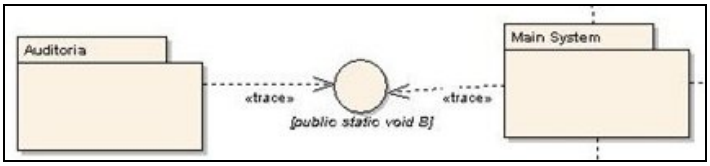


Fig. 3. Diagrama de package de aspecto.

IV. SOLUCIÓN PROPUESTA PARA EL PROYECTO

Para este trabajo se desarrollaron las fases de captura de requerimientos, análisis, identificación de casos de uso, identificación de aspectos, diseño y construcción de un módulo.

A. *Identificar rebanadas de casos de uso en el proyecto*

Para cada aspecto se identificaron las rebanadas de casos de uso que corresponde a una porción que contiene clases y extensiones a otras clases para un cierto aspecto.

Posteriormente se modelaron con UML Extendido las clases que corresponden a aspectos y que tenían extensiones a otras clases.

A continuación se puede ver en la Figura 4 la rebanada de caso de uso correspondiente al aspecto Auditoria.

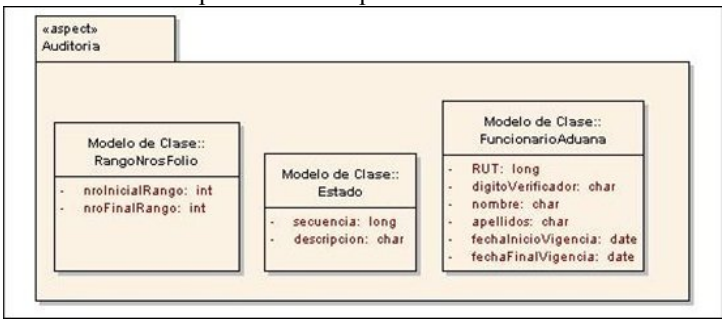


Fig. 4. Rebanada de caso de uso modelada con orientación a aspectos. El aspecto modelado corresponde a Auditoría.

B. *Diagrama con interacción de aspectos en proyecto*

El diagrama de interacción permite ver la la relación entre aspectos y componentes, además de mostrar el punto en donde ellos se cruzan, la cual es posible visualizarla en la Figura 5.

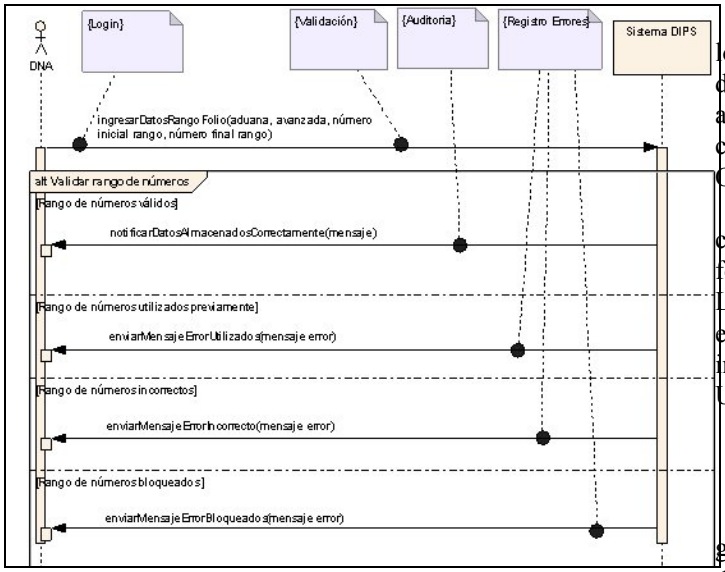


Fig. 5. Diagrama de interacción de ingreso de número de folio que incluye aspectos identificados. Los aspectos modelados son: login, validación, auditoria y registro errores.

C. *Construcción*

Para la construcción se utilizó el análisis y diseño del módulo de administración de números de folio.

Se escogió como lenguaje de programación para la orientación a aspectos el lenguaje AspectJ ya que actualmente es muy utilizado y viene como extensión de Java en la herramienta Eclipse.

Para construir este código se utilizó la herramienta My Eclipse Enterprise Workbench 6.0 con extensión AspectJ 1.5.0, ya que permite trabajar con Java y AspectJ a la vez.

En la construcción de este código fue de gran importancia identificar los puntos de unión, los cuales son la parte del código donde el aspecto incluye nuevo comportamiento, los puntos de unión identificados son agrupados en un punto de corte el cual indica en que lugar se ejecuta la acción del aspecto.

Finalizada la construcción del código con AspectJ se pudo ver que es un código que encapsula la acción del aspecto lo que lo hace reusable y más fácil de modificar.

V. MÉTRICAS Y MEDICIONES

Dentro de esta investigación se realizaron mediciones para medir beneficios entregados al incorporar orientación a aspectos.

A. *Mediciones realizadas a través de cuestionarios*

Como primera medición se realizó una encuesta que permitiera ver si se captaban beneficios en la incorporación de orientación de aspectos en las fases de análisis y diseño.

Esta encuesta fue creada basándose en las investigaciones de un grupo de estudio perteneciente a la PUC-Rio de Brasil el cual propone métricas de calidad [9].

El cuestionario propuesto incluye preguntas con respecto a los diagramas y modelos generados en las etapas de análisis y diseño, basándose en Orientación a Objetos del módulo de administración de números de folio del sistema DIPS, comparándolos con los modelos que integran Orientación a Objetos y Orientación a aspectos.

El universo de encuestados corresponde a 5 ingenieros con conocimientos en Orientación a Objetos, de estos dos de ellos forman parte de proyectos del Servicio Nacional de Aduanas. Los encuestados además cuentan con más de 5 años de experiencia en modelamiento y desarrollo de sistemas informáticos. Todos además han modelado utilizando lenguaje UML.

B. *Análisis de resultados de cuestionarios*

La primera sección del cuestionario está dirigida a ver el grado de conocimiento en Orientación a Objetos. El resultado del cuestionario demuestra que hay un alto grado de conocimiento de Orientación a Objetos ya que 4 personas (80% del total) del grupo encuestado lo conocen.

Se trató de conocer si los encuestados tenían nociones de Orientación a Aspectos, y si habían participado en algún proyecto en donde se hiciera uso de Orientación a Aspectos. Entre los encuestados había un parte que poseía conocimiento del término Orientación a Aspectos y con nociones de lo que significa, pero 2 personas (40%) no conocían el significado claro de Orientación a Aspectos.

Se sacó un promedio de las respuestas obtenidas de las consultas que pedían a encuestados ver que tan definidos estaban los aspectos Login, Auditoria, Validación y Registro de Errores en Casos de Uso que incluían OA, comparándolos con los casos de uso que sólo utilizaban OO. En el resultado se puede ver que un 30% está totalmente de acuerdo y un 45% está de acuerdo en que están más definidos los aspectos utilizando OA, esto señala que los encuestados encuentran que OA aporta un beneficio a los casos de uso para la definición de los aspectos.

Se preguntó a los encuestados si el modelo de clases mejoraba incorporando OA, principalmente centrando esta mejora para los aspectos (Login, Validación, Auditoria y Registro de Errores). Los encuestados en su mayoría se declararon estar parcialmente de acuerdo, esto debido a que por comentarios emitidos no era en forma clara el beneficio obtenido de incorporar OA, encontraron que el modelo se hacía más complejo.

Los beneficios se hubiesen visto en forma más clara si se incorporarán al modelo de clase todos los métodos de las clases aspectos, ya que un gran beneficio es la reducción de métodos (esto se podrá apreciar claramente más adelante en el capítulo de métricas).

Se consultó a los encuestados si apreciaban que la construcción del modelo de clases con OA requería de más tiempo. Dentro de las respuestas dadas por los encuestados 3 de estos (60%) está parcialmente de acuerdo que al incorporar OA en el modelo de clase requiere de más tiempo y 1 de los encuestados (20%) de acuerdo, por lo tanto 4 de los

encuestados encuentran que requiere de más tiempo incorporar OA en el modelo de clase.

Se consultó a los encuestados si incorporaría OA en modelos de clases de futuros proyectos. Hubo una aceptación por parte de los encuestados de incorporar OA en modelos de clases de proyectos futuros, 2 de los encuestados (40%) está totalmente de acuerdo y 1 de los encuestados (20%) de acuerdo, por lo que 3 de los encuestados adoptaría OA, lo que implica que los encuestados ven beneficios de adoptar OA sin ver como negativo el tiempo adicional de un modelo de clases con OA.

Se preguntó a los encuestados por medio del cuestionario si los diagramas de secuencia que incorporan OA muestran en forma más definida los aspectos identificados (Login, Validación, Auditoria y Registro de Errores) que los diagramas de secuencia que no poseen OA. Los resultados obtenidos muestran que los encuestados ven los aspectos en forma más definida incorporando OA en los diagramas de secuencia, lo que correspondería a un beneficio más dado por OA.

Se consultó además a los encuestados si desde su punto de vista se requiere de más tiempo para desarrollar diagramas de secuencia con OA. Los encuestados observan que incorporar OA a los diagramas de secuencia requiere de más tiempo comparándolo con los diagramas de secuencia que no incluyen notación OA.

Finalmente se consultó a los encuestados si incorporarían a futuros proyectos la notación OA en los diagramas de secuencia. A pesar de ver que requiere de más tiempo diseñar diagramas de secuencia con OA, 4 de los encuestados señalan que incorporarían OA en futuros diagramas de secuencia, ya que 2 de los encuestados (40%) está totalmente de acuerdo de incorporarlo y 2 de los encuestados (40%) está de acuerdo.

C. Análisis de resultados de cuestionarios

Para efectuar el análisis de los resultados de la encuesta se toma como referencia la hipótesis de este trabajo: "Incluir e integrar Orientación a Aspectos en las fases de análisis, diseño y construcción de proyecto de software construido en Orientación a Objetos trae consigo más beneficios en el desarrollo y mantención, que un producto de software sólo construido en base a Orientación a Objetos". A partir de esta hipótesis se consultó a los encuestados diferencias percibidas entre etapas de análisis y diseño de un proyecto que sólo utilizara OO y otro que utilizara OO complementado con OA. Los resultados del cuestionario indican que más de la mitad de los encuestados ven en forma más definida los requerimientos aspectos cuando se incorpora la notación de Orientación a Aspectos. Esto genera un beneficio a las etapas de análisis y diseño, principalmente si se busca dejar en forma más independiente los requerimientos correspondientes a aspectos, para un mejor análisis de estos y poder contar con un diseño de aspectos que permita implementarlos en la etapa de construcción.

Los encuestados indicaron un aumento en el tiempo de desarrollo de los casos de uso, modelo de clases y diagrama de secuencias, al incorporar la notación utilizada por Orientación a Aspectos; pero a pesar de este aumento en el tiempo se vieron muy inclinados a incorporar Orientación a Aspectos en futuros proyectos.

D. Medición de beneficios a través de métricas

Junto con las mediciones realizadas por medio del cuestionario se incorporó a esta investigación métricas que pudiesen medir la calidad y beneficio de incorporar orientación a aspectos. Estas métricas fueron obtenidas del modelo de calidad propuesto por Aida Atef Zacaria [10]. Este modelo se muestra a continuación en la Figura 6.

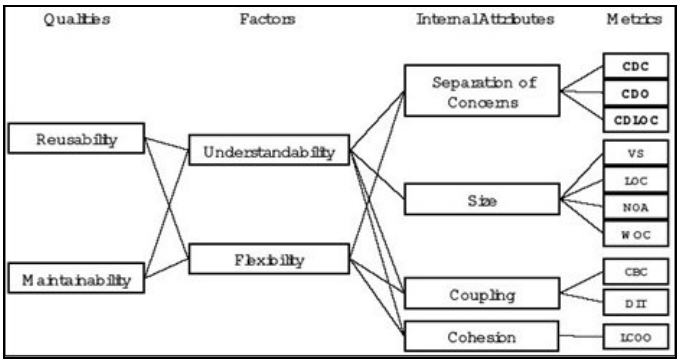


Fig. 6. Modelo de calidad que incluye métricas.

Las métricas utilizadas son las siguientes:

- CDC: Difusión del aspecto en los componentes (Concern Diffusion over Components)
- CDO: Difusión de aspectos en operaciones (Concern Diffusion over Operations)
- CDLOC: Difusión de aspectos en LOC (Concern Diffusion over LOC)
- VS: Tamaño de Vocabulario (Vocabulary Size)
- LOC: Líneas de Código (LOC Lines of Code)
- NOA: Número de atributos (Number of Attributes)
- WOC: Peso de las operaciones por componente (Weighted Operations per Component)
- CBC: Acoplamiento entre Componentes (Coupling between Components)
- DIT: Profundidad del árbol de herencia (Depth of Inherance Tree)
- LCOO: Falta de cohesión en las operaciones (Lack of Cohesion in Operations)

Se realizaron las mediciones utilizando las métricas del modelo de calidad aplicadas al módulo que fue construido sin aplicar OA y otro integrando OA, ambos se midieron y los resultados obtenidos se muestran en la Tabla I.

TABLA I
RESULTADO DE MÉTRICAS

	CDC	CDO	CDOLOC	VS	LOC	NOA	WOC	CBC	DIT	LCOO
Código sin OA	6	-	-	6	48	35	12	-	-	-
Código con OA	8	7	3	8	35	35	4	8	-	-

Tanto DIT como LCOO no fueron medidas porque no son medibles en código construido.

LOC muestra que hay una reducción las líneas de código al incorporar código con OA. Hay un tamaño en el vocabulario como muestra VS pero este aumento no desfavorece el tamaño del código del sistema.

Las métricas adicionales arrojaron los siguientes resultados:

$$\text{Beneficio\_POA} = \frac{\text{Tamaño\_sin\_aspectos} * \text{Factor\_de\_Desarrollo\_Java}}{\text{Tamaño\_con\_aspectos} * \text{Factor\_de\_Desarrollo\_AspectJ}} \quad (1)$$

Beneficio_POA = 6,86

$$\text{Limpieza} = \frac{(\text{Tamaño\_sin\_aspectos} - \text{Tamaño\_Funcionalidad\_Básica}) * 100}{\text{Tamaño\_sin\_aspectos}} \quad (2)$$

Limpieza = 56,25

Esto indica que existe un porcentaje de beneficio (1) al incorporar orientación a aspectos en el código y la limpieza (2) de código al utilizar OA es de un 56,25% lo que es un valor que indica que al menos se hace una reducción de la mitad del tamaño original del código.

VI. Conclusiones

Este estudio se enfocó en investigar la incorporación de Orientación a Aspectos en las etapas de análisis, diseño y construcción de un proyecto informático.

En la fase de captura de requerimientos se identificaron los aspectos candidatos a partir de los requerimientos obtenidos, los aspectos identificados fueron: Login (autentificación) ,Validación (de número de folio) , Auditoría y Registros de Errores. En las etapas de análisis y diseño se utilizó UML con OA para modelamiento de requerimientos.

Se utilizaron dos formas de medir los beneficios otorgados al incorporar OA en el proyecto. La primera medición consistió en un cuestionario realizado a ingenieros con conocimiento en Orientación a Objetos, y que buscaba ver la opinión de éstos. Los resultados del cuestionario indicaron mayoritariamente que utilizando la notación de Orientación a Aspectos los requerimientos se ven más definidos; esto genera un beneficio a las etapas de análisis y diseño, principalmente si se busca dejar en forma más independiente los requerimientos aspectos, para un mejor análisis de éstos y poder contar con un diseño de aspectos que permita implementarlos adecuadamente en la etapa de construcción. Los encuestados indicaron ver un aumento en el tiempo de desarrollo de los casos de uso, modelo de clases y diagramas de secuencia al incorporar la notación utilizada por Orientación a Aspectos; pero a pesar de este aumento en el tiempo se vieron muy inclinados a incorporar Orientación a Aspectos en futuros proyectos.

La segunda medición aplicada al proyecto correspondía a un conjunto de métricas que permitieran medir los beneficios que otorga el paradigma Orientación a Aspectos al incorporarlo en un proyecto de software que utiliza Orientación a Objetos con notación UML en la etapa de construcción de código. Analizando las métricas utilizadas se observa que el número de clases aumenta al incorporar las clases aspectos pero esto se compensa en el número de líneas de código, ya que una implementación con aspecto reduce significativamente este atributo. También se observa una reducción en el número de parámetros traspasados por método, debido a que con OA basta traspasarlos al aspecto.

Agradecimientos

Primeramente agradezco al Servicio Nacional de Aduanas, por permitirme realizar esta investigación tomando como caso de estudio un proyecto que está actualmente en desarrollo. También entrego mis agradecimientos a la profesora Silvana Roncagliolo perteneciente a la Escuela de Ingeniería Informática quien fue mi profesora guía en esta investigación. También debo agradecer a los ingenieros encuestados quienes se dieron el tiempo para responder los cuestionarios aplicados en este proyecto. Finalmente dar gracias a Dios y a mi familia que siempre me dan el apoyo necesario.

Referencias

[1] Lieber Lieberherr Karl, Adaptive Object-Oriented Software The Demeter Method, 1996 University Boston, http://www.ccs.neu.edu/research/demeter/biblio/dem-book.html
[2] Lieberherr Karl, Aspect-Oriented Programming with Aspectual Methods, 2001
[3] Mohamed M. Kandé, Jörg Kienzle and Alfred Strohmeier, From AOP to UML: Towards an Aspect-Oriented Architectural Modeling Approach, Software Engineering Laboratory, Swiss Federal Institute of Technology Lausanne, Switzerland, 2002
[4] Christina Chavez, Carlos Lucena, A Metamodel for Aspect-Oriented Modeling, 2002
[5] Betina Haak, Miguel Díaz, Claudia Marcos, Jane Prior, Identificación Temprana De Aspectos, ISISTAN Instituto de Sistemas Tandil, Facultad de Ciencias Exactas, UNICEN
[6] Frans Sanen, Eddy Truyen, Wouter Joosen, Andrew Jackson, Andronikos Nedos, Siobhán Clarke Distributed Systems Group, Trinity College Dublín, Neil Loughran, Awais, Rashid, Classifying and documenting aspect interactions, Distrinet Research Group, Department of Computer Science K.U.Leuven Celestynenlaan 200A, Computing Department Infolab 21,Lancaster University
[7] J. Suzuki, Y. Yamamoto. Extending UML with Aspect: Aspect Support in the Design Phase. 3er Aspect-Oriented Programming (AOP) Workshop at ECOOP'99
[8] Basch M., Sanchez A., Paper Incorporating Aspects into UML, Agenda of the Workshop on Aspect-Oriented Modeling with UML Marzo 18, 2003
[9] Claudio Sant'Anna, Alessandro Garcia, Christina Chavez, On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework, PUC-Rio, Computer Science Department, Brasil, 2003

[10] Zacaria A. y Dr. Hoda Hosny , Metrics for Aspect-Object Software Design, The American University in Cairo, Agenda of the Workshop on Aspect-Oriented Modeling with UML Marzo 18, 2003

RESUMEN

El presente trabajo es una investigación de los beneficios obtenidos al incorporar Orientación a Aspectos en un proyecto desarrollado en Orientación a Objetos. En un módulo de un proyecto en particular se utilizó en las etapas de análisis, diseño y codificación Orientación a Objetos sin incluir orientación a aspectos; y posteriormente en el mismo módulo se desarrollaron las mismas etapas pero incluyendo Orientación a Aspectos. Para la medición del beneficio de la incorporación de Orientación a Aspectos se realizaron encuestas a profesionales de informática con preguntas fijas que permitiesen dar una medición del beneficio. Junto a estas encuestas se utilizó además fórmulas que permitieron medir los beneficios en el diseño y codificación al incorporar Orientación a Aspectos.

ABSTRACT

This work is an investigation of the benefits obtained by incorporating Aspect-Orientation on a project developed in Object-Orientation. In a module of a particular project are used in the stages of analysis, design and coding Object-Orientation without included Aspect-Orientation, and later in the same module were the same stages but including Aspect-Orientation. To measure the benefit of incorporating Aspect-Orientation surveyed IT professionals to set questions which would enable a measurement of the benefit. Together these surveys are also used formulas to measure the benefits in the design and coding by incorporating Aspect-Orientation.

Tabla de Contenidos

GLOSARIO DE TÉRMINOS

Aspecto (Aspect): Diferentes temas de los que son necesario ocuparse para resolver un problema. Describen extensiones al comportamiento de los objetos. Hacen referencia a las clases de los objetos y definen en qué punto se colocan estas extensiones. Es un proceso realizado dentro de una clase pero que no es propio de la misma.

Asunto (Concern): Característica o preocupación importante para algún involucrado en el desarrollo de software que debe rastrearse desde la captura de requerimientos hasta el código.

Asuntos Transversales (Crosscutting Concerns): Son responsabilidades que aparecen diseminadas en código atravesando partes del sistema no relacionadas en el modelo. Un ejemplo de asunto transversal es el logging de un sistema. Existen asuntos que no pueden ser encapsulados fácilmente dentro de clases, módulos, etc. (según el tipo de lenguaje), y pueden afectar a varios módulos ya definidos. Estos asuntos requerirán su implementación en muchas clases o módulos, produciendo un código enmarañado (tangling) y/o mezclado (scattering), el cual será difícil de modificar y entender. A este tipo de asunto, se lo denomina crosscutting concern. Un aspecto es una abstracción que modela un crosscutting concern.

Desarrollo de Software Orientado a Aspectos (DSOA): Nuevo paradigma de programación cuya idea principal es proporcionar un soporte avanzado para la separación de *concerns* introduciendo una nueva unidad modular, llamada aspecto.

Entretejido (Weaving): Mecanismo por el cual se combinan los aspectos con el código base y puede hacerse en distintos momentos de la vida de un programa.

Puntos de Corte (Pointcut): Un conjunto de condiciones aplicadas a un punto de unión que, al cumplirse, activarán el punto de corte y se ejecutará el punto de ejecución asignado a dicho punto de corte.
Puntos de Ejecución (Advice): Fragmento de código que se ejecuta cuando se activa un punto de corte.

Puntos de Unión (Join Point): Una posición bien definida dentro del código orientado a objetos, por ejemplo, la declaración de un método , la ejecución de un constructor o la asignación de un miembro de un objeto.

Rebanadas de Caso de Uso: Una rebanada contiene clases y extensiones a otras clases para un cierto aspecto.

Tipos de rebanadas:

Específicas, contienen las clases necesarias para un cierto caso de uso.

No específicas, contienen clases del dominio del problema que se usan en muchos casos de uso.

Separación de Asuntos (Separation of concern): Cuando un problema dado involucra varios aspectos que deben ser identificados y separados.

1. INTRODUCCIÓN

La programación Orientada a Objetos (OO) es un gran avance para los requerimientos funcionales. Ofrece un buen nivel de abstracción pero no es suficiente, ya que en la codificación de los requerimientos hay código mezclado y diseminado. Esto dificulta el mantenimiento de un sistema.

La Orientación a Aspectos (OA) trata de resolver el problema de la separación de asuntos (separation of concerns). El principio de separación de asuntos fue identificado en la década de 1970, y plantea que un problema dado involucra varios aspectos que deben ser identificados y separados. Algunos de estos aspectos son:

- Sincronización
- Distribución
- Coordinación
- Persistencia
- Replicación

Los aspectos describen extensiones al comportamiento de los objetos. Hacen referencia a las clases de los objetos y definen en qué punto se colocan estas extensiones.

Las principales características del desarrollo de software con Orientación a Aspectos son flexibilidad, adaptabilidad y reusabilidad de los elementos usados para componer los sistemas. Orientación a Aspectos provee varios mecanismos para expresar, adaptar, aislar y reusar asuntos transversales en el desarrollo de software para obtener esas principales características [14].

Los primeros períodos en la evolución de la Orientación a Aspectos se centró principalmente en el nivel de implementación y codificación, es por esto que se encuentra más avanzado el estudio en lo que respecta a lenguajes para Orientación a Aspectos como es el caso del lenguaje AspectJ. Pero en los últimos tiempos surgen trabajos e investigaciones para llevar la separación de funcionalidades a nivel de análisis y diseño en un proyecto de software [21].

La ingeniería de requerimientos Orientada a Aspectos reduce el peligro de cambios no esperados en los productos de software, a través de una identificación temprana de aspectos y resolución de conflictos. O sea, si no se tratan los aspectos tempranamente, podrían surgir cambios en el sistema que se está desarrollando en etapas posteriores, aumentando los costos de desarrollo, mantenimiento y evolución. Es por esto que el presente trabajo tiene por

objetivo investigar la integración de Orientación a Aspectos en un proyecto de software que está siendo desarrollado con Orientación a Objetos en las etapas de Análisis, Diseño y Construcción. Se busca por medio de esta investigación poder incluir una evaluación de los beneficios que provee utilizar en conjunto la Orientación a Objetos y la Orientación a Aspectos.

Cuando se utilizan aspectos, pueden aparecer conflictos cuando varios aspectos afectan a un mismo elemento del modelo. Una posible situación conflictiva se presentará cuando un mismo elemento del modelo se vea afectado por más de un aspecto. En este punto, la mayoría de las veces no se tiene definido cómo se deben aplicar dichos aspectos. Se podrían priorizar y aplicar primeramente el de mayor prioridad, o quizás excluir alguno de ellos, etc. Recientemente un número de investigadores han propuesto el uso de conceptos Orientados a Aspectos en etapas tempranas del desarrollo de software. El hecho de identificar los asuntos en etapas tempranas del ciclo de desarrollo de software puede significar la reducción de costos de desarrollo, mantenimiento y acciones de mejoramiento [3].

En este trabajo se maneja la siguiente hipótesis “Incluir e integrar Orientación a Aspectos en las fases de análisis, diseño y construcción de un proyecto de software construido en Orientación a Objetos trae consigo más beneficios en el desarrollo y mantención, que un producto de software sólo construido en base a Orientación a Objetos”.

2. OBJETIVOS

En el presente capítulo se presentan los objetivos del trabajo.

2.1. OBJETIVO GENERAL

Realizar un estudio de los beneficios obtenidos al integrar la Orientación a Aspectos con Orientación a Objetos.

Para realizar este estudio se propone ejecutar las etapas de análisis, diseño y construcción de un módulo en un proyecto de software seleccionado para esta investigación; comparar los casos de uso en los que no se incluyó Orientación a Aspectos con los casos de uso en que si fueron incluidos. Con lo anterior verificar los beneficios obtenidos de esta forma realzar la importancia de incorporar en el área informática el uso de Orientación a Aspectos.

2.2. OBJETIVOS ESPECÍFICOS

Comprobar si ingenieros con conocimientos en Orientación a Objetos perciben diferencias y beneficios al incorporar Orientación a Aspectos en las etapas de análisis y diseño de un proyecto de software.

Estudiar, seleccionar y analizar resultados de mediciones que permitan cuantificar los beneficios en un proyecto de software integrando Orientación a Objetos con Orientación a Aspectos.

Promover el uso de Orientación a Aspectos en todas las etapas de los proyectos informáticos.

2.3. DESCRIPCIÓN DEL PROBLEMA DE PROYECTO DE SOFTWARE

Con el presente trabajo se busca poder medir la utilidad de incluir la Orientación a Aspectos en un proyecto de software en sus etapas de Análisis, Diseño y Construcción. Para este objetivo se seleccionó un proyecto que se desarrolla para el Servicio Nacional de Aduanas Chile, enfocándose en sus fases de Análisis, Diseño y Construcción.

El proyecto propuesto está destinado para el Servicio Nacional de Aduanas, institución pública que cumple funciones claves para el desarrollo del país, ya que tiene un rol preponderante en materia de comercio exterior, especialmente, en la facilitación y agilización de las operaciones de importación y exportación, a través de la simplificación de trámites y procesos.

El Servicio Nacional de Aduanas requiere contar con el Sistema para el Control de las Declaraciones de Importación y Pago Simultáneo de Viajeros (DIPS Viajeros) [6]. Estas operaciones, por ser de bajo valor y que generalmente

requieren ser despachadas con rapidez, necesitan de un sistema ágil que permita, por un lado satisfacer las necesidades de los usuarios y por otro lado, facilitar la fiscalización que a ellas realiza el servicio.

La importación de las mercancías podrá realizarse mediante Declaración de Importación de Pago Simultáneo (DIPS), cuyo formulario será proporcionado por el Servicio Nacional de Aduanas, en los siguientes casos:

- Las comprendidas en la sección 0 del arancel aduanero, con excepción de las señaladas en las partidas 0001; 0002; 0003; 0006; 0007, 0020; 0024; 0031; 0032 y 0033.
- Las donadas con ocasión de catástrofe o calamidad pública al Estado, a personas naturales o jurídicas, de derecho público o fundaciones o corporaciones de derecho privado y a las universidades reconocidas por el estado.
- Aquellas cuyo valor FOB no exceda de US$ 500 facturado.
- Aquellas que arriben conjuntamente con el viajero, consignadas a un tercero, siempre que su valor FOB facturado no exceda de US$ 500 y pertenezcan a una persona natural o jurídica.

El formulario de DIPS Viajero se deberá confeccionar al arribo del pasajero al país o al momento de hacer abandono de la zona franca regular o zona franca de extensión. Éste podrá ser confeccionado por Fiscalizadores o por Digitadores con los correspondientes permisos.

Este proyecto es desarrollado en sus fases de análisis y diseño con Orientación a Objetos siguiendo los pasos del proceso de desarrollo de software denominado Proceso Unificado (UP). El proceso UP es iterativo e incremental, centrado en la arquitectura y guiado por los casos de uso. El proceso UP está conformado por cuatro fases:

- Inicio: en donde se ve la vialidad del proyecto, se propone la arquitectura, se analizan los riesgos críticos y se presenta un prototipo inicial. Se generan casos de uso fundamentales.
- Elaboración: se analiza la factibilidad del proyecto viendo los costos y riesgos que pueden afectar. Se ven los niveles de calidad a los cuales se desea llegar y se hace una planificación que involucre personal y costos. Se genera gran parte de los casos de uso de los requerimientos funcionales.
- Construcción: se finaliza la construcción de los casos de uso con el detalle de cada uno y los diagramas de secuencias de éstos. Se finaliza el análisis, diseño, implementación y pruebas del proyecto.
- Transición: se genera el producto final. Se desarrollan y entregan los manuales y documentos que están destinados al usuario para el uso del producto final. Finalmente se instala en el entorno de usuario y se corrigen los errores detectados.

Las fases del proceso UP que se seguirán para este trabajo son Inicio, Elaboración y parte de fase de Construcción (no se realizarán pruebas de producto). En cada una de las fases se confeccionarán artefactos que son productos o salidas de las fases, para este trabajo se generarán los siguientes artefactos:
Captura de Requerimientos con documento único de requerimientos. Se analizarán para este trabajo sólo los requerimientos que deben ser modelados por Orientación a Aspectos.

- Generación de Casos de Uso
- Detalle de Casos de Uso
- Modelo de Clases
- Diagrama de Secuencias
- Código de Producto

Respecto de los requerimientos de este proyecto que provocan dificultad para ser modelados con Orientación a Objetos principalmente se tiene el requerimiento de Auditoría, que corresponde al registro de eventos producidos sobre los documentos de Aduana, que considera creación de documento, modificación de datos del documento, así como también, anulación y eliminación de datos del documento aduanero. Otro requerimiento que complica el modelamiento corresponde al ingreso de información con carácter privado, lo que hace que el sistema deba tener un acceso controlado y restringido, es decir seguridad. Estos dos requerimientos, tanto en la etapa de análisis como de diseño, han presentado dificultad al ser modelados con Orientación a Objetos; el presente trabajo tendrá como objetivo el investigar si la Orientación a Aspectos puede resultar de ayuda para modelar estos requerimientos, en forma más clara, para quienes utilizarán los documentos finales de análisis y diseño del sistema.

3. METODOLOGÍA DE INVESTIGACIÓN

La metodología de investigación seleccionada para este trabajo corresponde a la metodología propuesta por Roberto Samoieri, considerando las siguientes etapas:

- Concebir la idea a investigar: la investigación de este trabajo se centra en evaluar los beneficios de incorporar OA a un proyecto.
- Plantear el problema de investigación: para esta investigación se realizará una comparación entre un proyecto desarrollado en OO frente a un proyecto desarrollado en OO con incorporación de OA, realizando mediciones en ambos proyectos.
- Elaborar el marco teórico: se realiza estudio bibliográfico de fuentes de investigadores y profesionales que han estudiado la incorporación de OA.
- Tipo de investigación: para este trabajo se escogió investigación descriptiva ya que se detallará el atributo beneficio de incorporación de OA a proyectos informáticos, a través de mediciones efectuadas a este atributo.
- Establecer la hipótesis: la hipótesis central de este trabajo es que utilizar OA en proyectos informáticos trae consigo beneficios.
- Seleccionar el Diseño de investigación: el diseño de esta investigación está compuesto por las siguientes etapas:
  - Estudiar fuentes relacionadas con Orientación a Aspectos.
  - Desarrollar las etapas de análisis, diseño y construcción del módulo de un proyecto utilizando OO.
  - Desarrollar las etapas de análisis, diseño y construcción del módulo de un proyecto utilizando OO y OA.
  - Efectuar mediciones que permitan una comparación entre un proyecto que no utilice OA con otro que si la incorpore.
  - Realizar análisis de mediciones efectuadas para calcular beneficio entregado por OA.
- Recolectar los datos: los datos que serán utilizados para evaluar beneficios de OA se recopilarán por dos vías. La primera vía corresponde a un cuestionario que permitirá reunir las opiniones de encuestados correspondientes a ingenieros a los cuales se le presentarán modelos de análisis y diseño que incorporarán OA. La segunda vía de recolección de datos son métricas aplicadas al módulo del proyecto seleccionado, para esto se deberá generar una parte del código con Java y otro código que incorpore AspectJ (lenguaje de programación para Orientación a Aspectos).
- Analizar los datos: en esta investigación se analizaron los datos de los resultados del cuestionario realizado y además un análisis de los resultados por medio de las métricas aplicadas en el módulo del proyecto.

- Presentar los resultados: los resultados obtenidos del cuestionario serán presentados en gráficos que permiten ver las respuestas obtenidas de los encuestados. Para las métricas se presentarán los resultados de las fórmulas en tablas, para permitir comparación entre los resultados obtenidos.

4. ESTADO DEL ARTE

La Programación Estructurada fue creada a fines de los años sesenta, y hoy en día es la forma más sencilla de programar ya que utiliza tres estructuras: secuencial, selectiva e iterativa. Entre sus ventajas están que es de fácil entendimiento, los errores se detectan y corrigen en forma fácil, se generan programas sencillos y fáciles con bajo costo de mantenimiento.

Una desventaja en la programación estructurada es que debido a su estructura se puede llegar a generar un bloque de código extenso, el cual cuando se hace muy grande dificulta su mantenimiento. Para resolver esta desventaja un programa se divide en unidades más pequeñas llamadas funciones, cada una de éstas con un propósito bien definido y una tarea concreta. Pero las funciones no bastaron para resolver problemas complejos, es por esto que surgió la OO.

El principio básico de la Orientación a Objetos es que un sistema de software se ve como una secuencia de "transformaciones" en un conjunto de objetos. En la programación orientada a objetos es el lenguaje el que se adapta al problema, a diferencia de la programación estructurada en donde el problema se ajustaba al lenguaje. La programación estructurada enfatiza en los algoritmos, en cambio, la programación Orientada a Objetos enfatiza en los datos. En Orientación a Objetos para resolver el problema se descompone en objetos, y no como el caso de la programación estructurada, en donde el problema se descompone en funciones [23].

Posterior a la Orientación a Objetos se empezó a hablar del concepto de programación orientada a aspectos el cual fue introducido por Gregor Kiczales y su grupo, aunque el equipo Demeter había estado utilizando ideas orientadas a aspectos antes incluso de que se acuñara el término. El trabajo del grupo Demeter [15] estaba centrado en la Programación Adaptativa (PA), que no es más que una instancia temprana de la programación orientada a aspectos. El término Programación Adaptativa se introdujo recién en 1991. La relación entre OA y PA nace de la Ley de Demeter: "Sólo conversa con tus amigos inmediatos". Se le denominó Ley de Demeter debido a que el proyecto Demeter fue fuertemente influenciado por esta Ley. En el 2004 Karl Lieberherr redefinió la Ley dejándola de la siguiente forma: "Sólo conversa con tus amigos quienes compartes tus asuntos" [17]. Aplicando la ley de Demeter a objetos, se tendría que una operación de un objeto sólo tendría que utilizar:

- Las operaciones propias del objeto
- Los objetos que tenga asociados o sean atributos del objeto
- Los objetos que recibe como parámetro la operación
- Los objetos que cree la operación

Siguiendo estos principios de la Ley de Demeter en un proyecto de software se logra maximizar la encapsulación y se baja el nivel de acoplamiento.

La PA es una metodología que aplica la separación de asuntos transversales relacionados, los cuales navegan a través de un grupo de objetos relacionados con el propósito de cumplir una tarea. En PA los asuntos transversales son expresados por un gráfico de clases similar a un diagrama de clases de UML. La separación de asuntos en PA ofrece muchos beneficios como código menos disperso y menos enmarañado, lo que lo hace ser un código más fácil de reusar. No fue hasta 1995 cuando se publicó la primera definición temprana del concepto de aspecto, realizada también por el grupo Demeter.

Gracias a la colaboración de Cristina Lopes y Karl J. Lieberherr [16] (quienes participaban en el grupo Demeter), con Gregor Kiczales y su grupo se introdujo el término de Programación Orientada a Aspectos (POA). Entre los objetivos que POA ha propuesto están principalmente el de separar conceptos, lo que hace que cada cosa quede en su sitio, es decir que cada decisión se tome en un lugar concreto y no en forma diseminadas como aún sucede en OO; y el de minimizar las dependencias entre ellos, lo que lleva a una disminución del acoplamiento entre los distintos elementos.

De la consecución de estos objetivos se pueden obtener las siguientes ventajas:

- Un código menos enmarañado, más natural y más reducido.
- Una mayor facilidad para razonar sobre las materias, ya que están separadas y tienen una dependencia mínima.
- Más facilidad para depurar y hacer modificaciones en el código.
- Se consigue que un conjunto grande de modificaciones en la definición de una materia tenga un impacto mínimo en las otras.
- Se tiene un código más reusable y que se puede acoplar y desacoplar cuando sea necesario.

El Desarrollo de Software Orientado a Aspectos (DSOA) es una adaptación del proceso unificado [12] e incluye las actividades:

- Captura de Requerimientos
    - Identificar los requerimientos funcionales con Casos de Uso
    - Identificar los requerimientos no funcionales

- Contemplar posibles extensiones
- Detallar los Casos de Uso

- Análisis
  - Identificar la estructura de los elementos del análisis que serán:
    - Paquetes
    - Tipos de Clases: interfaz, control y entidad
    - Identificar la estructura de Casos de Uso en rebanadas de Caso de Uso, estas estructuras son ortogonales
    - Capa de Aplicación: paquetes con sus respectivas clases
    - Capa de Dominio: modelo de clases
    - Capa de Infraestructura: autorización, distribución y persistencia
    - Rebanada de Caso de Uso: contiene Clases y Extensiones a otras clases para un cierto aspecto
    - Tipos de Rebanadas:
      I. Específicas: contienen las clases necesarias para un cierto Caso de Uso
      II. No Específicas: contienen clases del dominio del problema que se usan en múltiples casos de uso
- Diseño
  - Distribuir en los nodos los paquetes
  - Reorganizar las estructuras de los elementos del análisis incluyendo detalles del diseño
  - Refinar las clases con detalles de implementación
  - Dividir las rebanadas incluyendo aspectos y las extensiones de clases
- Implementación
  - Generar el código de las clases
  - Codificar las rebanadas en un lenguaje orientado a aspectos, como por ejemplo AspectJ
- Pruebas: las pruebas se llevan a cabo desde los requerimientos hasta la codificación
  - Se diseñan pruebas para cada Caso de Uso
  - Se diseñan las pruebas para las rebanadas
  - Se crean rebanadas de pruebas que se puedan remover al completar las pruebas

Se han detectado problemas en la etapa de análisis de requerimientos de aspectos, específicamente en la construcción del modelo funcional de requerimientos, Silvio Meier [24] propone formas para solucionar problemas que se presentan al modelar requerimientos de aspectos.

Con respecto al modelamiento de clases existen investigaciones [4 y 18] que buscan modelar metamodelos para Orientación a Aspectos. Estas investigaciones buscan utilizar UML extendido que soporte Orientación a Aspectos. También buscan la transición de un modelo Orientado a Objetos a un modelo basado en Orientación a Objetos combinado con Orientación a Aspectos.

Existen propuestas de perfiles UML publicadas en Workshop on Aspect-Oriented Modeling with UML [19] que corresponden a un conjunto predefinido de mecanismos de extensión para un dominio, tecnología o metodología. En esta propuesta se utiliza UML extendido utilizando estereotipos que permiten incorporar aspectos.

Una propuesta para la fase de Diseño es la formulada por Mark Basch y Arturo Sánchez [1] que permite utilizar UML para diagramas de clase e interacción de aspectos. En esta propuesta se sugiere la notación para diagramas de paquetes, donde cada aspecto es encapsulado en su propio paquete, y todas las funcionalidades del aspecto son modeladas en el mismo. Esto puede incluir diagrama de interacción y diagrama de clases.

Hoy en día los proyectos que incorporan orientación a aspectos utilizan en la etapa de construcción mayoritariamente el lenguaje AspectJ. AspectJ fue creada en el año 1998 por Gregor Kiczales y corresponde a una extensión de Java que se distribuye en forma gratuita como Open Source. Esta extensión permite incluir en los programas Java las clases de tipo aspecto. Actualmente AspectJ es difundido por la organización Eclipse y su última versión actualmente es AspectJ 1.6.4.

En AspectJ existen dos tipos de asuntos transversales: los dinámicos y los estáticos. Un asuntos transversal dinámico es cuando el aspecto afecta al comportamiento de la aplicación, modificando o agregando nuevo comportamiento. Los constructores utilizados son:

- Puntos de Enlace (*join point*): son los puntos de ejecución donde el aspecto agrega un nuevo comportamiento.
- Puntos de Corte (*cut point*): permite agrupar los puntos de unión y exponer su contexto. Indica el lugar donde va la acción del aspecto.
- Avisos (*advice*): especifica el código que se ejecutará en el punto de unión y que satisface a cierto punto de corte. Indica cuál es la acción que hace el aspecto.

Un asunto transversal estático ocurre cuando el aspecto afecta la estructura estática del programa, como ocurre con las clases e interfaces. Su principal función es dar soporte al código del asunto transversal dinámico. Los constructores de asuntos transversales estáticos son:

- Declaraciones inter-tipo: permiten añadir miembros (campos, métodos o constructores) a clases, interfaces o aspectos de una aplicación.
- Declaraciones de parentesco: logran indicar que ciertas clases implementan una interfaz o extienden una nueva clase.
- Declaraciones en tiempo de compilación: se utilizan para agregar advertencias o errores en tiempo de compilación.
- Declaraciones de precedencia: son usadas para especificar relaciones de precedencia entre aspectos.
- Excepciones suavizadas: ayudan a ignorar el sistema de chequeo de excepciones de Java, no dejando que actúen las excepciones que tiene lugar en puntos de unión.

Tanto los constructores pertenecientes a los asuntos transversales dinámicos como los de asuntos transversales estáticos se encapsulan en un meta modelo denominado Aspecto. Un aspecto es la unidad básica de AspectJ, como la clase lo es para un lenguaje Orientado a Objetos. Un aspecto puede también contener atributos, métodos y clases anidadas.

Actualmente existe Spring que es un framework de código abierto creado el 2003 para aplicaciones en Java que soporta las versiones 1.3 y posteriores. Una de las ventajas de Spring es que se encarga del acoplamiento entre objetos de la aplicación pero sin llegar a formar dependencias con el framework. Para no invadir el código se basa en interfaces en lugar de clases. Además posee una infraestructura que permite desarrollar los componentes como POJOs (Plain Old Java Object), conteniendo sólo la lógica del negocio y ocultando la complejidad de la arquitectura [29].

Spring está conformado por varios sub frameworks uno de ellos llamado Aspect-Orietnted Programming framework que maneja la funcionalidad que cruza transversalmente una aplicación, invocando métodos específicos en objetos específicos, para dar solución a requerimientos que se usan en múltiples lugares. El framework de Spring se integra con frameworks más poderosos como es AspectJ. En Spring un punto de enlace es la invocación a un método. Y un punto de corte es seleccionar un punto de enlace en el objetivo. Un avisador es el que relaciona al punto de enlace con el punto de corte.

Además de la medición realizada a la etapa de análisis y diseño se requerirá incorporar una medición que vaya relacionada con la implementación del sistema, para esto será utilizado un modelo de calidad que aplique métricas.

Uno de los primeros modelos de calidad es el propuesto por Barry Boehm en 1978, el cual realiza la medición utilizando 7 factores de calidad, estos son: portabilidad, confiabilidad, eficiencia, usabilidad, testeabilidad, facilidad de entendimiento y flexibilidad a modificaciones. Este modelo se centra en la medición del producto final.

En el año 1992 se introduce un estándar de calidad internacional propuesto por Internacional Organization for Standardization (ISO) , la que en ese año emitió la ISO 9126 la cual contiene 6 factores de calidad : funcionalidad, confiabilidad, eficiencia, usabilidad, mantenibilidad y portabilidad.

Una métrica para medir incorporación de OA es medir tamaño del software; existen dos formas de medir tamaño, una de éstas es contando las líneas de código, y una segunda forma es contando los puntos de función por medio del método de Albretcht, este método va relacionado con los casos de uso que posee el diseño de un sistema.

Otra forma de medición son las métricas de complejidad del software midiendo la cohesión y acoplamiento, una métrica propuesta es la dada por Jianjun Zhao [26 y 27] que plantea fórmulas de medición de la cohesión y acoplamiento de un sistema que incorpora OA.

Junto a las anteriores métricas se encuentran las métricas externas las cuales apuntan a las interfaces dejando de lado la estructura interna del sistema, un ejemplo es contabilizar el número de invocaciones que recibe un aspecto o el número de puntos de corte desde donde se invoca.

Aida Atef Zacaria [25] propone métricas para medir Orientación a Aspectos en un proyecto de software. Para esto utiliza un modelo de calidad como se ve en la Figura 1, que incluye algunas de las métricas propuestas por Aida Atef Zacaria y que se centran en ver reusabilidad, mantenibilidad, entendimiento y flexibilidad del sistema, que son parte de los factores de calidad propuesto por la ISO 9126. Se escogió este modelo propuesto debido a que se centra en los factores de calidad que contienen como atributos internos separación de asuntos que es propio de OA, tamaño, acoplamiento y cohesión, estos últimos mejorados con OO y ahora mejorados con OA.

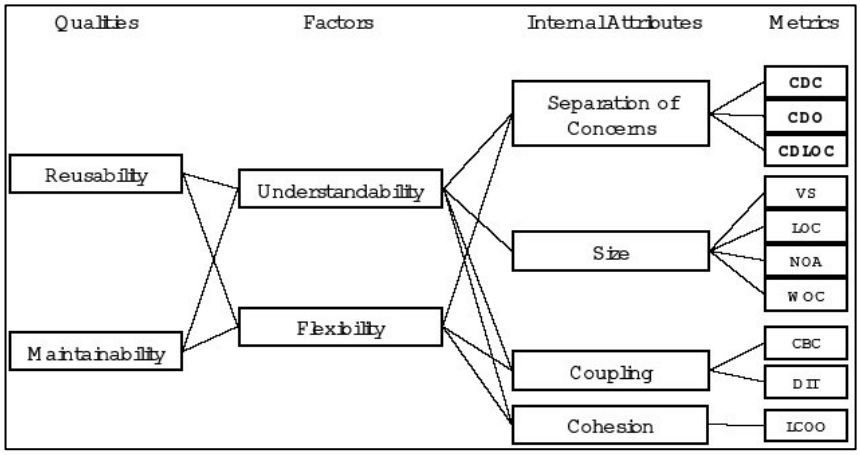


Figura 1: Modelo de Calidad

Las métricas propuestas para medir incorporación de Orientación a Aspectos en proyecto son las que se describen a continuación.

a) Métricas pertenecientes a Modelo de Calidad (Modelo de Calidad de Figura 1)

1. Difusión del aspecto en los componentes (CDC Concern Diffusion over Components): CDC es la métrica de diseño que cuenta el número de componentes primarios con el objetivo principal de contribuir a la implementación de un aspecto. Además, cuenta el número de componentes que acceden al componente principal usando declaraciones de atributos, parámetros formales, tipos de retorno, declaraciones y variables locales, o invocando sus métodos.

   CDC = Número de clases aspecto + Número de clases que acceden a clase aspecto

2. Difusión de aspectos en operaciones (CDO Concern Diffusion over Operations): CDO cuenta el número de operaciones primarias cuyo principal objetivo es contribuir a la implementación de un asunto. Además, cuenta el número de métodos y notificaciones que acceden al componente primario invocando sus métodos o usándolos en parámetros formales, tipos de retorno, declaraciones o variables locales. Los constructores también son contados como operaciones.

   CDO = Número de métodos de una clase aspecto + Número de métodos que invocan métodos de clase aspecto

3. Difusión de aspectos en LOC (CDLOC Concern Diffusion over LOC): CDLOC cuenta el número de puntos de transición para cada aspecto a través de las líneas de código. El uso de esta métrica requiere un proceso compartido que particiona el código en áreas compartidas y no compartidas. Las áreas compartidas son líneas de código que implantan un asunta dado. Los puntos de transición son puntos en el código donde está la transición de un área compartida a un área no compartida y viceversa. Se analiza línea por línea del código para identificar y contar los puntos de transición.

   CDLOC = Número de puntos de transición

4. Tamaño de Vocabulario (VS Vocabulary Size): VS cuenta el número de componentes del sistema. Esta métrica mide el tamaño del vocabulario del sistema. Se cuenta cada nombre de componente como parte del vocabulario del sistema. Las instancias de componentes no son contabilizadas.

   VS = Número de clases existentes

5. Líneas de Código (LOC Lines of Code): LOC cuenta el número de líneas de código. Las líneas en blanco no son contabilizadas como parte del código.

   LOC = Número de líneas de código

6. Número de atributos (NOA Number of Attributes): NOA cuenta el vocabulario interno de cada componente.

   NOA = Número de atributos de las clases

7. Peso de las operaciones por componente (WOC Weighted Operations per Component): WOC mide la complejidad de un componente en términos de sus operaciones. La medición de la complejidad de una operación se obtiene al contar el número de parámetros de la operación, asumiendo que una operación con más parámetros que otra es más compleja.

   WOC = Número de parámetros por método

8. Acoplamiento entre Componentes (CBC Coupling Between Components): CBC esta definido para medir en un componente (clase o aspecto) cual es el número de otros componentes acoplados a éste. Cuenta el número de clases que son usados en las declaraciones de atributo. También cuenta el número de componentes declarados

como parámetros formales, tipos de retorno, declaraciones y variables locales, clases y aspectos en los cuales están creados dentro de los atributos y métodos seleccionados. En resumen CBC cuenta el número de instancias de acoplamiento entre clases. A mayor acoplamiento el sistema es más sensible a los cambios y esto hace más difícil su mantención.

CBC = Número de clases acopladas a cada clase

9. Profundidad del árbol de herencia (DIT Depth of Inherance Tree): DIT es el máximo largo desde la clase nodo hasta la raíz del árbol. Éste permite medir el número de ancestros de una clase.

   DIT = Número de ancestros de una clase (en herencias)

10. Falta de cohesión en las operaciones (LCOO Lack of Cohesion in Operations): LCOO mide la falta de cohesión de una clase con respecto a sus operaciones o métodos. LCOO mide la cantidad de métodos pares que no acceden a una misma variable instanciada. Esta métrica mide eficiencia y reusabilidad. Baja cohesión incrementa la complejidad en el código.

    LCOO = Número de métodos de cada clase que no están cohesionados

b) Métricas Adicionales

Otra métrica para medir los beneficios de Orientación a Aspectos incorporada en fase de construcción es la métrica *Beneficio_POA* [7]. Esta métrica mide la relación de esfuerzo de desarrollo entre una implementación utilizando aspectos con otra que no la incluye. Se basa en: tamaño del producto y el esfuerzo de desarrollo. Se utiliza el factor de desarrollo propuesto por Rich Rice, este factor establece que para un sistema pequeño el tiempo de desarrollo promedio en Java es de 10 horas/programador, mientras que en AspectJ es de 2 horas/programador, se evalúa el tamaño físico de los archivos de las clases y de los aspectos en Kb. La métrica se define como un cuociente entre los valores aplicados en ambas implementaciones. Si el valor de resultado es mayor a 1 significa que hay un impacto positivo al usar Orientación a Aspectos:

$$\text{Beneficio\_POA} = \frac{\text{Tamaño\_sin\_aspectos} * \text{Factor\_de\_Desarrollo\_Java}}{\text{Tamaño\_con\_aspectos} * \text{Factor\_de\_Desarrollo\_AspectJ}}$$

Tamaño_con_aspectos = Tamaño_Funcionalidad_Básica + Tamaño_Aspecto

Cuando se incorpora aspectos en la construcción de un sistema se tiene como resultado un código que no contiene código entrelazado ya que éste queda encapsulado en los aspectos. Para medir el código que se reduce al introducir aspectos se utilizará la métrica propuesta, llamada *Limpieza*. Esta métrica mide el porcentaje de código que se reduce al introducir aspectos, utilizando el tamaño de las clases, medido en Kb. La formula es la siguiente:

$$\text{Limpieza} = \frac{(\text{Tamaño\_sin\_aspectos} - \text{Tamaño\_Funcionalidad\_Básica}) * 100}{\text{Tamaño\_sin\_aspectos}}$$

El resultado corresponderá al porcentaje de código que se reduce al incorporar aspectos. Finalmente se propone contar las líneas de código de las clases incluyendo y no incluyendo aspectos, comparando ambos valores para ver tamaño.

5. DESARROLLO DE PROPUESTA

Para esta investigación de las fases propuestas por Jacobson [12] se aplicarán Captura de Requerimientos, Análisis, Identificar la estructura de Casos de Uso en rebanadas de Caso de Uso, Rebanada de Caso de Uso, Diseño (Diagrama de secuencia o interacción y Modelo de Clases) y construcción de un módulo.

5.1. CAPTURA DE REQUERIMIENTOS

Se seguirán los pasos propuestos [3] para la identificación de aspectos de la etapas de Capturas de Requerimientos y Análisis Orientado a Aspectos. En dicho trabajo se propone una forma de poder identificar los aspectos en la etapa de análisis de requerimientos. Se enfoca además a modularizar los aspectos identificados.

5.1.1. PASOS PROPUESTOS APLICADOS A PROYECTO

Para el desarrollo de esta investigación se escogió el módulo de administración de número de folio, que forma parte del sistema DIPS Viajero. A continuación se mostrarán los pasos a seguir para la captura de requerimientos.

Paso 1: Analizar lista de Casos de Uso del Proyecto.
Se escogió el módulo correspondiente a Rango de Números de Folio, que permite asignar un rango de números a una Aduana y posteriormente a su asignación se utilizan estos números en folios confeccionados en forma manual, para ser registrados en el sistema.

Los casos de uso de este módulo son:

- Caso de Uso 1: Ingresar Rango de Números de Folio
- Caso de Uso 2: Modificar Rango de Números de Folio
- Caso de Uso 3: Bloquear Rango de Números de Folio
- Caso de Uso 4: Validar Número de Folio utilizado

Paso 2: Elección de Aspectos Candidatos

Aspecto Candidato	Casos de Uso	Actor	Responsabilidad
Login	1,2,3	Administrador, Fiscalizador	Dar seguridad en acceso a interfaz
Auditoría	1,2,3	Administrador, Fiscalizador	Realizar un registro de Auditoria de los eventos que producen cambios en los datos

Validación	1,2,4	Administrador, Fiscalizador	Validar que número de folio indicado se encuentre correcto
Registro de Errores	1,2,3,4	Administrador, Fiscalizador, Sistema	Cuando se produzcan errores durante la ejecución de un caso de uso se deberá registrar

Paso 3: Identificar relaciones entre Casos de Uso y Aspectos Candidatos

	Caso de Uso 1	Caso de Uso 2	Caso de Uso 3	Caso de Uso 4
Login	X	X	X	
Auditoria	X	X	X	
Validación	X	X		X
Registro de Errores	X	X	X	X

Paso 4: Modelar en UML

El diagrama de casos de uso inicial desarrollado sólo con Orientación a Objetos, y excluyendo orientación a aspectos, se muestra en la Figura 2.

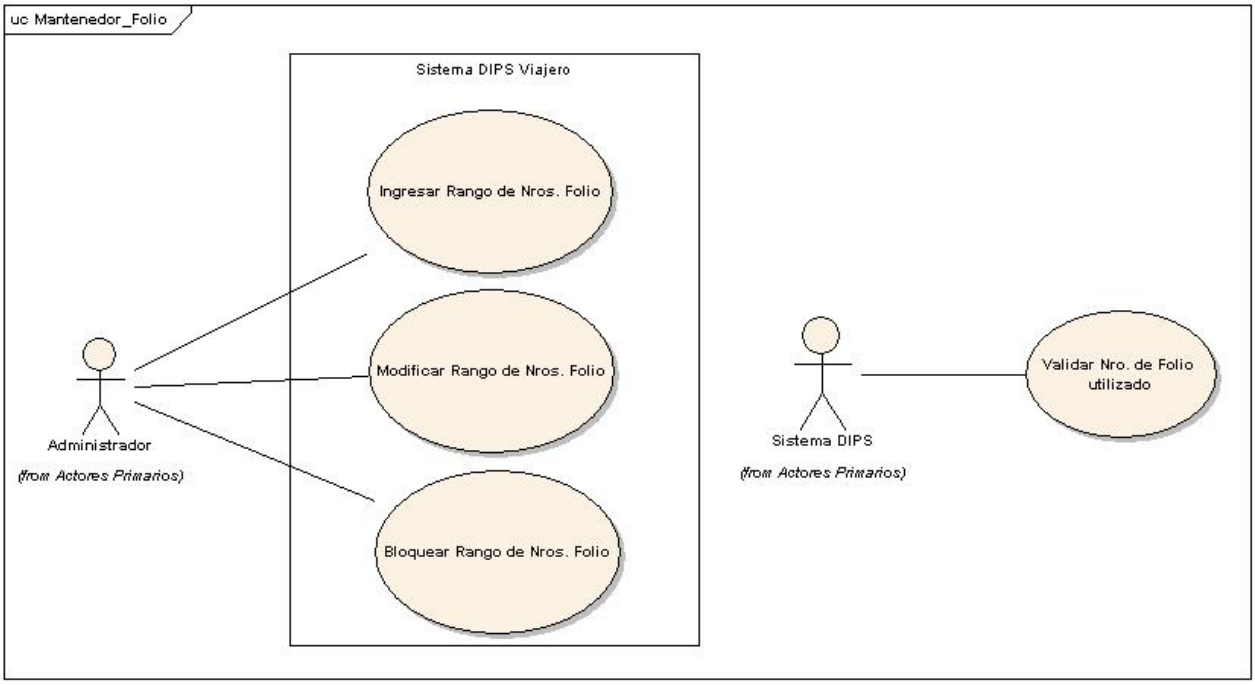


Figura 2: Casos de Uso OO

Los casos de uso desarrollados en Orientación a Objetos para el Módulo de Número de Folio se encuentran en Anexo A.

Desarrollados los pasos anteriores se llega al modelo que incorpora los aspectos capturados, y en el cual se incluye Orientación a Aspectos combinado con Orientación a Objetos como se ve en la Figura 3.

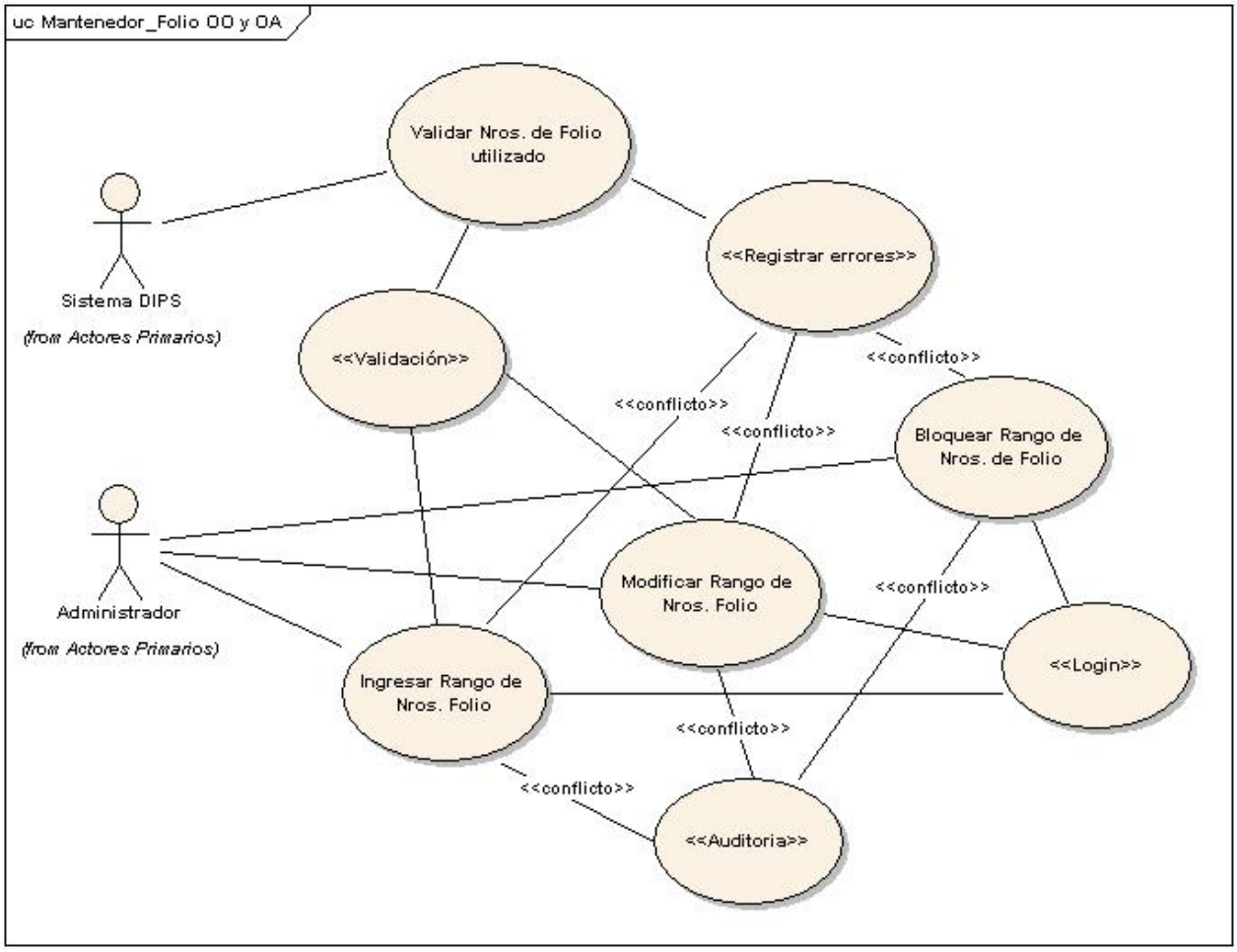


Figura 3: Casos de Uso con OA y OO

La notación para los aspectos en este modelo es <<nombre aspecto>> según notación propuesta en [3]. En este tipo de modelos se produce lo que se llama el fenómeno de los conflictos entre aspectos, que es cuando dos o más aspectos compiten por su activación, esto sucede sobretodo en el caso de que estos aspectos no son independientes entre si. La notación utilizada para este caso es <<conflicto>>.

Los casos de uso desarrollados en Orientación a Aspectos para captura de requerimientos de aspectos se pueden ver en Anexo B.

5.1.2. IDENTIFICACIÓN DE INTERACCIÓN DE ASPECTOS EN PROYECTO

Además de la identificación de aspectos propuesta en [3], se usa para las descripciones de las interacciones entre aspectos la descripción propuesta en [8], como se ve en la Figura 5. Esta propuesta permite describir e identificar los aspectos que están incluidos dentro de los requerimientos del sistema. Se ha concluido en estudios recientes que es de gran importancia identificar los requerimientos de aspectos en etapas tempranas del desarrollo de software, así se evita dejarlos excluidos.

En el proyecto, tras identificar las interacciones entre aspectos, no se detectaron conflictos entre aspectos ya que cada uno actúa en tiempos distintos. Se utilizó un cuadro como se muestra en la Figura 4, para la descripción de aspecto identificado, y complementar de esta forma el modelo UML de caso de uso que incluía al aspecto. El resto de las interacciones se pueden ver en Anexo B.

Nombre	Registrar eventos
Aspecto Involucrado	Auditoría
Tipo	Conflicto
Ejemplo	Cuando se ingresa nuevo rango de números de folio se debe registrar datos de evento
Explicación predicados	_____
Tiempo de Respuesta	No bloquear acción de siguientes eventos registro debe ser transparente para usuario

Figura 4: Interacciones entre aspectos

5.2. ANÁLISIS

En la etapa de análisis, considerando que el proyecto se modela con Orientación a Objetos utilizando como lenguaje UML, en Orientación a Aspectos se utilizó una versión extendida del mismo, propuesto por J. Suzuki y Yamamoto [13], posteriormente se realizó una comparación entre un modelo con Orientación a Aspectos y otro que no lo incluye, ambos utilizando UML, para de esta forma facilitar el trabajo de medición de beneficios otorgados. Esta propuesta está basada en la idea de una nueva metaclase en el metamodelo UML, llamado “aspecto”. El aspecto se deriva como un nuevo constructor del elemento Clasificador, así se define al nuevo elemento dentro de UML como: “elementos que describen características de comportamientos estructurales”, por ello se les identifica como una clase especial, tal como se muestra en la Figura 5.

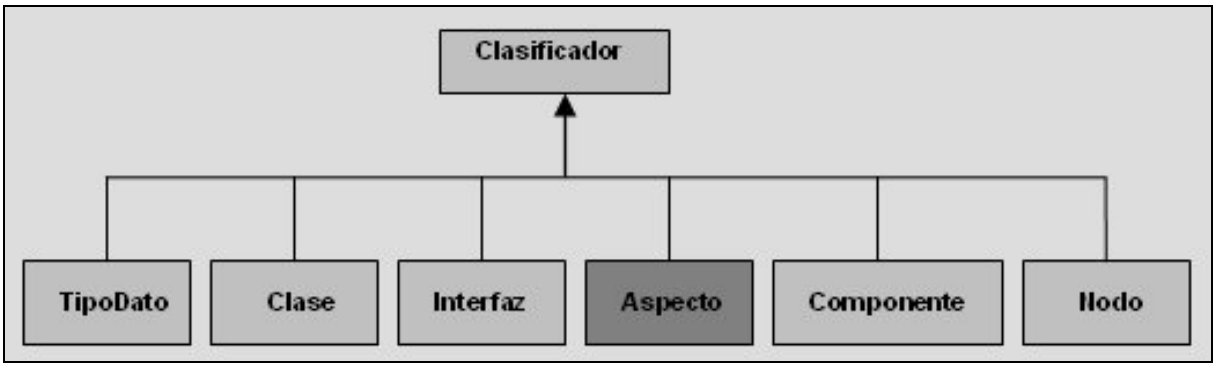


Figura 5: Clase especial "Aspecto"

Dentro de esta propuesta de diseño, un aspecto se representa como un rectángulo de clase con el estereotipo <<aspecto>>; en la parte de operaciones de aspecto se muestran las declaraciones tejidas. Cada tejido o lo que son equivalentes a la introducción (introduction) y avisos (advices), se muestra como una operación con el estereotipo <<tejido>>. Una declaración de tejido representa los elementos que están afectados por el aspecto (clases, métodos o variables); esto se realiza colocando el nombre de la clase que afecta el aspecto, los métodos del aspecto que se tejan y el valor de retorno que se obtiene, tal como lo muestra la Figura 6.

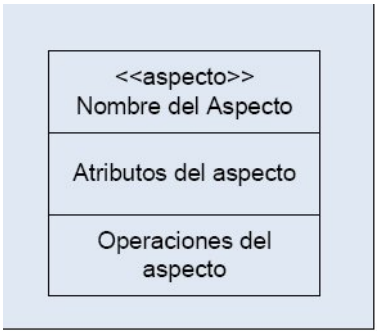


Figura 6: Clase Aspecto

Dentro del análisis pueden existir varias clases a las cuales afecta un aspecto, esto hace que se tengan especificaciones de métodos similares afectando a diferentes clases, por esta razón en el modelado se utiliza un cuadro que contienen las diferentes clases y qué está relacionado a un aspecto, como muestra la Figura 7. A dicho cuadro se le da un nombre genérico de tal forma, que dentro de las operaciones dicho nombre sustituye a todas las clases a las que el aspecto afecte, así será la comprensión más efectiva.

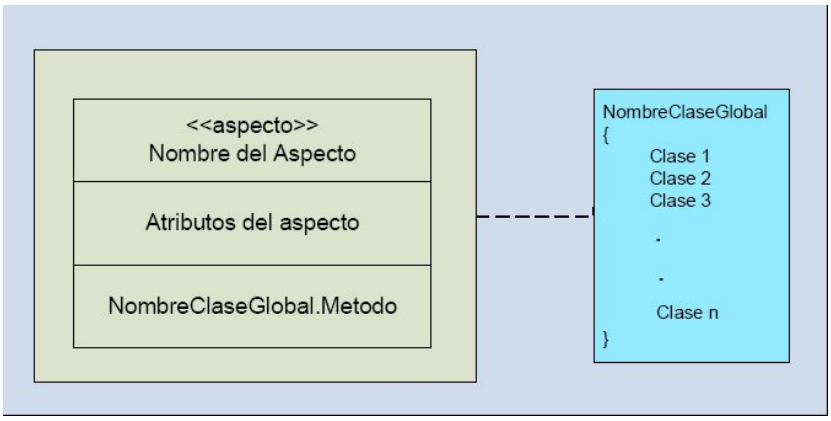


Figura 7: Aspecto asociado a varias clases

La relación entre los aspectos y las clases a las que afectan es un tipo de relación de dependencia, se representa mediante el estereotipo <<realiza>>. La idea en si con este estereotipo es que dentro del aspecto se ubiquen las operaciones que afectarán a la clase, y con dicha relación todos estos métodos sean parte de una forma lógica de la clase, aunque físicamente estén definidos dentro del aspecto. Esta relación se representa mediante la notación de la relación Aspecto-Clase mostrada en la Figura 8.

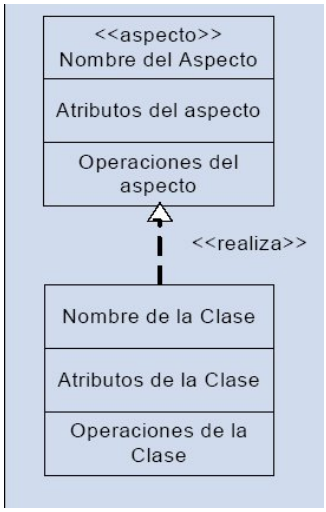


Figura 8: Relación Aspecto-Clase

Después de enlazar los aspectos con las clases afectadas los autores Susuki y Yamamoto recomiendan que se especifique en la clase resultante, la clase origen y el aspecto que han intervenido obteniendo así una nueva estructura, la cual se representa con un paquete, como se muestra en la Figura 9. Para modelar esto se utiliza el estereotipo “clase tejida”.

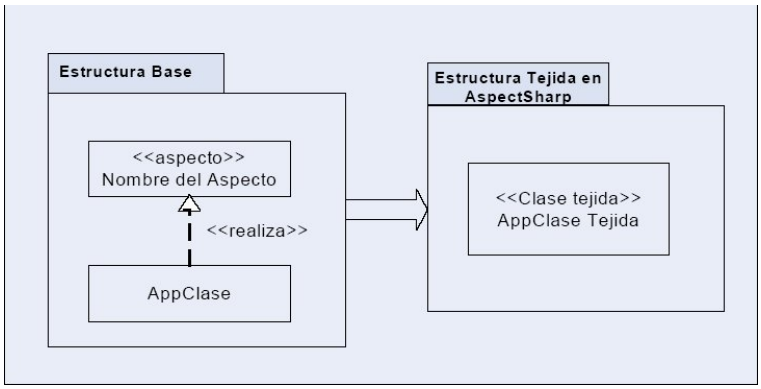


Figura 9: Enlace Clase y Aspecto

En esta propuesta no hay una definición para los puntos de corte, lo que podría ser un inconveniente para que no sea adoptada como idónea para los conceptos de Orientación a Aspectos, pero por el esteriotipado que se ha adoptado es muy fácil comprender la ubicación de dichos conceptos, principalmente por las especificaciones que se hacen dentro del aspecto. Esta propuesta podría considerarse como el primer intento de extender UML para soportar Orientación a Aspectos.

También existe una notación en casos de uso de aspectos propuesta por Iris Groher y Stefan Schulze [11]. Esta propuesta utiliza UML extendido y corresponde a avances desarrollados a los trabajos de investigación que se hicieron a partir del trabajo de Jacobson [12] y de lo que se ha avanzado con AspectJ, viendo cómo generar código a partir de modelos UML para Orientación a Aspectos.

Un ejemplo concreto de modelamiento en Orientación a Aspectos efectuado por Gefei Zhang [10] muestra una aplicación Web modelada mediante Orientación a Aspectos, este ejemplo es de gran ayuda para el desarrollo de esta investigación y de su aplicación en el proyecto, ya que permite ver un claro ejemplo de modelamiento de aspectos de seguridad de un sistema Web, los cuales también son requerimientos del proyecto en desarrollo.

5.2.1. IDENTIFICAR REBANADAS DE CASO DE USO EN EL PROYECTO

Para cada aspecto se identificaron las rebanadas de caso de uso que corresponde a una porción que contiene clases y extensiones a otras clases para un cierto aspecto. Para esto primero se identificaron las clases que no son aspectos utilizados por el módulo de Administración de Número de Folio. Las clases correspondientes a este módulo que fueron modeladas sólo con Orientación a Objetos se ven en la Figura 10.

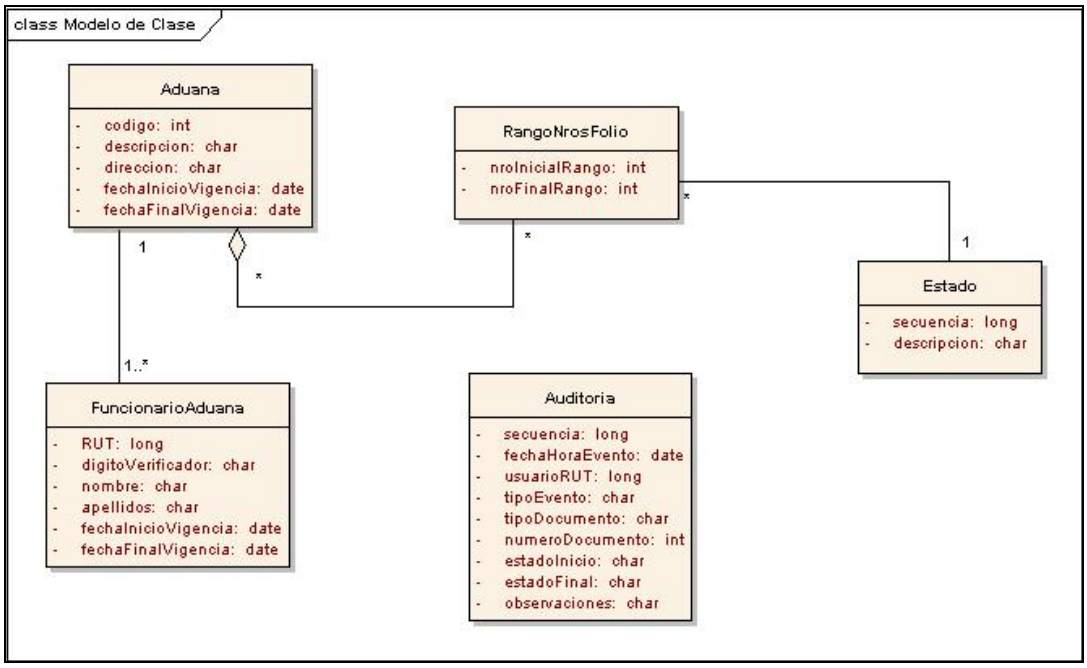


Figura 10: Clases modeladas con Orientación a Objetos

Se observa que para el aspecto Auditoria se utiliza una clase Auditoria el cual contendrá los atributos y métodos necesarios para auditar los eventos. No se ve reflejada el vínculo entre la clase aspecto y las otras clases.

Posteriormente se crearon las clases que correspondían a aspectos y que tenían extensiones a las otras clases. Para esto se utilizó las propuestas de modelamiento de rebanadas de caso de uso con Orientación a Aspectos. La Figura 11 muestra una de las rebanadas correspondiente al aspecto de Auditoria, para el resto de las rebanadas ver Anexo B.

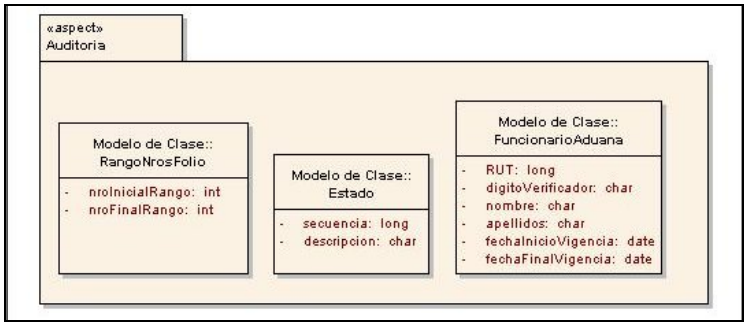


Figura 11: Rebanada de Caso de Uso modelada con Orientación a Aspectos

5.3. DISEÑO Y DIAGRAMA DE PACKAGES DE ASPECTOS

Para que un lenguaje permita Orientación a Aspectos debe incluir el concepto de puntos de enlace (join points). Un punto de enlace es la parte del código que se ve afectado por cierto aspecto. La propuesta [1] incorpora los diagramas de clases dentro de los paquetes de aspectos. El diagrama de clase muestra cual componente de clase es cruzado por el aspecto, dibujando una relación entre el punto de enlace y el componente de la clase, como muestra la Figura 12.

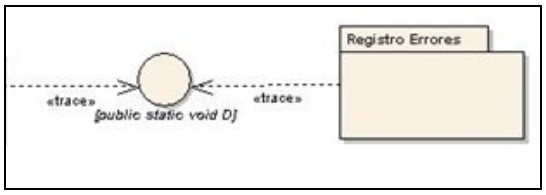


Figura 12: Diagrama de clase

Se confeccionó un diagrama de paquetes correspondientes a los aspectos identificados, el que se despliega en la Figura 13.

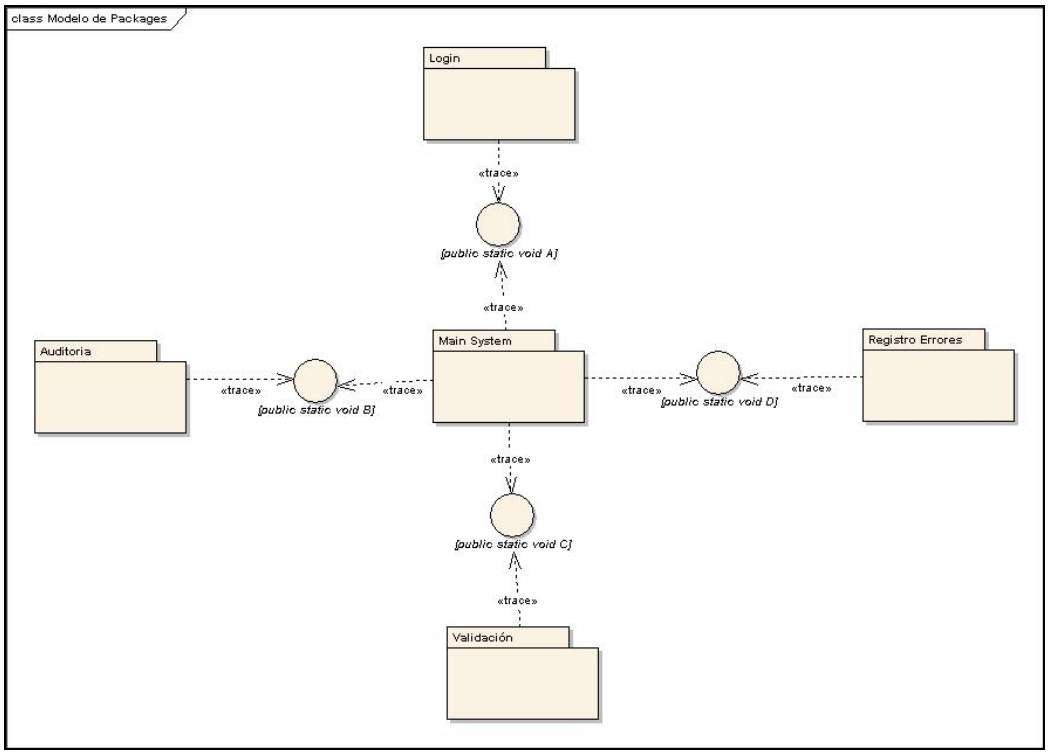


Figura 13: Diagrama de paquetes de aspectos

5.3.1. DIAGRAMA DE INTERACCIÓN CON ASPECTOS EN EL PROYECTO

Los diagramas de interacción describen la relación específica entre los aspectos y los componentes y en los puntos en los cuales ellos se cruzan. Estos deben ser modelados en diagramas de interacción dentro del paquete principal, tal como se muestra en la Figura 14. Los diagramas de interacción o secuencia para este proyecto fueron generados para los sub-módulos pertenecientes al módulo de Administración de Número de Folio.

Primero se modelaron los diagramas correspondientes al módulo escogido utilizando sólo Orientación a Objetos. En la Figura 14 se muestra el sub-módulo Ingresar Número de Folio.

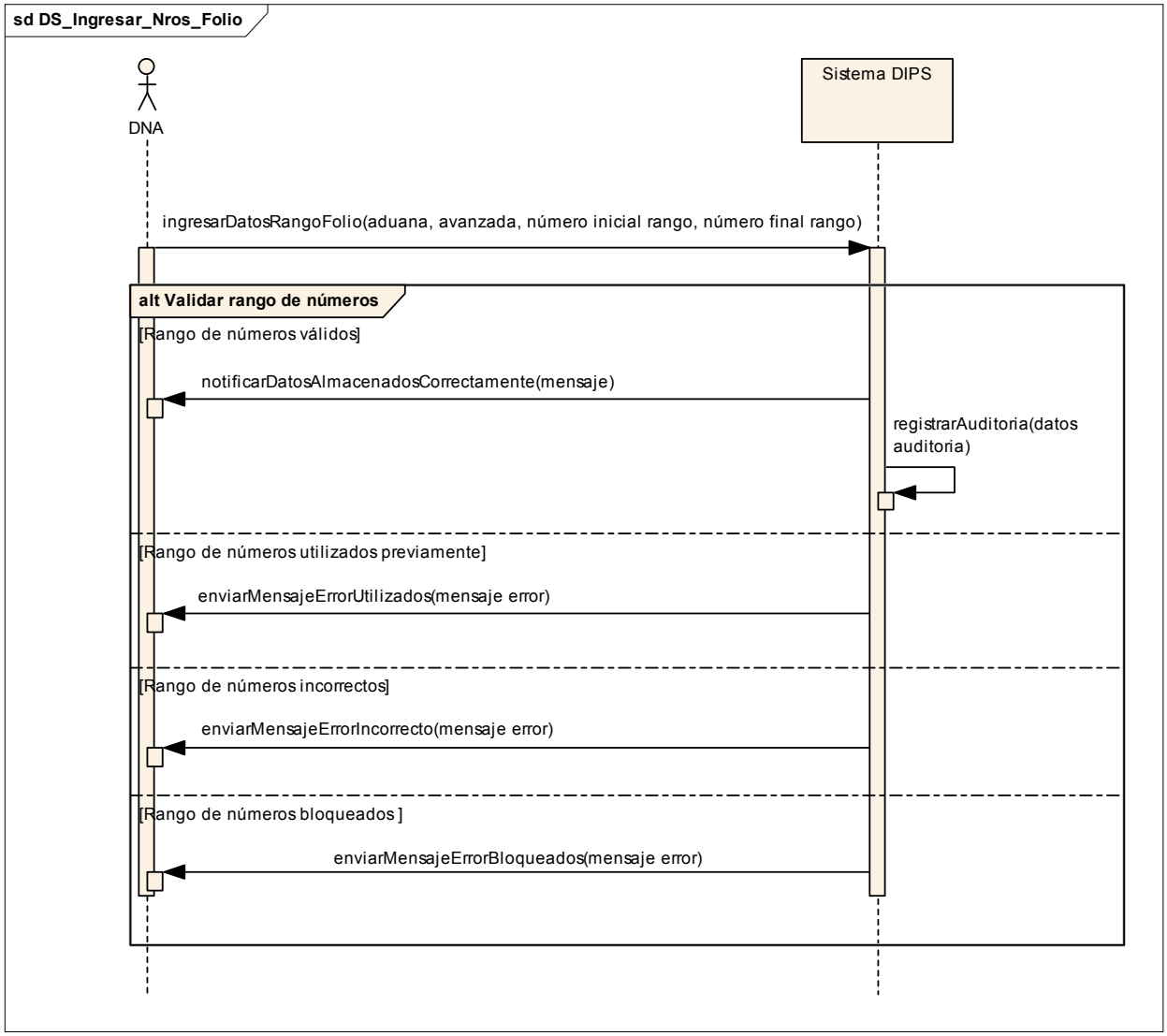


Figura 14: Diagrama de interacción de Ingreso de Número de Folio con Orientación a Objetos

El resto de los diagramas de interacción realizados sólo con Orientación a Objetos y que corresponden a los sub módulos del módulo de Administración de Número de Folio junto con su descripción, pueden rescatarse del Anexo C.

Siguiendo las propuestas para diagramas de interacción con aspectos se confeccionaron los diagramas de interacción que incluyen modelamiento con Orientación a Aspectos. En la Figura 15 se muestra el diagrama de interacción del sub-módulo Ingreso de Número de Folio utilizando Orientación a Aspectos.

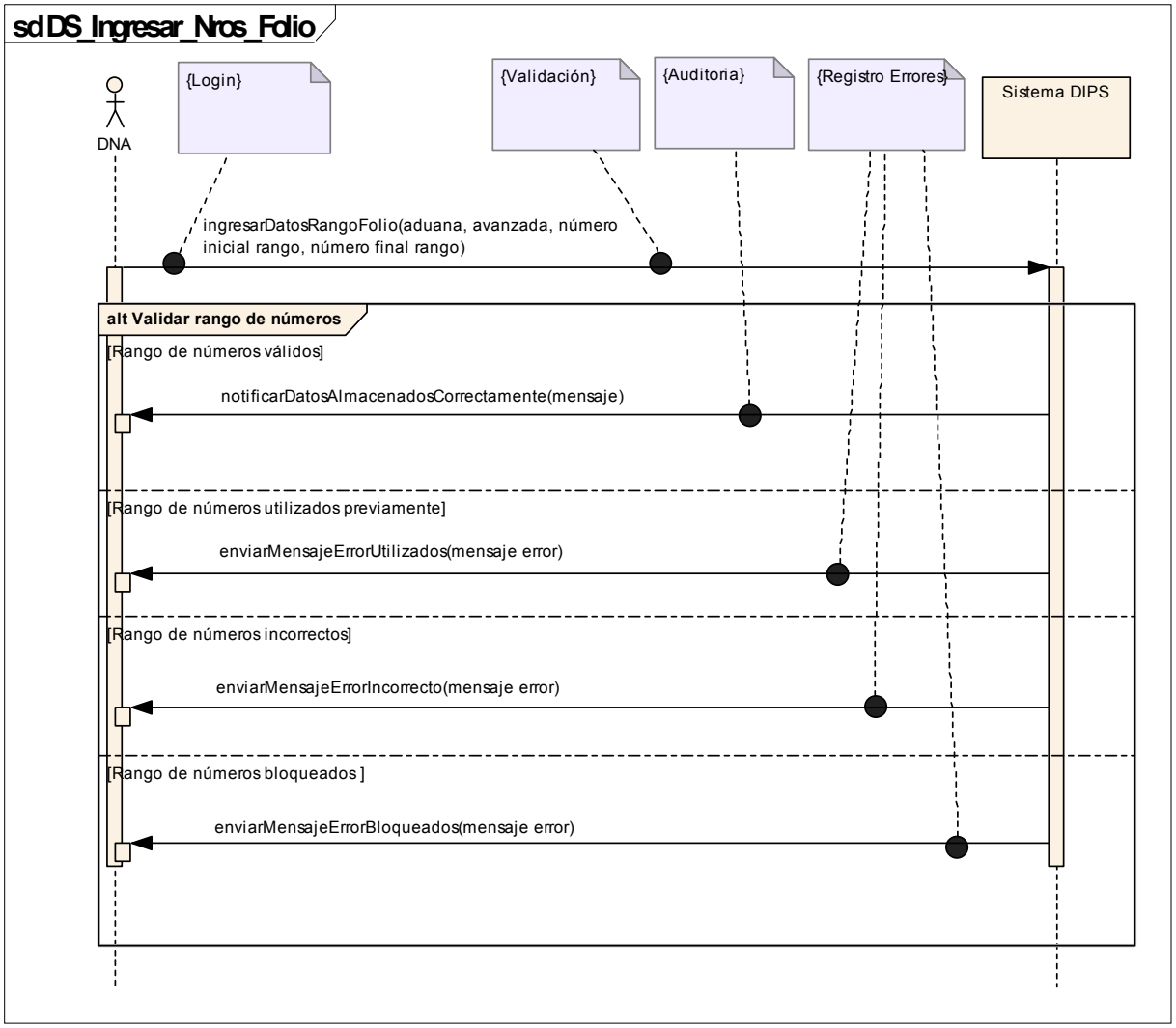


Figura 15: Diagrama de interacción de Ingreso de Número de Folio con Orientación a Aspectos

Los demás diagramas de interacción que incluyen Orientación a Aspectos y que corresponden a los sub-módulos del módulo de Administración de Número de Folio, junto con su descripción, se encuentran en el Anexo D.

5.4. CONSTRUCCIÓN

Para este proyecto se escogió construir el Módulo de Administración de Números de Folios utilizados por las Declaraciones de Importación y Pago Simultáneo de Viajeros (DIPS Viajeros). Este sistema será desarrollado en su totalidad en lenguaje Java con J2EE, lo que permite utilizar para esta investigación el lenguaje AspectJ el cual se agrega como extensión a Java, utilizando como herramienta de desarrollo My Eclipse Enterprise Workbench 6.0 con extensión AspectJ versión 1.5.0, cuya ventana de entrada se ve en la Figura 16. Junto a esta herramienta se utilizó el software BEA WebLogic 1.8.4. Se construyó un código utilizando sólo Java y otro uasando Java con Aspectos; ambos códigos se utilizaron para las mediciones de beneficio a través de métricas, detalladas más adelante en la sección Métricas Aplicadas en el Proyecto.

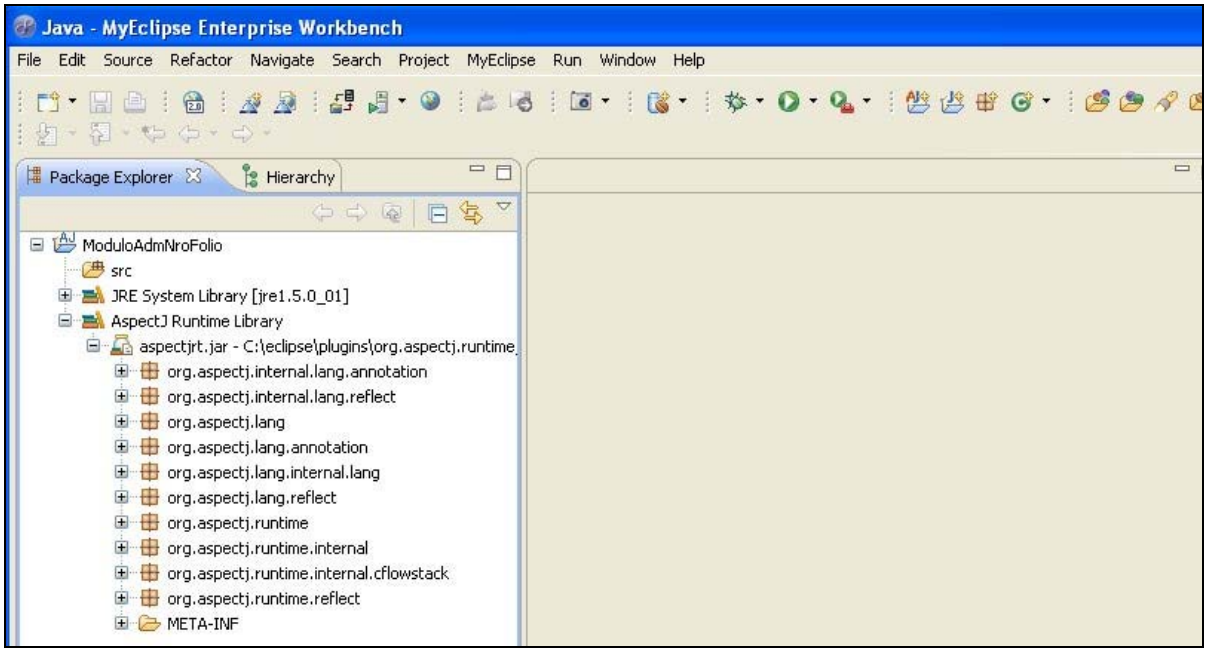


Figura 16: Ventana de Presentación de la Herramienta My Eclipse Enterprise Workbench 6.0 con extensión AspectJ

Para la construcción, primero se identificaron los puntos de unión detectados en la fase de diseño. Es en esos puntos de unión donde se añade el comportamiento del aspecto, todos los cuales se deben agrupar en un mismo punto de corte. Posteriormente por medio del aviso se implementó el comportamiento transversal, el cual se ejecuta cuando se

llegue a un punto de unión que satisfaga la definición del punto de corte. Todo lo anterior queda dentro de la clase aspecto, parte de la construcción realizada en lenguaje AspectJ se puede ver en Anexo G.

6. RESULTADOS

Se requiere en esta investigación obtener métricas que determinen la efectividad de incorporar Orientación a Aspectos. Para esto se realizaron dos tipos de mediciones, la primera consiste en un cuestionario que contiene parte del análisis y diseño de un módulo de un proyecto de software seleccionado, comparando casos de uso donde no se incluye Orientación a Aspectos con otros casos de uso donde si se incluyen. La segunda medición corresponde a métricas aplicadas a código donde no se incluye Orientación a Aspectos y medición a código donde si se incluye.

6.1. MEDICIÓN DE BENEFICIOS A TRAVÉS DE CUESTIONARIO

Como primera medición se aplicará una encuesta que permita evaluar el impacto de incorporar Orientación a Aspectos en las etapas de análisis y diseño desde el punto de vista de encuestados que correspondan a analistas y desarrolladores de software con conocimiento en Orientación a Objetos. El cuestionario aplicado en esta tesis está basado en investigaciones realizadas por un grupo perteneciente a la universidad PUC-Rio de Brasil y el cual propone distintas métricas de calidad.

Los objetivos de este cuestionario son los siguientes:

- Verificar si los encuestados ven los beneficios obtenidos por medio de la incorporación de Orientación a Aspectos en las etapas de análisis (Casos de Uso) y diseño (Modelo de Clases y Diagrama de Secuencias).
- Medir el grado de conocimiento que los encuestados tienen de la Orientación a Aspectos.

Se optó por un cuestionario individual por escrito en donde no hay intervención del encuestador ya que se utilizan preguntas con imágenes y términos que son familiares para el encuestado y que no requieren ser explicados. Este cuestinoario incluye preguntas con respecto a los diagramas y modelos generados en las etapas de análisis y diseño, basándose en Orientación a Objetos del módulo de Administración de Número de Folio del sistema DIPS Viajero, y comparándolos con los diagramas que incluyen Orientación a Aspectos. El cuestionario utilizado es el que se muestra en el Anexo E.

El cuestionario está dividido en las siguientes secciones: la primera sección corresponde a los datos personales del encuestado, principalmente datos laborales. Se incluyen en esta sección preguntas con respecto a los conocimentos del encuestado con respecto a OO y OA. Dentro de esta misma sección se muestran las variables categóricas numerales para cada pregunta, estás variables conforman la siguiente escala de medición:

0: No corresponde

1: En desacuerdo

2: Parcialmente de acuerdo
3: De acuerdo
4: Totalmente de acuerdo

La segunda sección apunta a preguntas de información que permitan medir el grado de conocimiento del encuestado con respecto a Orientación a Objetos y con respecto a Orientación a Aspectos.

La tercera sección del cuestionario denominado Casos de Uso está conformada por preguntas que apuntan a determinar el nivel de claridad con que el encuestado puede ver más definido ciertos requerimientos en etapas de análisis de un proyecto utilizando OA.

En la cuarta sección se pueden observar preguntas con respecto al Modelo de Clases que indagan sobre si el encuestado encuentra más óptimo y de más fácil de entender el modelo utilizando OA.

La quinta sección del cuestionario se refiere a los Diagramas de Secuencia, consultando al encuestado si ciertos requerimientos se ven más definidos utilizando diagramas que incorporan OA.

La última sección del cuestionario otorga un espacio para que el encuestado realice observaciones con respecto a las respuestas dadas.

6.1.1. UNIVERSO DE ENCUESTADOS

La muestra corresponde al universo de encuestados, que es un subgrupo de personas seleccionadas para participar en el estudio. En general se puede afirmar que las características de la muestra permiten realizar cálculos que llevan a probar una hipótesis. Dos evidentes limitaciones para el uso de muestras son el presupuesto y el tiempo. Para la obtención de una muestra se realizan los siguientes pasos [28]:

1.- Definición de la población objetivo
Se buscó un universo de ingenieros informáticos con conocimientos en modelamiento con Orientación a Objetos. La muestra debía contener ingenieros con más de cinco años de experiencia de trabajo, dentro de este grupo más de uno de los encuestados que fuera actualmente jefe de proyectos desarrollados en Orientación a Objetos.

Se buscó además que todos los encuestados hayan desarrollado proyectos en Orientación a Objetos utilizando lenguaje UML.

2.- Determinación del marco de muestreo

En el marco de muestreo se buscó ingenieros de la rama informática que formaran parte de proyectos que se están diseñando para Aduana en forma paralela al proyecto del sistema DIPS Viajero. Abarcó ingenieros que actualmente trabajarán en la Región Metropolitana y en Valparaíso.

3.- Elección de una técnica de muestreo

Se utilizó una técnica de muestreo no probabilística ya que se seleccionaron ingenieros que no fueron escogidos en forma aleatoria, sino que cumplieran ciertas características.

4.- Determinación del tamaño de la muestra

Para los diseños de investigación exploratorios, como el caso de este trabajo que utiliza investigación cualitativa, el tamaño de la muestra es pequeño. Debido a que en los proyectos de investigación el tiempo y los recursos son limitados, el tamaño de la muestra también dependerá de estos factores, para este caso debido al tiempo no fue posible tomar una muestra más grande llegando sólo a ser cinco encuestados. Por tratarse de un muestreo no probabilístico se tomó en cuenta encuestados que cumplieran las características de la población objetivo.

5.- Ejecución del proceso de muestreo

Para llevar a cabo el proceso de muestreo se utilizó el correo electrónico enviando la encuesta a personas seleccionadas que cumplieran las características de la muestra. En algunos casos que se necesitó, se enviaron además explicaciones anexas para llenar la encuesta.

6.2. RESULTADOS DE CUESTIONARIOS

Los resultados de las distintas preguntas que conforman el cuestionario fueron diagramadas utilizando gráfico de Pie, en donde el 100% corresponde a los cinco encuestados.

6.2.1. RESULTADOS DE CUESTIONARIOS CONSULTA DE CONOCIMIENTOS OO

La primera sección del cuestionario está dirigida a ver el grado de conocimiento en Orientación a Objetos. El resultado del cuestionario demuestra que hay un alto grado de conocimiento de Orientación a Objetos ya que 4 personas (80% del total) del grupo encuestado lo conocen.

6.2.2. RESULTADO DE CONSULTAS EN CONOCIMIENTOS EN OA

Dentro de la primera sección del cuestionario se trató de conocer si los encuestados tenían nociones de Orientación a Aspectos, y si habían participado en algún proyecto en donde se hiciera uso de Orientación a Aspectos. Entre los encuestados había un parte que poseía conocimiento del término Orientación a Aspectos y con nociones de lo que significa, pero 2 personas (40%) no conocían el significado claro de Orientación a Aspectos.

Uno de los encuestados indica haber participado en un proyecto con Orientación a Aspectos, pero la mayoría señala no haber participado en proyectos que la incorporan. Es de gran valor conocer que en Chile existan proyectos que han utilizado Orientación a Aspectos, sería necesario realizar futuras investigaciones que midieran los beneficios en proyectos ya construidos bajo Orientación a Aspectos.

6.2.3. RESULTADO CONSULTAS DE OA EN CASOS DE USO

La segunda sección del cuestionario mide los beneficios de incorporar Orientación a Aspectos en los casos de uso. Para realizar esta sección de preguntas se utilizó el modelo de casos de uso del módulo de Administración de Números de Folio, utilizando sólo Orientación a Objetos y un segundo modelo que incluyera además Orientación a Aspectos.

Para los modelos de casos de uso con Orientación a Aspectos se utilizaron los aspectos detectados para el módulo de Administración de Números de Folio, los cuales son: Login, Validación, Auditoria y Registro de Errores. Las consultas buscan determinar si incluyendo OA se ve más claramente en los casos de uso los aspectos identificados. La totalidad de los resultados que corresponden a las consultas de cada uno de los aspectos se encuentran en el Anexo F, aquí se analizan algunos de ellos.

Se sacó un promedio de las respuestas obtenidas de las consultas que pedían a encuestados ver que tan definidos estaban los aspectos Login, Auditoria, Validación y Registro de Errores en Casos de Uso que incluían OA, comparándolos con los casos de uso que sólo utilizaban OO. En el resultado se puede ver que un 30% está totalmente de acuerdo y un 45% está de acuerdo en que están más definidos los aspectos utilizando OA, esto señala que los encuestados encuentran que OA aporta un beneficio a los casos de uso para la definición de los aspectos.

Dentro de esta misma sección se consultó a los encuestados que si incluir OA requeriría de más tiempo para la confección de los casos de uso.

La última pregunta relacionada a casos de uso consulta a los encuestados si incorporarían OA para los casos de uso de proyectos futuros. Los encuestados no negaron utilizar la posibilidad de utilizar a futuro OA en casos de uso, ya que 40% estuvo totalmente de acuerdo y 40% de acuerdo, esto demuestra una aceptación de OA por parte de los encuestados.

6.2.4. RESULTADO CONSULTAS DE OA EN MODELO DE CLASES

En la tercera sección del cuestionario se buscaba ver el beneficio que los encuestados perciben al incorporar OA en el modelo de clases. Se utilizó el modelo de clases del módulo de Administración de Número de Folio, comparando el modelo de clases sin OA por el modelo de clases con OA, teniendo presente los aspectos Login, Validación, Auditoria y Registro de Errores.

Se preguntó a los encuestados si el modelo de clases mejoraba incorporando OA, principalmente centrando esta mejora para los aspectos (Login, Validación, Auditoria y Registro de Errores). Los encuestados en su mayoría se declararon estar parcialmente de acuerdo, esto debido a que por comentarios emitidos no era en forma clara el beneficio obtenido de incorporar OA, encontraron que el modelo se hacía más complejo.

Los beneficios se hubiesen visto en forma más clara si se incorporarán al modelo de clase todos los métodos de las clases aspectos, ya que un gran beneficio es la reducción de métodos (esto se podrá apreciar claramente más adelante en el capítulo de métricas).

Se consultó a los encuestados si apreciaban que la construcción del modelo de clases con OA requería de más tiempo. Dentro de las respuestas dadas por los encuestados 3 de estos (60%) está parcialmente de acuerdo que al incorporar OA en el modelo de clase requiere de más tiempo y 1 de los encuestados (20%) de acuerdo, por lo tanto 4 de los encuestados encuentran que requiere de más tiempo incorporar OA en el modelo de clase.

Se consultó a los encuestados si incorporaría OA en modelos de clases de futuros proyectos. Hubo una aceptación por parte de los encuestados de incorporar OA en modelos de clases de proyectos futuros, 2 de los encuestados (40%) está totalmente de acuerdo y 1 de los encuestados (20%) de acuerdo, por lo que 3 de los encuestados adoptaría OA, lo que implica que los encuestados ven beneficios de adoptar OA sin ver como negativo el tiempo adicional de un modelo de clases con OA.

6.2.5. RESULTADO CONSULTAS DE OA EN DIAGRAMA DE SECUENCIAS

La cuarta sección del cuestionario se enfoca en los diagramas de secuencia que incorporan OA.

Se preguntó a los encuestados por medio del cuestionario si los diagramas de secuencia que incorporan OA muestran en forma más definida los aspectos identificados (Login, Validación, Auditoria y Registro de Errores) que los diagramas de secuencia que no poseen OA.

Se consultaron en forma separada cada uno de los aspectos y si estos se veían más claramente en los diagramas de secuencia con OA, los resultados se encuentran en Anexo F.

Los resultados obtenidos muestran que los encuestados ven los aspectos en forma más definida incorporando OA en los diagramas de secuencia, lo que correspondería a un beneficio más dado por OA.

Se consultó además a los encuestados si desde su punto de vista se requiere de más tiempo para desarrollar diagramas de secuencia con OA. Los encuestados observan que incorporar OA a los diagramas de secuencia requiere de más tiempo comparándolo con los diagramas de secuencia que no incluyen notación OA.

Finalmente se consultó a los encuestados si incorporarían a futuros proyectos la notación OA en los diagramas de secuencia. A pesar de ver que requiere de más tiempo diseñar diagramas de secuencia con OA, 4 de los encuestados señalan que incorporarían OA en futuros diagramas de secuencia, ya que 2 de los encuestados (40%) está totalmente de acuerdo de incorporarlo y 2 de los encuestados (40%) está de acuerdo.

6.2.6. OBSERVACIONES INDICADAS DENTRO DE ENCUESTA

Se añadió una sección adicional en donde los encuestados podían indicar observaciones con respecto al cuestionario y la incorporación de Orientación a Aspectos en análisis y diseño. Dentro de las observaciones se señaló que al incorporar OA se requiere de un análisis y diseño más extenso, pero también se permite ver en forma más detallada y sencilla los requerimientos de tipo aspecto, disminuyendo el factor de cambios que se produce durante las etapas de construcción.
Otra de las observaciones es que se tiene conocimiento de que aún está en etapa inicial todo con respecto a OA pero que en el futuro se ve su incorporación a los proyectos. No se ve como un reemplazo de OO sino más bien como un complemento.

También se señaló que desconociendo OA le fue fácil entender los casos de uso y diagramas de secuencia que incorporaban OA, pero no así en el modelo de clases.

6.2.7. ANÁLISIS DE RESULTADOS DE CUESTIONARIO

Para efectuar el análisis de los resultados de la encuesta se toma como referencia la hipótesis de este trabajo: "Incluir e integrar Orientación a Aspectos en las fases de análisis, diseño y construcción de proyecto de software construido en Orientación a Objetos trae consigo más beneficios en el desarrollo y mantención, que un producto de software sólo construido en base a Orientación a Objetos". A partir de esta hipótesis se consultó a los encuestados diferencias percibidas entre etapas de análisis y diseño de un proyecto que sólo utilizara OO y otro que utilizara OO complementado con OA. Los resultados del cuestionario indican que más de la mitad de los encuestados ven en forma más definida los requerimientos aspectos cuando se incorpora la notación de Orientación a Aspectos. Esto genera un beneficio a las etapas de análisis y diseño, principalmente si se busca dejar en forma más independiente los requerimientos correspondientes a aspectos, para un mejor análisis de estos y poder contar con un diseño de aspectos que permita implementarlos en la etapa de construcción.

Los encuestados indicaron un aumento en el tiempo de desarrollo de los casos de uso, modelo de clases y diagrama de secuencias, al incorporar la notación utilizada por Orientación a Aspectos; pero a pesar de este aumento en el tiempo se vieron muy inclinados a incorporar Orientación a Aspectos en futuros proyectos.

Debido a que el universo de encuestados es reducido se requiere a futuro realizar más encuestas tanto entre profesionales informáticos como a profesionales de otras áreas, posibles contrapartes de proyectos informáticos desarrollados con UML y Orientación a Objetos.

6.3. MEDICIÓN DE BENEFICIOS A TRAVÉS DE MÉTRICAS

6.3.1. MÉTRICAS APLICADAS EN EL PROYECTO

Las métricas serán aplicadas a las clases aspectos identificadas en este proyecto. Se utilizarán dos tipos de métricas, las primeras denominadas métricas pertenecientes al Modelo de Calidad y que corresponden a métricas que miden la calidad del sistema. Las segundas denominadas Métricas Adicionales, estando compuestas por dos fórmulas, en donde la primera permite medir el beneficio obtenido al incorporar OA y la segunda calcula el porcentaje de limpieza de código al utilizar OA.

a) Métricas pertenecientes a Modelo de Calidad

No se aplicaron la totalidad de los factores de la ISO 9126, ya que el resto de los factores no tienen relación directa con OA. Para la medición del factor de calidad de tamaño se optó por contar las líneas de código y no la medición por medio de los casos de uso debido a que sólo se desarrollaría un solo módulo del sistema, para la medición de casos de uso se requiere de mayor cantidad de módulos de sistemas desarrollados con casos de uso. Las métricas de Jianjun Zhao sólo se centran en acoplamiento y cohesión, no se utilizaron en este estudio ya que las métricas aplicadas por Zacaria están basadas en las métricas propuestas por Zhao, siendo más simples de aplicar las de Zacaria.

1. Difusión del aspecto en los componentes (CDC)

Proyecto sin OA

Nombre de Clase	Tipo de Clase
FuncionarioAduana	Clase normal
RangoNrosFolio	Clase normal
Estado	Clase normal
Aduana	Clase normal
Auditoria	Clase normal
Errores	Clase normal

CDC_sin_OA = Número de clases aspecto + Número de clases que acceden a clase aspecto

CDC_sin_OA = 0 + 6

CDC_sin_OA = 6

Proyecto con OA

Nombre de Clase	Tipo de Clase
FuncionarioAduana	Clase normal
RangoNrosFolio	Clase normal
Estado	Clase normal
Aduana	Clase normal
Login	Clase Aspecto
Validcion	Clase Aspecto
Auditoria	Clase Aspecto
RegistroErrores	Clase Aspecto

CDC_con_OA = Número de clases aspecto + Número de clases que acceden a clase aspecto

CDC_con_OA = 4 + 4

CDC_con_OA = 8

2. Difusión de aspectos en operaciones (CDO)

Tomando como referencia aspecto Auditoria

Nombre de método que invoca Auditoria	Clase
addNroFolio()	RangoNroFolio
transferNroFolio()	RangoNroFolio
blockNroFolio()	RangoNroFolio

Nombre de método de Auditoria	Clase aspecto
logAddNroFolio()	Auditoria
logTransferNroFolio()	Auditoria
logBlockNroFolio()	Auditoria
logGeneral()	Auditoria

CDO = Número de métodos de una clase aspecto + Número de métodos que invocan métodos de clase aspecto

CDO = 3 + 4

CDO = 7

3. Difusión de aspectos en LOC (CDLOC)

Tomando como evaluación aspecto Auditoria

Puntos de transición de Auditoria	Clase
Pointcut addNroFolio()	Auditoria
Pointcut transferNroFolio()	Auditoria
Pointcut blockNroFolio()	Auditoria

CDLOC_con_OA = Número de puntos de transición

CDLOC_con_OA = 3

4. Tamaño de Vocabulario (VS)

Proyecto sin OA

Nombre de Clase	Tipo de Clase
FuncionarioAduana	Clase normal
RangoNrosFolio	Clase normal
Estado	Clase normal
Aduana	Clase normal
Auditoria	Clase normal
Errores	Clase normal

VS_sin_OA = Número de clases existentes

VS_sin_OA = Clases no aspectos + Clases aspectos

VS_sin_OA = 6 + 0

VS_sin_OA = 6

Proyecto con OA

Nombre de Clase	Tipo de Clase
FuncionarioAduana	Clase normal
RangoNrosFolio	Clase normal
Estado	Clase normal
Aduana	Clase normal
Login	Clase Aspecto
Validcion	Clase Aspecto
Auditoria	Clase Aspecto
RegistroErrores	Clase Aspecto

VS_con_OA = Número de clases existentes

VS_con_OA = Clases no aspectos + Clases aspectos

VS_con_OA = 4 + 4

VS_con_OA = 8

5. Líneas de Código (LOC)

Tomando de referencia aspecto Auditoria y la interacción de éste con las otras clases

Nombre de método que invoca Auditoria	Clase
addNroFolio()	RangoNroFolio
transferNroFolio()	RangoNroFolio
blockNroFolio()	RangoNroFolio

LOC = Número de líneas de código

LOC (sin aspectos) = 20 + 16 + 12

LOC (sin aspectos) = 48

Nombre de método de Auditoria	Clase aspecto
logAddNroFolio()	Auditoria
logTransferNroFolio()	Auditoria
logBlockNroFolio()	Auditoria
logGeneral()	Auditoria

LOC (con aspectos) = 11 + 7 + 3 + 14

LOC (con aspectos) = 35

6. Número de atributos (NOA)

NOA_1 (Aduana) = 5

NOA_2 (RangoNrosFolio) = 2

NOA_3 (FuncionarioAduana) = 6

NOA_4 (Estado) = 2

NOA_ 5 (Login) = 4

NOA_6 (Auditoria) = 9

NOA_7 (RegErrores) = 4

NOA_8 (Validacion) = 3

Total NOA = NOA_1 + NOA_2 + NOA_3 + NOA_4 + NOA_5 + NOA_6 + NOA_7 + NOA_8

Total NOA = 5 + 2 + 6 + 2 + 4 + 9 + 4 + 3

Total NOA = 35

7. Peso de las operaciones por componente (WOC)

   WOC (sin aspectos) = 12

   WOC (con aspectos)= 4

   Estos parámetros corresponden a los datos de auditoría que deben ser registrados. Para el caso de no utilizar OA, los parámetros deben estar traspasados por cada uno de los métodos, al ser tres métodos los cuatro parámetros pasan a ser doce. En cambio, utilizando OA sólo se traspasan por medio del método aspecto.

8. Acoplamiento entre Componentes (CBC)

   CBC_1 (Login) = 1

   CBC_2 (Auditoria) = 3

   CBC_3 (RegErrores) = 2

   CBC_4 (Validacion) = 2

   Total CBC = CBC_1 + CBC_2 + CBC_3 + CBC_4

   Total CBC = 1 + 3 + 2 + 2

   Total CBC = 8

9. Profundidad del árbol de herencia (DIT)

   Esta métrica no será calculada debido a que dentro del módulo de Administración de Número de Folio no existen herencias.

10. Falta de cohesión en las operaciones (LCOO)

    Esta métrica no será aplicada en esta evaluación.

    LCOO = Número de métodos de cada clase que no están cohesionados

b) Métricas adicionales

   Para estas métricas sólo es tomada en cuenta el aspecto Auditoria.

   La primera métrica adicional mide el porcentaje de código que se reduce al introducir aspectos, utilizando el tamaño de las clases, medido en Kb. Si el valor del resultado es mayor a 1 significa que hay un impacto positivo al usar Orientación a Aspectos:

La fórmula de esta métrica es la siguiente:

Factor de Desarrollo Java = 10 horas/programador
Factor de Desarrollo AspectJ = 2 horas/programador

$$Beneficio\_POA = \frac{Tamaño\_sin\_aspectos * Factor\_de\_Desarrollo\_Java}{Tamaño\_con\_aspectos * Factor\_de\_Desarrollo\_AspectJ}$$

$$Beneficio\_POA = \frac{48 * 10}{35 * 2} = 6,86$$

Tamaño_con_aspectos = Tamaño_Funcionalidad_Básica + Tamaño_Aspecto
Tamaño_con_aspectos = 21 + 14

La segunda métrica adicional corresponde a la medición del nivel de limpieza de código al utilizar OA. La fórmula es la siguiente:

$$Limpieza = \frac{(Tamaño\_sin\_aspectos - Tamaño\_Funcionalidad\_Básica) * 100}{Tamaño\_sin\_aspectos}$$

$$Limpieza = \frac{(48 - 21) * 100}{48} = 56,25$$

6.3.2. RESUMEN DE RESULTADOS DE MÉTRICAS APLICADAS

a) Métricas pertenecientes a Modelo de Calidad

Los resultados obtenidos a través de las métricas se presentan en Tabla 1 para permitir una comparación entre implementación que utiliza OA con una implementación que no la incorpora.

	CDC	CDO	CDOLOC	VS	LOC	NOA	WOC	CBC	DIT	LCOO
Código sin OA	6	-	-	6	48	35	12	-	-	-
Código con OA	8	7	3	8	35	35	4	8	-	-

Tabla 1: Tabla de resultados de métricas

b) Métricas adicionales

Resultados arrojados por las métricas adicionales son los siguientes:

- Beneficio_POA = 6,86
- Limpieza = 56,25

6.3.3. ANÁLISIS DE RESULTADOS DE MÉTRICAS

En el caso de implementación sin OA se utilizó código Java para implementar el módulo de Administración de Número de Folio, para esta investigación sólo se implementó el aspecto de Auditoría. El código fue construido utilizando la herramienta Eclipse 3.3 junto con JDK 1.5.0. Para el caso del código con OA se utilizó herramienta My Eclipse Enterprise Workbench 6.0 con extensión AspectJ 1.5.3 para utilizar AspectJ como lenguaje de aspecto.

Analizando las métricas utilizadas se observa que el número de clases aumenta al incorporar las clases aspectos pero esto se compensa en el número de líneas de código, ya que una implementación con aspectos reduce significativamente el número de líneas de código de 48 a 35 (tomando en cuenta sólo implementación de aspecto de Auditoría). El número de parámetros traspasados por métodos también se reduce al utilizar OA, ya que basta sólo pasar los parámetros por medio del aspecto y no repetirlos en cada método.

La primera métrica adicional demuestra que existe un beneficio de incorporar OA ya que dio como resultado 6,86, al obtener un número positivo mayor a 1 corresponde a existencia de beneficio. En cuanto a la segunda métrica adicional se obtuvo un 56,25 % de limpieza de código incorporando OA.

7. CONCLUSIONES

Este estudio se enfocó en investigar la incorporación de Orientación a Aspectos en las etapas de análisis, diseño y construcción de un proyecto informático.

En la fase de captura de requerimientos se identificaron los aspectos candidatos a partir de los requerimientos obtenidos, los aspectos identificados fueron: Login (autentificación) ,Validación (de número de folio) , Auditoría y Registros de Errores. En las etapas de análisis y diseño se utilizó UML con OA para modelamiento de requerimientos.

Para la parte de construcción se utilizó el análisis y diseño del módulo de Administración de Números de Folio, del cual se codificó una parte en código Java, y además la misma parte utilizando Java con extensión AspectJ. Se pudo mostrar que hubo una reducción en la cantidad de líneas de código y un código más encapsulado que permitía que los requerimientos de tipo aspectos fueran reusables y de fácil modificación.

Se utilizaron dos formas de medir los beneficios otorgados al incorporar OA en el proyecto. La primera medición consistió en un cuestionario realizado a ingenieros con conocimiento en Orientación a Objetos, y que buscaba ver la opinión de éstos. Los resultados del cuestionario indicaron mayoritariamente que utilizando la notación de Orientación a Aspectos los requerimientos se ven más definidos; esto genera un beneficio a las etapas de análisis y diseño, principalmente si se busca dejar en forma más independiente los requerimientos aspectos, para un mejor análisis de éstos y poder contar con un diseño de aspectos que permita implementarlos adecuadamente en la etapa de construcción. Los encuestados indicaron ver un aumento en el tiempo de desarrollo de los casos de uso, modelo de clases y diagramas de secuencia al incorporar la notación utilizada por Orientación a Aspectos; pero a pesar de este aumento en el tiempo se vieron muy inclinados a incorporar Orientación a Aspectos en futuros proyectos.

La segunda medición aplicada al proyecto correspondía a un conjunto de métricas que permitieran medir los beneficios que otorga el paradigma Orientación a Aspectos al incorporarlo en un proyecto de software que utiliza Orientación a Objetos con notación UML en la etapa de construcción de código. Analizando las métricas utilizadas se observa que el número de clases aumenta al incorporar las clases aspectos pero esto se compensa en el número de líneas de código, ya que una implementación con aspecto reduce significativamente este atributo. También se observa una reducción en el número de parámetros traspasados por método, debido a que con OA basta traspasarlos al aspecto.

7.1. TRABAJO FUTURO

Como trabajo futuro sería de gran avance desarrollar un estándar para confeccionar casos de uso, modelos de clases y diagramas de interacción que permitan incorporar aspectos recopilados en los requerimientos. Incorporando aspectos en etapas tempranas del proyecto se permitirá construir aplicaciones basados en Orientación a Aspectos.

Dentro del diseño Orientado a Aspectos se debe avanzar en el desarrollo arquitectónico de los sistemas tomando en cuenta los últimos lenguajes que incorporan aspectos.

Se debe trabajar además en la forma de implementar de manera eficiente los aspectos en las bases de datos, trabajando sobre bases de datos Orientados a Objetos.

Sería interesante a futuro realizar mediciones que permitan ver los beneficios de OA en la fase de pruebas, realizada por el equipo de aseguramiento de calidad (QA). Verificar si utilizando OA se facilita el desarrollo de las pruebas en los distintos proyectos en que participe el equipo QA; además medir si se produce una disminución de errores en los proyectos que integran OA. Hacer mediciones que permitan verificar si se reduce el tiempo de corrección de errores que posean aspectos, comparando los tiempos de proyectos que posean OA versus los que no tengan incluido OA.

Junto a lo anterior realizar un cuestionario y aplicarlo en empresas informáticas del país que permitiera recopilar el grado de conocimiento y uso de Orientación a Aspectos, de igual forma medir el grado de aceptación a su uso.

8. REFERENCIAS

[1] Basch M., Sanchez A., Paper Incorporating Aspects into UML, Agenda of the Workshop on Aspect-Oriented Modeling with UML Marzo 18, 2003

[2] Bases de Licitación de proyecto DIPS Viajero publicadas en sitio de Servicio Nacional de Aduanas http://www.aduana.cl/prontus_aduana/site/artic/20071119/pags/20071119101716.html

[3] Betina Haak, Miguel Díaz, Claudia Marcos, Jane Prior, Identificación Temprana De Aspectos, ISISTAN Instituto de Sistemas Tandil, Facultad de Ciencias Exactas, UNICEN

[4] Christina Chavez, Carlos Lucena, A Metamodel for Aspect-Oriented Modeling, 2002

[5] Claudio Sant'Anna, Alessandro Garcia, Christina Chavez, On the Reuse and Maintenance of Aspect-Oriented Software: An Assessment Framework, PUC-Rio, Computer Science Department, Brasil, 2003

[6] Compendio de Normas de Aduana, Capítulo 3 punto 12, publicado en http://www.aduana.cl

[7] Fernando Asteasuain, Bernardo E. Contreras, Elsa Estévez, Pablo R. Fillottrani, Programación Orientada a Aspectos: Metodología y Evaluación, Departamento de Ciencias e Ingeniería de la Computación Universidad Nacional del Sur, Argentina, 2003.

[8] Frans Sanen, Eddy Truyen, Wouter Joosen, Andrew Jackson, Andronikos Nedos, Siobhán Clarke Distributed Systems Group, Trinity College Dublín, Neil Loughran, Awais, Rashid, Classifying and documenting aspect interactions, Distrinet Research Group, Department of Computer Science K.U.Leuven Celestynenlaan 200A, Computing Department Infolab 21,Lancaster University

[9] G. Kiczales, C. Lopes. Aspect-Oriented Programming with AspectJTM –Tutorial
Xerox Parc, http://www.eclipse.org/aspectj/

[10] Gefei Zhang , Hubert Baumeister, Nora Koch, Alexander Knapp, Aspect-Oriented Modeling of Access Control en Web Aplications, Institut f ür Informatik, Ludwig Maximiliano Universit at Müchen, 2005

[11] Groher I., Schulze S., Generating Aspect Code from UML Models, Agenda of the Workshop on Aspect- Oriented Modeling with UML Marzo 18, 2003

[12] Jacobson 2005, http://www.jaczone.com/papers/, Dr. Ivar Jacobson, co-founder of Jaczone AB, is one of the great thought-leaders in the software world where he has made several seminal contributions. He is one of the fathers of components and component architecture, use cases, modern business engineering, the Unified Modeling Language and the Rational Unified Process

[13] J. Suzuki, Y. Yamamoto. Extending UML with Aspect: Aspect Support in the Design Phase. 3er Aspect-Oriented Programming (AOP) Workshop at ECOOP'99

[14] Kiczales, G. Aspect-Oriented Programming. 1997. Procedings of ECOOP, Springer erang. LNCS 1241

[15] Lieberherr Karl, Adaptive Object-Oriented Software The Demeter Method, 1996 University Boston http://www.ccs.neu.edu/research/demeter/biblio/dem-book.html

[16] Lieberherr Karl, Aspect-Oriented Programming with Aspectual Methods, 2001

[17] Lieberherr Karl, Controlling the Complexity of Software Designs, 2004, 12 Mayo (Publicación Web)

[18] Mohamed M. Kandé, Jörg Kienzle and Alfred Strohmeier, From AOP to UML: Towards an Aspect-Oriented Architectural Modeling Approach, Software Engineering Laboratory, Swiss Federal Institute of Technology Lausanne, Switzerland, 2002

[19] OMG Unified Modeling Language Specification, version 1.4, September 2001

[20] Paper UML Profile for Aspect-Oriented software development, Agenda of the Workshop on Aspect-Oriented Modeling with UML, Marzo 18, 2003

[21] Reina Quintero, Antonia María 2000 Departamento de Lenguajes y Sistemas Informáticos, Universidad de Sevilla

[22] Samoieri, Roberto Hernández, 1998, Metodología de la Investigación

[23] Schaum Joyanes, Director de Departamento de Lenguajes y Sistemas Informáticos, Universidad Pontificia de Salamanca, Madrid, Programación orientada a objetos versus programación estructurada: C++ y algoritmos, 2006.

[24] Silvio Meier, Martin Gliz, Problems when Introducing Aspect-Oriented Constructs in Models Functional Requeriments and Possible Solution to these Problems, Departament of Informatics, University, Switzerland of Zurich, 2005

[25] Zacaria A. y Dr. Hoda Hosny , Metrics for Aspect-Object Software Design, The American University in Cairo, Agenda of the Workshop on Aspect-Oriented Modeling with UML Marzo 18, 2003

[26] Zhao Jianjun, Measuring Coupling in Aspect-Oriented Systems, Department of Computer Science and Engineering,Fukuoka Institute of Technology, Japan 2004

[27] Zhao Jianjun, Measuring Aspect Cohesion, Department of Computer Science and Engineering,Fukuoka Institute of Technology, Japan 2004

[28] Naresh K. Malhotra, José Francisco Javier Dávila Martínez, Enrique (ed.) Quintanar Duarte, Magda Elizabeth Treviño Rosales, Rocío Moreno Sanabria, Investigación de mercados: un enfoque aplicado, Pearson Educación, 2004

[29] Media Net Software, Spring Framework (presentación power point), Junio 2008, Madrid España

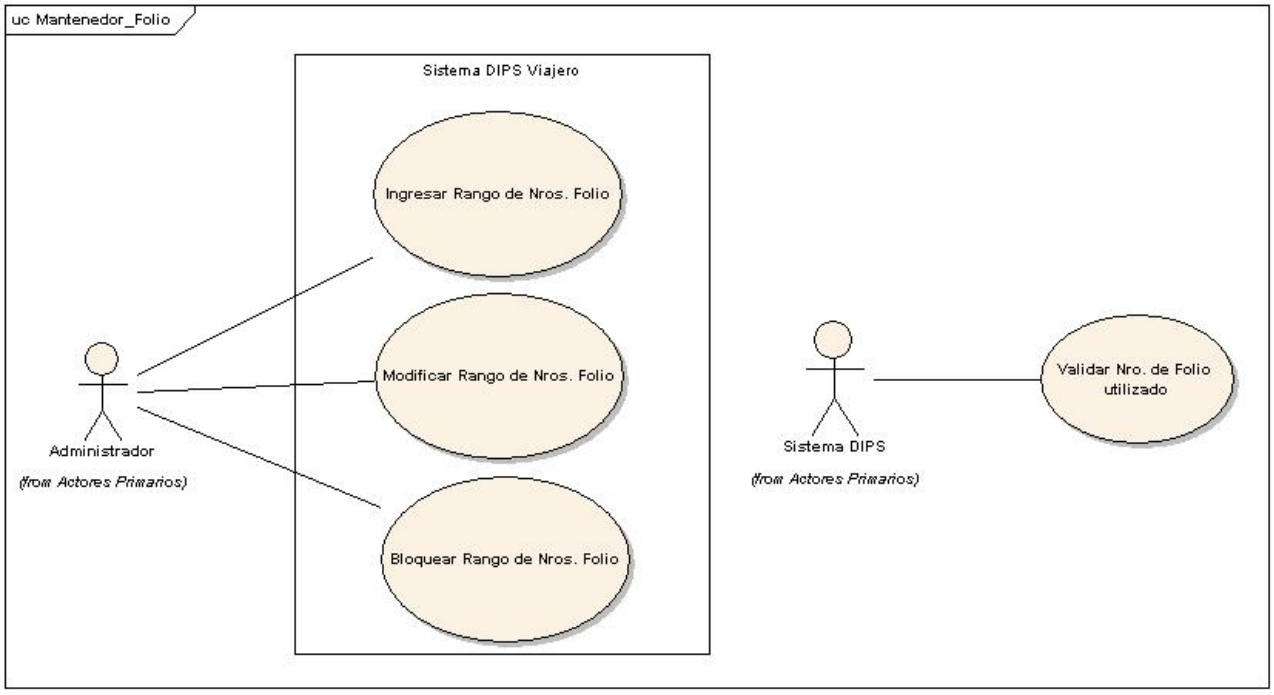
ANEXO A
Análisis
Orientación a Objetos

1.1. CASOS DE USO

1.1.1. Lista de Casos de Uso de módulo seleccionado (Módulo de Rangos de Nros. De Folio)

- Caso de Uso 1: Ingresar Rango de Números de Folio
- Caso de Uso 2: Modificar Rango de Números de Folio
- Caso de Uso 3: Bloquear Rango de Números de Folio
- Caso de Uso 4: Validar Número de Folio utilizado

1.1.2. Diagrama de Caso de Uso: Módulo Mantenedor Números de Folio



1.1.3. Descripción de Caso de Uso: Ingresar Rango de Nros. Folio

Objetivo:	Este caso de uso permite ingresar números de folio repartidos para una cierta aduana. En los casos de Avanzada se debe especificar a cual corresponde. El ingreso debe realizarse a través de la Dirección Nacional de Aduanas (DNA).	
Escenarios:		
Flujo Básico	Flujo Normal	1. El actor debe indicar Aduana a la cual se va asignar rango de número de folio. 2. El actor debe indicar número inicial de rango de números de folio. 3. El actor debe indicar número final de rango de números de folio. 4. El actor solicita a sistema que guarde datos de rango de números de folio asignado a aduana.
Flujo Alternativo	Error - Rango a bloquear	4. Si dentro del rango de números indicados existen números que han sido usados como número de identificación en DIPS Viajero aceptadas, rechazadas o anuladas.

	posee números utilizados	4.1 El sistema despliega mensaje indicando que dentro de rango a ingresar existen números de folio que ya han sido utilizados por DIPS Viajero.
Flujo Alternativo	Error - Rango de números incorrecto	4. Si los datos de rango de números esta incorrecto. - Si se ingresa número final de rango menor a número inicial de rango, el sistema deberá indicar error en rango indicado ya que número de rango final debe ser mayor a número de rango inicial. - Si formato de número de folio indicado en reglas de negocio no corresponde a números de rango se debe indicar error.
Flujo Alternativo	Error - Rango de números bloqueados	4. Si se intenta bloquear rango de números bloqueados. 4.1 Si administrador trata de bloquear números de rango que han sido bloqueados el sistema deberá indicar error.
Restricciones:		
Pre-Condición	Usuario Logeado	El usuario debe estar logeado y con permisos en el sistema.
Post-Condición	Rango de Números de Folio ingresado	El Rango de números de folio ha sido ingresado en el sistema.

1.1.4. Descripción de Caso de Uso: Modificar Rango de Nros. Folio

Objetivo:	Este caso de uso permite a actor poder modificar cierto rango de números de folio de una Aduana a otra. La modificación debe realizarse a través de la Dirección Nacional de Aduanas (DNA).
Escenarios:	
Flujo Básico	Flujo Normal	1. El actor debe seleccionar Aduana que se desea modificar. 2. El actor debe indicar nuevo número final de rango, los restantes número serán asignados a otra aduana. Siempre se asignarán los últimos números del rango, no podrá ser asignado los números del centro del rango. 3. El actor debe seleccionar a que Aduana será traspasado los números de folio. 4. El actor solicita a sistema sean traspasados el rango de números de folio indicado. 5. El sistema solicita confirmación desplegando mensaje con nombre de aduana y rango de números que serán traspasados. 6. El actor confirma traspaso de rango de números.
Flujo Alternativo	Error - Rango indicado ha sido utilizado previamente	4. Si dentro del rango de números indicados existen números que han sido usados como número de identificación en DIPS Viajero aceptadas, rechazadas o anuladas. 4.1 El sistema despliega mensaje indicando que dentro de rango a traspasar existen números de folio que ya han sido utilizados por DIPS Viajero.
Flujo Alternativo	Error - Número de rango indicado no pertenece a Aduana	4. Si el número indicado de rango a traspasar no pertenece a la Aduana origen indicada. 4.1 Si el número de folio de rango ha traspasar de una Aduana origen no corresponde a esa Aduana, el sistema indicará error.
Flujo Alternativo	Error - Número de rango bloqueado	4. Si se intenta traspasar rango de números bloqueados. 4.1 Si administrador trata de traspasar números de rango que han sido bloqueados el sistema deberá indicar error.
Restricciones:		
Pre-Condición	Usuario Logeado	El usuario debe estar logeado y con permisos en el sistema.
Post-Condición	Rango de Número de Folio modificado	El Rango de números de folio ha sido modificado en el sistema.

1.1.5. Descripción de Caso de Uso: Bloquear Rango de Nros. Folio

Objetivo:	Este caso de uso permite bloquear rango de números de folio, al bloquear estos números se impide sean utilizados como números provisorio para DIPS Viajero. Los rangos de números de folio son bloqueados en casos como por ejemplo pérdida de folios antes de que lleguen a destino. El bloqueo debe realizarse a través de la Dirección Nacional de Aduanas (DNA).	
Escenarios:		
Flujo Básico	Flujo Normal	1. El actor indica número inicial de rango de números de folio. 2. El actor indica número final de rango de números de folio. 3. El actor solicita a sistema sean bloqueado números de folio que pertenecen a rango indicado. 4. El sistema indica que rango de números de folio fueron bloqueados.
Flujo Alternativo	Error - Rango a bloquear posee números utilizados	3. Si dentro de rango de números de folio a bloquear existen números que han sido utilizados como número de identificación en DIPS Viajero aceptadas, rechazadas o anuladas. 3.1 El sistema despliega mensaje indicando que números no pueden ser utilizados porque números fueron utilizados como número manual de aceptación en DIPS Viajero.
Flujo Alternativo	Error - Rango de números bloqueados	3. Si se intenta bloquear rango de números bloqueados. 3.1 Si administrador trata de bloquear números de rango que han sido bloqueados el sistema deberá indicar error.
Restricciones:		
Pre-Condición	Usuario Logeado	El usuario debe estar logeado y con permisos en el sistema.
Post-Condición	Rango de Números de Folio bloqueado	El Rango de números de folio ha sido bloqueado en el sistema.

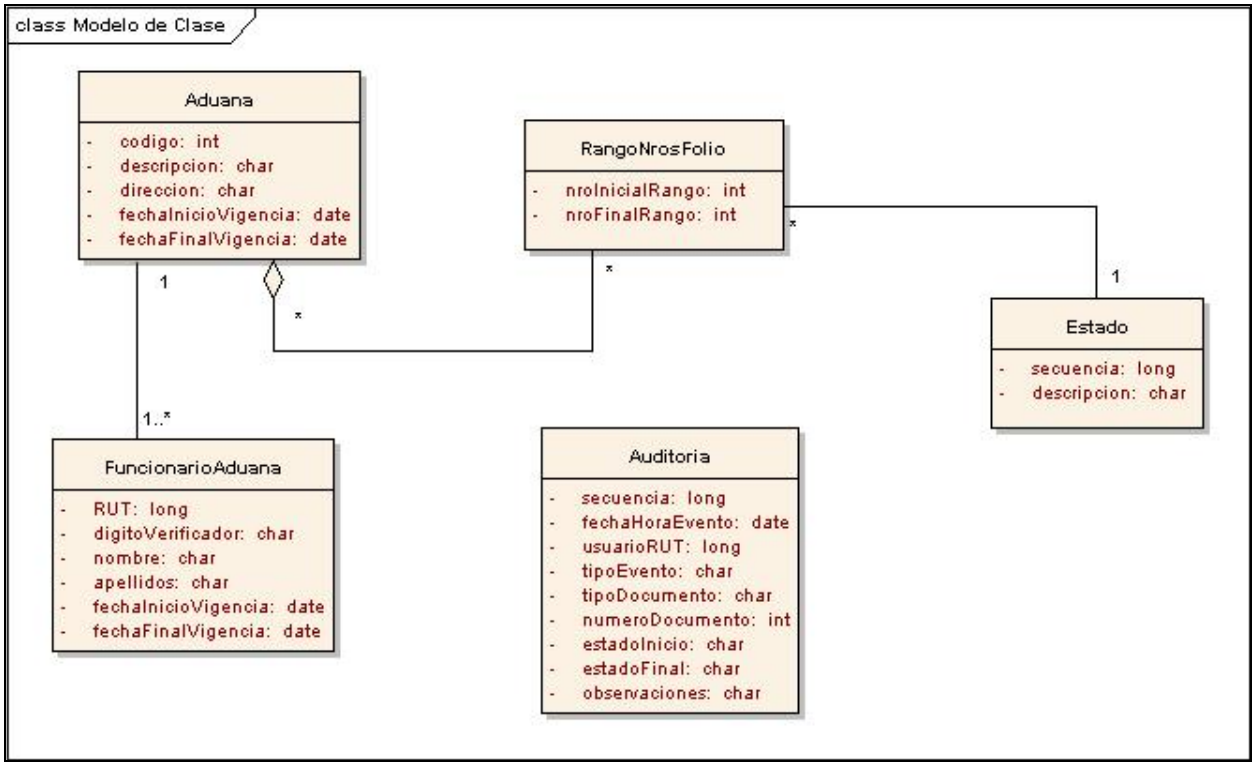
1.1.6. Descripción de Caso de Uso: Validar Nro. de Folio utilizado

Objetivo:	Este caso de uso permite verificar que número de folio utilizado como número de identificación en forma manual en una DIPS Viajero pertenezca a rango de números entregados en Aduana en la cual se confeccionó la DIPS.	
Escenarios:		
Flujo Básico	Flujo Normal	1. El sistema recibe como parámetro número de folio utilizado en DIPS Viajero. 2. El sistema verifica que número de folio esté corrrecto y pertenezca a rango de números asignados a Aduana donde se confecciona la DIPS Viajero. 3. El sistema indica que número de folio es correcto.
Flujo Alternativo	Error - Número de Folio incorrecto	2. El sistema DIPS valida que número sea correcto 2.1 Si número de folio no corresponde a números que pertenecen a rango de números entregados a Aduana donde se confecciona DIPS Viajero. 2.2 El sistema DIPS indica que número de folio es incorrecto.
Restricciones:		

1.2. Requerimientos No Funcionales

Nombre	Descripción
RNF - Acceso a Internet	Se requiere que todas las Avanzadas Fronterizas donde se generan DIPS Viajeros tengan acceso a Internet
RNF - Auditoría	El sistema debe dejar un registro de auditoria cuando se produzcan cualquiera de los siguientes eventos: - Confeccionar DIPS Viajero - Modificar DIPS Viajero - Anular DIPS Viajero - Eliminar DIPS Viajero - Ingresar Viajero - Modificar Viajero - Anular Viajero - Desbloquear Franquicia Se debe registrar: - fecha y hora en que se produce evento - usuario que realiza evento - evento realizado - tipo de evento si fue realizado en forma automática o manual - tipo de documento afectado por evento (cuando corresponda) - número de documento afectado por evento (cuando corresponda) - estado del documento antes de efectuado evento (cuando corresponda) - estado del documento después de efectuado evento (cuando corresponda) En el caso de tratarse de una anulación de DIPS Viajero se deberán registrar además los siguientes datos: - número de resolución (número de resolución que da autorización de anulación de DIPS Viajero) - fecha de resolución (fecha en que se autorizó resolución) - valor de cuenta 191 - motivo de anulación
RNF - Seguridad de Acceso	El sistema debe contar con un acceso de seguridad que permita solo acceso a: - Fiscalizadores - Jefe SubDepartamento de Regímenes Especiales - Administradores del sistema Este acceso de seguridad debe seguir el esquema actual utilizado en el Servicio Nacional de Aduana, que se encuentra administrado por el área de Informática de Aduana.
RFN - Registro de errores	Cuando el sistema detecte errores en la confección de DIPS Viajero estos deberán ser registrados.

1.3. Modelo de Clase



Aduana

Aduanas que existen a lo largo del país.

Attributos

Nombre	Tipo	Notas
codigo	int	Código que identifica a Aduana.
descripcion	char	Descripción indicando nombre de Aduana.
direccion	char	Dirección donde se localiza Aduana.
fechaInicioVigencia	date	Fecha inicio de vigencia de datos de Aduana.
fechaFinalVigencia	date	Fecha de término de vigencia de datos de Aduana.

Operaciones

Métodos	Descripción	Parámetros
getCodigo()		
setCodigo()		
getDescripcion()		
setDescripcion()		
getDireccion()		
setDireccion()		

Auditoria

Permite registrar auditorias de los distintos eventos que realice el sistema. Principalmente de los eventos que ejecuten modificaciones a los datos almacenados.

Attributos

Nombre	Tipo	Notas
secuencia	long	Código que identifica a auditoria.
fechaHoraEvento	date	Fecha y Hora en que se realizó el evento.
usuarioRUT	long	Usuario que gatilló el evento. Se registra RUT de usuario sin digito verificador.
tipoEvento	char	Tipo de evento ejecutado, si corresponde a ingreso, modificación, anulación o eliminación de datos.
tipoDocumento	char	Tipo de documento en los casos que corresponda al cual se aplica la acción del evento.
numeroDocumento	int	Número que identifica al documento que esta siendo afectado por el evento, en los casos que corresponda.
estadoInicio	char	Estado del documento antes del evento.
estadoFinal	char	Estado final del documento después de efectuado el evento.
nroResolucion	int	Para los casos en que el evento requiera de una resolución de autorización para efectuar el evento deberá registrarse el número de

Nombre	Tipo	Notas
		aquella resolución.
fechaResolucion	date	Fecha en que fue autorizada la resolución que autoriza la ejecución del evento.
valorCuenta191	float	Monto total de cuenta 191 en los casos de que el documento afectado por el evento corresponda a una DIPS Viajero se deberá registrar este monto.
observaciones	char	Se registran observaciones como en el caso de anulación donde se debe registrar motivo de anulación.

Operaciones

Métodos	Descripción	Parámetros
addAuditoria()	Método que crea un nuevo registro de auditoria.	
getSecuencia()		
setSecuencia()		
getFechaHoraEvento()		
set FechaHoraEvento()		
getUsuarioRUT()		
set UsuarioRUT()		
getTipoEvento()		
set TipoEvento()		
getTipoDocumento()		
set TipoDocumento()		
getNumeroDocumento()		
set NumeroDocumento()		
getEstadoInicio()		
set EstadoInicio()		
getEstadoFinal()		
setEstadoFinal()		
getNroResolucion()		
setNroResolucion()		
getFechaResolucion()		
setFechaResolucion()		
getValorCuenta191()		
set ValorCuenta191()		
getObservaciones()		
setObservaciones()		

Estado

Posibles estados que pueden obtener en el tiempo.

Attributos

Nombre	Tipo	Notas
secuencia	long	Código que identifica al estado.
descripcion	char	Descripción breve de estado.

Operaciones

Métodos	Descripción	Parámetros
getSecuencia()		
setSecuencia()		
getDescripcion()		
setDescripcion()		

FuncionarioAduana

Corresponde a funcionario de Aduana encargado de realizar control de mercancías ingresadas al país por viajeros.

Attributos

Nombre	Tipo	Notas
RUT	long	RUT que identifica a fiscalizador sin dígito verificador.
digitoVerificador	int	Dígito verificador de RUT de fiscalizador.
nombre	char	Nombre de fiscalizador.
apellidos	char	Apellidos Paterno y Materno de fiscalizador.
fechaInicioVigencia	date	Fecha de inicio de vigencia de datos de fiscalizador.
fechaFinalVigencia	date	Fecha de término de vigencia de datos de fiscalizador.

Operaciones

Métodos	Descripción	Parámetros
getRUT()		
setRUT()		
getDigitoVerificador()		
setDigitoVerificador()		
getNombre()		

Métodos	Descripción	Parámetros
setNombre()		
getApellidos()		
setApellidos()		

ANEXO B
Análisis
Orientación a Aspectos

1.1 CAPTURA DE REQUERIMIENTOS

Paso 1: Analizar lista de Casos de Uso del Proyecto. Se escogió el módulo correspondiente a Rango de Números de Folio que permite asignar un rango de números a un Aduana y poder utilizar estos números en folios confeccionados en forma manual y que son registrados en el sistema posteriormente. Los casos de uso de este módulo son:

- Caso de Uso 1: Ingresar Rango de Números de Folio
- Caso de Uso 2: Modificar Rango de Números de Folio
- Caso de Uso 3: Bloquear Rango de Números de Folio
- Caso de Uso 4: Validar Número de Folio utilizado

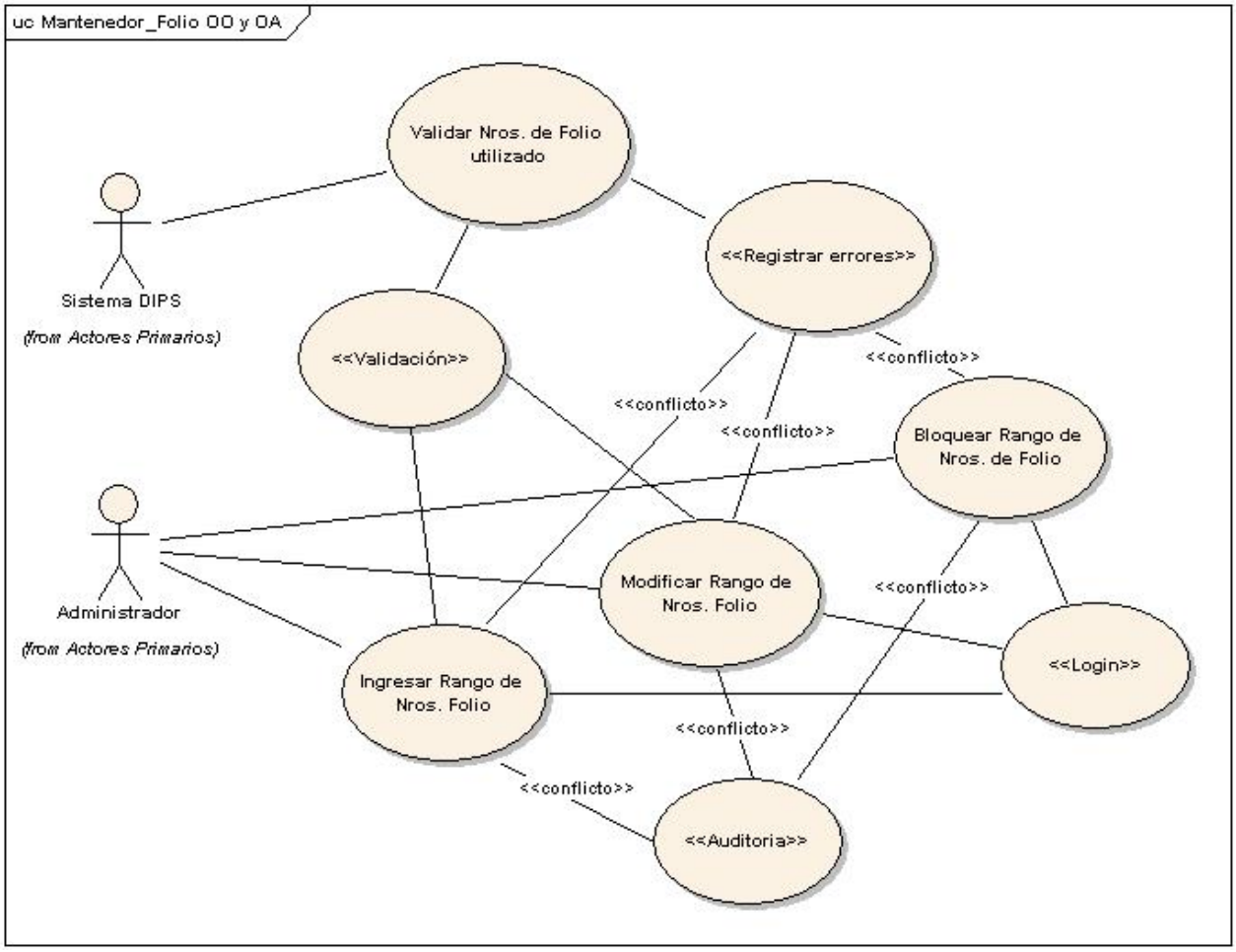
Paso 2: Elección de Aspectos Candidatos

Aspecto Candidato	Casos de Uso	Actor	Responsabilidad
Login	1,2,3	Administrador, Fiscalizador	Dar seguridad en acceso a interfaz
Auditoria	1,2,3	Administrador, Fiscalizador	Realizar un registro de auditoria de los eventos que producen cambios en los datos
Validación	1,2,4	Administrador, Fiscalizador	Validar que número de folio indicado se encuentre correcto
Registro Errores	1,2,3,4	Administrador, Fiscalizador, Sistema	Cuando se produzcan errores durante la ejecución de un caso de uso se deberá registrar

Paso 3: Identificar relaciones entre Casos de Uso y Aspectos Candidatos

	Caso de Uso 1	Caso de Uso 2	Caso de Uso 3	Caso de Uso 4
Login	X	X	X	
Auditoria	X	X	X	
Validación	X	X		X
Registro Errores	X	X	X	X

Paso 4: Modelar en UML



Paso 5: Interacciones entre aspectos

Nombre	Proveer seguridad de acceso a interfaces
Aspecto Involucrado	Login
Tipo	Conflicto
Ejemplo	Dar acceso sólo a administradores y fiscalizadores
Explicación predicados	
Tiempo de Respuesta	Abrir nueva sesión

Nombre	Registrar eventos
Aspecto Involucrado	Auditoria
Tipo	Conflicto
Ejemplo	Cuando se ingresa nuevo rango de números de folio se debe registrar datos de evento
Explicación predicados	
Tiempo de Respuesta	No bloquear acción de siguientes eventos registro debe ser transparente para usuario

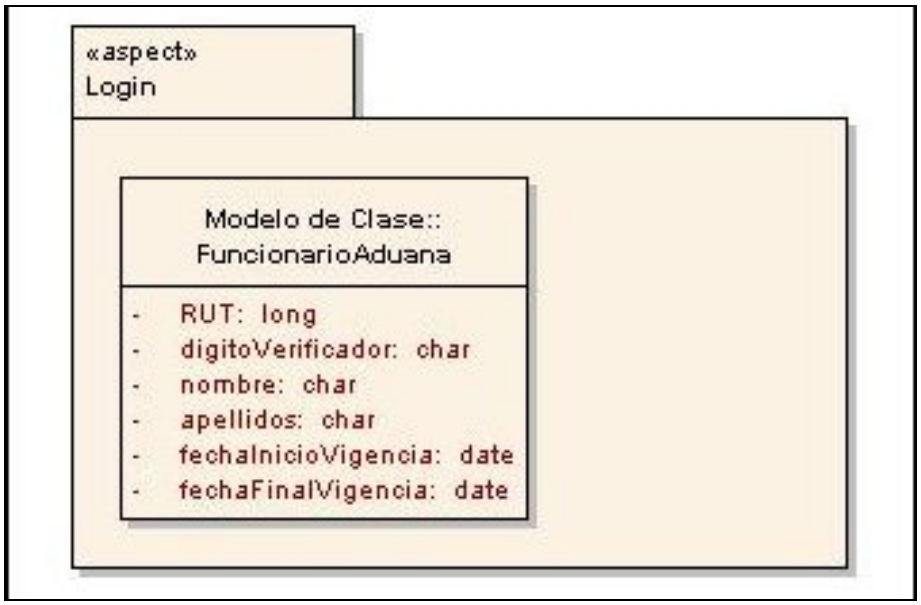
Nombre	Validar número de folio
Aspecto Involucrado	Validación
Tipo	Conflicto
Ejemplo	No registrar dato si es incorrecto número de folio, indicar mensaje de error
Explicación predicados	
Tiempo de Respuesta	Desplegar mensaje error en forma automática y bloquear acción

Nombre	Registrar errores cuando se produzcan
Aspecto Involucrado	Registrar error
Tipo	Conflicto
Ejemplo	Si se produce problemas en sistema se debe registrar error
Explicación predicados	
Tiempo de Respuesta	Cerrar sesión actual y registrar error

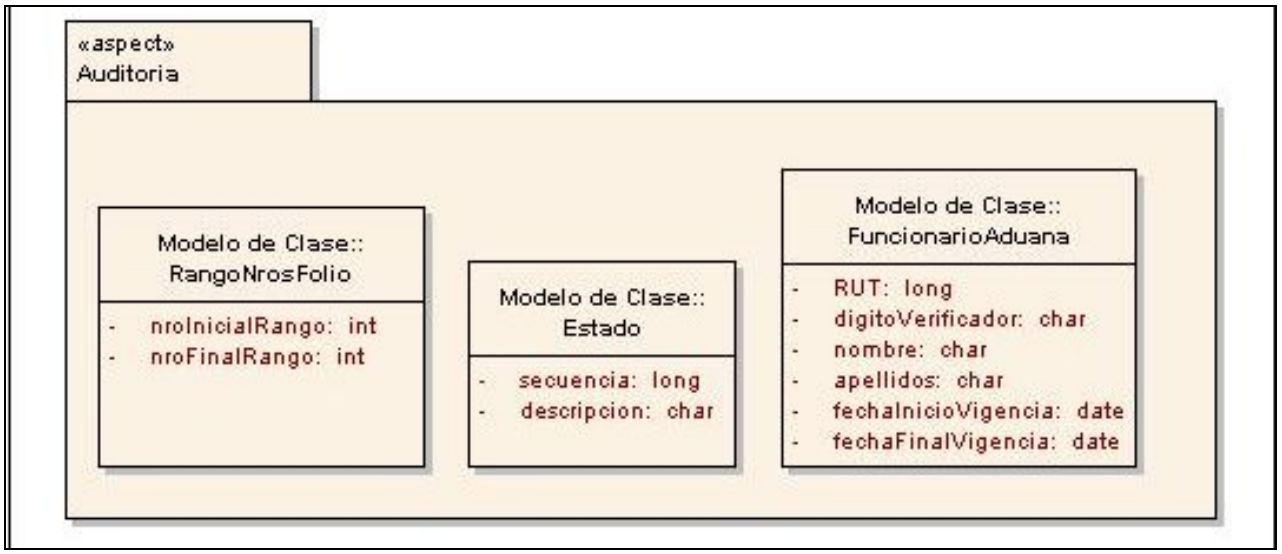
1.2 Análisis

1.2.1 Modelo de Aspectos y Clases

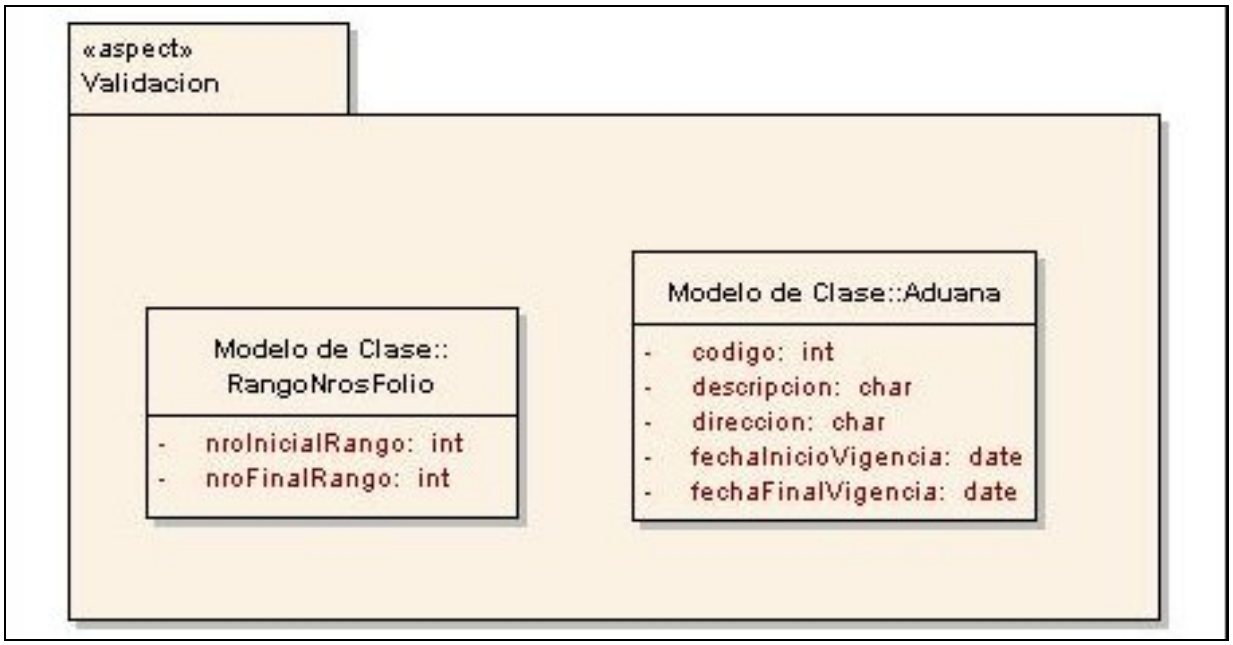
Aspecto: Login



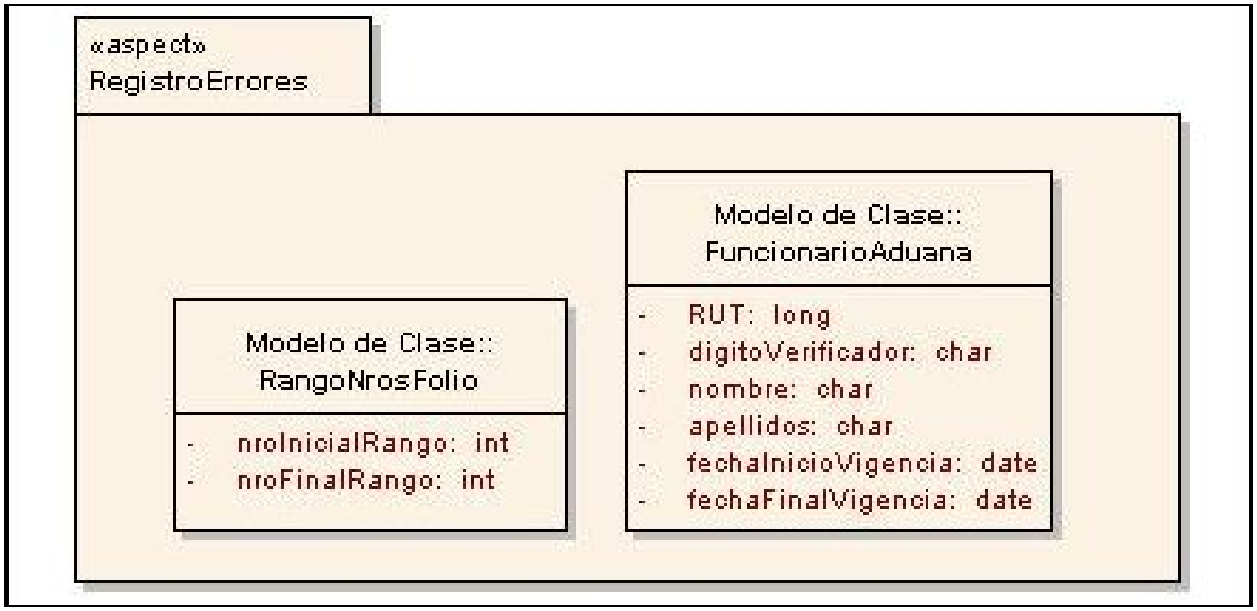
Aspecto: Auditoria



Aspecto: Validación



Aspecto: Registro Errores

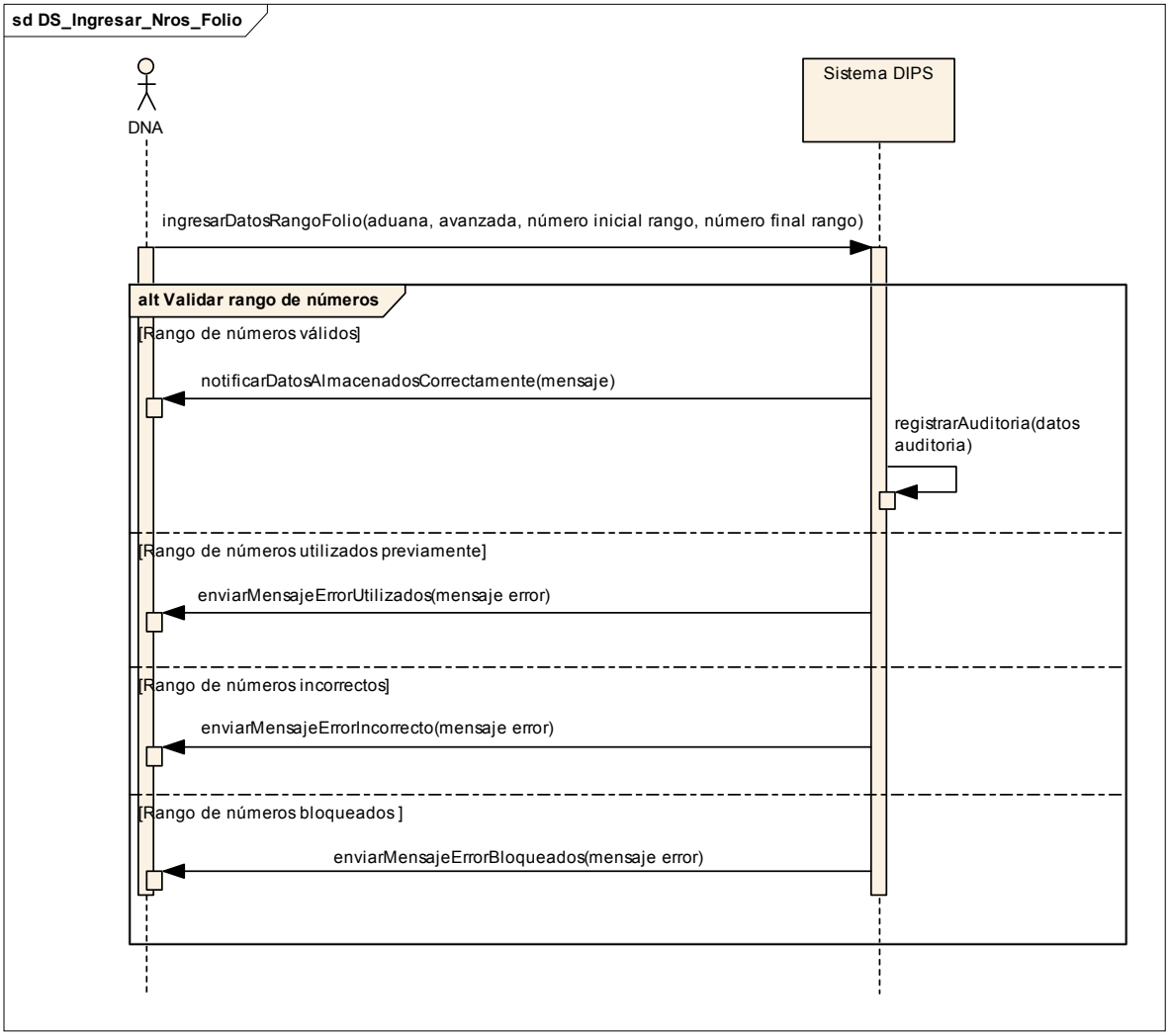


ANEXO C
Diseño
Orientación a Objetos

1.1 DIAGRAMA DE SECUENCIAS

1.1.1 Diagramas de Interacción o Secuencia: Módulo Mantenedor Números de Folio

1.1.1.1 Diagrama de Secuencia: Ingresar Rango Nros. Folio

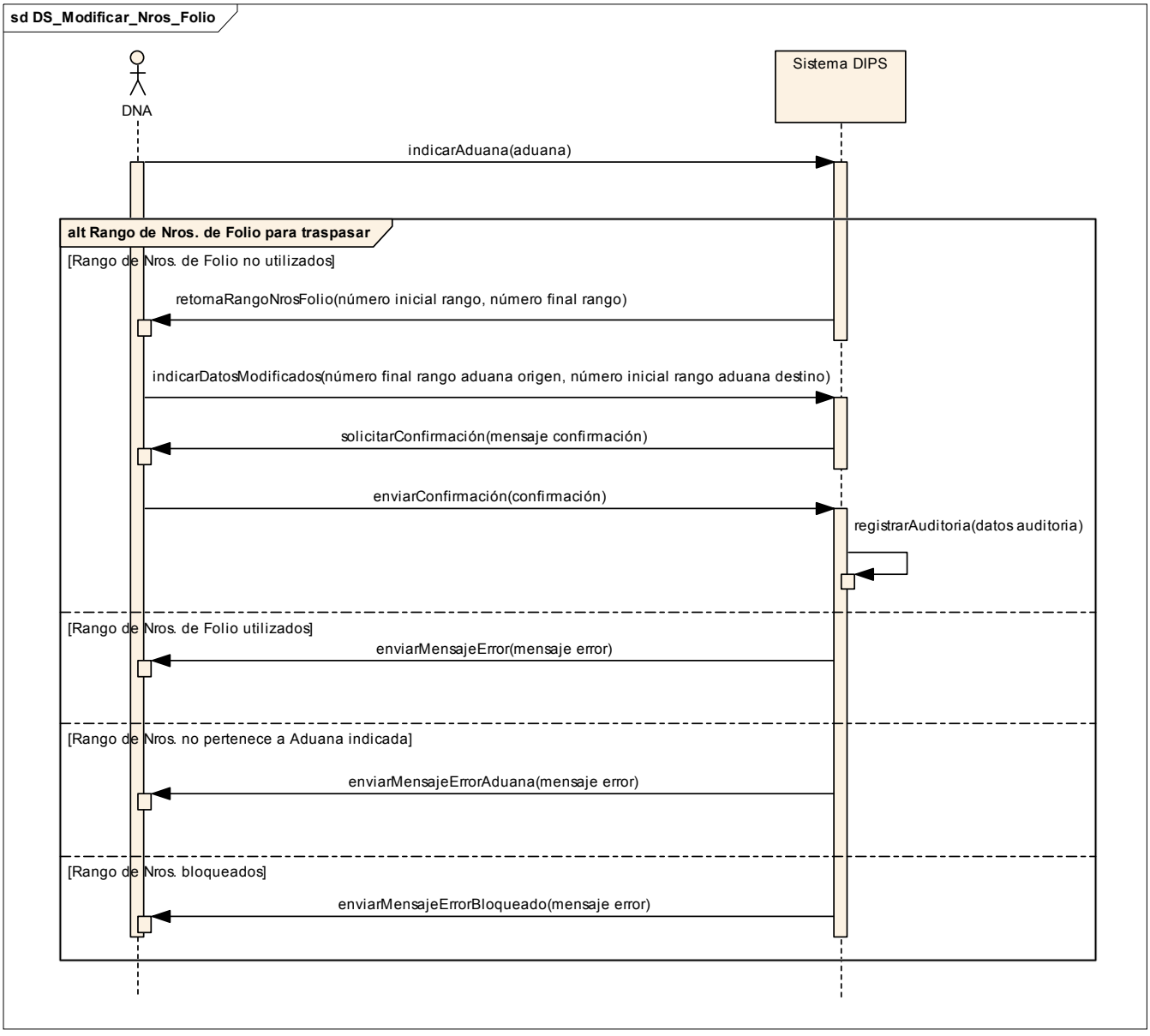


1.1.1.2 Descripción de Secuencias Relevantes:

Secuencias	Parámetros	Origen	Destino	Notas
ingresarDatosRangoFolio	aduana, avanzada, número inicial rango, número final rango	DNA	Sistema DIPS	El administrador ingresa rango de números de folio que hayan sido distribuidas a una Aduana específica.
notificarDatosAlmacenadosCorrectamente	mensaje	Sistema DIPS	DNA	El sistema notifica que los datos de rango de números de folio han sido almacenados correctamente.
registrarAuditoria	datos auditoria	Sistema DIPS	Sistema DIPS	Se deben registrar los siguientes datos de auditoria: - fecha y hora - aduana - número inicial y final de rango - fiscalizador que realiza evento - evento - tipo evento (automático o manual)
enviarMensajeErrorUtilizados	mensaje error	Sistema DIPS	DNA	Si dentro de rango de números indicados existen números que han sido previamente utilizados en DIPS Viajero aceptadas, rechazadas o anuladas, el sistema debe notificar error.
enviarMensajeErrorIncorrecto	mensaje error	Sistema DIPS	DNA	Si los datos de rango de números esta incorrecto. - Si se ingresa número final de rango menor a número inicial de rango, el sistema deberá indicar error en rango indicado ya que número de rango final debe ser mayor a número de rango inicial. - Si formato de número de folio indicado en reglas de negocio no corresponde a números de rango se debe indicar error.

enviarMensajeErrorBloqueados	mensaje error	Sistema DIPS	DNA	Si administrador trata de ingresar números de rango que han sido bloqueados el sistema deberá indicar error.

1.1.1.3 Diagrama de Secuencia: Modificar Rango Nros. Folio

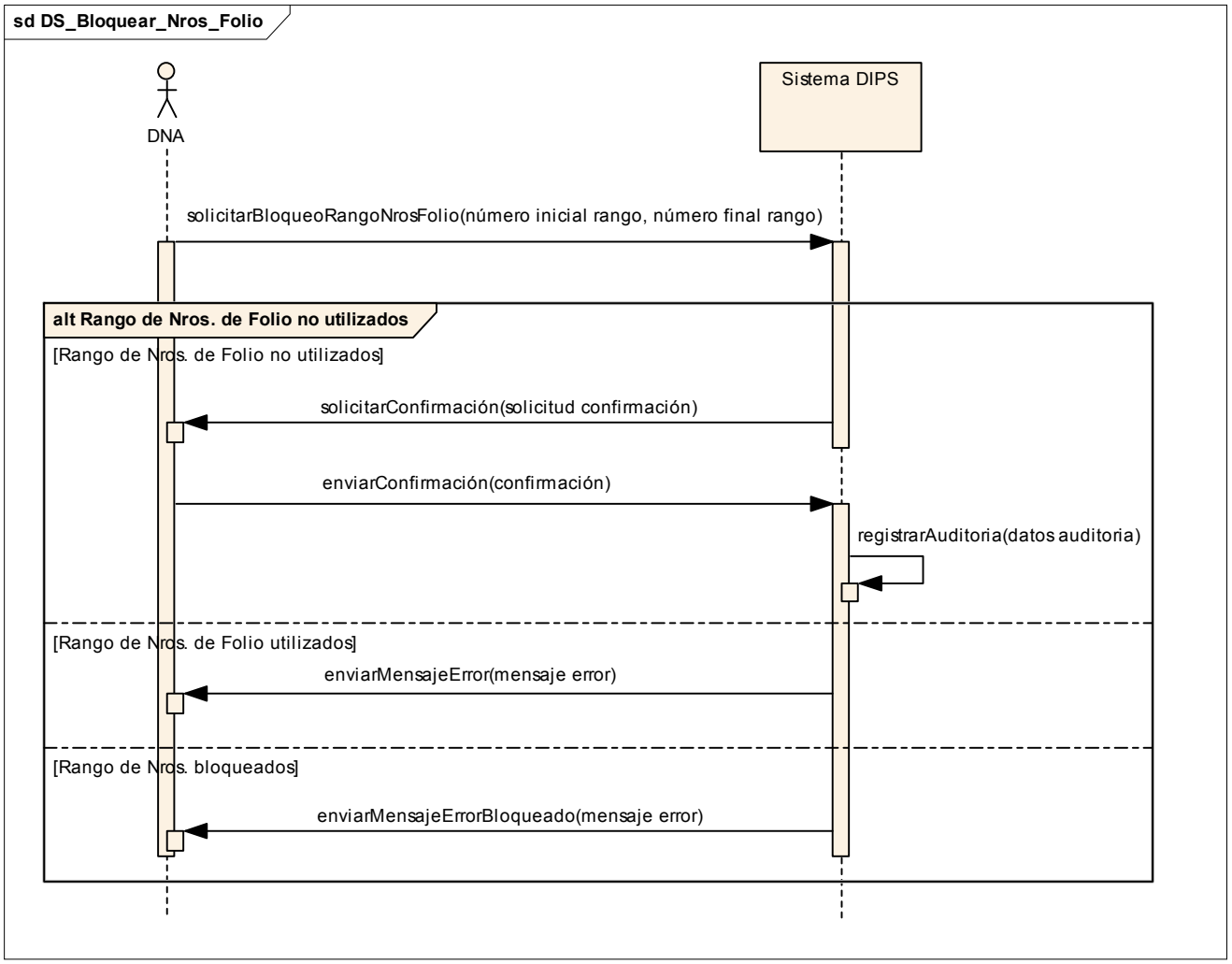


1.1.1.4 Descripción de Secuencias Relevantes:

Secuencias	Parámetros	Origen	Destino	Notas
indicarAduana	aduana	DNA	Sistema DIPS	El administrador debe indicar a sistema Aduana a la cual se le quitará números de folio para ser trasladados a otra Aduana.
retornaRangoNrosFolio	número inicial rango, número final rango	Sistema DIPS	DNA	El sistema retorna rango de números correspondientes a Aduana indicada.
indicarDatosModificados	número final rango aduana origen, número inicial rango aduana destino	DNA	Sistema DIPS	El administrador debe indicar número final que mantendrá la Aduana y debe indicar número inicial del rango que será traspasado a segunda Aduana.
solicitarConfirmación	mensaje confirmación	Sistema DIPS	DNA	El sistema solicita confirmar traslado de rango de números de folio desde Aduana origen a Aduana destino.
enviarConfirmación	confirmación	DNA	Sistema DIPS	El administrador confirma traslado de rango de números de folio a Aduana destino.
registrarAuditoria	datos auditoria	Sistema DIPS	Sistema DIPS	Se deben registrar los siguientes datos de auditoria: - fecha y hora - aduana - número inicial y final de rango - fiscalizador que realiza evento - evento - tipo evento (automático o manual)
enviarMensajeError	mensaje error	Sistema DIPS	DNA	Se notifica error cuando se desea traspasar números de folio que ya han sido utilizados como números manuales de aceptación en DIPS Viajero aceptadas, rechazadas o anuladas.

enviarMensajeErrorAduana	mensaje error	Sistema DIPS	DNA	Si el número de folio de rango ha traspasar de una Aduana origen no corresponde a esa Aduana, el sistema indicará error.
enviarMensajeErrorBloqueado	mensaje error	Sistema DIPS	DNA	Si administrador trata de traspasar números de rango que han sido bloqueados el sistema deberá indicar error.

1.1.1.5 Diagrama de Secuencia: Bloquear Rango Nros. Folio



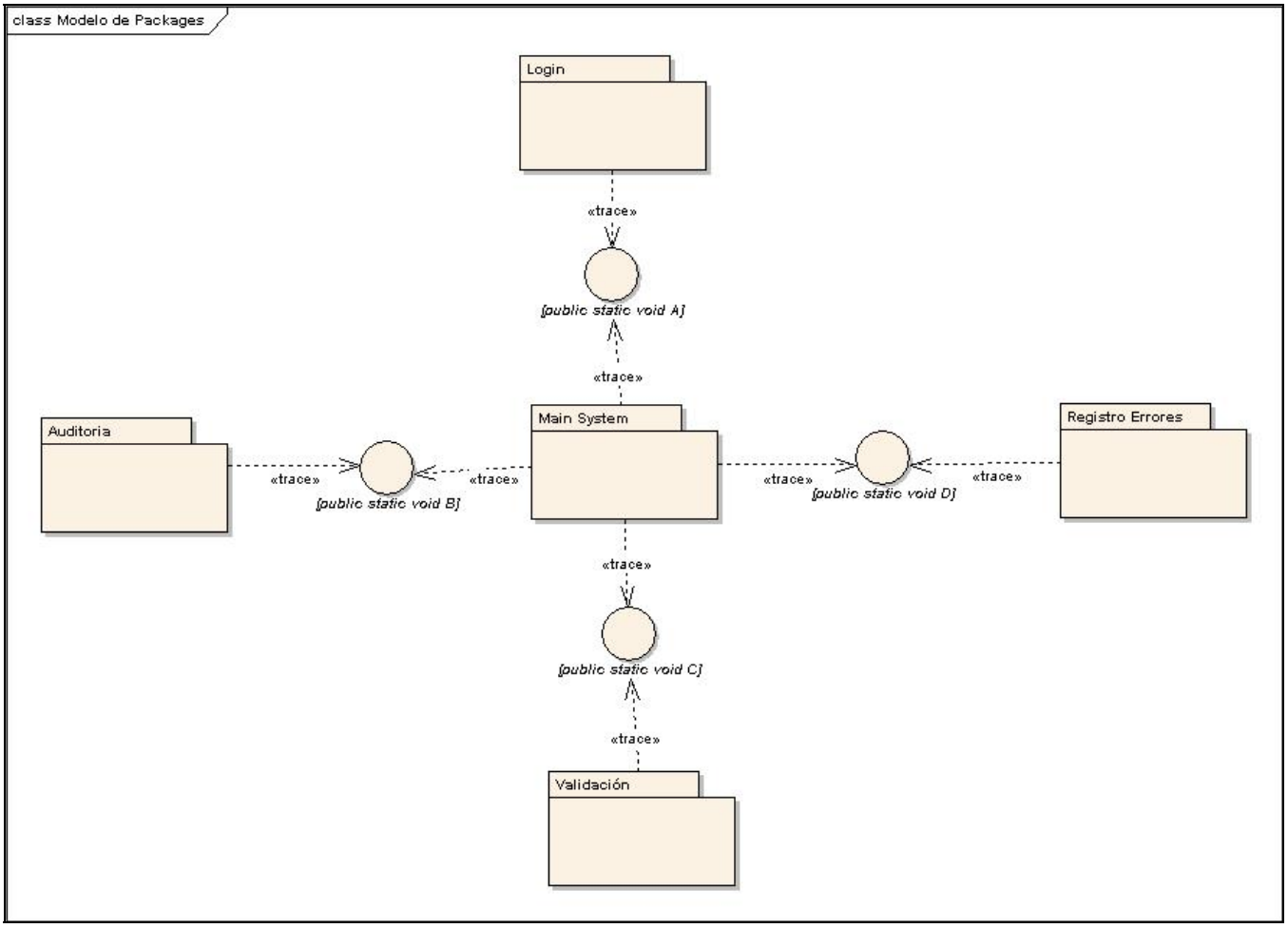
1.1.1.6 Descripción de Secuencias Relevantes:

Secuencias	Parámetros	Origen	Destino	Notas
solicitarBloqueoRangoNrosFolio	número inicial rango, número final rango	DNA	Sistema DIPS	El administrador debe indicar número inicial de rango y número final de rango. Este bloqueo será independiente de la Aduana. Los números bloqueados no podrán ser utilizados como número manual.
solicitarConfirmación	solicitud confirmación	Sistema DIPS	DNA	El sistema solicita a administrador confirma bloqueo de rango de números de folio.
enviarConfirmación	confirmación	DNA	Sistema DIPS	El administrador confirma bloqueo de rango de números de folio.
registrarAuditoria	datos auditoria	Sistema DIPS	Sistema DIPS	Se deben registrar los siguientes datos de auditoria: - fecha y hora - aduana - número inicial y final de rango - fiscalizador que realiza evento - evento - tipo evento (automático o manual)
enviarMensajeError	mensaje error	Sistema DIPS	DNA	El sistema debe indicar error cuando se intente bloquear números de folio que hayan sido utilizados como número manual de aceptación en DIPS Viajero aceptadas, rechazadas o anuladas.
enviarMensajeErrorBloqueado	mensaje error	Sistema DIPS	DNA	Si administrador trata de bloquear números de rango que han sido bloqueados el sistema deberá indicar error.

ANEXO D
Diseño
Orientación a Aspectos

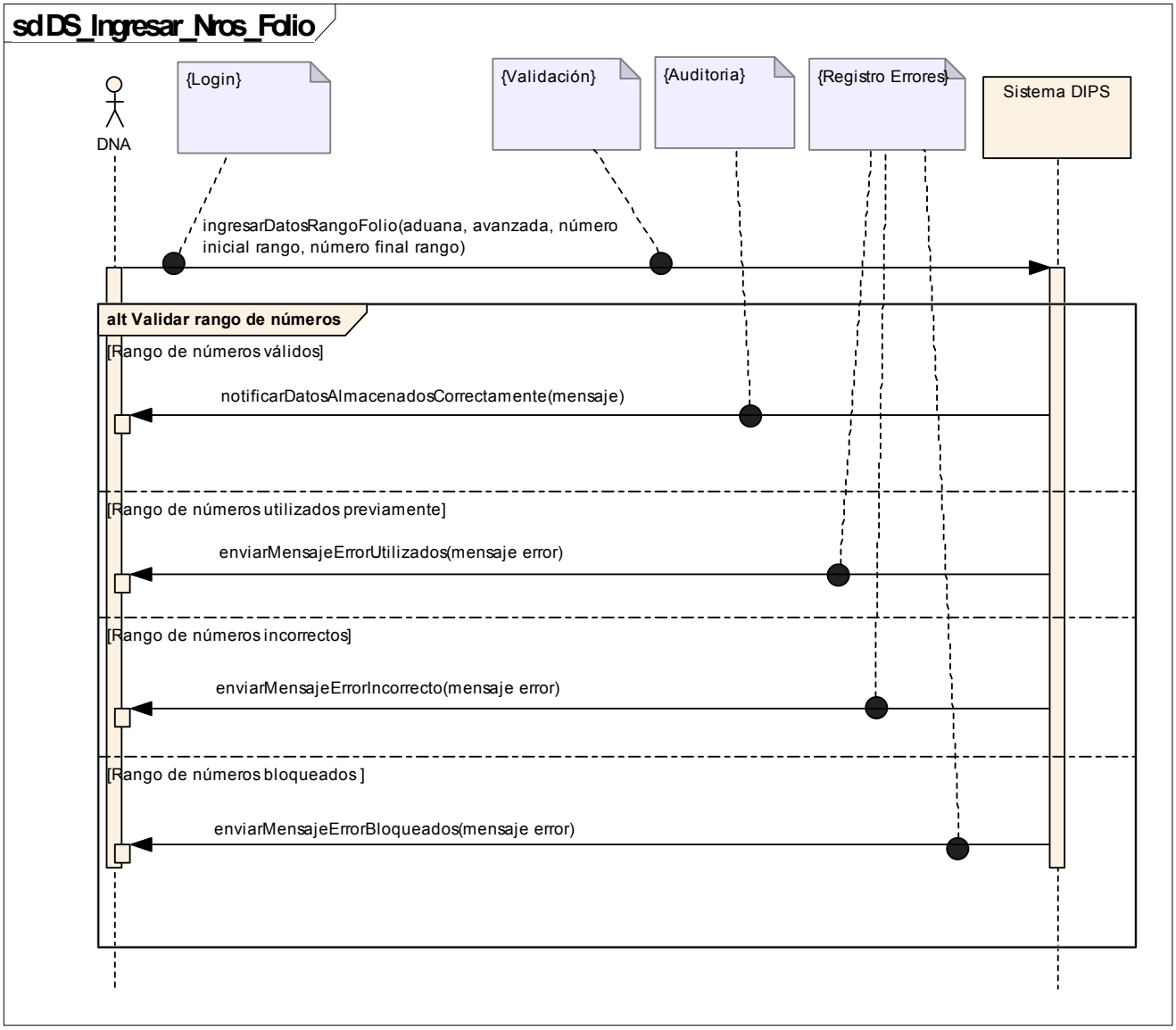
1.1 DISEÑO

1.1.1 Diagrama de packages



1.1.2 Diagrama de Secuencia

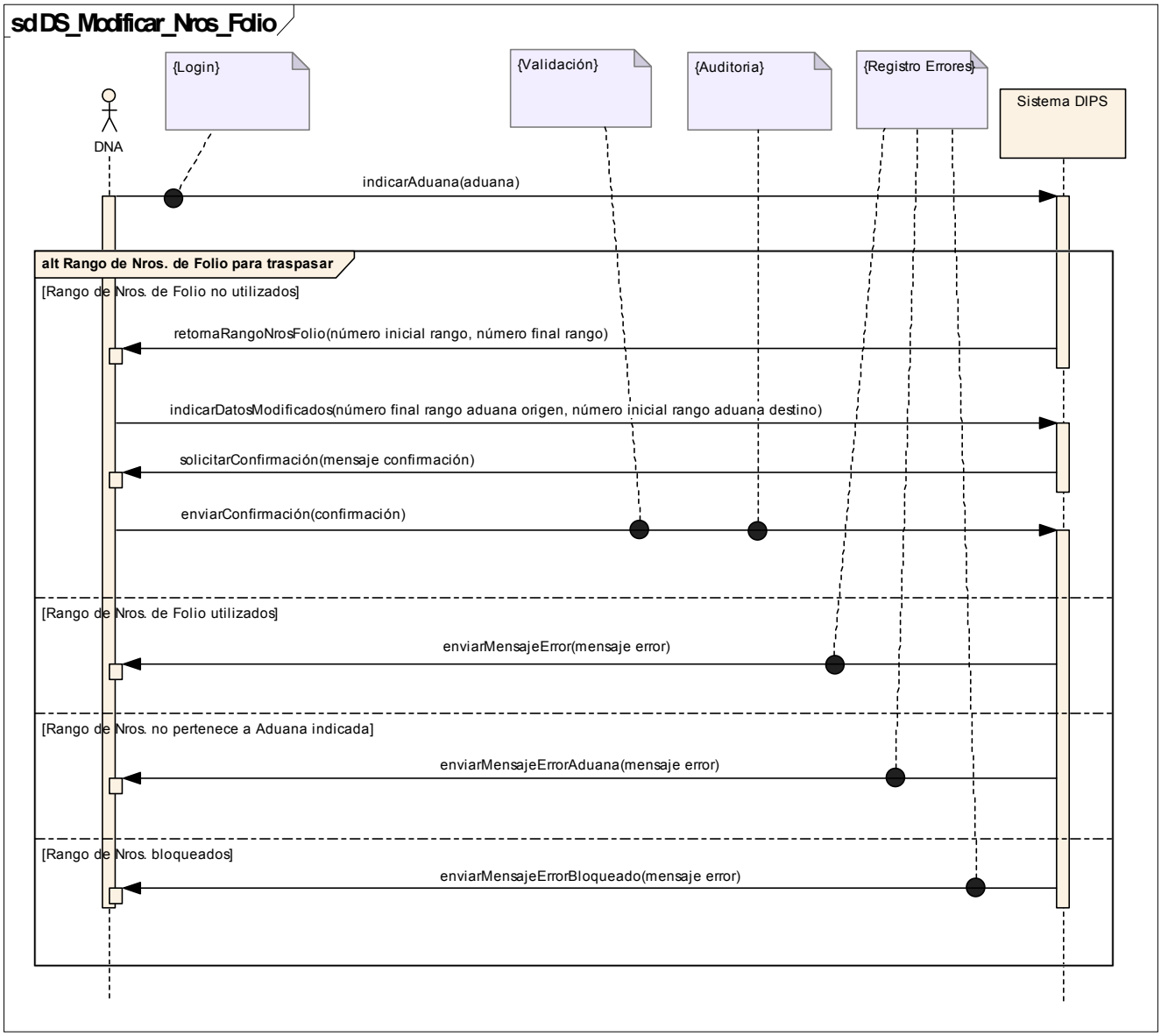
1.1.2.1 Diagrama de Secuencia: Ingresar Rango Nros. Folio



1.1.2.2 Descripción de Secuencias Relevantes:

Secuencias	Parámetros	Origen	Destino	Notas
ingresarDatosRangoFolio	aduana, avanzada, número inicial rango, número final rango	DNA	Sistema DIPS	El administrador ingresa rango de números de folio que hayan sido distribuidas a una Aduana específica.
notificarDatosAlmacenadosCorrectamente	mensaje	Sistema DIPS	DNA	El sistema notifica que los datos de rango de números de folio han sido almacenados correctamente.
enviarMensajeErrorUtilizados	mensaje error	Sistema DIPS	DNA	Si dentro de rango de números indicados existen números que han sido previamente utilizados en DIPS Viajero aceptadas, rechazadas o anuladas, el sistema debe notificar error.
enviarMensajeErrorIncorrecto	mensaje error	Sistema DIPS	DNA	Si los datos de rango de números esta incorrecto. - Si se ingresa número final de rango menor a número inicial de rango, el sistema deberá indicar error en rango indicado ya que número de rango final debe ser mayor a número de rango inicial. - Si formato de número de folio indicado en reglas de negocio no corresponde a números de rango se debe indicar error.
enviarMensajeErrorBloqueados	mensaje error	Sistema DIPS	DNA	Si administrador trata de ingresar números de rango que han sido bloqueados el sistema deberá indicar error.

1.1.2.3 Diagrama de Secuencia: Modificar Rango Nros. Folio

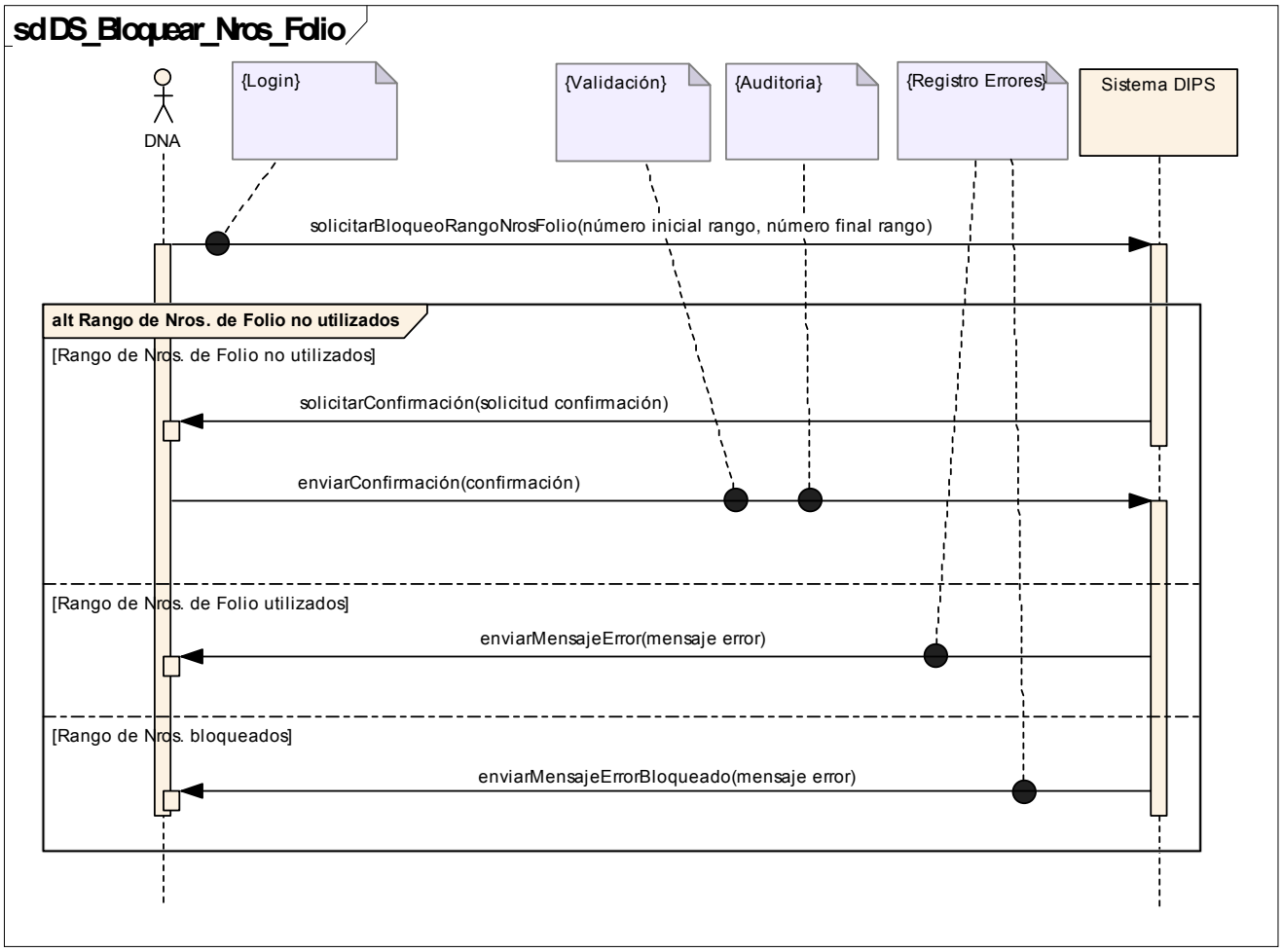


1.1.2.4 Descripción de Secuencias Relevantes:

Secuencias	Parámetros	Origen	Destino	Notas
indicarAduana	aduana	DNA	Sistema DIPS	El administrador debe indicar a sistema Aduana a la cual se le quitará números de folio para ser trasladados a otra Aduana.
retornaRangoNrosFolio	número inicial rango, número final rango	Sistema DIPS	DNA	El sistema retorna rango de números correspondientes a Aduana indicada.
indicarDatosModificados	número final rango aduana origen, número inicial rango aduana destino	DNA	Sistema DIPS	El administrador debe indicar número final que mantendrá la Aduana y debe indicar número inicial del rango que será traspasado a segunda Aduana.
solicitarConfirmación	mensaje confirmación	Sistema DIPS	DNA	El sistema solicita confirmar traslado de rango de números de folio desde Aduana origen a Aduana destino.
enviarConfirmación	confirmación	DNA	Sistema DIPS	El administrador confirma traslado de rango de números de folio a Aduana destino.
enviarMensajeError	mensaje error	Sistema DIPS	DNA	Se notifica error cuando se desea traspasar números de folio que ya han sido utilizados como números manuales de aceptación en DIPS Viajero aceptadas, rechazadas o anuladas.
enviarMensajeError Aduana	mensaje error	Sistema DIPS	DNA	Si el número de folio de rango ha traspasar de una Aduana origen no corresponde a esa Aduana, el sistema indicará error.

enviarMensajeErrorBloqueado	mensaje error	Sistema DIPS	DNA	Si administrador trata de traspasar números de rango que han sido bloqueados el sistema deberá indicar error.

1.1.2.5 Diagrama de Secuencia: Bloquear Rango Nros. Folio



1.1.2.6 Descripción de Secuencias Relevantes:

Secuencias	Parámetros	Origen	Destino	Notas
solicitarBloqueoRangoNrosFolio	número inicial rango, número final rango	DNA	Sistema DIPS	El administrador debe indicar número inicial de rango y número final de rango. Este bloqueo será independiente de la Aduana. Los números bloqueados no podrán ser utilizados como número manual.
solicitarConfirmación	solicitud confirmación	Sistema DIPS	DNA	El sistema solicita a administrador confirma bloqueo de rango de números de folio.
enviarConfirmación	confirmación	DNA	Sistema DIPS	El administrador confirma bloqueo de rango de números de folio.
enviarMensajeError	mensaje error	Sistema DIPS	DNA	El sistema debe indicar error cuando se intente bloquear números de folio que hayan sido utilizados como número manual de aceptación en DIPS Viajero aceptadas, rechazadas o anuladas.
enviarMensajeErrorBloqueado	mensaje error	Sistema DIPS	DNA	Si administrador trata de bloquear números de rango que han sido bloqueados el sistema deberá indicar error.

ANEXO E
Cuestionario

1.1 CUESTIONARIO

Nombre:
Profesión:
Empresa u Organización donde se desempeña:
Cargo dentro de la empresa:

OA: Orientación a Aspectos

0: No corresponde
1: En desacuerdo
2: Parcialmente de acuerdo
3: De acuerdo
4: Totalmente de acuerdo

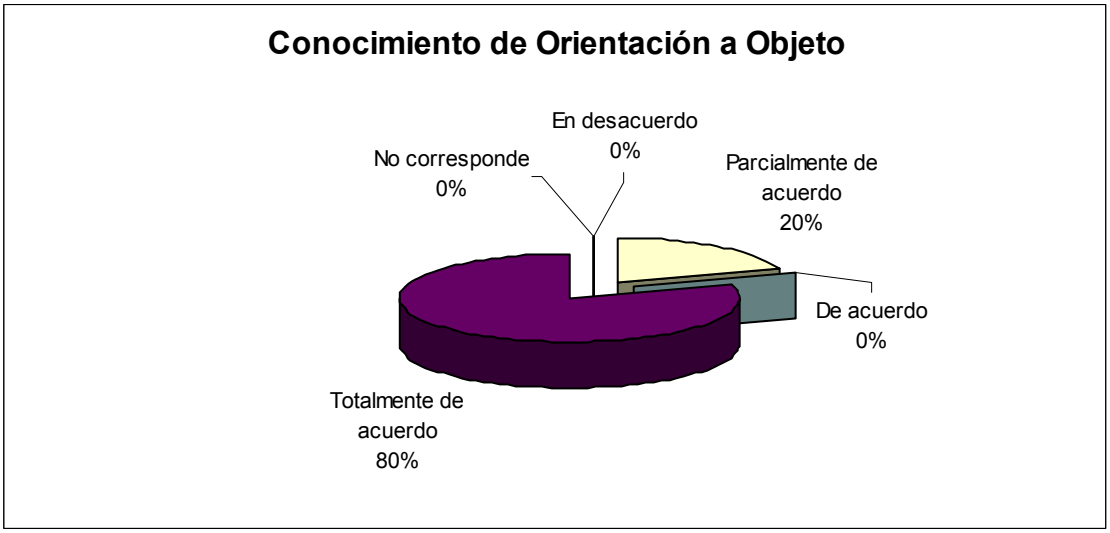
GENERALES	0	1	2	3	4
¿Ha trabajado en proyectos donde se utilizó Orientación a Objetos?					
¿Conoce el uso de Orientación a Aspectos?					
¿Ha trabajado en proyectos donde se utilizó Orientación a Aspectos?					
CASOS DE USO	**0**	**1**	**2**	**3**	**4**
¿Se ve más definido el requerimiento de Login en casos de uso que incluyen OA?					
¿Se ve más definido el requerimiento de Auditoria en casos de uso que incluyen OA?					
¿Se ve más definido el requerimiento de Validación en casos de uso que incluyen OA?					
¿Se ve más definido el requerimiento de Registro de Errores en casos de uso que incluyen OA?					
¿Cree que requiere más tiempo de trabajo trabajar casos de uso que incluyen OA?					

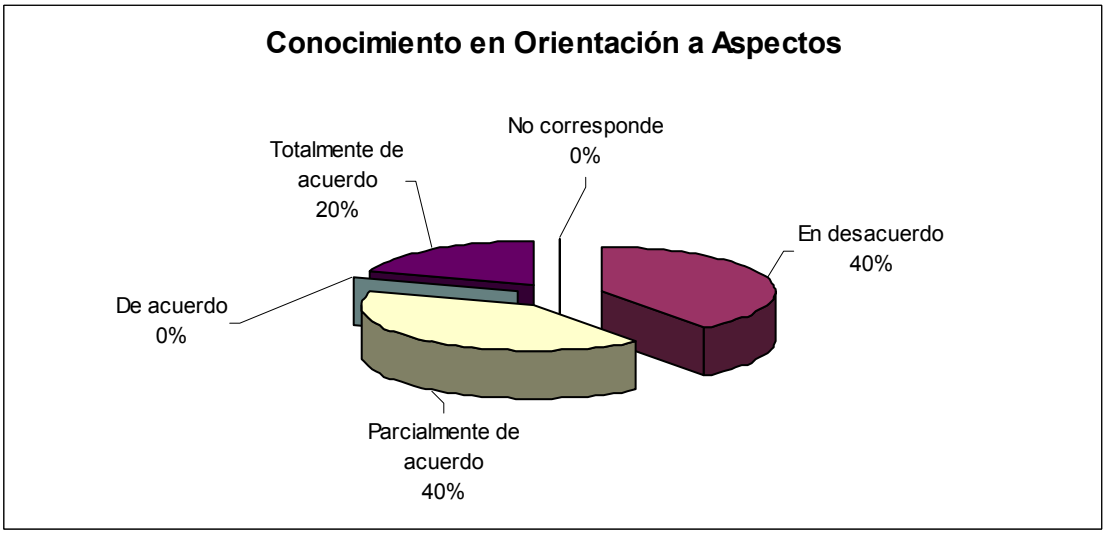
¿Incorporaría en futuros proyectos casos de uso que incluyen OA?						
MODELO DE CLASES	0	1	2	3	4	
¿Es más óptimo a nivel de métodos de las clases el modelo propuesto que incluye OA? (métodos incorporados en login, auditoria, validación de casos de uso y registro de errores)						
¿Es de fácil entendimiento modelo de clases propuesto que incluye OA?						
¿Cree que requiere más tiempo de trabajo trabajar modelo de clases que incluye OA?						
¿Incorporaría en futuros proyectos trabajar modelo de clases que incluya OA?						
DIAGRAMAS DE SECUENCIA	0	1	2	3	4	
¿Se ve más definido el requerimiento de Login en diagramas de secuencia que incluyen OA?						
¿Se ve más definido el requerimiento de Auditoria en diagramas de secuencia que incluyen OA?						
¿Se ve más definido el requerimiento de Validación en diagramas de secuencia que incluyen OA?						
¿Se ve más definido el requerimiento de Registro de Errores en diagramas de secuencia que incluyen OA?						
¿Cree que requiere más tiempo de trabajo trabajar diagrama de secuencia que incluyen OA?						
¿Incorporaría en futuros proyectos diagrama de secuencia que incluyen OA?						

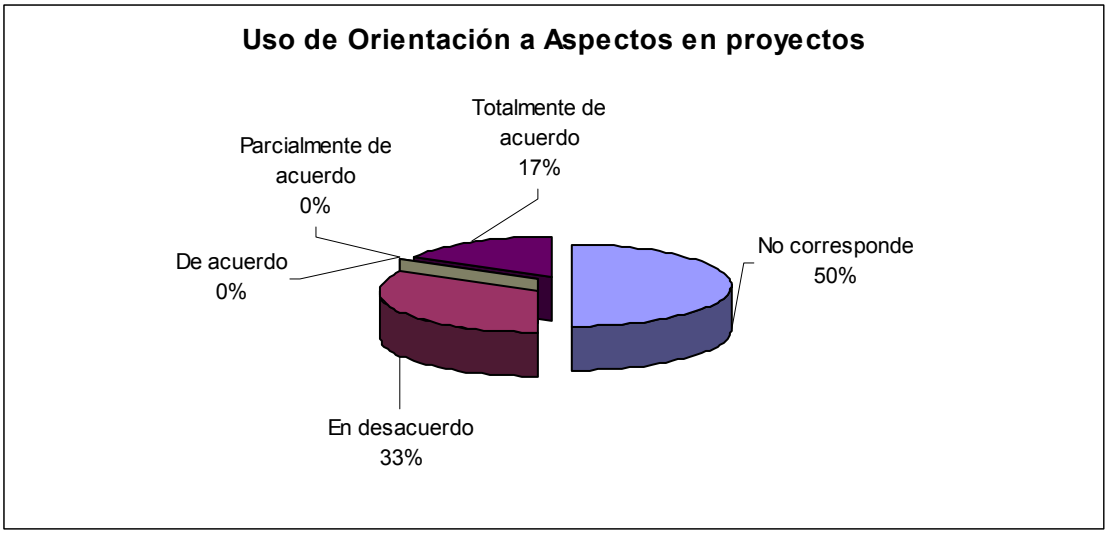
Indicar Observaciones con respecto a respuestas dadas en cuestionario:

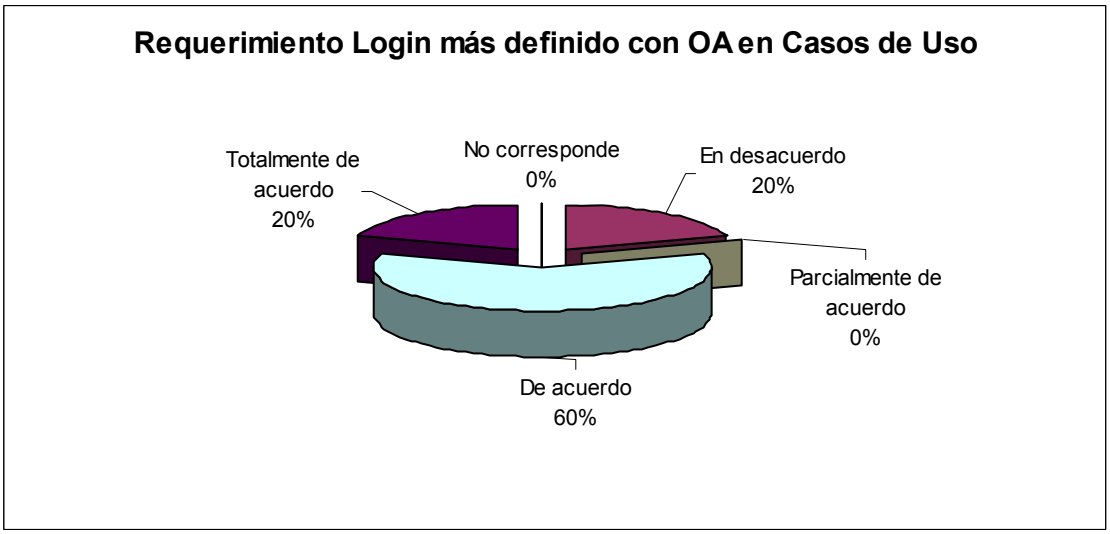
ANEXO F
Resultados Gráficos de Cuestionarios

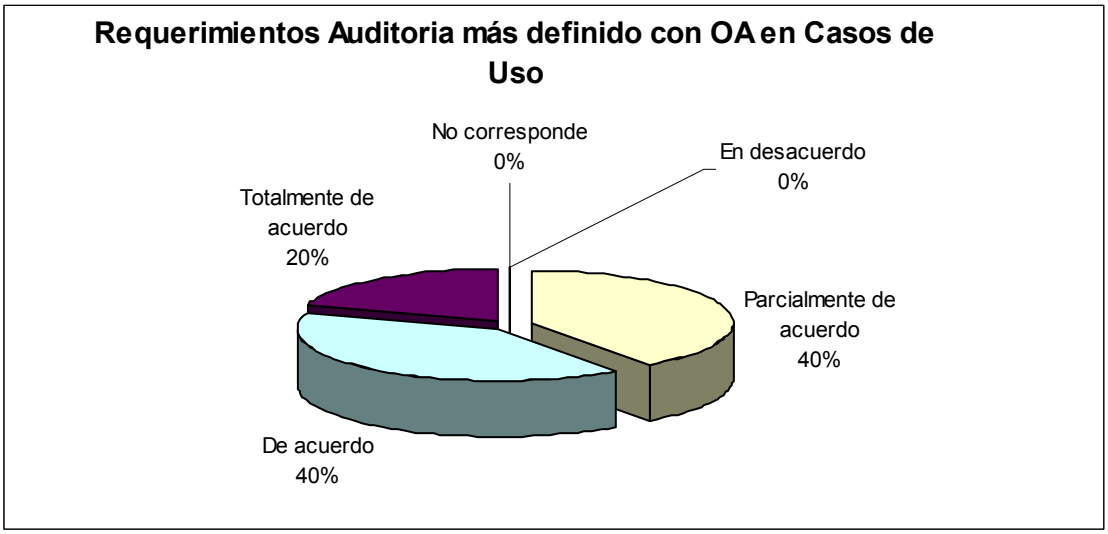
1.1 GRÁFICOS DE RESULTADOS DE CUESTIONARIOS



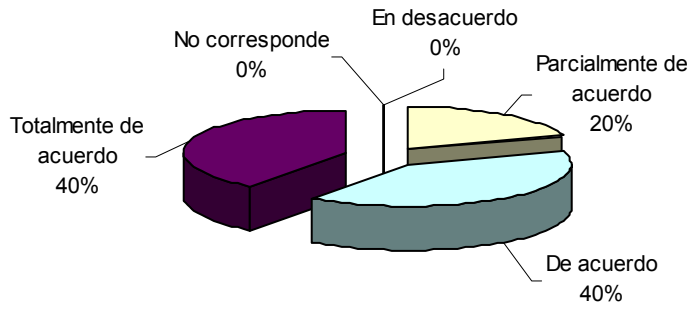



Uso de Orientación a Aspectos en proyectos
Totalmente de acuerdo 17%
Parcialmente de acuerdo 0%
De acuerdo 0%
No corresponde 50%
En desacuerdo 33%

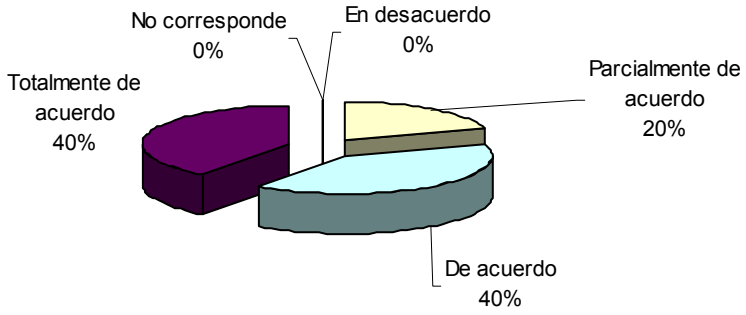

Requerimiento Login más definido con OA en Casos de Uso
Totalmente de acuerdo 20%
No corresponde 0%
En desacuerdo 20%
Parcialmente de acuerdo 0%
De acuerdo 60%


Requerimientos Auditoria más definido con OA en Casos de Uso
No corresponde 0%
En desacuerdo 0%
Totalmente de acuerdo 20%
Parcialmente de acuerdo 40%
De acuerdo 40%

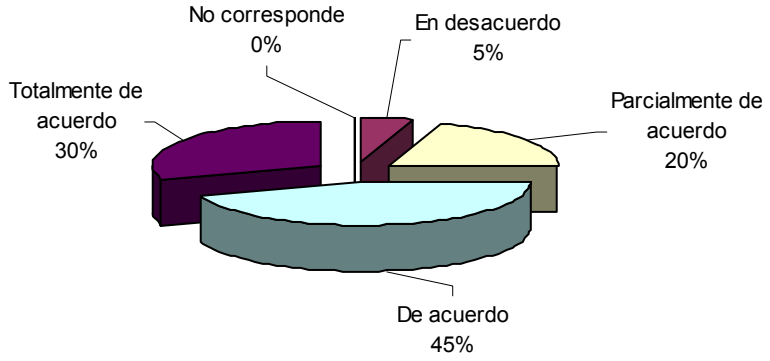
Requerimiento Validación más definido con OA en Casos de Uso



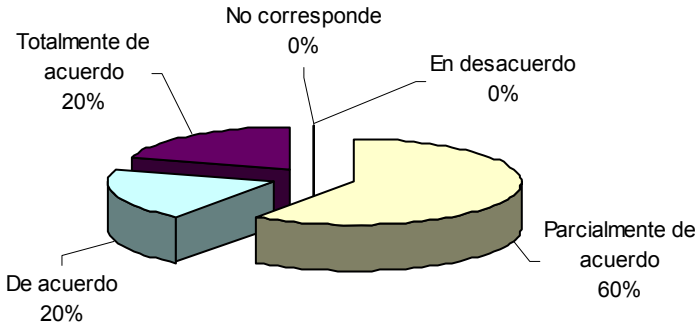
Requerimiento Registro de Errores más definido con OA en Casos de Uso



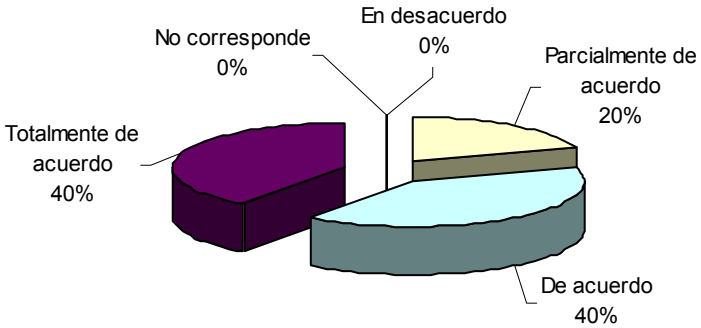
Aspectos más definidos con OA en Casos de Uso



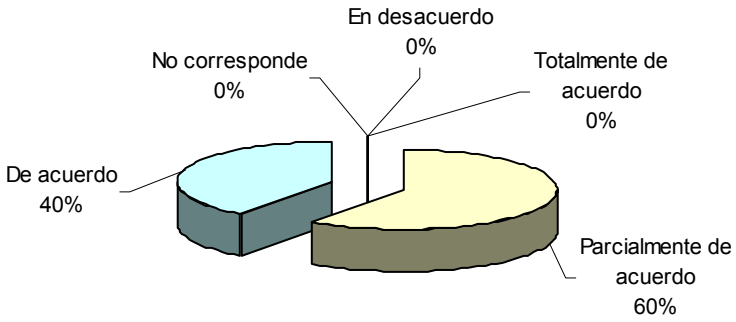
Incorporar OA en Caso de Uso ¿requiere más tiempo?



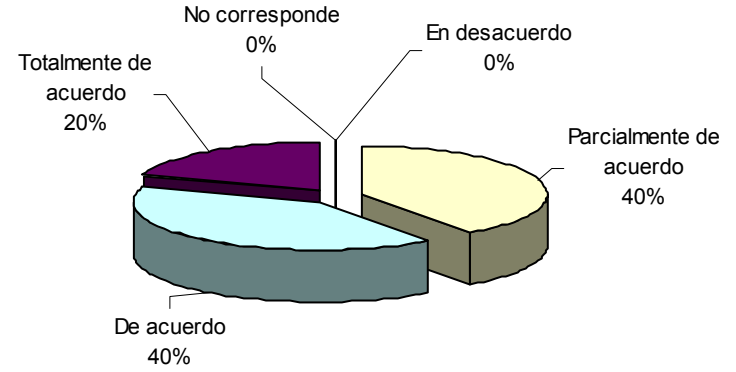
¿Incorporaría OA en Casos de Uso en proyectos futuros?



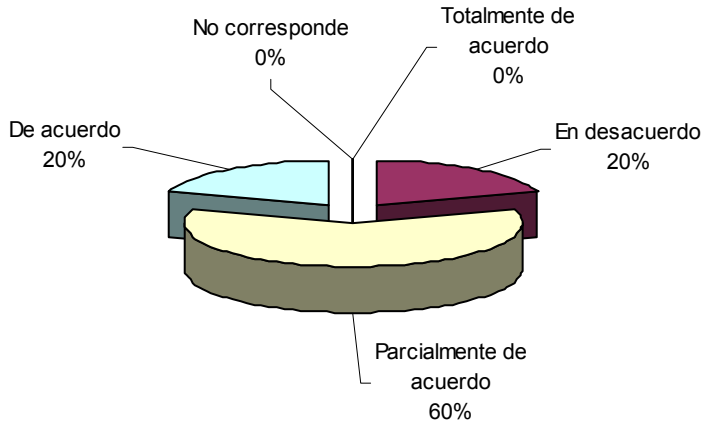
¿Mejora el modelo de clase al incorporar OA?



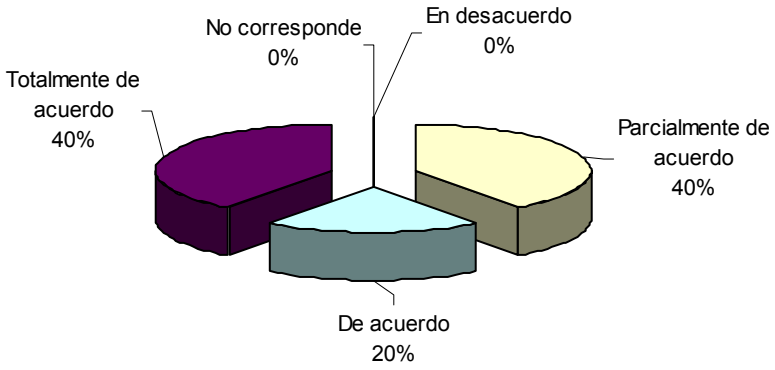
¿Es de fácil entendimiento el modelo de clase con OA?



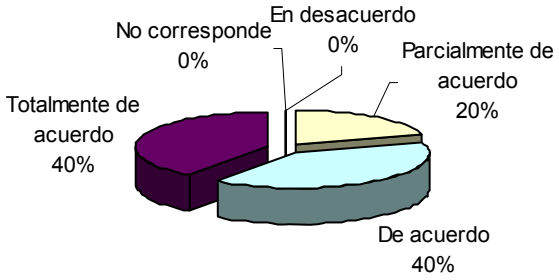
Incorporar OA en modelo de clase ¿requiere más tiempo?



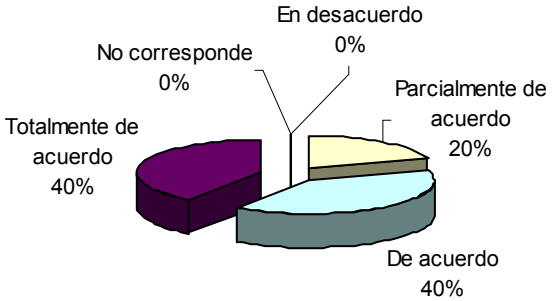
¿Incorporaría OA en modelo de clase en proyectos futuros?



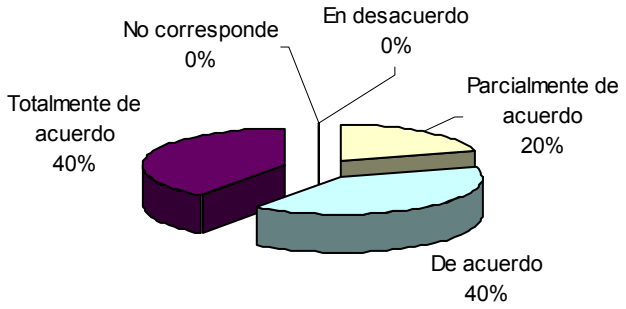
Requerimiento Login más definido con OA en diagrama de secuencia



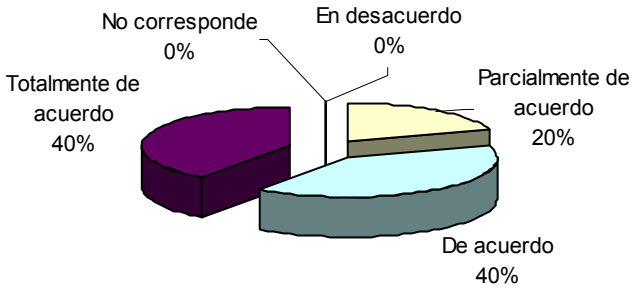
Requerimiento Auditoria más definido con OA en diagrama de secuencia



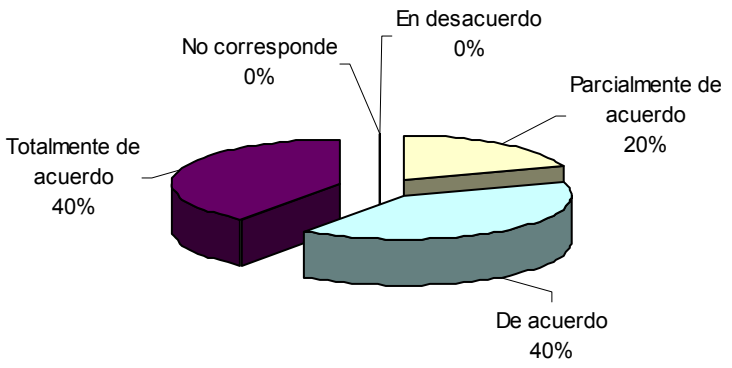
Requerimiento Validación más definido con OA en diagrama de secuencia



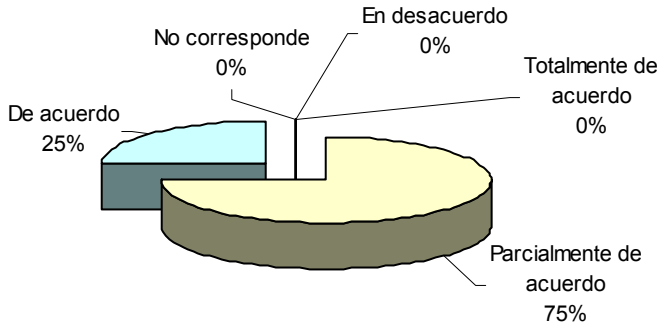
Requerimiento Registro de Errores más definido con OA en disgrama de secuencia



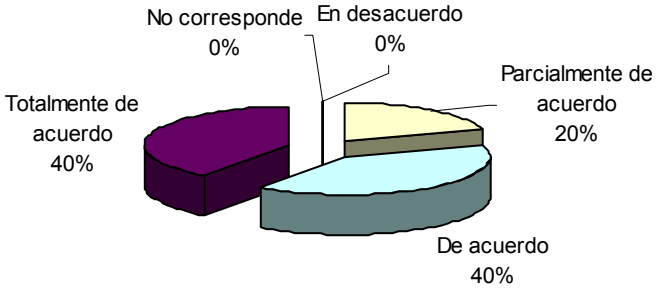
Aspectos más definidos con OA en diagrama de secuencias



Incorporar OA en diagrama de secuencia ¿requiere de más tiempo?



¿Incorporaría OA en diagrama de secuencia en proyectos futuros?



ANEXO G
Código AspectJ

```
import org.aspectj.lang.*;
import java.util.*;

aspect AuditoriaElem {
  Hashtable joinPointsSeen = new Hashtable();

  pointcut methodCall() : call(* Rango.*(..));
  pointcut constructorCall() : call(Rango.new(..));

  pointcut outsideCall() :
    (methodCall() || constructorCall()) && !within(Rango);

  pointcut interestingJoinPoints() :
    outsideCall() || within(Rango);

  pointcut mainExecution() :
    execution(public static void Rango.main(String[]));

  pointcut modificarExecution() :
    execution(public static void Rango.modificar(String[]));

  pointcut bloquearExecution() :
    execution(public static void Rango.bloquear(String[]));

  protected void registrarAuditoria(String accion, String  usuario, Date
fechahora){
     //registroAuditoria
     .
     .
     .
  }

  after() : mainExecution() {
       //registroAuditoria de creación de rango
       registrarAuditoria(“crear”,usuario,fechahora);
       .
       .
  }

  after() : modificarExecution() {
       //registroAuditoria de modificación rango
       registrarAuditoria(“crear”,usuario,fechahora);
       .
       .
  }

  after() : bloquearExecution() {
       //registroAuditoria de bloqueao de rango
       registrarAuditoria(“crear”,usuario,fechahora);
       .
       .
  }

}
```

```
class Rango {
  void crearRango() {
        .
        .
        .
  }
  void modificarRango() {
        .
        .
        .
  }
  void bloquearRango() {
        .
        .
        .
  }

  public static void main(String[] args) {
    new Rango().crearRango();
  }

  public static void modificar(String[] args) {
    new Rango().modificarRango ();
  }

  public static void bloquear(String[] args) {
    new Rango().bloquearRango ();
  }

}
```