Universidad de las Ciencias Informáticas

Facultad 3

Título: Modelación e implementación del módulo Documentos del subsistema Depósitos de Aduana.

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Juan Javier Dans Moreno.

Tutores: Ing. Lianet Pineda De la Nuez.

Ing. Yuniel Rodríguez Bello.

La Habana, Junio 2011

"Año 53 de la Revolución"

DECLARACIÓN DE AUTORÍA

Declaro ser autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año_____.

Juan Javier Dans Moreno.

_____

Firma del Autor

Ing. Lianet Pineda de la Nuez.

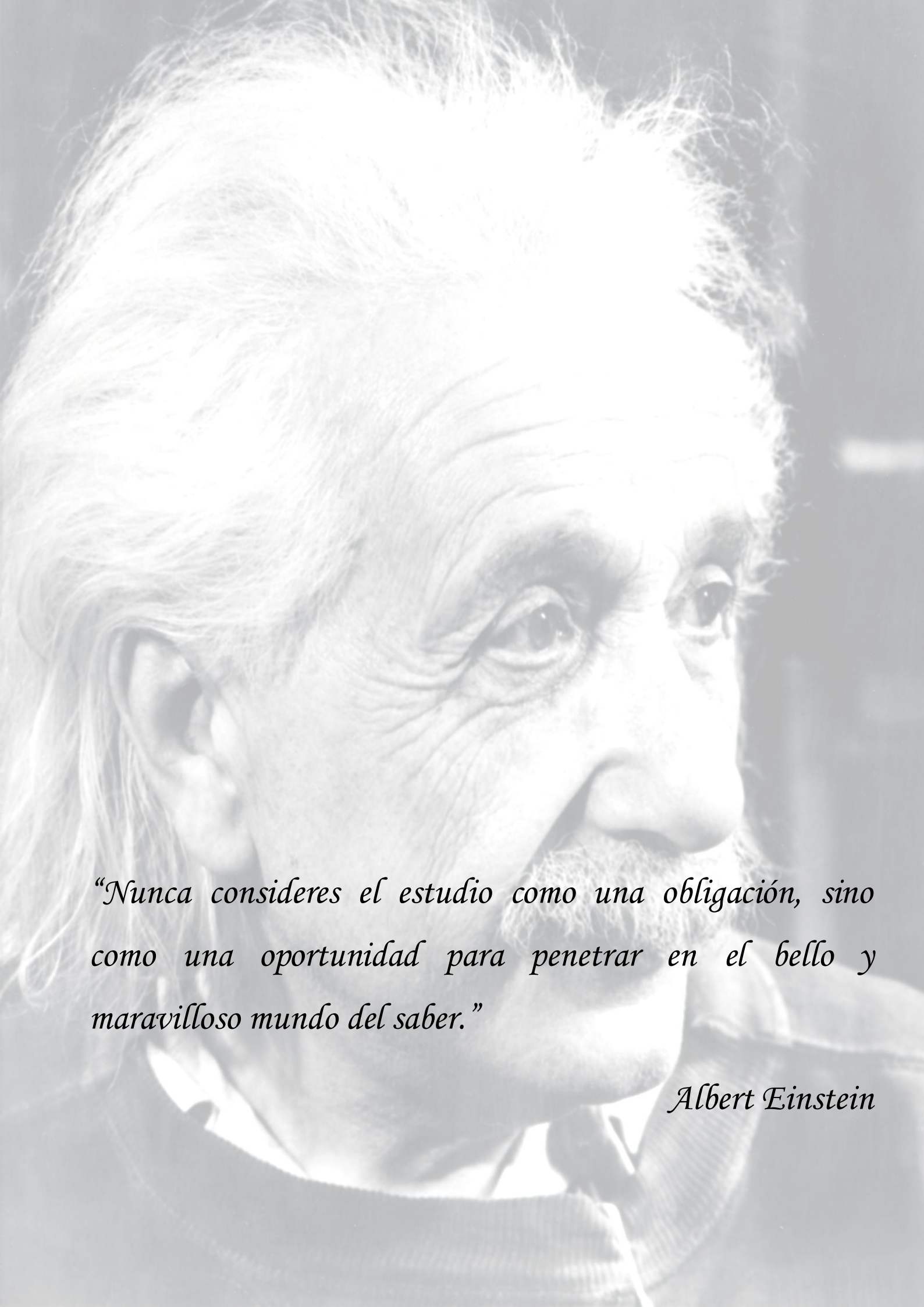
_____

Firma del Tutor

Ing. Yuniel Rodríguez Bello.

_____

Firma del Tutor



"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber."

Albert Einstein

AGRADECIMIENTOS

A mis padres Bárbara Moreno y Enrique Juan Dans, por haberme traído al mundo, por confiar en mí en todo momento y sobre todo por su ejemplo, su amor y su cariño, sin ellos no hubiera sido posible seguir.

A mis tíos Cecilia y Guillermo por su preocupación constante, por apoyarme y quererme como un hijo más.

A mis hermanos y en especial Enrique por su cariño, confianza y lo que representas para mí.

A toda mi familia en general, vivos y fallecidos, los quiero mucho.

A mi tutora Lianet Pineda por ser una persona especial, por comprenderme y enseñarme, por su paciencia y por estar siempre dispuesta a ayudarme.

A mi tutor Yuniel Bello por sus consejos, su forma de ser y por toda la ayuda brindada.

A ambos tutores muchas gracias, sin su ayuda no hubiera podido lograrlo.

A todo el equipo de desarrollo y amigos del subsistema Depósito nunca los olvidaré por ayudarme tanto: Leonardo, mi hermanita Elizabeth, Yaksel, Gabriela e Irina.

A la persona que considero mí compañera de tesis: Yusney Martínez Estrada.

A mis amigos por soportarme, entenderme, quererme y ayudarme siempre que lo necesite: Boris Enrique, Yordani, Jorge Carlos Driggs, Ricardo Pablo, Beatriz Molina, Marlon Niebla, Franklin, Eyeris, Dayán, Raymond Weeden, Lázaro Gil, en fin a todo el que de una forma u otra hicieron posible que un sueño hoy se convierta en realidad.

A la Revolución Cubana y a Fidel, por crear universidades e instituciones que brindan la oportunidad de penetrar en el bello y maravilloso mundo del saber.

A todos mis compañeros de aula. Muchas Gracias!!!

Juan Javier Dans Moreno.

DEDICATORIA

Dedico este trabajo de diploma a todas las personas que quiero, en especial mis padres, hermanos, abuelos, tíos, primos y demás familiares por ser ustedes una fuente de inspiración para mí.

A todos los amigos y personas que he conocido en el transcurso de estos cinco años de mi vida y a la Universidad de las Ciencias Informáticas, institución donde logré cumplir varios de mis sueños.

Juan Javier Dans Moreno.

RESUMEN

El uso de las Tecnologías de la Información y las Comunicaciones permiten a las organizaciones en gran medida la optimización de sus recursos y mejora de sus operaciones, logrando aumentar su eficiencia y competitividad.

La Aduana General de la República de Cuba se ha trazado como meta la automatización e informatización de la mayoría de los procesos de sus diversos espacios de trabajo. El área correspondiente a los Depósitos de Aduana presenta varias dificultades debido a que los inspectores aduaneros efectúan de manera manual el procesamiento de la documentación y la gestión del inventario. El presente trabajo de diploma tiene como objetivo desarrollar el módulo Documentos para el subsistema Depósitos de Aduana que facilite la gestión de los procesos asociados al control de la documentación.

Con este fin se utilizó la metodología de desarrollo RUP y el lenguaje de modelado UML para elaborar todos los artefactos. Para obtener un mayor entendimiento del negocio se realizó el modelado de procesos con la notación BPMN empleando la herramienta CASE Visual Paradigm. Mediante el empleo de las técnicas de captura y validación de requerimientos se definieron los requisitos funcionales del sistema. Las tecnologías que serán utilizadas para su implementación son la herramienta de desarrollo NetBeans, el lenguaje de programación PHP (*del inglés, Hypertext Preprocessor*), utilizando el framework Symfony, la librería ExtJS de JavaScript y Oracle como sistema gestor de base de datos.

Palabras claves: Aduana**,** Depósitos, Procesamiento de documentos

TABLA DE CONTENIDOS

AGRADECIMIENTOS I

DEDICATORIA I

RESUMEN I

TABLA DE CONTENIDOS I

INTRODUCCIÓN 1

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA 5

1.1 Introducción 5
1.2 Generalidades de los Depósitos de Aduana 5
1.3 Soluciones informáticas para la gestión de la documentación de los Depósitos de Aduana 7
1.3.1 Soluciones Extranjeras 7
1.3.1.1 Depósito aduanero 7
1.3.1.2 S4 tr@nsERP 8
1.3.1.3 Sistema Aduanero Automatizado SIDUNEA 10
1.3.1.4 CLIP™ 14
1.3.1.5 TICA 14
1.3.2 Soluciones Cubanas 17
1.3.2.1 SUA 17
1.4 Ingeniería de Requerimientos 17
1.4.1 Actividades de la Ingeniería de Requerimiento 18
1.4.2 Técnicas de captura de requerimientos 19
1.4.3 Técnicas de validación de requerimientos 21
1.5 Diseño de Software 22
1.5.1 Arquitectura de Software 22
1.5.2 Estilos Arquitectónicos 23
1.5.3 Patrones 25
1.5.3.1 Patrones de Arquitectura 26
1.5.3.2 Patrones de Diseño 27
1.6 Metodología, Lenguaje y Herramientas de modelado utilizadas para el desarrollo 28
1.6.1 Metodología de Desarrollo de Software 28
1.6.2 Notación para el Modelado de Procesos de Negocio 31

1.6.3 Lenguaje de Modelado 31
1.6.4 Herramienta CASE para el modelado 32
1.6.5 Leguaje de Programación....32
1.6.6 Marco de Trabajo (Framework) 33
1.6.7 Interfaz de usuario 34
1.6.8 Entorno de Desarrollo Integrado (IDE)....34
1.6.9 Gestor de Base de Datos 35
1.7 Conclusiones....36
CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA....38
2.1 Introducción....38
2.2 Modelado de los procesos del Negocio....38
2.2.1 Descripción del proceso Gestionar Depósitos de Aduana ..**¡Error! Marcador no definido.**
2.2.2 Descripción del subproceso Gestionar Remisión de Entrada 40
2.3 Análisis Crítico de la Ejecución de los Procesos....40
2.4 Modelo Conceptual 42
2.5 Especificación de los Requerimientos del Sistema....43
2.5.1 Técnicas para la Captura de Requerimientos....43
2.5.2 Requerimientos Funcionales 43
2.5.3 Técnicas para la Validación de Requerimientos 46
2.6 Aportes de la Solución y Beneficios Esperados....46
2.7 Conclusiones....47
CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DEL SISTEMA....48
3.1 Introducción....48
3.2 Diseño del sistema....48
3.2.1 Patrones utilizados....48
3.2.1.1 Patrones GRASP....49
3.2.1.2 Patrones GoF....49
3.2.1.3 Patrón MVC según Symfony 50
3.2.2 Diagrama de clases del Diseño con Estereotipos Web....51
3.2.3 Paquetes lib e Interfaces JS....53
3.2.4 Diagrama de interacción....56
3.2.4.1 Diagrama de Secuencia....57
3.2.4.2 Diagrama de Secuencia Orientado a Actividades del Negocio 57

3.2.5 Diagrama de clases persistentes 58
3.2.6 Diseño de la Base de Datos 60
3.2.7 Métricas para la evaluación del diseño 61
3.2.7.1 Métrica de Tamaño Operacional de Clase (TOC) 61
3.3 Implementación del sistema 64
3.3.1 Estándar de Codificación 64
3.3.2 Tratamiento de Errores 66
3.3.3 Diagrama de Despliegue 66
3.3.4 Diagrama de Componentes 67
3.4 Conclusiones 68
CONCLUSIONES 69
RECOMENDACIONES 70
REFERENCIAS BIBLIOGRÁFICAS 71
BIBLIOGRAFÍA .... **¡ERROR! MARCADOR NO DEFINIDO.**
ANEXOS .... **¡ERROR! MARCADOR NO DEFINIDO.**
GLOSARIO 73

INTRODUCCIÓN

Las Tecnologías de la Información y las Comunicaciones (TIC) conforman el conjunto de recursos necesarios para gestionar la información y particularmente los ordenadores, programas informáticos y redes necesarias para convertirla, almacenarla, administrarla, transmitirla y encontrarla. Con la utilización de las TIC las organizaciones han logrado grandes beneficios, como son la optimización de sus recursos y mejora de sus operaciones, logrando aumentar su eficiencia y competitividad.

La información se encuentra en diferentes medios como: el papel, las cintas magnéticas, los discos duros, los diskettes, entre otros. Cada día, es mayor la cantidad de empresas, entidades y personas que presentan la necesidad de informatizar los servicios, incluyendo el control de los documentos. Teniendo un buen control de la administración de la información se pueden tomar mejores decisiones.

Una de las metas que tiene Cuba, en el presente, y para los próximos años es la informatización de entidades, empresas y la sociedad. El desarrollo del software cubano es un paso importante para lograr independencia de soluciones informáticas de empresas privadas y contribuir al desarrollo del país. La Universidad de las Ciencias Informáticas (UCI) juega en esto un papel importante porque en ella se forman los futuros profesionales del software, se realizan productos y proyectos de cooperación con entidades nacionales.

La Aduana General de la República (AGR) forma parte del sistema de control estatal y actúa en función de ello, recaudando los derechos de aduanas y facilitando respuesta dentro de su jurisdicción y competencia a los hechos que incidan en el tráfico internacional de mercancías, viajeros, postal y los medios que los transportan, previniendo, detectando y enfrentando el fraude y el contrabando, así como contribuyendo a la protección nacional e internacional del medio ambiente. (1)

La AGR apoyado en su Centro de Automatización de la Dirección y la Información (CADI), y la UCI, se encuentra trabajando en la informatización de la mayoría de sus procesos y la digitalización de los documentos que se utilicen en ellos, constituyendo de notable importancia la gestión y control del inventario de los Depósitos de Aduana; entiéndase por estos, lugar o instalación autorizado por la

Aduana para el almacenamiento de mercancías sujetas al régimen aduanero[1] de Depósito de Aduanas[2], el que puede ser público o privado. (2)

En la actualidad los Depósitos de Aduana poseen sistemas informáticos para la gestión del inventario que cumplen los requerimientos de la normativa vigente. Los depositarios[3] proporcionan la información correspondiente a la Aduana mediante soportes digitales o documentos generados en el formato tradicional debido a que la Aduana no dispone de una solución informática que le permita recepcionar y manipular la información proveniente de estos sistemas con la finalidad de vincularla al proceso de despacho aduanero[4], lo cual provoca gastos en recursos, tales como: combustible, materiales de oficina, entre otros. Como consecuencia la gestión y control de las operaciones por parte de los inspectores aduaneros se vuelve un proceso engorroso e ineficiente y en numerables ocasiones la información no se tiene en el momento preciso para la toma de decisiones oportuna y las acciones correctivas necesarias.

A partir de la problemática planteada se define como **problema a resolver**: ¿cómo gestionar los procesos asociados al control de la documentación de los Depósitos de Aduana?

Definiéndose como **objeto de estudio**: los procesos de los Depósitos de Aduana.

Según lo planteado anteriormente se establece como **campo de acción**: los procesos asociados al control de la documentación de los Depósitos de Aduana.

Una vez planteada la problemática se define como **objetivo general**: desarrollar un sistema informático que gestione los procesos asociados al control de la documentación de los Depósitos de Aduana.

[1] Tratamiento aplicable a las mercancías sometidas al control de la Aduana, de acuerdo con la Normativa Aduanera, según la naturaleza y objetivos de la operación.

[2] Régimen aduanero mediante el cual las mercancías importadas se almacenan bajo control aduanero en un lugar designado a este efecto (Depósito de Aduana), sin el pago de los derechos de aduanas y en espera de que se le otorgue un nuevo régimen.

[3] Persona natural o jurídica habilitada por la Aduana para operar Depósitos de Aduana o almacenes de mercancías bajo control aduanero.

[4] Cumplimiento de las formalidades aduaneras necesarias para exportar, importar o para colocar las mercancías bajo otro régimen aduanero.

Se plantea la siguiente **idea a defender**: modelar e implementar un sistema informático permitirá la gestión de los procesos asociados al control de la documentación de los Depósitos de Aduana.

Del objetivo general se derivan los siguientes **objetivos específicos**:

- Elaborar el marco teórico relativo a las soluciones de software que gestionan los Depósitos de Aduana y el diseño de sistemas informáticos.
- Modelar los procesos asociados al negocio, objeto de informatización.
- Especificar los requisitos funcionales.
- Modelar el diseño del sistema para definir las clases y componentes necesarios para su implementación.
- Implementar un sistema informático que gestione el control de la documentación de los Depósitos de Aduana.

Con el fin de dar cumplimiento a los objetivos del presente trabajo, se concibieron las siguientes **tareas de investigación**:

- Análisis del estado del arte de las soluciones de software que gestionan los Depósitos de Aduana.
- Análisis de la metodología de desarrollo de software, lenguajes, herramientas de modelado y tecnologías utilizadas en el proyecto.
- Análisis de la normativa aduanera vigente correspondiente al control de la documentación de los Depósitos de Aduana.
- Definición de los requisitos funcionales del sistema a desarrollar.
- Modelación de los artefactos del diseño: modelo de datos, diagrama de clases, diagrama de secuencia orientado a actividades del negocio, diagrama de despliegue.
- Validación técnica y funcional del diseño obtenido.
- Implementación del sistema.

Los **posibles resultados** son**:**

- Modelo de procesos del negocio.
- Descripción de los procesos del negocio.

- Especificación detallada de los requisitos funcionales.
- Modelo de datos.
- Diagrama de clases del diseño.
- Diagrama de secuencia orientado a actividades del negocio.
- Diagrama de despliegue.
- Diagrama de componentes.
- Código fuente del sistema, que permitirá el control de la documentación de los Depósitos de Aduana.

El trabajo está estructurado por 3 capítulos en los que se encuentra el contenido distribuido de la siguiente forma:

El **primer capítulo** muestra la fundamentación teórica de la investigación: el estado del arte de algunas de las soluciones informáticas existentes para la gestión de los Depósitos de Aduana, a nivel internacional y nacional. Detalla la metodología, lenguaje y herramientas utilizadas en el desarrollo de la solución, técnicas en la ingeniería de requerimientos, y los patrones de software más utilizados.

El **segundo capítulo** se refiere a las características del sistema, se expone y se describe de manera general los procesos que se ejecutan en los Depósitos de Aduana de la AGR, las técnicas utilizadas para la captura y validación de los requerimientos, además se muestra un listado de los requerimientos funcionales con una pequeña descripción, el modelo conceptual y los beneficios esperados con esta solución.

El **tercer capítulo** aborda temas correspondientes al diseño e implementación de la solución, contiene los patrones de diseño utilizados en el desarrollo del sistema, se muestran los diagramas de secuencia orientado a actividades, de clases del diseño con estereotipos Web, de clases persistentes, diseño de la base de datos, métricas para la evaluación del diseño, estándar de codificación y los diagramas de despliegue y componentes.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En el capítulo se describen los elementos principales que fundamentan el contenido de este trabajo. Se realiza un estudio profundo de las soluciones informáticas nacionales e internacionales para la gestión de los Depósitos de Aduana, de algunas de las técnicas de captura y validación de los requisitos. Se describen aspectos importantes de los estilos arquitectónicos, patrones de desarrollo de software, herramientas, metodología y lenguaje para el desarrollo del producto.

1.2 Generalidades de los Depósitos de Aduana

La República de Cuba es signataria del *Convenio para Simplificación y Armonización de los Regímenes Aduaneros*, conocido por *Convenio de Kyoto*, desde 1995, y en consecuencia la legislación aduanera nacional relativa a la aplicación de los distintos regímenes previsto en este cuerpo legal se ha adecuado íntegramente a lo que en él se dispone.

El Decreto Ley No. 162, de 3 de abril de 1996, De Aduanas, establece en su artículo 156, que el régimen de depósito de aduanas es aquel con arreglo al cual las mercancías importadas se almacenan bajo control aduanero en un lugar designado a este efecto, sin el pago de los derechos de aduana y en espera de que se les otorgue un nuevo régimen. (3)

Como puede apreciarse, existen dos características esenciales en este régimen: la primera: que las mercancías se depositan en almacenes bajo el control de la autoridad aduanera; y la segunda, sin el pago de los derechos e impuestos a la importación.

El 26 de junio de 1999, fue enmendado el referido convenio en Bruselas, Bélgica, quedando integrado por tres partes: el Cuerpo del Convenio, el Anexo General y los Anexos Específicos. El Anexo D, del citado Convenio, se refiere a los depósitos aduaneros definiéndolos como: el régimen aduanero según el cual las mercancías importadas son almacenadas bajo control de la Aduana en un lugar habilitado para esta finalidad (depósito aduanero) sin pagar derechos o impuestos a la importación. (4)

Así mismo, este cuerpo legal clasifica los depósitos aduaneros en públicos y privados, especificando que los últimos serán previstos por las legislaciones nacionales, cuando las necesidades particulares del sector comercial lo justifiquen.

El Régimen de Depósito de Aduanas en Cuba se encuentra regulado por la Resolución No. 12, de 3 de julio de 2002, del Jefe de la Aduana General de la República, Normas para la aplicación del Régimen Aduanero de Depósito de Aduanas y para los Locales, Instalaciones y Áreas destinadas a depositar las mercancías sujetas a dicho régimen. (5)

En Cuba los depósitos se clasifican en públicos y privados, existiendo tres modalidades a las que pueden acogerse los comerciantes a quienes interese disfrutar de las facilidades que brinda este régimen. Estas son: (5)

- **Depósitos de carácter público:** destinados a depositar mercancías de cualquiera que ostente la titularidad del régimen, mediante el arrendamiento de espacios o la contratación de servicios de almacenamiento, entre la administración del depósito y los titulares del régimen. En ellos el depositario y el depositante son personas jurídicas distintas, que establecen entre sí una relación contractual, sin perjuicio de los derechos y obligaciones derivados del disfrute de la autorización otorgada por la Aduana.

- **Bodega Pública:** es un espacio dentro del Depósito de carácter público que, formando parte de éste, está destinado a prestar los servicios de recepción, almacenamiento, custodia y facturación de las mercancías de varios depositantes.

- **Depósito de carácter privado:** destinados al uso exclusivo del titular que disfruta del régimen, al que se autoriza para el almacenamiento de mercancías propias de su actividad, no permitiéndose en ellos el arrendamiento de locales ni la contratación de servicios de almacenamiento a terceros. En ellos el depositario y el depositante o titular del régimen coinciden en una misma persona jurídica.

La utilización del Régimen de Depósito de Aduanas en Cuba, reporta numerosos beneficios tanto para las empresas importadoras y exportadoras cubanas, como para los empresarios extranjeros que realizan actividades de comercio exterior en nuestro país.

Las mercancías que se encuentran bajo el régimen de depósito de aduanas pueden transferirse de un depósito a otro dentro de la misma o hacia distinta jurisdicción aduanera mediante el régimen especial de tránsito aduanero.

El almacenaje de las mercancías en el depósito permite la realización de actividades tales como: toma de muestras para su correcta clasificación, operaciones de conservación y mantenimiento, reempaque,

formación de lotes y otros cambios con el objetivo de mejorar su apariencia y facilitar su transportación o comercialización siempre que no se realicen transformaciones sustanciales.

1.3 Soluciones informáticas para la gestión de la documentación de los Depósitos de Aduana

En la actualidad los Depósitos de Aduana emiten gran cantidad de documentación debido a los altos niveles de movimiento de mercancías que presentan, el control y gestión de dicha información es un aspecto necesario e importante para las Aduanas.

En el mundo actual existe un gran desarrollo, avance y conocimiento de las TIC, lo cual posibilita a las Aduanas poseer sistemas informáticos para el control de la documentación e información que se genera en los Depósitos de Aduana.

A continuación se realiza un análisis de algunas de las soluciones extranjeras y cubanas existentes actualmente, utilizadas con este fin.

1.3.1 Soluciones Extranjeras

1.3.1.1 Depósito aduanero

Aplicación que permite la explotación de Depósitos Aduaneros en cualquiera de sus facetas:

- Depósito aduanero (público o privado).
- Depósito distinto del aduanero.
- Almacén de avituallamiento.
- Almacén de depósito temporal (A.D.T.).
- Local autorizado para mercancías de exportación (L.A.M.E.).
- Almacenaje mercancías nacionales.

Todas estas facetas se encuentran dentro de una sola aplicación, los registros de entradas y salidas se realizan de forma única, las mercancías se pueden mover de un recinto a otro dentro de la misma aplicación, en principio la aplicación se contempla como estándar, aunque existe la posibilidad de efectuar las modificaciones o ampliaciones en la medida en que los usuarios requieran para la perfecta integración de la aplicación con la general de la empresa mediante la importación o exportación de

ficheros con la información necesaria, el envío y recepción de documentos EDI[5] (*del inglés, Electronic Data Interchange*) es totalmente transparente para el usuario y presenta un servicio denominado: *Solicitud de Permiso,* con el objetivo de asesorar en la redacción y confección de los escritos y documentos a los clientes para facilitarles la transmisión de los permisos para la autorización del tipo de Depósito en el que esté interesado.

Brinda una serie de informes que presentan como objetivo principal informar a la Aduana de Control de los movimientos habidos en un período de tiempo determinado, estos son:

- Existencias por Ubicaciones.
- Existencias por Clientes.
- Existencias por Recintos.
- Existencias a una fecha.
- Existencias por Código Mercancía.
- Búsqueda por artículo.
- Existencias por Cliente / Remitente.
- Inventario Valorado.
- Cálculo estadías.
- Estadías por Clientes.
- Stocks Remitentes / Destinatarios.
- Facturación Movimientos.
- Saldos.
- Salidas Documento Único Aduanero (DUA) Recapitulativo.

1.3.1.2 S4 tr@nsERP

Presenta distintos módulos de depósito basados en los requerimientos de cada una de las aduanas que gestiona, de modo que, además de los controles clásicos de un almacén, se incorporan las

[5] Intercambio electrónico de datos: intercambio entre sistemas de información, por medios electrónicos, de datos estructurados de acuerdo con normas de mensajes acordadas.

transmisiones EDI y los diferentes listados de control requeridos por la Aduana. Pueden funcionar como una aplicación independiente o integrada con el resto de sus módulos.

Módulos

- Depósito Aduanero.
- Depósito para Restitución.
- Depósito Fiscal.
- Almacén de Depósito Temporal (ADT).
- Depósito Distinto del Aduanero.

Características Técnicas:

- Multiempresa, Multidelegación.
- Parametrizable:
    - En acceso y búsqueda de la información.
    - En trazabilidad.
    - En funcionalidad y modelización de datos.
    - En reportes y análisis.
- Control del acceso a la información por módulos de seguridad.
- Comunicación por mail, fax e internet.
- Mensajería electrónica integrada.
- Integración:
    - Con su gestión de almacén.
    - Con su web, personalizado para clientes, usuarios y colaboradores.
    - Con otros módulos de S-4.
- Construido con herramientas Microsoft, SQL Server como almacén de datos. Intercambio con XML (*del inglés, Extensible Markup Language*) y herramientas del sistema Office de Microsoft para importación y exportación de información.

Características específicas del módulo Depósito Aduanero:

- Dos versiones: Público y Privado.
- Inclusión en Depósito Aduanero Privado (IDA).
- Registro de entradas y salidas.
- Control por referencias de producto.
- Ventas en Depósito.
- Transformaciones en depósito (RUN).
- Traspasos entre Depósitos.
- Mensajes EDI.
- Listado de existencias.
- Contabilidad de existencias.
- Trazabilidad.
- Integración con módulo de ADT.
- Integración con módulo de Aduanas.
- Integración con facturación emitida y soportada (costes).
- Exportación de datos a sistema Office.

1.3.1.3 Sistema Aduanero Automatizado SIDUNEA

Desarrollada por la Conferencia de las Naciones Unidas sobre el Comercio y el Desarrollo (UNCTAD), es la herramienta informática para el control y administración de la gestión aduanera, actualmente es usada con éxito en más de 80 países.

En la actualidad la UNCTAD ofrece tres versiones del sistema, las cuales son:

- SIDUNEA World
- SIDUNEA++
- SIDUNEA v2

De las versiones presentadas anteriormente se analiza principalmente SIDUNEA World debido a que es el último resultado presentado a nivel mundial, se basa en las experiencias exitosas de SIDUNEA++, que fue diseñado para funcionar en entornos de telecomunicaciones difícil y operar a través de redes GSM (*del inglés, Global System for Mobile Communications*) que están generalizadas en los países en desarrollo. Estar basado en la web, el sistema SIDUNEA World permitirá a las administraciones aduaneras y los operadores realizar la mayoría de sus transacciones - Declaraciones de la Aduana de Manifiestos de Carga y los documentos de tránsito - a través de Internet.

Características

Las funcionalidades básicas del sistema están diseñadas para:

- Facilitar y mejorar el cálculo, recaudación y contabilidad de los derechos de aduana y otros cargos relacionados con las operaciones aduaneras.
- Acelerar el despacho de las mercancías y evitar el contrabando.
- Proporcionar la gestión aduanera con información oportuna y precisa.

Esto se logra por:

- Verificación automática, la validación y la contabilidad de las declaraciones de aduanas.
- Aplicación uniforme de la ley y de los tipos de corrección.
- Capacidad para monitorear y rastrear el pago de derechos e impuestos.
- Control del valor y la descripción de la corrección de mercancías a través del uso de la valoración en línea y bases de datos arancelarias.
- Selección de operaciones de riesgo a efectos de control.
- Un procesamiento más rápido de documentos y reducir al mínimo la manipulación manual de documentos.
- Aplicación sistemática de métodos de trabajo y la asignación óptima de las funciones y recursos.

Ventajas

Algunos de los beneficios que brinda este sistema son:

- Aumento de las capacidades de control de aduanas (por ejemplo, la evaluación del riesgo, el acceso en línea a bases de datos externas, control de imágenes).
- Aumento de las capacidades para la contabilidad, la auditoría a posteriori, las estadísticas y gestión de la información.
- Más fácil la agregación de datos a nivel regional e internacional.
- Más rápida la toma de decisiones.
- Pone en práctica las normas establecidas por la Organización Internacional para la Estandarización (ISO), de las Naciones Unidas, de la Organización Mundial de Aduanas (OMA) y de la Organización Mundial de Comercio (OMC).
- Soporte completo de la administración electrónica y la capacidad de interoperar en línea con el exterior, sistemas de gobierno y bases de datos.
- El sistema asegura la cobertura completa del proceso de liquidación e incluye la función de la capacidad para apoyar a los requisitos nacionales específicos y los frecuentes cambios de regulación.
- Capaz de adaptarse a las diferentes estructuras de organización, generando economías de escala financiera.
- Completa evaluación del riesgo y capacidad de selectividad, las nuevas funciones como la integración de las imágenes o el uso de códigos de barras están disponibles, así como el acceso en línea a bases de datos externas.
- Los cambios y actualizaciones del sistema se realizan sin necesidad de programación, completa flexibilidad.
- El código fuente es propiedad del país usuario del sistema.

- Es totalmente compatible con SIDUNEA++, asegurando una transición sin problemas a las aduanas electrónicas.
- Permite la posibilidad de modificar o desarrollar módulos adicionales a nivel nacional utilizando plantillas existentes.

Tecnología

Características:

- Se encuentra escrito íntegramente en Java, se ejecuta de forma local o mediante la red.
- La arquitectura se encuentra basada en niveles a base de productos modulares.
- Se ejecuta en cualquier base de datos: Oracle, Informix, Sybase, Interbase, SQL Server, donde exista un puente JDBC v3.0.
- Funciona en cualquier sistema operativo equipado con una máquina virtual de Java: Linux, Solaris, AIX, HP / UX, Windows XP / NT, Mac o cualquier lugar donde haya una máquina virtual Java.
- Presenta el uso de XML que permite el intercambio de cualquier documento dentro y fuera del sistema, entre las administraciones aduaneras y los comerciantes y entre las administraciones aduaneras en los distintos países.
- Permite soluciones destinadas a los entornos de clúster de alta disponibilidad.

La plataforma SIDUNEA World cumple con los requisitos generales siguientes:

- Basada en Internet.
- Independiente de plataformas de hardware.
- Independiente de RDBMS (*del inglés, Relational Data Base Management System*).
- Resistencia a las averías de telecomunicaciones.
- Escalabilidad.
- Construido en funciones de seguridad.

La plataforma se basa en una arquitectura sofisticada, que elimina la necesidad de mantener conexiones permanentes con un servidor nacional, algo que es particularmente importante para los países en materia de telecomunicaciones poco fiables. Donde las telecomunicaciones son más fiables, el enfoque de la Web tradicional puede ser utilizado con mayor amplitud. La plataforma SIDUNEA World también explota el potencial de los dispositivos móviles de acceso a Internet.

1.3.1.4 CLIP™

Solución informática para la gestión de Depósitos de Aduanas desarrollada por la empresa Cotecna, no sólo proporciona una imagen a tiempo real de los niveles del inventario, sino que además asegura que se apliquen controles aduaneros continuos y medidas de seguridad a lo largo de la cadena logística del depósito. Es una solución muy eficaz que une en una sola plataforma los depósitos y la logística, asegurando así la coexistencia óptima de la gestión del espacio, los informes y las funciones de control aduanero. Realiza informes sobre el inventario de los niveles de las existencias en cualquier momento e incorpora la determinación de aranceles e impuestos y el pago a las aduanas.

Algunas de las funciones y servicios que gestiona de manera general son:

- Recepción de mercancías en depósito.
- Gestión y logística.
- Almacenaje.
- Consolidación y desconsolidación.
- Seguridad y protección.
- Controles de inventario en tiempo real hasta su cesión.
- Gestión del pago de impuestos y aranceles.
- Documentación.
- Informes sobre la gestión.

1.3.1.5 TICA

El proyecto de Tecnología de Información para el Control Aduanero (TICA) es un componente del Plan Estratégico del Servicio de Aduanas que busca modernizar la gestión aduanera mediante el uso intensivo de la tecnología. Sistema aduanero que funciona en la Aduana de Costa Rica y desarrollado

por la misma, opera desde el 2007, es considerado un ejemplo de los beneficios de la modernización de los sistemas fiscales, elaborado con la herramienta de desarrollo GeneXus 9.0 y SQL Server 2008 como gestor de base de datos.

Las principales características que tiene el nuevo modelo al cual responde el sistema de información TICA son las siguientes:

- Implementación de las mejores prácticas en los procedimientos aduaneros.
- Utilización de un formato único de declaración aduanera.
- Automatización del proceso aduanero (recepción, validación, arancel integrado, pago, aceptación, selectividad y levante electrónico, entre otros).
- Autodeterminación y pago electrónico.
- Un sistema de información centralizado (una sola base de datos, registro de todas las operaciones).
- Control centralizado, basado en inteligencia del negocio y en la aplicación eficiente del análisis de riesgo.
- Conexión electrónica obligatoria con organismos públicos y privados (exenciones, notas técnicas, bancos).
- Eliminación de la presentación de papeles.
- Operación del sistema las 24 horas y 365 días al año.
- Adaptabilidad del sistema a las necesidades del comercio exterior (aplicación de convenios internacionales, tratados comerciales, entre otros).

Entre otros beneficios, la Administración aduanera podrá:

- Mejorar su servicio para cumplir con las exigencias de la sociedad costarricense.
- Disminuir sus costos de operación.
- Lograr mayor efectividad en el control aduanero.

- Aumentar la capacidad de los funcionarios para hacer cumplir la ley.

¿En términos generales, cuáles son los principales cambios que introducirá el TICA en los procedimientos aduaneros?

En Depósito:

- Asignación de número de inventario al ingreso de mercancías (mantiene relación con los documentos de ingreso y ubicación dentro del depósito).
- Mensaje de fin de tránsito (captura fecha y hora, consulta participación aduana, programa descarga).
- Reempaque y distribución (fraccionamiento y agrupamiento).
- Control de cambio de ubicación de mercancías con levante autorizado (tres días después de autorización de levante).
- Control de salidas de mercancías.
- Control de plazos y cambio a mercancías en abandono.
- Despacho a bordo (autorización y control).

Después de haber analizado las soluciones informáticas extranjeras para la gestión y control de las operaciones que se realizan en los diferentes depósitos bajo control aduanero, se puede concluir que a pesar de que brindan una amplia gama de funcionalidades, no son factibles para utilizar en la AGR, debido a diferentes argumentos como son:

- El sistema S4 tr@nsERP es construido con herramientas de Microsoft, es privativo por lo cual para poder utilizarlo es necesario comprar la licencia, utiliza como gestor de base de datos SQL Server y el paquete Office de Microsoft para importación y exportación de información.
- En el caso de Deposito Aduanero es necesario comprar la licencia para poder utilizarlo, es una aplicación de entorno escritorio y funciona solamente sobre plataforma Windows.
- Por otra parte tenemos que CLIP™ es desarrollado y comercializado por la empresa Cotecna y utilizado solamente por las empresas que realicen convenios con esta.

- El TICA implementa las legislaciones y normas específicas de Costa Rica, desarrollado con herramientas privativas como son: la herramienta de desarrollo GeneXus 9.0 y SQL Server 2008 como gestor de base de datos.

- SIDUNEA World es reconocido como el sistema más completo para las Aduanas ya que es adaptable y cumple con las normas establecidas por la ISO, OMA y OMC, pero para poder utilizarlo es necesario comprarlo y está desarrollado en el lenguaje de programación Java.

1.3.2 Soluciones Cubanas

1.3.2.1 SUA

La Aduana de Cuba presenta su propia solución informática denominada Sistema Único de Aduanas (SUA). El SUA automatiza los cinco procesos aduaneros (despacho comercial, no comercial, de viajeros, de medios de transporte y de postal y envío), posee una base de datos única y centralizada, a la que acceden en línea todas las aduanas del país. En la actualidad especialistas del CADI, estudiantes y profesores de la UCI se han centrado en la realización de la reingeniería del mismo con el objetivo de crear un sistema con herramientas más modernas y potentes.

Dentro de los diferentes módulos que presenta pueden encontrar: *Almacén*, el cual solo gestiona una parte de los procesos que se realizan en los depósitos temporales, los cuales son: la extracción y recepción de mercancías bajo los regímenes aduaneros de tránsitos y transferencias. Es una aplicación de entorno web, desarrollada con el lenguaje de programación PHP, paradigma de programación estructurada y utiliza como gestor de base de datos Oracle.

Actualmente el sistema SUA no presenta un módulo o subsistema para la gestión y control de las operaciones que se realizan en los Depósitos de Aduana, la AGR tampoco cuenta con la presencia de algún otro sistema que gestiones estos procesos, de ahí la necesidad que presenta el país de una solución informática que solucione los problemas planteados anteriormente.

1.4 Ingeniería de Requerimientos

En el glosario de Terminología de Ingeniería de Software de la IEEE se define requerimiento como:

"Una condición o necesidad de un usuario para resolver un problema o alcanzar un objetivo." (6)

"Una condición o capacidad que debe estar presente en un sistema o componente de un sistema para satisfacer un contrato, estándar, especificación u otro documento formal." (6)

También Ian Sommerville presenta una definición de acerca de lo que es requerimiento.

"Un requerimiento es simplemente una declaración abstracta de alto nivel de un servicio que debe proporcionar el sistema o una restricción de éste." (7)

La Ingeniería de Requerimientos (IR) cumple un papel primordial en el proceso de producción de software, ya que se enfoca en un área fundamental: definición de lo que se desea producir. Su principal tarea consiste en la generación de especificaciones correctas que describan con claridad, sin ambigüedad, en forma consistente y compacta, las necesidades de los usuarios o clientes; de esta manera, se pretende minimizar los problemas relacionados por la mala gestión de requisitos en el desarrollo de sistemas. (8)

Otros de los conceptos de la ingeniería de requerimientos son propuestos por Pressman el cual plantea.

"Ingeniería de Requerimientos ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajan. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactúan los usuarios finales con el software." (9)

1.4.1 Actividades de la Ingeniería de Requerimiento

Para realizar este proceso de Ingeniería de Requerimientos, no existe una única técnica estandarizada y estructurada que ofrezca un marco de desarrollo que garantice la calidad del resultado. Existe en cambio un conjunto de técnicas, cuyo uso proponen las diferentes metodologías para el desarrollo de aplicaciones web. Se debe tener en cuenta que la selección de las técnicas y el éxito de los resultados que se obtengan, depende en gran medida tanto del equipo de análisis y desarrollo, como de los propios clientes o usuarios que en ella participen.

Este se puede dividir en tres grandes actividades:

1. **Captura de requisitos**

   La captura de requisitos es la actividad mediante la que el equipo de desarrollo de un sistema de software extrae, de cualquier fuente de información disponible, las necesidades que debe cubrir dicho sistema. El proceso de captura de requisitos puede resultar complejo, principalmente si el entorno de trabajo es desconocido para el equipo de analistas, y depende mucho de las personas que participen en él. Por la complejidad que todo esto puede implicar, la ingeniería de requisitos ha trabajado desde hace años en desarrollar técnicas que permitan hacer este proceso de una forma más eficiente y precisa. (10)

2. **Especificación de Requerimientos**

   En esta fase se documentan los requerimientos acordados con el cliente, en un nivel aprobado de detalles. En la práctica esta actividad se va desarrollando en conjunto con el análisis debido a que en esta se plasman en documentos los requerimientos del cliente.

3. **Validación de Requerimientos**

   Los requisitos una vez definidos necesitan ser validados. La validación de requisitos tiene como misión demostrar que la definición de los requisitos define realmente el sistema que el usuario necesita o el cliente desea. Es necesario asegurar que el análisis realizado y los resultados obtenidos de la etapa de definición de requisitos son correctos. Pocas son las propuestas existentes que ofrecen técnicas para la realización de la validación y muchas de ellas consisten en revisar los modelos obtenidos en la definición de requisitos con el usuario para detectar errores o inconsistencias. (10)

1.4.2 Técnicas de captura de requerimientos

Entrevistas: resultan una técnica muy aceptada dentro de la ingeniería de requisitos y su uso está ampliamente extendido. Las entrevistas le permiten al analista tomar conocimiento del problema y comprender los objetivos de la solución buscada. A través de esta técnica el equipo de trabajo se acerca al problema de una forma natural. Existen muchos tipos de entrevistas y son muchos los autores que han trabajado en definir su estructura y dar guías para su correcta realización.

Básicamente, la estructura de la entrevista abarca tres pasos: identificación de los entrevistados, preparación de la entrevista, realización de la entrevista y documentación de los resultados (protocolo de la entrevista).

A pesar de que las entrevistas son esenciales en el proceso de la captura de requisitos y con su aplicación el equipo de desarrollo puede obtener una amplia visión del trabajo y las necesidades del usuario, es necesario destacar que no es una técnica sencilla de aplicar. Requiere que el entrevistador sea experimentado y tenga capacidad para elegir bien a los entrevistados y obtener de ellos toda la información posible en un período de tiempo siempre limitado. (10)

Tormenta de ideas: es una técnica de reuniones en grupo cuyo objetivo es que los participantes muestren sus ideas de forma libre. Consiste en la mera acumulación de ideas y/o información sin evaluar las mismas. El grupo de personas que participa en estas reuniones no debe ser muy numeroso (máximo 10 personas), una de ellas debe asumir el rol de moderador de la sesión, pero sin carácter de controlador. Como técnica de captura de requisitos es sencilla de usar y de aplicar, puesto que no requiere tanto trabajo en grupo.

Además suele ofrecer una visión general de las necesidades del sistema, pero normalmente no sirve para obtener detalles concretos del mismo, por lo que suele aplicarse en los primeros encuentros. (10)

Mapa Conceptual: son grafos en los que los vértices representan conceptos y las aristas representan posibles relaciones entre dichos conceptos. Estos grafos de relaciones se desarrollan con el usuario y sirven para aclarar los conceptos relacionados con el sistema a desarrollar. Son muy usados dentro de la ingeniería de requisitos, pues son fáciles de entender por el usuario, más aún si el equipo de desarrollo hace el esfuerzo de elaborarlo en el lenguaje de éste. (10)

Sketches y Storyboards: está técnica es frecuentemente usada por los diseñadores gráficos de aplicaciones en el entorno web. La misma consiste en representar sobre papel en forma muy esquemática las diferentes interfaces al usuario (sketches). Estos sketches pueden ser agrupados y unidos por enlaces dando idea de la estructura de navegación (storyboard). (10)

Casos de Uso: aunque inicialmente se desarrollaron como técnica para la definición de requisitos, algunos autores proponen casos de uso como técnica para la captura de requisitos. Los casos de uso permiten mostrar el contorno (actores) y el alcance (requisitos funcionales expresados como casos de uso) de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Los actores son elementos

externos (personas, otros sistemas, entre otros) que interactúan con el sistema como si de una caja negra se tratase. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores.

La ventaja esencial de los casos de uso es que resultan muy fáciles de entender para el usuario o cliente, sin embargo carecen de la precisión necesaria si no se acompañan con una información textual o detallada con otra técnica como pueden ser los diagramas de actividades. (10)

Cuestionarios y Checklists: esta técnica requiere que el analista conozca el ámbito del problema en el que está trabajando. Consiste en redactar un documento con preguntas cuyas respuestas sean cortas y concretas, o incluso cerradas por unas cuantas opciones en el propio cuestionario (Checklist). Este cuestionario será cumplimentado por el grupo de personas entrevistadas o simplemente para recoger información en forma independiente de una entrevista. (10)

Comparación de terminología: uno de los problemas que surge durante la elicitación de requisitos es que usuarios y expertos no llegan a entenderse debido a problemas de terminología. Esta técnica es utilizada en forma complementaria a otras técnicas para obtener consenso respecto de la terminología a ser usada en el proyecto de desarrollo. Para ello es necesario identificar el uso de términos diferentes para los mismos conceptos (correspondencia), misma terminología para diferentes conceptos (conflictos) o cuando no hay concordancia exacta ni en el vocabulario ni en los conceptos (contraste). (10)

Existen más técnicas para la captura de requerimientos (análisis de otros sistemas, estudio de la documentación, entre otras), incluso también es común encontrar alternativas que combinen varias de estas técnicas. Sin embargo, las presentadas ofrecen un conjunto representativo de las más utilizadas. (10)

1.4.3 Técnicas de validación de requerimientos

Revisiones de requerimientos: los requerimientos son analizados sistemáticamente por un equipo de revisores. Una revisión de requerimientos es un proceso manual que involucra a personas tanto de la organización del cliente como de la contratista. Las revisiones de requerimientos pueden ser informales o formales. (7)

Auditorías: la revisión de la documentación con esta técnica consiste en un chequeo de los resultados contra una Checklist predefinida o definida a comienzos del proceso, es decir sólo una muestra es revisada. (10)

Matrices de trazabilidad: esta técnica consiste en marcar los objetivos del sistema y chequearlos contra los requisitos del mismo. Es necesario ir viendo qué objetivos cubre cada requisito, de esta forma se podrán detectar inconsistencias u objetivos no cubiertos. (10)

Prototipos: algunas propuestas se basan en obtener de la definición de requisitos prototipos que, sin tener la totalidad de la funcionalidad del sistema, permitan al usuario hacerse una idea de la estructura de la interfaz del sistema con el usuario. Esta técnica tiene el problema de que el usuario debe entender que lo que está viendo es un prototipo y no el sistema final. (10)

Generación de casos de prueba: los requerimientos deben poder probarse. Si las pruebas para éstos se conciben como parte del proceso de validación, a menudo revela los problemas en los requerimientos. Si una prueba es difícil o imposible de diseñar, normalmente significa que los requerimientos serán difíciles de implementar y deberían ser considerados nuevamente. Desarrollar pruebas para los requerimientos del usuario antes que se escriba código es una parte fundamental de la programación extrema. (7)

1.5 Diseño de Software

En el diseño se modela el sistema y se encuentra la forma para que soporte todos los requerimientos, incluyendo los requisitos no funcionales y otras restricciones, se asienta en el núcleo técnico del proceso de ingeniería del software y se aplica independientemente del paradigma de desarrollo utilizado. En la elaboración de la solución es preciso tener presente ciertos estándares, estilos y patrones que son muy útiles en la obtención de un diseño de software robusto.

1.5.1 Arquitectura de Software

En el desarrollo de aplicaciones informáticas la arquitectura de software es un término muy frecuente y de gran referencia ya que sirve para guiar la implementación y el desarrollo del sistema desde etapas tempranas del diseño. Existen muchas definiciones acerca del tema, y aunque todas son consideraciones especializadas de distintos autores, coinciden en que a grandes rasgos se refiere a la estructura del sistema, constituida por componentes de software y relaciones entre estos. (11)

Algunas definiciones formales han sido expuestas como por ejemplo.

"La arquitectura del software de un programa o sistema de computación es la estructura o estructuras del sistema que comprende los elementos del software, las propiedades externamente visibles de esos elementos, y las relaciones entre ellos". (11)

"La Arquitectura de Software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones". (12)

Esta definición es un poco amplia, por lo que una de las más aplicadas es la establecida por la IEEE STD 1471-2000 (*Software Engineering Standards Committee of the IEEE Computer Society, 2000*): *"La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución."*

1.5.2 Estilos Arquitectónicos

Los estilos arquitectónicos no son más que patrones pertenecientes a un tipo de vista concreto, que definen una serie de restricciones a los tipos de elementos y relaciones de la vista arquitectónica. Algunos estilos son universales, mientras que otros definen un tipo particular de software. En cualquier caso, la arquitectura de un sistema está compuesta por vistas pertenecientes a múltiples estilos.

Brevemente descritos, los estilos conjugan elementos, conectores, configuraciones y restricciones. Al estipular los conectores como elemento de juicio de primera clase, el concepto de estilo, incidentalmente, se sitúa en un orden de discurso y de método que el modelado orientado a objetos en general y UML (*del inglés, Unified Modeling Language*) en particular no cubren satisfactoriamente. La descripción de un estilo se puede formular en lenguaje natural o en diagramas, pero lo mejor es hacerlo en un lenguaje de descripción arquitectónica o en lenguajes formales de especificación.

A diferencia de los patrones de diseño, que son centenares, los estilos se ordenan en seis o siete clases fundamentales y unos veinte ejemplares, como máximo. Es digno de señalarse el empeño por incluir todas las formas existentes de aplicaciones en un conjunto de dimensiones tan modestas. Las arquitecturas complejas o compuestas resultan del agregado o la composición de estilos más básicos.

Estilos de Flujo de Datos

- Tubería y filtros.
- Secuencial en lote.

Estilos Centrados en Datos

- Arquitecturas de Pizarra o Repositorio.
- Bases de Datos.
- Sistemas de Hipertextos.

Estilos de Llamada y Retorno

- Model-View-Controller (MVC).
- Arquitecturas en Capas.
- Arquitecturas Orientadas a Objetos.
- Arquitecturas Basadas en Componentes.

Estilos Derivados

- C2.
- GenVoca.
- REST.

Estilos de Código Móvil

- Arquitectura de Máquinas Virtuales.

Estilos heterogéneos

- Sistemas de control de procesos.
- Arquitecturas Basadas en Atributos.

Estilos Peer-to-Peer

- Arquitecturas Basadas en Eventos.
- Arquitecturas Orientadas a Servicios (SOA).

- Arquitecturas Basadas en Recursos.

1.5.3 Patrones

Cada patrón describe un problema que ocurre y una y otra vez en un entorno determinado y describe también el núcleo de la solución del problema, de forma que puede utilizarse un millón de veces sin tener que hacer dos veces lo mismo.

Durante el desarrollo de software suelen surgir problemas comunes, los cuales son resueltos con el uso de alternativas de soluciones que pueden ser muy efectivas, pues en la práctica y en el transcurrir de los años se ha demostrado que son eficientes frente a dificultades presentadas, estas variantes son como esquemas de situaciones que se presentan con frecuencia, y traen asociada la manera en que se solventan, estos elementos son llamados patrones.

Dentro de las definiciones de patrones se encuentra la propuesta por Craig Larman:

"Un patrón es un par problema/solución con nombre que se puede aplicar en nuevos contextos, con consejos sobre cómo aplicarlo en nuevas situaciones, o sea, un patrón es una descripción de un problema bien conocido que suele incluir: descripción, escenario de uso, solución concreta, consecuencias de utilizar el patrón, ejemplos de implementación y lista de patrones relacionados."

¿Qué son los patrones?

- Solucionan un problema: los patrones capturan soluciones, no sólo principios o estrategias abstractas.
- Son un concepto probado: capturan soluciones demostradas, no teorías o especulaciones.
- La solución no es obvia: los mejores patrones generan una solución a un problema de forma indirecta.
- Describen participantes y relaciones entre ellos: describen módulos, estructuras del sistema y mecanismos complejos.
- El patrón tiene un componente humano significativo: todo software proporciona a los seres humanos confort y calidad de vida (estética y utilidad).

1.5.3.1 Patrones de Arquitectura

Los patrones de arquitectura están relacionados fundamentalmente con la estructura de un sistema de software. Especifican un conjunto predefinido de subsistemas con sus responsabilidades y una serie de recomendaciones para organizar los distintos componentes.

Describen un problema particular y recurrente del diseño, que surge en un contexto específico, y presenta un esquema genérico y probado de su solución. (13)

Patrón Modelo Vista Controlador (MVC)

Es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres capas o componentes distintos. El patrón MVC se utiliza frecuentemente en aplicaciones web, donde la vista es la página HTML (*del inglés, Hypertext Markup Language*) que se visualiza en el navegador y el código que proporciona de datos dinámicos a la página, el modelo es el Sistema de Gestión de Base de Datos y la lógica de negocio y el controlador es el responsable de recibir los eventos de entrada desde la vista, enviarlos al modelo y manejar también las respuestas del modelo y transmitirlas hacia la vista. (14)

Modelo

El modelo es un conjunto de clases que representan la información del mundo real que el sistema debe procesar. El modelo desconoce la existencia de las vistas y del controlador. Ese enfoque suena interesante, pero en la práctica no es aplicable pues deben existir interfaces que permitan a los módulos comunicarse entre sí.

Esta es la recepción específica del dominio de la información sobre la cual funciona la aplicación. El modelo es una forma de llamar la capa de dominio. La lógica de dominio añade significados a los datos.

Vista

Las vistas son el conjunto de clases que se encargan de mostrar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo. Una vista obtiene del modelo solamente la información que necesita para desplegar y se actualiza cada vez que el modelo del dominio cambia por medio de notificaciones generadas por el modelo de la aplicación.

Controlador

El controlador es un objeto que se encarga de dirigir el flujo del control de la aplicación debido a mensajes externos, como datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encarga de modificar el modelo o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no conocen de la existencia del controlador. (14)

1.5.3.2 Patrones de Diseño

Los Patrones de Diseño no son más que soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan, son descripciones de clases cuyas instancias colaboran entre sí y brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares.

Patrones GRASP

GRASP es un acrónimo que significa General Responsibility Assignment Software Patterns (*en español, patrones generales de software para asignar responsabilidades*). El nombre se eligió para indicar la importancia de captar *(grasping)* estos principios, si se quiere diseñar eficazmente el software orientado a objetos.

Los patrones GRASP describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones.

Tienen como motivación:

- La asignación de responsabilidades como la habilidad más importante en el análisis y diseño orientado por objetos.
- Respetar los principios fundamentales como uno de los factores críticos, para obtener diseños reutilizables, mantenibles y extendibles.

GRASP: descripción de los principios fundamentales de la asignación de responsabilidades expresados como patrones. (15)

Patrones GoF

El grupo de GoF (*del inglés, Gang of Four*) clasificaron los patrones en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento.

- Creacionales: los patrones creacionales se relacionan con las formas de crear instancias de objetos. El objetivo de estos patrones es abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.

- Estructurales: los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.

- Comportamiento: los patrones de comportamiento ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

1.6 Metodología, Lenguaje y Herramientas de modelado utilizadas para el desarrollo

1.6.1 Metodología de Desarrollo de Software

Las metodologías de desarrollo de software describen los pasos que se deben seguir para la producción de un determinado producto informático, a través de patrones, que surgieron gracias a la experiencia en un determinado campo del desarrollo de software, como es el caso de los patrones de diseño, los cuales sugieren una serie de pasos a la hora de realizar un diseño determinado.

Rational Unified Process (RUP)

RUP especifica un framework para el desarrollo de un proyecto, en particular un proyecto de desarrollo de software, definiendo: etapas, actividades a realizar por un equipo de desarrollo, secuencia y lógica necesaria para obtener el producto final.

RUP es un proceso formal: provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando cronograma y presupuesto). Fue desarrollado por Rational Software, y está integrado con toda la suite Rational de herramientas. Puede ser adaptado y extendido para satisfacer las necesidades de la

organización que lo adopte. Es guiado por casos de uso y centrado en la arquitectura, y utiliza UML como lenguaje de notación.

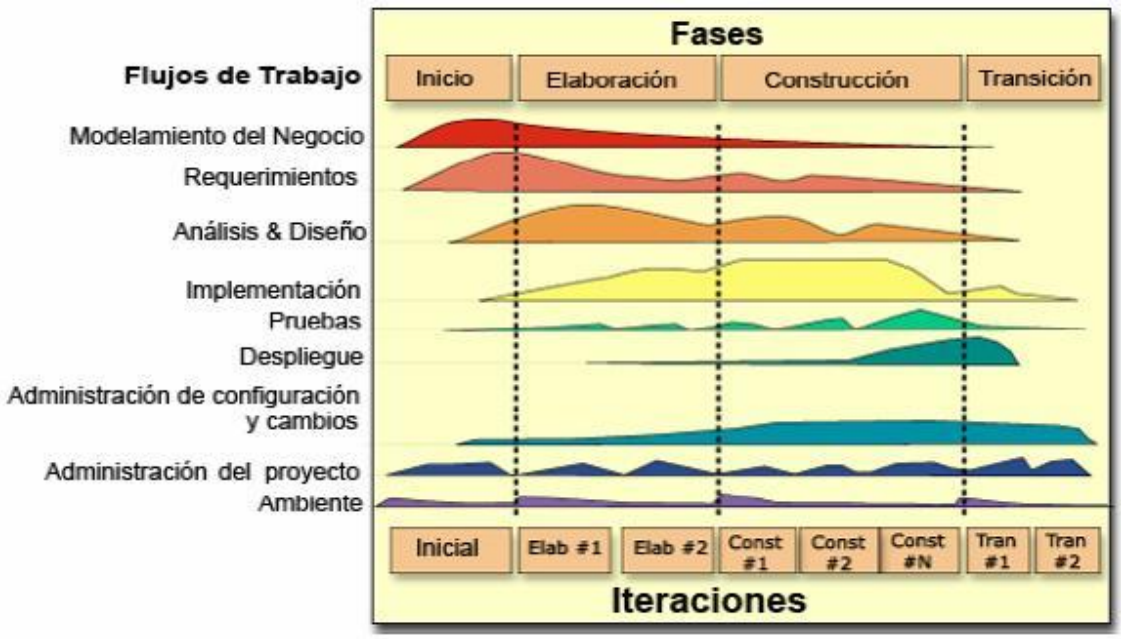


Fig. 1.1 Fases y flujos de trabajo de RUP

Está formado por cuatro fases y 9 flujos de trabajo. Cada una de estas etapas o fases es desarrollada mediante el ciclo de iteraciones, la cual consiste en reproducir el ciclo de vida en cascada a menor escala. Los objetivos de una iteración se establecen en función de la evaluación de las iteraciones precedentes. Además de ser guiado por casos de uso, centrado en la arquitectura y se caracteriza por ser iterativo e incremental.

Flujos de Trabajo

Los flujos de trabajo son nueve, de ellos:

Seis son de ingeniería

1. Modelamiento del Negocio: es donde se entiende las necesidades del negocio.
2. Requerimientos: trasladando las necesidades del negocio a un sistema automatizado.
3. Análisis y Diseño: trasladando los requerimientos dentro de la arquitectura de software.

4. Implementación: creando software que se ajuste a la arquitectura y que tenga el comportamiento deseado.

5. Pruebas: asegurándose que el comportamiento requerido es el correcto y que todo lo solicitado está presente.

6. Instalación: produce entregables del producto y realiza actividades (empaque, instalación, asistencia a usuarios, entre otros) para entregar el software a los usuarios finales.

Y tres son de soporte.

7. Administración del proyecto: involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

8. Administración de configuración y cambios: describe cómo controlar los elementos producidos por todos los integrantes del equipo de proyecto en cuanto a: utilización/actualización concurrente de elementos, control de versiones, entre otros.

9. Ambiente: contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

Ventajas

- Evaluación en cada fase que permite cambios de objetivos.
- Funciona bien en proyectos de innovación.
- Es sencillo, ya que sigue los pasos intuitivos necesarios a la hora de desarrollar el software.
- Seguimiento detallado en cada una de las fases.

Desventajas

- La evaluación de riesgos es compleja.
- Excesiva flexibilidad para algunos proyectos.
- Se corre el riesgo de poner al cliente en una situación que puede ser muy incómoda para él.

- El cliente deberá ser capaz de describir y entender a un gran nivel de detalle para poder acordar un alcance del proyecto con él.

1.6.2 Notación para el Modelado de Procesos de Negocio

Business Process Modeling Notation (BPMN)

En español, Notación para el Modelado de Procesos de Negocio, es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo (workflow). BPMN fue inicialmente desarrollada por la organización Business Process Management Initiative (BPMI), y es actualmente mantenida por el OMG (*del inglés, Object Management Group*), luego de la fusión de las dos organizaciones en el año 2005. Su versión actual es la 1.2 y hay una versión futura propuesta, la 2.0.

El principal objetivo de BPMN es proveer una notación estándar que sea fácilmente legible y entendible por parte de todos los involucrados e interesados del negocio (stakeholders). Entre estos interesados están los analistas de negocio (quienes definen y redefinen los procesos), los desarrolladores técnicos responsables de implementar los procesos) y los gerentes y administradores del negocio (quienes monitorizan y gestionan los procesos). En síntesis BPMN tiene la finalidad de servir como lenguaje común para cerrar la brecha de comunicación que frecuentemente se presenta entre el diseño de los procesos de negocio y su implementación. (16)

1.6.3 Lenguaje de Modelado

Lenguaje Unificado de Modelado (UML) 2.0

Es uno de los lenguajes de modelado de procesos más conocidos y utilizados en la actualidad. Está respaldado por el OMG. El UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. Además, ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio y funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables.

Es importante resaltar que UML es un "lenguaje de modelado" para especificar o para describir métodos o procesos. Se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir. En otras palabras, es el lenguaje en el que está descrito el modelo. (17)

1.6.4 Herramienta CASE para el modelado

Hoy en día muchas empresas se han extendido a la adquisición de herramientas CASE, con el fin de modelar los aspectos claves de todo el proceso de desarrollo de un sistema, desde el principio hasta el final. Una de estas herramientas es Visual Paradigm.

Visual Paradigm for UML 6.4

Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo de desarrollo de software: análisis y diseño orientado a objetivos, construcción, prueba y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagrama y generar documentación. (18)

1.6.5 Leguaje de Programación

PHP 5.3

PHP es un lenguaje de programación generalmente usado en la programación de sitios web dinámicos y actualmente es casi el lenguaje de desarrollo de sitios más usado en todo el mundo. Fue originalmente creado por Rasmus Lerdorf para presentar su portafolio de trabajo en el año 1994. Originalmente fue desarrollado en Perl. PHP al principio significaba Personal Home Page pero con el tiempo como ya es desarrollado por otro grupo se llama PHP Hypertext Preprocesor. (19)

Características del lenguaje PHP

- PHP es un lenguaje interpretado, solo se necesita un navegador web para ejecutarlo.
- Es un lenguaje del lado del servidor, por lo que los script se ejecutan remotamente y el resultado aparece en la máquina cliente (local).
- Tiene soporte para muchos tipos de bases de datos, entre las principales están MySQL, PostgreSQL, SQLite, Oracle, entre otras.
- La sintaxis es parecida a la del lenguaje C (Que también tiene un parecido a Perl).
- Es embebido en código HTML.

- Soporte de orientación a objetos.

1.6.6 Marco de Trabajo (Framework)

Symfony 1.2.8

Es un completo marco de trabajo diseñado para optimizar el desarrollo de las aplicaciones web mediante algunas de sus principales características. Para empezar, separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Además, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación.

Está desarrollado completamente en PHP 5. Ha sido probado en numerosos proyectos reales y se utiliza en sitios web de comercio electrónico de primer nivel. Symfony es compatible con la mayoría de gestores de bases de datos, como MySQL, PostgreSQL, Oracle y Microsoft SQL Server. Se puede ejecutar tanto en plataformas *nix (Unix, Linux, entre otros) como en plataformas Windows. (20)

Symfony se diseñó para que se ajustara a los siguientes requisitos: (20)

- Fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Independiente del sistema gestor de bases de datos.
- Sencillo de usar en la mayoría de casos, pero lo suficientemente flexible como para adaptarse a los casos más complejos.
- Basado en la premisa de "convenir en vez de configurar", en la que el desarrollador solo debe configurar aquello que no es convencional.
- Sigue la mayoría de mejores prácticas y patrones de diseño para la web.
- Preparado para aplicaciones empresariales, adaptable a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.

- Código fácil de leer que incluye comentarios de phpDocumentor y que permite un mantenimiento muy sencillo.
- Fácil de extender, lo que permite su integración con librerías desarrolladas por terceros.

1.6.7 Interfaz de usuario

ExtJS 3.0

Es un framework diseñado para la creación de páginas web dinámicas del lado del cliente basado en lenguaje JavaScript, permite una gran reutilización de componentes, personalización de los mismos, haciendo uso del manejo de tecnologías como AJAX (*del inglés, Asynchronous JavaScript And XML*) para el intercambio asincrónico de los datos entre el cliente y el servidor, además de lo ventajoso que puede representar la similitud de su aspecto con el de una aplicación de escritorio. Sus características principales son: gran desempeño, componentes de interfaz de usuario personalizables, con buen diseño y documentación. (21)

1.6.8 Entorno de Desarrollo Integrado (IDE)

Un IDE en un programa informático compuesto por un conjunto de herramientas, utilizadas por los programadores para desarrollar código, consisten en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Pueden dedicarse en exclusiva a un sólo lenguaje de programación o bien para varios.

NetBeans IDE 6.9

NetBeans IDE es una herramienta libre y gratuita sin restricciones de uso, pensada para escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extenderlo.

Algunas de las características que presenta integrado a PHP son:

- Creación de proyectos PHP.
- Integración con Symfony y ZenFramework.
- Editor de código fuente.
- Integración con PHPUnit Testing

- Depuración de PHP
- Integración con MySQL, PostgreSQL y Oracle.
- Integración con Sistemas de Control de Versiones.

1.6.9 Gestor de Base de Datos

Oracle 11g

Es un sistema de gestión de base de datos objeto-relacional (*ORDBMS por el acrónimo en inglés de Object-Relational Data Base Management System*), desarrollado por la empresa Oracle.

Se considera a Oracle como uno de los sistemas de bases de datos más completos, destacando:

- Soporte de transacciones.
- Estabilidad.
- Escalabilidad.
- Soporte multiplataforma.

Su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySQL o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

Las principales funcionalidades aportadas por todo el SGBD Oracle son:

- Soporte y tratamiento de una gran cantidad de datos (Gbytes).
- Soporte de una gran cantidad de usuarios accediendo concurrentemente a los datos.
- Seguridad de acceso a los datos, restringiendo dicho acceso a las necesidades de cada usuario.
- Integridad referencial en su estructura de base de datos.
- Conectividad entre las aplicaciones de los clientes en sus puestos de trabajo y el servidor de base de datos Oracle (estructura cliente/servidor).

- Conectividad entre bases de datos remotas (estructura de bases de datos distribuidas).
- Portabilidad.
- Compatibilidad.

1.7 Conclusiones

En el capítulo se presenta un estudio de los conceptos, normas y resoluciones principales sobre el régimen de depósito de aduanas, así como el funcionamiento, estructura y clasificación en Cuba. Se demuestra la necesidad e importancia que tiene para la Aduana contar con un sistema informático que facilite el control de los documentos en los Depósitos de Aduana, se realiza un análisis de los sistemas extranjeros que son utilizados para tales fines, los cuáles no son factibles porque en su totalidad son privativos, lo cual provoca gastos de grandes sumas de dinero por concepto de licencia, soporte y compra, además no cumplen con las legislaciones y normativas aduaneras cubanas vigentes (excepto SIDUNEA, que es adaptable) para el funcionamiento y control de los Depósitos de Aduana, y en el caso del sistema nacional SUA como se explica anteriormente no satisface las necesidades existentes en la Aduana correspondiente a los depósitos.

Con la premisa de obtener un sistema de alto nivel de calidad y funcionamiento se analizaron diferentes temáticas importante en el desarrollo de un software como son: la ingeniería de requerimientos, el diseño y arquitectura de software, patrones de arquitectura y diseño, metodologías, lenguajes y tecnologías a utilizar.

La solución que se propone desarrollar debe integrarse a los restantes módulos del sistema informático para la gestión de los Depósitos de Aduana (DEPAD), el cual constituye a su vez un subsistema del producto GINA (Gestión Integral de Aduanas) desarrollado por el Departamento de Desarrollo de Soluciones para la Aduana del Centro para la Informatización de la Gestión de Entidades (CEIGE).

Dicho departamento presenta un procedimiento para la Ingeniería de Requisitos, definido en el trabajo de diploma de la Ing. Jenni Manso Martínez, esta investigación determina que la metodología de desarrollo para la realización de las actividades del procedimiento es RUP pero con algunas modificaciones ya que es un proceso de desarrollo de software configurable que puede ser adaptado a las características propias del proyecto.

A continuación se detallan las modificaciones realizadas:

- En vez de utilizar el Modelo de Casos de Uso del negocio y los diagramas de actividades, se propone el uso del Modelo de Proceso de Negocio con la notación BPMN.
- Para los conceptos del negocio, en vez del Modelo de Objeto del Negocio, se propone confeccionar el Modelo Conceptual.
- Se debe tomar cada requisito funcional como un caso de uso.
- No se realizarán las actividades que propone RUP en el análisis, sino que se pasará directamente al diseño ya que los analistas no tienen los conocimientos necesarios de los frameworks que se utilizan en el proyecto, por lo que los artefactos que se generen no ayudarían a los diseñadores, constituyendo una pérdida de tiempo. (22)

Se determinó utilizar la herramienta Visual Paradigm para el modelado debido principalmente a la característica del proyecto de que se realice el desarrollo de la solución sobre el sistema operativo Linux en su distribución Nova o Ubuntu y además se encuentra en la lista de recursos necesarios para la ejecución del proceso que propone el Programa de Mejora IPP -3510:2009 Libro de Proceso para la Administración de Requisitos, y BPMN como la notación estándar ya que es soportada por numerosas herramientas, además de que tiene una interfaz gráfica amigable y sus diagramas son más fáciles de comprender por parte de los clientes, lo que facilita la comunicación.

La línea base arquitectónica del departamento se encuentra especificada en el trabajo de diploma del Ing. José Antonio Cobo Rodríguez, la misma define el uso de programación orientada a objetos, marco de trabajo el framework Symfony, el framework de persistencia a datos Propel como ORM (*del inglés, Object Relational Mapping*) debido a que se encuentra fuertemente vinculados a Symfony y el framework ExtJS para el desarrollo de las interfaces de usuario. Se describe con claridad que el lenguaje a utilizar en el desarrollo sea PHP y Oracle como gestor de base de datos debido a que son restricciones planteadas por el CADI.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En el capítulo se presenta una propuesta del sistema a desarrollar, se describen las características principales del negocio, se muestra el flujo principal de procesos, el modelado de los mismos y el modelo conceptual, se plasma un análisis crítico sobre cómo se ejecutan los procesos en la actualidad, se exponen los requerimientos funcionales definidos para el sistema, y se expresan además las técnicas de captura y validación de requerimientos utilizadas.

2.2 Modelado de los procesos del Negocio

A continuación se describe el proceso *Gestionar Depósitos de Aduana* y el subproceso *Gestionar Remisión de Entrada*, con la finalidad de lograr un mayor entendimiento del negocio que posibilite el desarrollo de un software con la mayor calidad posible.

2.2.1 Descripción del proceso Gestionar Depósitos de Aduana

El siguiente diagrama (ver Fig. 2.1) presenta el proceso Gestionar Depósitos de Aduana a partir del cual se derivan cada uno de los subprocesos que se efectúan en los Depósitos de Aduana los cuales son: Gestionar Remisión de Entrada, Gestionar Informe de Recepción, Gestionar Factura de Venta, Gestionar Remisión de Salida, Gestionar Ajuste, Gestionar Notificación de Avería, Gestionar Cancelación, Identificar Incidencia y Otorgar Prórroga.

Como pueden observar, se inicia con el subproceso *Identificar Incidencia* de infracciones de las normativas aduaneras que se comenten en los depósitos o el subproceso Otorgar Prórroga o la recepción de la información electrónica enviada por el subsistema Recepción Electrónica.

El depositario realiza el envío de los documentos sobre los movimientos de inventario al subsistema Recepción Electrónica, dichos documentos son: Remisión de Entrada, Informe de Recepción, Remisión de Salida, Factura de Venta, Ajuste, Notificación de Avería y Cancelación, los cuales son revisados de acuerdo al formato de intercambio electrónico establecido, se garantiza que se corresponda con la plantilla definida, estructura, tamaño y tipo de los campos que se intercambian; en caso de que existan errores en el documento se le notifican al depositario sino la información es enviada al subsistema Depósitos de Aduana, el cual la recepciona y según el tipo de documento enviado realiza las validaciones del negocio correspondientes, concluida las validaciones se envía una

respuesta al subsistema Recepción Electrónica, el cual la recibe y le notifica al depositario la respuesta emitida por el subsistema Depósitos de Aduana.

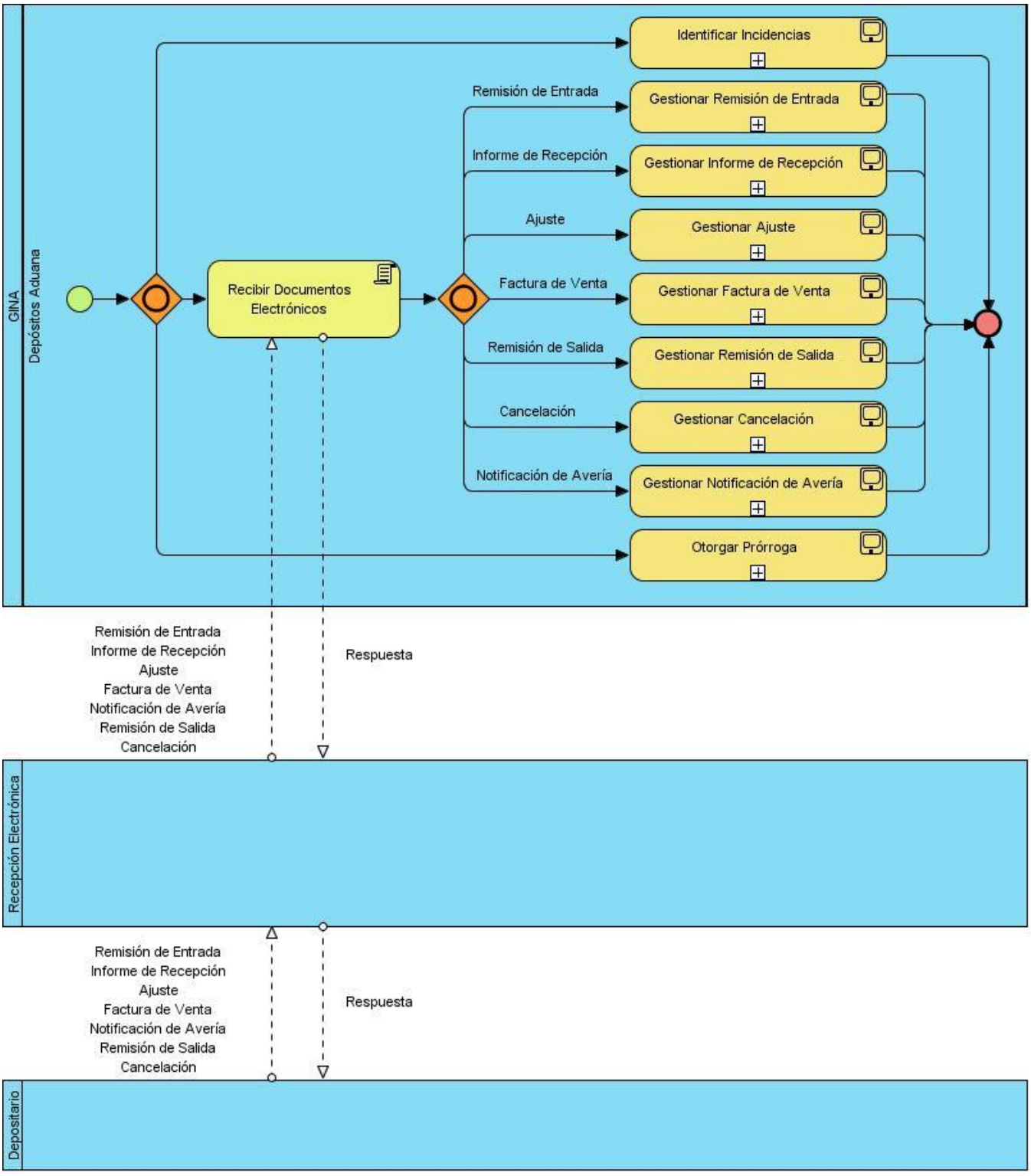


Fig. 2.1 Proceso Gestionar Depósitos de Aduana.

2.2.2 Descripción del subproceso Gestionar Remisión de Entrada

El subproceso se inicia (ver Fig. 2.2) una vez recibida la información del documento Remisión de Entrada con la verificación de los datos generales del mismo, debido a que el código de la aduana de entrada/salida tiene que corresponder con algún código nomenclado en las Tablas de Control y pertenecer a una aduana de despacho, que el código de la aduana de despacho y del depósito se corresponda con algún código nomenclado en las Tablas de Control, que la fecha del documento sea menor que la fecha actual y el número del documento no esté registrado, se comprueba que cada Declaración de Mercancías (DM) está registrada, asociada a todas las Facturas Comerciales y que ampara el BL/GA y los contenedores, si las DM presentan contenedores se confirma que estén asociados a una chapa y en caso de que se detecte algún error en las validaciones anteriores, se almacena el error correspondiente.

Si no ha sido almacenado ningún error, se registran todos los datos de la Remisión de Entrada y se envía una respuesta positiva al subsistema Recepción Electrónica de que los datos se han registrado correctamente, en caso contrario se envía una respuesta negativa del registro al subsistema Recepción Electrónica, notificándole los errores detectados.

2.3 Análisis Crítico de la Ejecución de los Procesos

En las operaciones que se realizan en los Depósitos de Aduana se mueven grandes cantidades de productos, de los cuales para conocer sus cifras reales en el inventario, realizar resúmenes estadísticos y reportes, los inspectores de la Aduana se auxilian de los sistemas informáticos que poseen los operadores de los depósitos.

Los inspectores aduaneros presentan como principal deficiencia la incapacidad de poder gestionar y controlar en tiempo la totalidad de las operaciones, debido a que se ejecutan de manera manual, provocando que en numerables ocasiones la información no exista en el momento necesario para la toma de decisiones oportuna y las acciones correctivas necesarias.

Si se automatizaran todas las operaciones se lograría una mayor eficiencia y un mejor acceso a la información para la toma oportuna de decisiones.

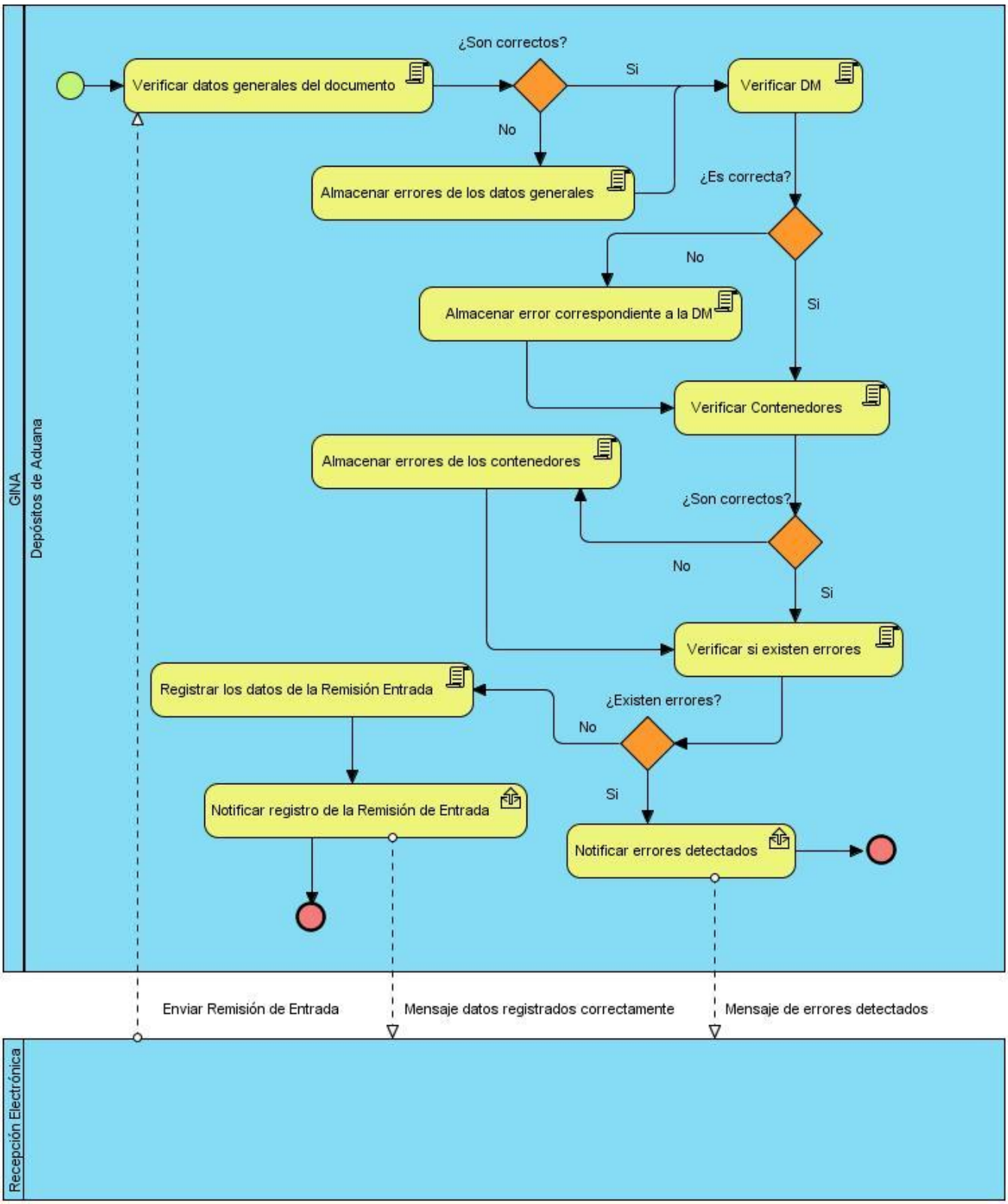


Fig. 2.2 Subproceso Gestionar Remisión de Entrada

2.4 Modelo Conceptual

A continuación se observa (ver Fig. 2.3) el modelo conceptual del negocio que presenta las entidades importantes y sus relaciones.

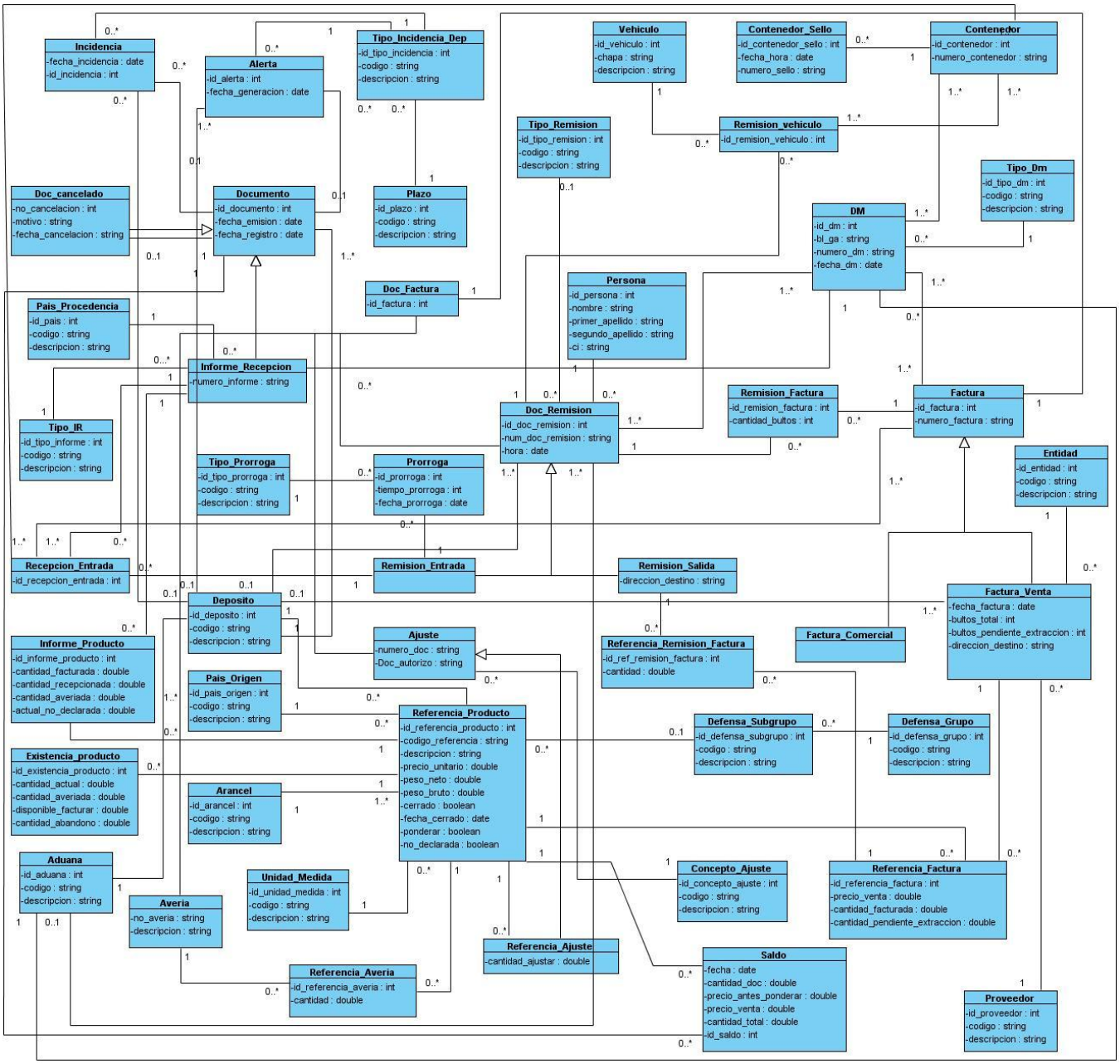


Fig. 2.3 Modelo Conceptual de Depósitos de Aduana

2.5 Especificación de los Requerimientos del Sistema

Especificar los requerimientos del sistema sirve como base para las actividades de ingeniería de software subsecuentes, describe la función y el desempeño de un sistema basado en computadora y las restricciones que regirán su desarrollo, por lo cual deben tenerse en cuenta las técnicas que son más factibles utilizar para la captura y la validación de estos requisitos; a continuación se exponen las técnicas empleadas además de un listado de los mismos.

2.5.1 Técnicas para la Captura de Requerimientos

Siguiendo el procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE, el cual plantea sobre las técnicas para la captura de requerimientos lo siguiente:

Teniendo en cuenta las características del cliente, la AGR, se decidió utilizar las siguientes técnicas que permitirán a los analistas, la recopilación y obtención de la información necesaria para la elicitación de requisitos, las mismas son: entrevista, observación, tormenta de ideas y talleres. (22)

Se combinan las siguientes técnicas: entrevista, observación y tormenta de ideas. Se **entrevistaron** inspectores de la aduana de los depósitos, funcionarios del Departamento de Técnicas Aduaneras de la AGR, a varios Jefes de los depósitos, en visitas realizadas a las instalaciones, ya que las entrevistas permiten un intercambio más abierto con los especialistas, mediante preguntas que esclarecen con precisión el funcionamiento de todo el trabajo en los Depósitos de Aduana. La **observación** se llevó a cabo para percibir como se desarrollan las operaciones en los distintos depósitos, pudiendo captar directamente las particularidades de estos procesos en las visitas realizadas; y la **tormenta de ideas** donde en conjunto con la especialista de la Aduana que atiende los Depósitos de Aduana, inspectores, los jefes del depósito y el personal de técnica aduanera, se acumularon ideas para tener una perspectiva general de las necesidades del sistema.

2.5.2 Requerimientos Funcionales

Los requerimientos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requerimientos funcionales de los sistemas también pueden declarar explícitamente lo que el sistema no debe hacer. (7)

Son capacidades o condiciones que el sistema debe cumplir, describen con detalle la función de éste, sus entradas y salidas, excepciones, entre otros, estos deben estar bien determinados y ser convenientemente comprendidos tanto por los implicados para con el sistema como por los desarrolladores del mismo para que exista un entendimiento común.

A continuación se exponen los requerimientos definidos para el sistema modelado:

RF1 Procesar información de la Remisión de Entrada: el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.

RF2 Procesar información del Informe de Recepción: el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.

RF3 Procesar información de la Remisión de Salida: el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.

RF4 Procesar información de la Factura de Venta: el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.

RF5 Procesar información de Ajuste: el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.

RF6 Procesar información de Cancelación: el sistema debe ser capaz de procesar la información proveniente de un fichero XML y actualizar la misma en la Base de Datos.

RF7 Mostrar datos de la Remisión de Entrada: el sistema debe mostrar todos los datos asociados a una Remisión de Entrada.

RF8 Mostrar datos de la Remisión de Salida: el sistema debe mostrar todos los datos asociados a una Remisión de Salida.

RF9 Buscar Remisiones: el sistema debe buscar todas las Remisiones.

RF10 Mostrar datos del Informe de Recepción Detallado: el sistema debe mostrar todos los datos asociados a un Informe de Recepción.

RF11 Mostrar datos del Informe de Recepción Realizado: el sistema debe mostrar los datos generales asociados a un Informe de Recepción.

RF12 Mostrar Informes de Recepción parciales: el sistema debe agrupar los Informes de Recepción, de cada depósito, según la parcialidad de los mismos.

RF13 Mostrar Informes de Recepción por descripción de producto: el sistema debe mostrar los Informes de Recepción que presenta un determinado producto.

RF14 Mostrar Informes de Recepción por criterio (Avería, Faltante, Sobrante o No Declarada): El sistema debe mostrar de cada depósito, todos los Informes de Recepción que presenten mercancías con avería, faltante, sobrante o no declarada.

RF15 Mostrar Informes de Recepción por país de procedencia: el sistema debe mostrar los Informes de Recepción que se han realizado según el país de procedencia.

RF16 Mostrar datos de la Factura de Venta: el sistema debe mostrar todos los datos asociados a una Factura de Venta.

RF17 Mostrar Facturas de Venta por descripción de producto: el sistema debe mostrar las Facturas de Venta que presenta un determinado producto.

RF18 Mostrar estado de la Factura de Venta: el sistema debe mostrar el estado en que se encuentran las Facturas de Venta de cada depósito.

RF19 Mostrar datos del Ajuste: el sistema debe ser capaz de mostrar los datos asociados a los Ajustes que se realizan en cada depósito.

RF20 Mostrar Ajustes por descripción de producto: el sistema debe mostrar los Ajustes que se han realizado para un determinado producto.

RF21 Mostrar datos de la Notificación de Avería: el sistema debe ser capaz de mostrar los datos asociados a las Notificaciones de Avería que se realizan en cada depósito.

RF22 Mostrar Documentos Cancelados: el sistema debe mostrar los datos de los documentos que han sido cancelados en cada depósito.

2.5.3 Técnicas para la Validación de Requerimientos

Siguiendo el procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE, el cual plantea sobre las técnicas para la validación de los requerimientos lo siguiente:

En el procedimiento propuesto se recomienda el uso de varias técnicas para la correcta validación de los requisitos en el Departamento de Soluciones para la Aduana las cuales son: revisiones del documento de requerimientos, construcción de prototipos, matrices de trazabilidad y generación de casos de pruebas. (22)

La validación de los requerimientos se realizó con la combinación las siguientes técnicas: revisiones del documento de requerimientos y construcción de prototipos. Se realizaron diversas **revisiones al documento de requerimientos** con la participación de los analistas del proyecto, jefe del subsistema, jefe del proyecto y especialistas del CADI para logar una correcta interpretación de la información transmitida, los señalamientos planteados fueron recogidos y aplicados posteriormente. Además se efectuó la **construcción de prototipos**, los cuales fueron presentados por cada requerimiento de software, aclarando con la Especialista Principal de Sistemas Automatizados del CADI, si las necesidades del área de trabajo Depósitos de Aduana fueron cubiertas por el sistema.

2.6 Aportes de la Solución y Beneficios Esperados

La solución propuesta posee las funcionalidades necesarias para gestionar los procesos asociados al control de la documentación de los Depósitos de Aduana de forma segura, eficaz y eficiente, contribuyendo notablemente en la eficiencia de la labor de los inspectores de aduana que en ellos intervienen.

El sistema permitirá mediante el uso de ficheros XML, recepcionar, manipular y presentar un control efectivo de toda la información proveniente de los Depósitos de Aduana con la finalidad de vincularla al proceso de despacho aduanero.

Proveerá reportes que garanticen la toma de decisiones oportuna y rapidez en el procesamiento de informes, que sirvan de contrapartida a la información ofrecida por el operador del depósito y eviten al máximo la infracción de leyes.

2.7 Conclusiones

En este capítulo se realizó la modelación y descripción de los procesos y subprocesos del negocio con el objetivo de lograr una comprensión y entendimiento de los mismos, para el posterior desarrollo de un sistema que los implemente con la calidad requerida. Se efectuó un análisis crítico de la ejecución de dichos procesos, se muestra el modelo conceptual donde se presentan las entidades importantes que se encuentran en el ámbito del problema y sus relaciones, se presentaron las técnicas de captura y validación de requerimientos definidas en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE y las utilizadas. Se exponen de manera clara los requisitos funcionales que debe cumplir el software para que el producto cumpla con las expectativas del cliente y se especifican los aportes de la solución y los beneficios esperados. De esta manera el presente capítulo facilita la posterior construcción del sistema.

CAPÍTULO 3: DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

3.1 Introducción

En el capítulo se desarrollan los flujos de trabajo de diseño e implementación, teniendo en cuenta como objetivos fundamentales la creación del modelo de diseño incluyendo los diagrama de clases del diseño con estereotipos web, diagrama de paquetes y los diagrama de secuencia orientado a actividades del negocio de cada uno de los requisitos funcionales de manera que se garantice que el diseño esté lo más orientado posible a la programación y asegurar así una mayor rapidez y calidad en este proceso. Se muestran los patrones utilizados en la solución, el diagrama de clases persistentes, el diseño de la base de datos, los resultados de la evaluación de la métrica de tamaño operacional de clase, el estándar de codificación empleado, el tratamiento de errores, el diagrama de despliegue y diagrama de componentes.

3.2 Diseño del sistema

La importancia del diseño del software puede describirse con una sola palabra: calidad. El diseño es la etapa en la que se fomentará la calidad en la ingeniería del software, proporciona las representaciones del software susceptibles de evaluar respecto de la calidad y es la única forma en que, de manera exacta, un requisito del cliente se puede convertir en un sistema o producto de software terminado.

El diseño del software sirve como fundamento para todas las actividades subsecuentes de la ingeniería de software y del soporte de éste. Sin diseño se corre el riesgo de construir un sistema inestable, el cual fallará cuando se realicen cambios pequeños; que será difícil de probar; cuya calidad no podrá evaluarse sino hasta etapas tardías del proceso del software, cuando queda poco tiempo y ya se ha gastado mucho dinero en él.

3.2.1 Patrones utilizados

Con la utilización del marco de trabajo Symfony, el sistema a desarrollar implementa varios de los patrones de diseño y arquitectónicos utilizados en la actualidad, ofreciendo a los desarrolladores la utilización de buenas prácticas en la implementación y la no preocupación de la aplicación de los mismos, brindándole al equipo de desarrollo y al producto cuantiosas ventajas.

3.2.1.1 Patrones GRASP

- **Controlador:** Symfony implementa un controlador frontal para atender todas las peticiones web (*sfActions*), es el único punto de entrada de toda la aplicación en un determinado entorno. Cuando recibe una petición, utiliza el sistema de enrutamiento para enviar la acción al controlador responsable de la solución y se encarga de manejar la seguridad y ejecución de los filtros. Este patrón se evidencia en las clases *sfFrontController*, *sfWebFrontController*, *sfContex*, los "*actions*" y el *index.php* del ambiente.

- **Alta Cohesión:** Symfony permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase *aduanaActions*, es responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades. Está formada por varias funcionalidades que están estrechamente relacionadas.

- **Bajo Acoplamiento:** la clase *aduanaActions* hereda únicamente de *sfActions* para alcanzar un bajo acoplamiento de clases. Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo, las cuales no tienen asociaciones con las de la vista o el controlador, lo que proporciona que la dependencia en este caso sea baja.

- **Experto:** Symfony utiliza Propel para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos (Peer del Modelo) poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan.

- **Creador:** en la clase *aduanaActions* se localizan las acciones definidas para el módulo, en las cuales se crean los objetos de las clases que representan las entidades, confirmando que la clase es "creador" de dichas entidades.

3.2.1.2 Patrones GoF

- **Singleton:** la clase *sfRouting* presenta el método *getInstance()*, es utilizada por el controlador frontal (*sfWebFrontController*) y se encarga de enrutar todas las peticiones que se hagan a la aplicación. *sfRouting* garantiza la existencia de una única instancia y la creación de un mecanismo de acceso global a dicha instancia.

- **Command:** se encuentra presente en la clase *sfWebFrontController*, en el método *dispatch()*,es la encargada de establecer el módulo y la acción que se va a usar según la petición del usuario. Además se aplica en la clase *sfRouting*, que está desactivada por defecto y procede según las necesidades del administrador del sistema donde se aplique el framework, la cual se puede activar o desactivar.

- **Decorator**: la clase abstracta *sfView*, padre de todas las vistas, contienen un decorador para permitir agregar funcionalidades dinámicamente. El archivo *layout.php*, conocido como plantilla global, contiene el código HTML que es tradicional en todas las páginas del sistema, para no tener que repetirlo. El contenido de la plantilla se integra en el layout, o si se mira desde el otro punto de vista, el layout decora la plantilla.

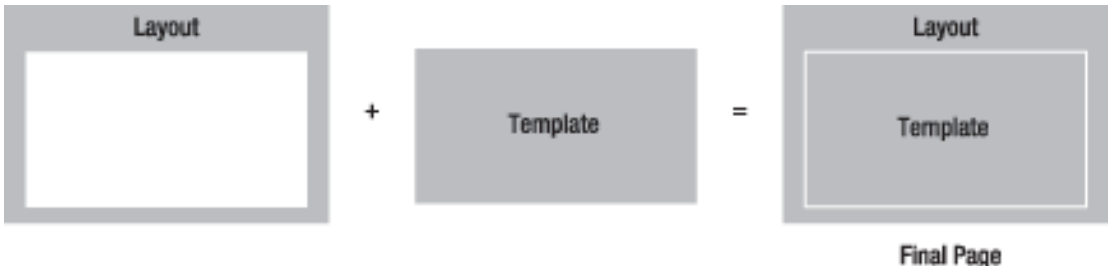


Fig. 3.1 Implementación del patrón Decorator por Symfony (20)

3.2.1.3 Patrón MVC según Symfony

El patrón MVC obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el framework, todo lo relacionado con la interfaz de usuario se guarda en la vista, la manipulación de datos se guarda en el modelo y el procesamiento de las peticiones constituye el controlador. El empleo del patrón MVC en un sistema resulta bastante útil además de restrictivo.

Symfony está basado en el patrón de arquitectura conocido MVC, formado por tres niveles: el *Modelo* representa la información con la que trabaja la aplicación, es decir, su lógica de negocio, la *Vista* es la encargada de originar las páginas que son mostradas como resultado de las acciones y el *Controlador* se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista. (20)

La implementación que realiza Symfony de la arquitectura MVC incluye varias clases como son:

- **sfController**: es la clase del controlador y se encarga de decodificar la petición y transferirla a la acción correspondiente.
- **sfRequest**: guarda todos los elementos que integran la petición (parámetros, cookies, cabeceras, entre otros).
- **sfResponse**: posee las cabeceras de la respuesta y los contenidos. El contenido de este objeto se convierte en la respuesta HTML que se remite al usuario.
- El singleton de contexto (que se obtiene mediante ***sfContext****::getInstance()*): guarda una referencia a todos los objetos que constituyen el núcleo de Symfony y puede ser accedido desde cualquier parte de la aplicación.

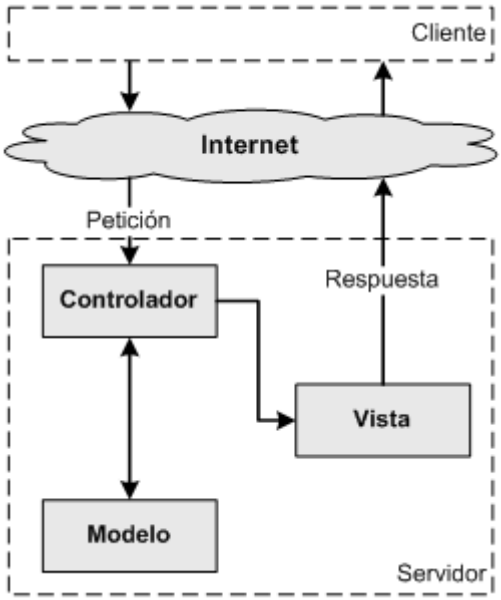


Fig. 3.2 El patrón MVC (20)

3.2.2 Diagrama de clases del Diseño con Estereotipos Web

Las extensiones UML para el diseño Web, exponen la solución para el modelamiento de diagramas de clases del diseño sobre tecnologías Web. De forma que quedan presentados los siguientes estereotipos.

Estereotipos	Imagen	Descripción
Server Page		Simboliza una página Web dinámica que posee el código construido por el servidor, normalmente porta scripts que se ejecutan en el servidor y que interactúan con los recursos de este lado: las bases de datos, los componentes de la lógica del negocio y sistemas externos.
Client Page		Representa una instancia de página cliente. Figura una página Web con código HTML, que interpreta y muestra el navegador del cliente.
Form		Simboliza un formulario, es el elemento que se encarga de enviar datos desde la página cliente hacia la página servidora.

Fig. 3.3 Estereotipos Web

La utilización del framework Symfony y las librerías ExtJS permiten desarrollar un diseño con una estructura similar para varios requerimientos. La página servidora *aduanaActrion.class* se encargará de atender las peticiones que realiza el controlador frontal; luego estas se redireccionan al layout. Como se explicó anteriormente el layout también se denomina plantilla global, almacena el código HTML que es común a todas las páginas de la aplicación, para no tener que repetirlo en cada página. El layout carga en el cuerpo la plantilla o pagina cliente (*indexSuccess.php*) que importa todas las interfaces de usuario del módulo, estas interfaces están definidas en el fichero de configuración *view.yml* y se encuentran representadas dentro del paquete Interfaces JS. Es muy frecuente la existencia de una relación de agregación entre la página cliente y los formularios, que contienen los componentes para la entrada de datos que serán enviados al servidor y procesados por las funciones de la clase aduanaActrion.class con la utilización de las clases del paquete lib y Nucleo_Symfony*, que garantizan el registro y obtención de la información (ver figura 3.4).

Nucleo_Symfony: *representa las clases que forman el núcleo de Symfony, dentro de las cuales podemos mencionar: sfController, sfRequest, sfResponse y el contexto que se obtiene mediante sfContext::getInstance().*

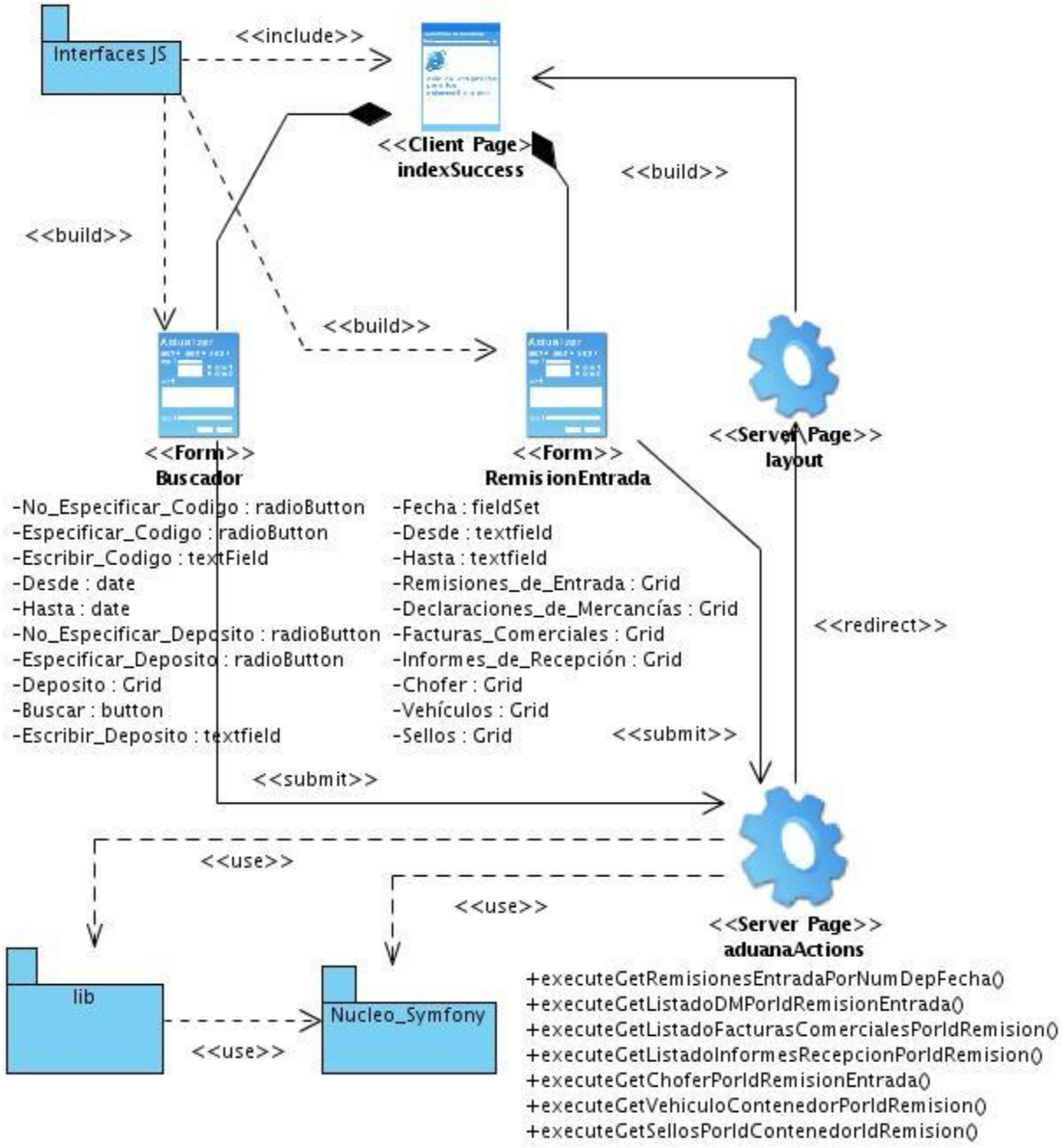


Fig. 3.4 Diagrama de clases Mostrar datos de la Remisión de Entrada

3.2.3 Paquetes lib e Interfaces JS

El diagrama de paquetes es utilizado para mostrar las agrupaciones lógicas en las que se encuentra dividido un sistema y las dependencias entre ellas. Suministran una descomposición de la jerarquía lógica de un sistema. Los paquetes contienen elementos del modelo al más alto nivel, tales como clases y sus relaciones, máquinas de estado, diagramas de casos de uso, interacciones, y

colaboraciones: cualquier elemento que no esté contenido en otro. Los elementos como atributos, operaciones, estados, líneas de vida, y mensajes están contenidos en otros elementos, y no aparecen como contenido directo de los paquetes. (23)

El paquete "lib" está compuesto por la clase sfAuxiliarTC y un paquete de Acceso a Datos (ver figura 3.5).

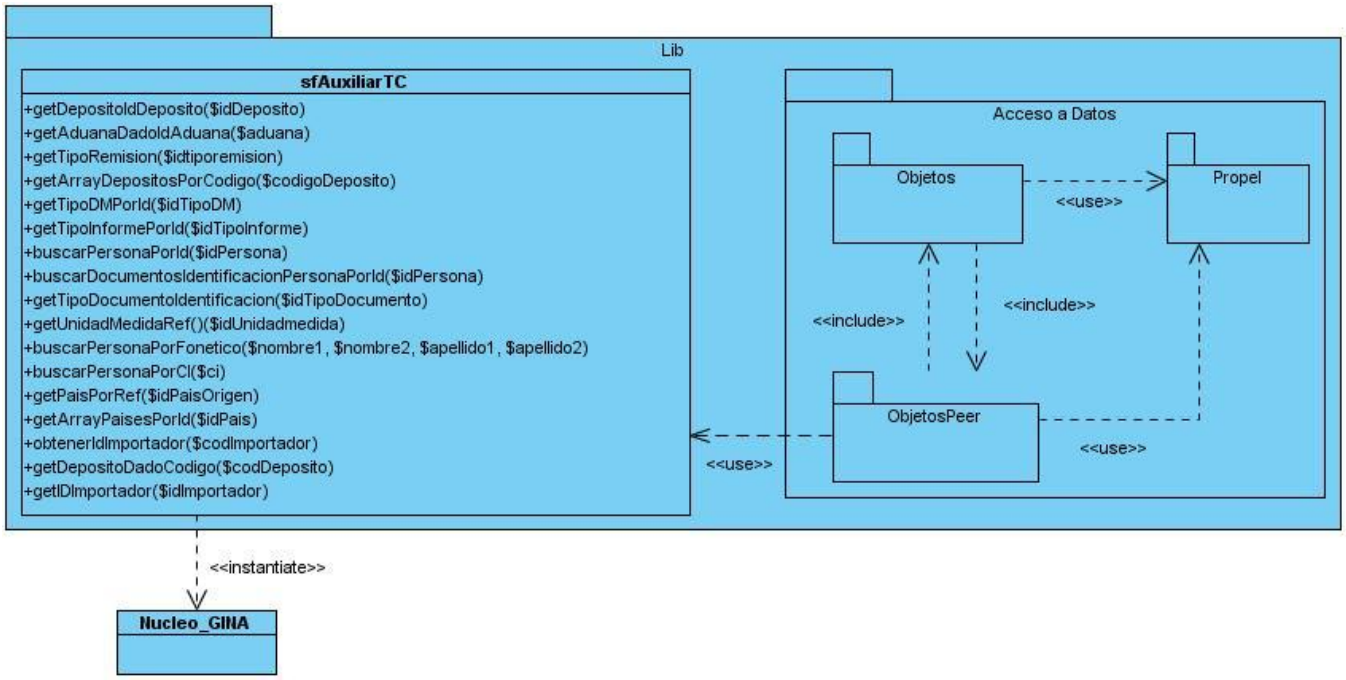


Fig. 3.5 Paquete lib

***Nucleo_GINA**: representa el fragmento del GINA encargado del intercambio de información entre subsistemas, los cuales son expertos en el manejo de información específica. El subsistema de Depósitos de Aduana tiene fuerte intercambio de dato con los siguientes subsistemas: recepción electrónica que se encarga de manejar los ficheros electrónicos, tablas de control que gestiona los nomencladores y persona que maneja los datos propios de las personas relacionadas con la Aduana.

La clase *sfAuxiliarTC* tiene como función principal devolver los datos necesarios de las tablas de otros subsistemas por lo que presenta un grupo de funcionalidades que satisfacen esta necesidad y el paquete Acceso a Datos contiene tres paquetes en su interior Objetos, ObjetosPeer y Propel, dentro de estos paquetes están las clases necesarias para efectuar la conexión y el intercambio de información de la aplicación con su propia base de datos.

El paquete Propel representa la capa de abstracción de objetos/relacional utilizada por Symfony, implementa una de las mejores capas para la abstracción a bases de datos disponible en PHP5, y se encuentra completamente integrado al framework.

El paquete Objetos tiene las clases de la lógica de la aplicación que contienen los atributos y métodos necesarios, mediante el uso de Propel permiten la inserción y actualización de información persistente. Por cada entidad o tabla del modelo físico de datos, existen dos clases que pertenecen al paquete Objetos, por ejemplo: la clase *DaRemisionEntrada* que hereda de *BaseDaRemisionEntrada*. Puede ser necesario añadir métodos y propiedades personalizadas en los objetos, estos se efectúan sobre las clases hijas (*DaRemisionEntrada*), pues las clases padre (*BaseDaRemisionEntrada*) poseen las funciones para acceder a la base de datos mediante Propel, e incluye del paquete *ObjetosPeer* su clase correspondiente (ejemplo *DaRemisionEntradaPeer*).

Las clases del paquete *ObjetosPeer* proporcionan los medios necesarios para obtener los registros de las tablas. Sus métodos devuelven normalmente un objeto o una colección de objetos de la clase objeto relacionada. Por cada una de las tablas en la base de datos se presentan dos clases siguiendo el mismo ejemplo se tendría que: *DaRemisionEntradaPeer* hereda de *BaseDaRemisionEntradaPeer*, así solo se modificará la clase hija y se podrán utilizar todos los métodos y atributos de la clase padre para lograr las consultas. Las clases Base de ambos paquetes son eliminadas siempre que se realice algún cambio en la base de datos de la aplicación y sea generado el modelo.

El paquete “Interfaces JS” contiene los códigos JavaScript que son leídos por el navegador en la representación de la página cliente (ver figura 3.6). Los archivos *Ext-base.js* y *Ext-all.js* constituyen el núcleo de las librerías ExtJS, y el resto representan las interfaces del módulo Documentos del Subsistema Depósitos de Aduana.

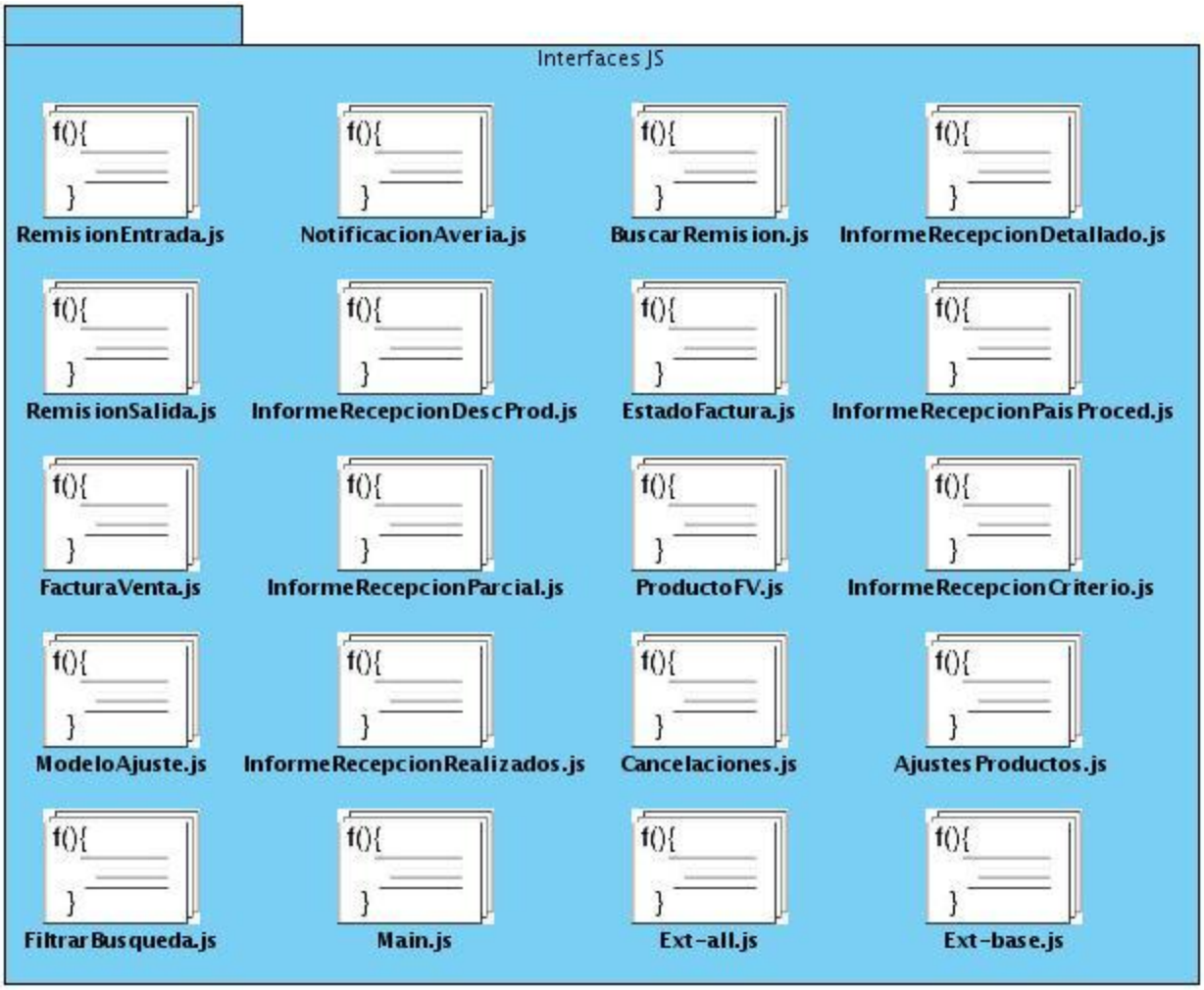


Fig. 3.6 Paquete Interfaces JS

3.2.4 Diagrama de interacción

Los diagramas de interacción muestran como los objetos se comunican unos con los otros para satisfacer los requerimientos, explica gráficamente las interacciones existentes entre las instancias de las clases. Son importantes para modelar los aspectos dinámicos de un sistema y para construir sistemas ejecutables a través de ingeniería hacia adelante e ingeniería inversa. Comúnmente contienen objetos, enlaces y mensajes. Hay dos tipos de diagrama de interacción, ambos basados en la misma información, pero cada uno enfatizando un aspecto particular: diagramas de secuencia y diagramas de colaboración.

3.2.4.1 Diagrama de Secuencia

Un diagrama de secuencia representa una interacción como un gráfico bidimensional. La dimensión vertical es el eje de tiempo, que avanza hacia abajo de la página. La dimensión horizontal muestra los roles de clasificador que representan objetos individuales en la colaboración. Cada rol de clasificador se representa mediante una columna vertical-línea de vida. Durante el tiempo que existe un objeto, el rol se muestra por una línea discontinua. Durante el tiempo que dura una activación de un procedimiento en el objeto, la línea de vida se dibuja como una línea doble. Se muestra un mensaje como una flecha desde la línea de vida de un objeto a la del otro. Las flechas se organizan en el diagrama en orden cronológico hacia abajo. (23)

3.2.4.2 Diagrama de Secuencia Orientado a Actividades del Negocio

El Departamento de Soluciones para la Aduana con el propósito de orientar el diseño lo mejor posible a la etapa de implementación utiliza los diagramas de secuencia orientados a actividades del negocio. El objetivo principal de este diagrama es especificar la interacción entre las clases, así como el flujo a seguir de las actividades a realizar, experimentando un nivel de especificación tal que permita a los programadores lograr la implementación de la solución en menor tiempo y con mejor calidad. Se representa por calles, actividades y sus relaciones, incluyendo también comentarios para mayor comprensión. Cada requisito funcional puede tener asociado uno o varios diagramas en dependencia del nivel de complejidad y la cantidad de funcionalidades comunes. Es considerado un híbrido entre los diagramas de actividades y los de secuencias propuestos por RUP en la etapa de diseño.

La figura 3.7 muestra un ejemplo de diagrama de este tipo, pertenece al RF *Mostrar datos de la Remisión de Entrada*, como se puede observar está compuesto por tres calles, cada una de ellas representa las áreas que abarca el negocio de esta funcionalidad, se inicia cuando se da clic en el botón buscar del filtro y son enviados los datos mediante el método POST al servidor, estos son obtenidos por el controlador *aduanaAction.class.php*, el cual es el encargado de realizar una llamada a la funcionalidad *getListadoFacturasPorIdRemision($idRem)* que se encuentra en la clase *DaFacturaPeer.php* y se encarga de realizar las validaciones pertinentes y devolver un listado de facturas asociadas a la Remisión.

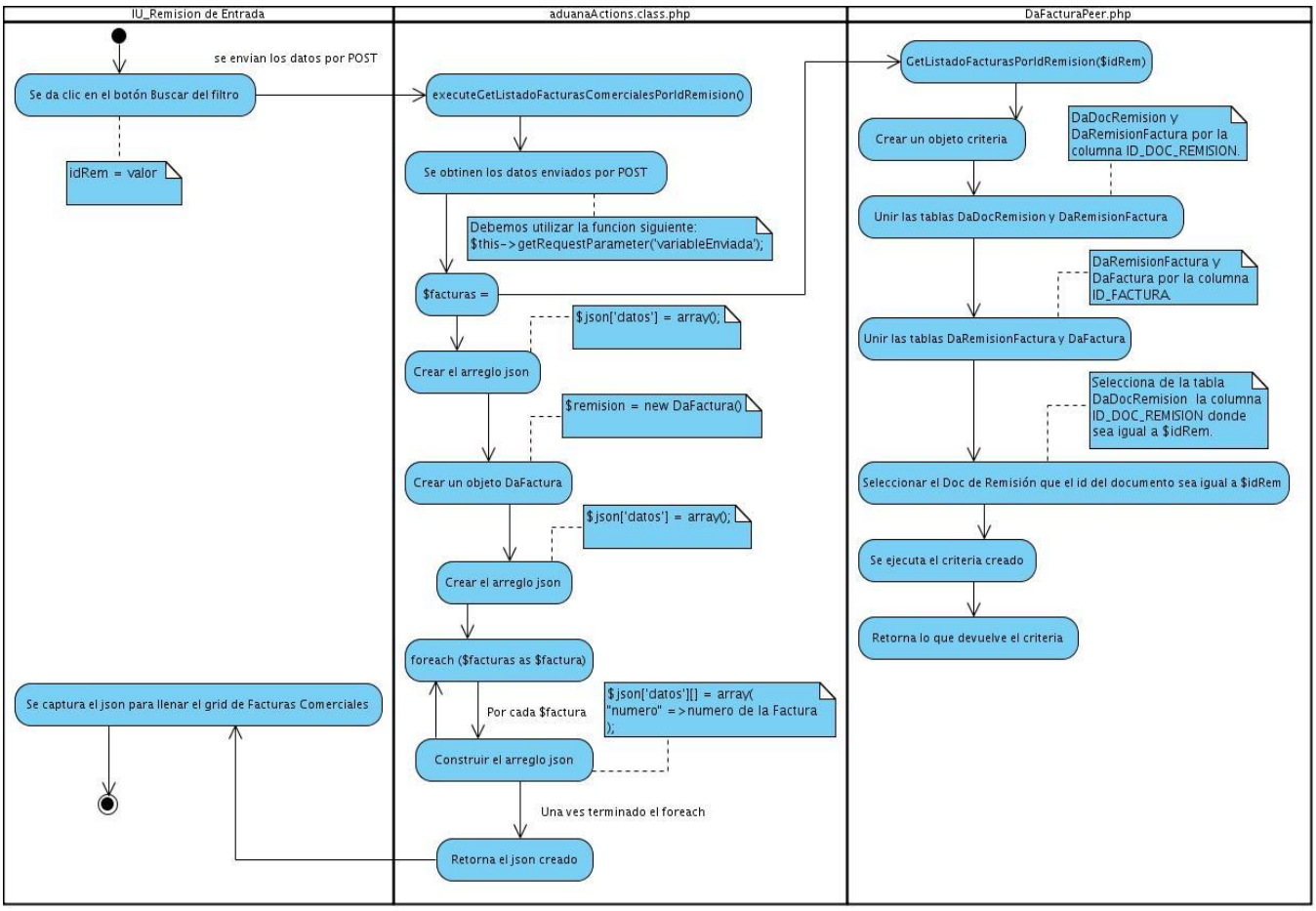


Fig. 3.7 Diagrama de Secuencia Orientado a Actividades del Negocio del RF Mostrar datos de la Remisión de Entrada.

3.2.5 Diagrama de clases persistentes

La figura 3.8 muestra el diagrama de diseño de las clases persistentes del sistema, sirve como una primera aproximación al diseño definitivo del modelo de datos.

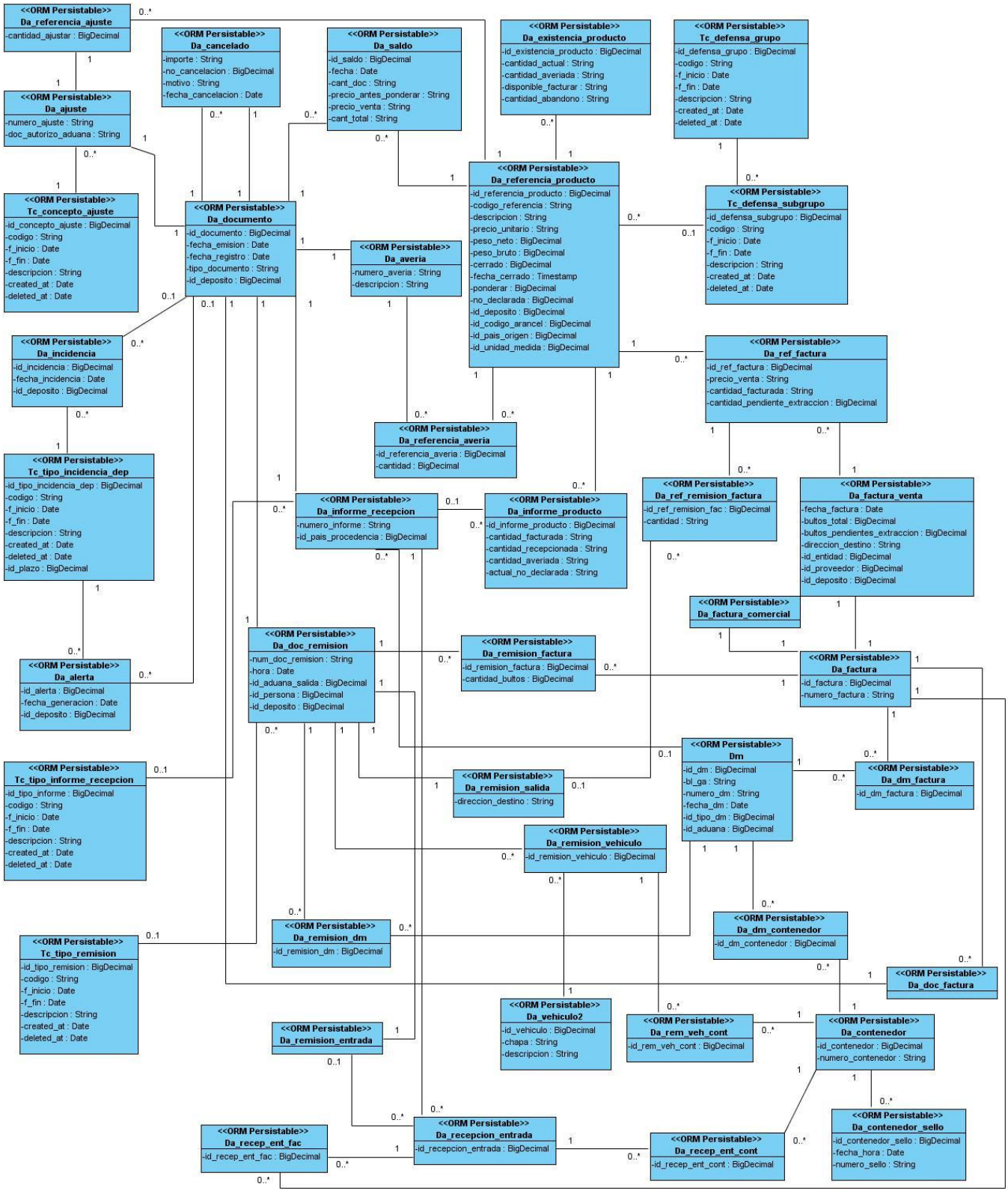


Fig. 3.8 Diagrama de clases persistentes

3.2.6 Diseño de la Base de Datos

A continuación se muestra el diagrama Entidad – Relación correspondiente al módulo.

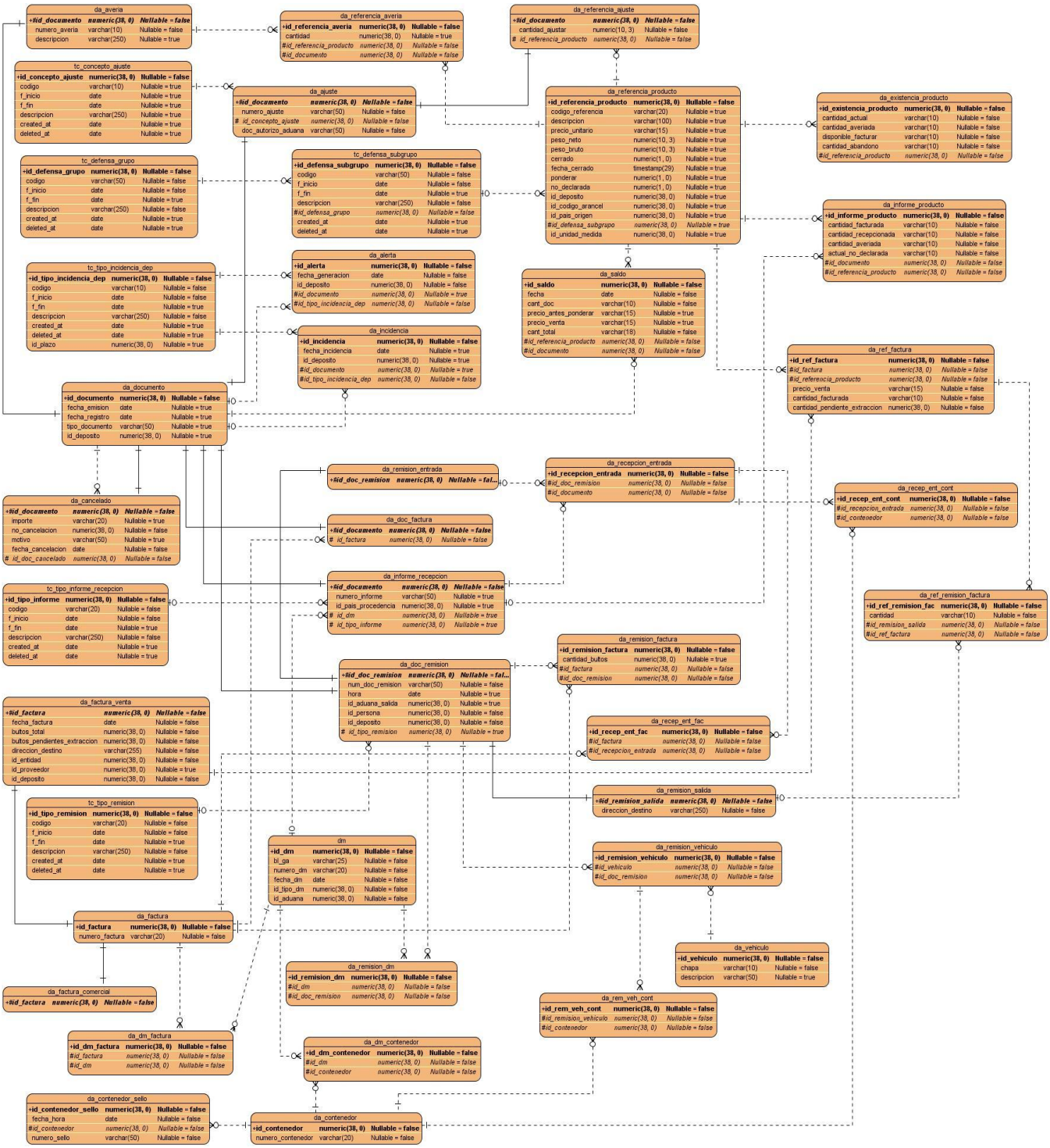


Fig. 3.8 Diagrama Entidad – Relación

3.2.7 Métricas para la evaluación del diseño

La métrica es una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado, las medidas y las métricas son componentes clave de cualquier disciplina de la ingeniería, los objetivos principales de las métricas son: comprender mejor la calidad del producto, estimar la efectividad del proceso y mejorar la calidad del trabajo realizado a nivel del proyecto.

3.2.7.1 Métrica de Tamaño Operacional de Clase (TOC)

La métrica TOC se basa esencialmente en la cantidad de funcionalidades contenidas en las clases, a partir de las cuales se determina la afectación que ejerce en el diseño.

A continuación se describen los tipos de afectaciones que se pueden observar al evaluar el diseño según la métrica.

Tamaño Operacional de Clase (TOC)	
Atributos	**Afectación**
Responsabilidad	El aumento del TOC provoca un aumento de la responsabilidad asignada a la clase.
Complejidad de Implementación	El aumento del TOC provoca un aumento de la complejidad de implementación de la clase.
Reutilización	Un aumento del TOC provoca una disminución en el grado de reutilización de la clase.

Tabla 3.1 Afectaciones en el diseño según la métrica TOC.

Para la evaluación de cada atributo se establece un rango de valores que determinan la complejidad en la aplicación del TOC. A continuación se muestra el rango de valores y la complejidad por cada uno de los atributos.

	Categoría	**Criterio**
Responsabilidad	Baja	< =Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio

Complejidad implementación	Baja	< = Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	> 2* Promedio

Reutilización	Baja	> 2* Promedio
	Media	Entre Promedio y 2* Promedio
	Alta	<= Promedio

Tabla 3.2 Rango de valores de para la evaluación técnica de los atributos de calidad relacionados con la métrica TOC.

A continuación se muestran los resultados obtenidos una vez aplicada la métrica:

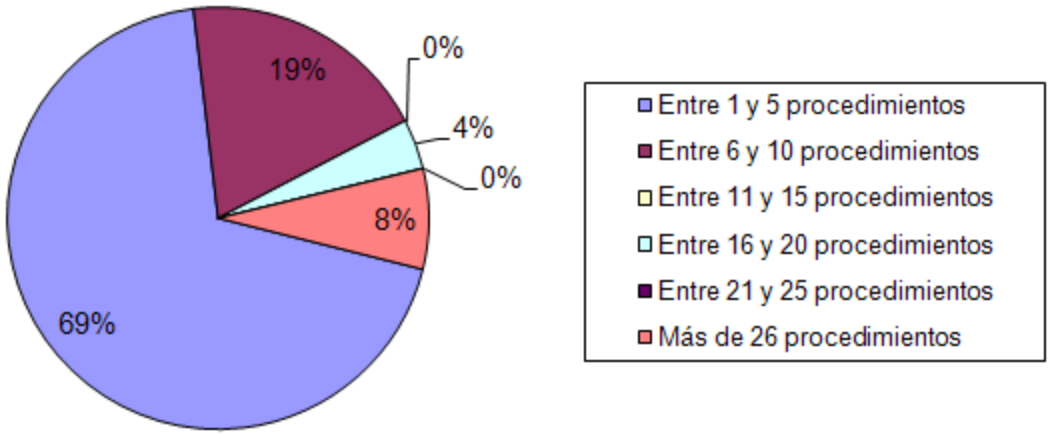


Fig. 3.9 Representación en % de los resultados obtenidos en el instrumento agrupados en los intervalos definidos según la métrica TOC.

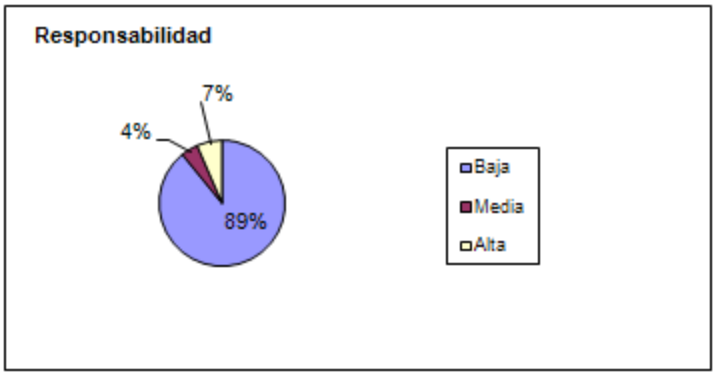


Fig. 3.10 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Responsabilidad.

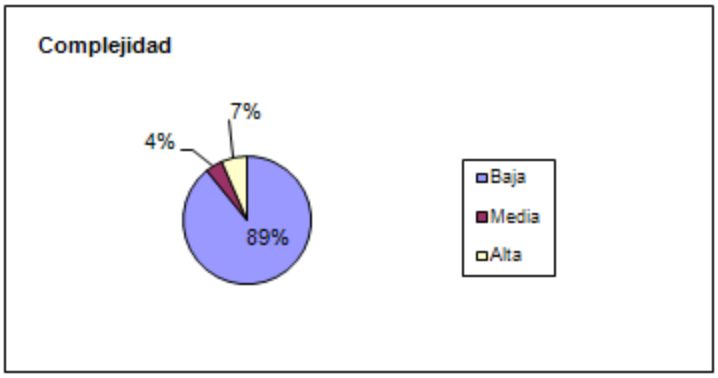


Fig. 3.11 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Complejidad de Implementación.

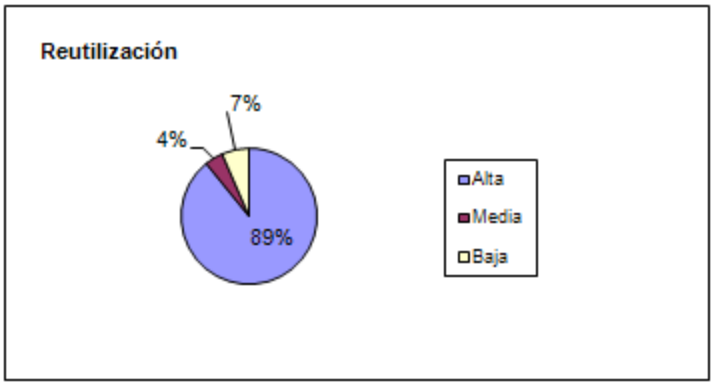


Fig. 3.12 Representación de la incidencia de los resultados de la evaluación de la métrica TOC en el atributo Reutilización.

Durante la evaluación de la métrica TOC los resultados obtenidos demuestran que los atributos de calidad de las clases se encuentran en un nivel satisfactorio (89%); de manera que se puede confirmar

la elevada reutilización (elemento clave en el proceso de desarrollo de software) y cómo se reducen la responsabilidad y la complejidad de implementación.

3.3 Implementación del sistema

Concluido el diseño se inicia el flujo de trabajo implementación, presentando como objetivo fundamental el desarrollo de la arquitectura y el sistema como un todo. El propósito de la implementación es definir la organización del código, en términos de los subsistemas de implementación organizados en capas, implementar los elementos de diseño en términos de los elementos de implementación (archivos de origen, binarios, programas ejecutables y otros), probar y desarrollar componentes como unidades y por último integrar los resultados producidos por los desarrolladores individuales (o equipos) en un sistema ejecutable.

3.3.1 Estándar de Codificación

En la actualidad existen diferentes estándares para cada uno de los lenguajes, su utilización permite una comunicación fluida y directa entre los programadores de manera que se favorece la reutilización y mantenimiento de los sistemas.

El Departamento de Soluciones para la Aduana de CEIGE presenta su propia propuesta de estándar de codificación para todas las aplicaciones a desarrollar. Comprende todo el código generado bajo la tecnología y el lenguaje PHP y que utilicen la arquitectura regida por la utilización del Framework Arquitectónico Symfony. (24)

A continuación citamos algunos de los aspectos relevantes que presenta dicho documento:

Acciones (métodos o funcionalidades de la clase *nombreDelModuloAction.class.php*):

- Dentro de las especificaciones del Framework está que cada una de estas acciones debe comenzar con la palabra ***execute***.
- Todos los nombres de acciones deben estar en la nomenclatura "***CamelCase*"** comenzando por la palabra ***execute***.
- En caso de ser acciones referentes a un módulo de CRUD de una tabla deben ser nombres específicos como ***executeNuevo***, ***executeEditar***, entre otros.

- Los nombres de las acciones deben especificar con la menor cantidad de palabras cual es el objetivo de la acción, de ser posible estar en infinitivo. Debe especificar bien claro cuál es la acción que se pretende ejecutar con la acción pero sin especificar los parámetros que recibe.

Nombre de las clases:

- Los nombres de las clases deben estar expresados en notación *UpperCamelCase*.
- No se deben utilizar guiones bajos en su nombre "_".
- Deben expresar con claridad cuál es el alcance y la responsabilidad de la clase.
- Los nombres de las clases no deben estar atados a las clases de las que se deriva, cada clase debe tener un significado por ella misma, no en dependencia de la clase de la que deriva.
- En los nombres compuestos por más de tres palabras se debe revisarse el diseño, no sea que se le estén dando a la clase más responsabilidades de las que realmente tiene.

Nombres de las funciones:

- Todas las funciones definidas por los desarrolladores deben seguir la nomenclatura ***UpperCamelCase***, a no ser que para cierto ámbito se especifiquen características específicas.

Además deben cumplir con las siguientes disposiciones de forma general:

- Los nombres de las funciones deben dejar reflejado claramente cuál es la acción que realiza el mismo.
- Se debe apoyar en la utilización de sufijos que ayuden a identificar el resultado final de la ejecución de un método.
- Se debe apoyar en los prefijos para expresar la acción que realiza sobre un elemento determinado.

Variables:

- Los nombres de las variables deben expresar claramente el contenido de la misma.
- Pueden estar referidas en singular o plural.

- Se definen al principio de las estructuras donde son utilizadas.
- En caso de que no se le asigne un valor inicial se deben inicializar con un valor que indique el tipo de dato más general al que debe pertenecer.
- De esta nomenclatura se exceptúan las variables generadas por el Framework.

3.3.2 Tratamiento de Errores

Un aspecto importante a tener en cuenta a la hora de desarrollar un software es el tratamiento de errores, debido que muchos son los errores cometidos por los usuarios, unos por accidentes y otros no tan accidentales. Para garantizar el correcto funcionamiento del sistema es fundamental identificar y controlar los posibles problemas que se pueden presentar a la hora de interactuar con el software.

En el módulo Documentos se gestiona información importante que influye sobre el inventario real de los Depósitos de Aduana y en la toma de decisiones, es por eso que se realizan varios tipos de validaciones de errores.

Las validaciones del negocio se encargan de controlar el flujo de los datos recibidos en el controlador para evitar consecuencias alarmantes. Se llevan a cabo en las diferentes clases con el uso de funciones propias del lenguaje. En caso de que exista algún error, es notificado al usuario para que pueda corregirlos.

Las validaciones realizadas en las vistas juegan un papel primordial en la entrada correcta de los datos a la aplicación. Tienen como función principal evitar que se introduzcan datos incorrectos en los diferentes campos de los formularios para impedir ataques contra el sistema.

Otro de los aspectos importantes utilizado en el tratamiento de errores del sistema desarrollado es la utilización de los formularios generados por Symfony para insertar, eliminar o modificar valores de la base de datos, utilizando cada uno de los validadores implementados de manera que se garantiza una mayor coherencia de los mismos.

3.3.3 Diagrama de Despliegue

El diagrama de despliegue (ver Fig. 3.13) es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo.

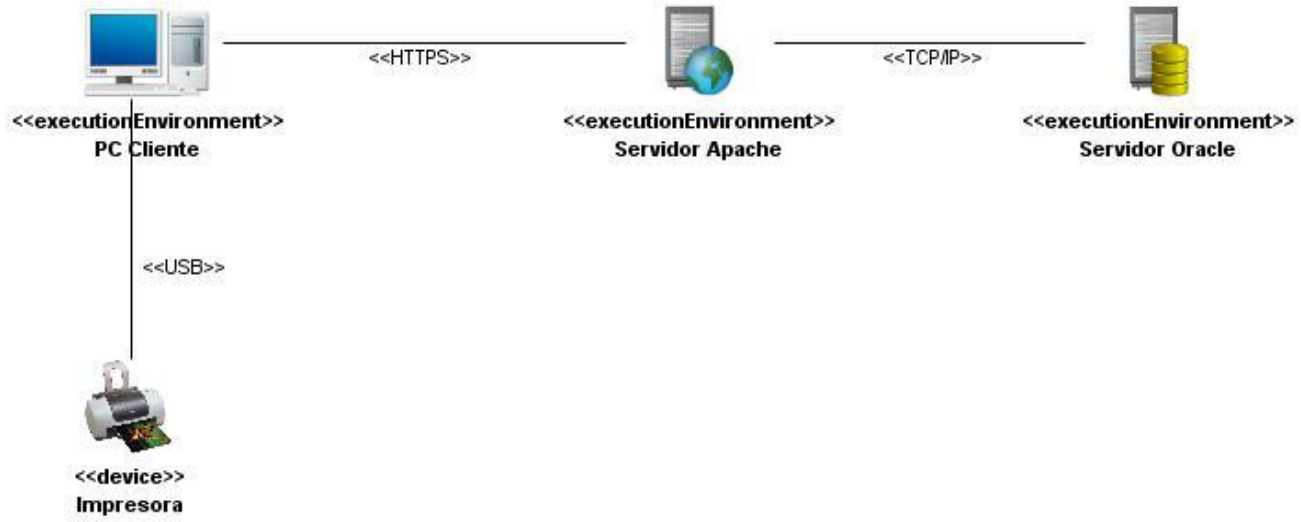


Fig. 3.13 Diagrama de Despliegue

3.3.4 Diagrama de Componentes

En el diagrama que muestra la Fig. 3.14 se presentan los principales componentes que se utilizan en el módulo Documentos. Se accede al sistema por medio del controlador frontal el cual según la petición realizada se comunica con la clase *actions*; esta clase interactúa con las interfaces que utilizan las librerías del Framework ExtJS y que son las encargadas de vincularse directamente con el usuario; también con las clases del negocio creadas por Propel y la clase sfAuxiliarTC que por medio de eventos es capaz de obtener información de los diferentes plugins y subsistemas. Los componentes de configuración que están presentes en todas las acciones realizadas en el módulo.

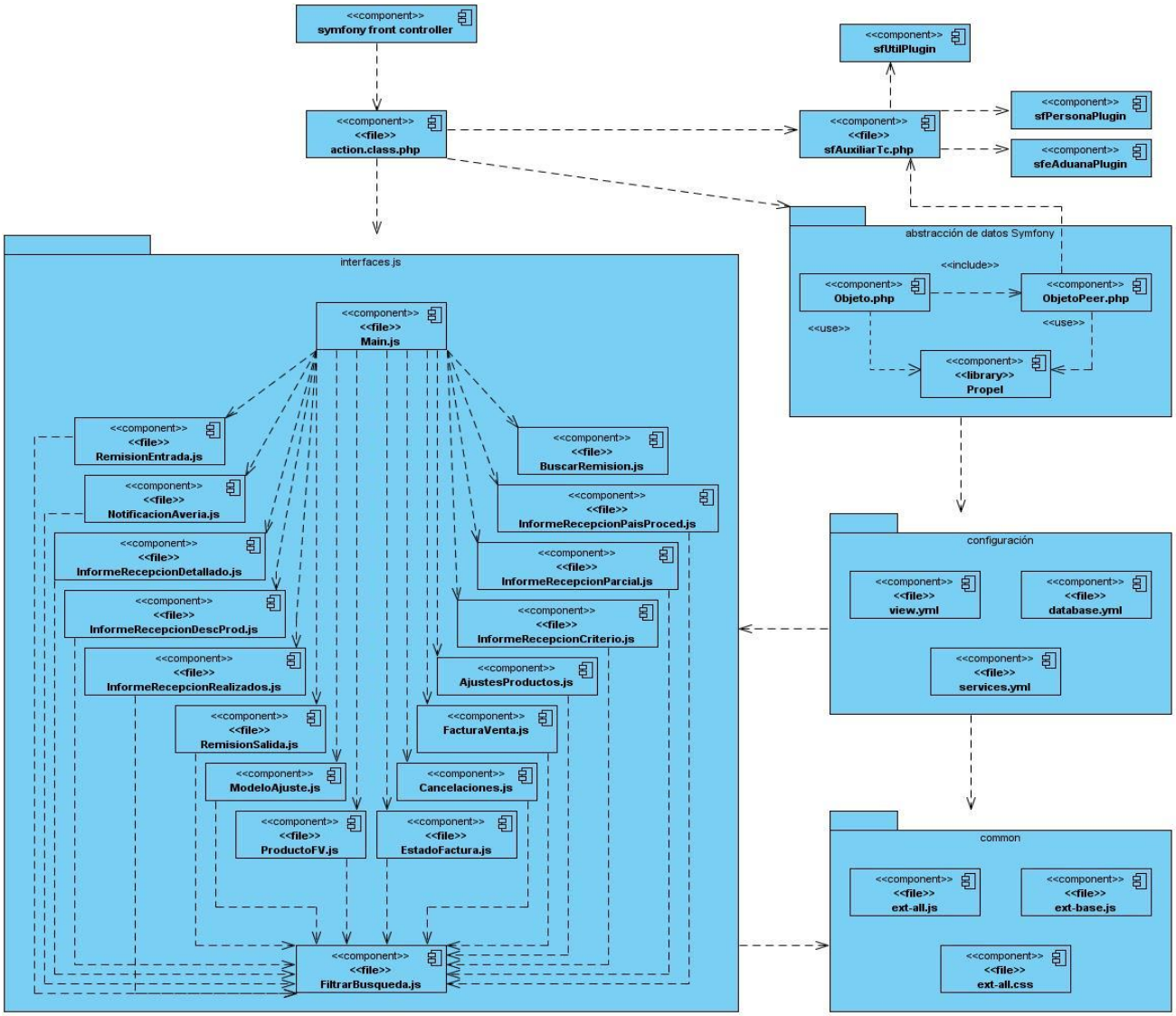


Fig. 3.14 Diagrama de Componentes

3.4 Conclusiones

En este capítulo se presentaron los artefactos más significativos generados durante la etapa de diseño e implementación de la solución, exponiéndose varios diagramas entre ellos: clases del diseño con estereotipos web, secuencia orientado a actividades del negocio, paquetes, clases persistentes, diseño de la base de datos, despliegue y componentes, los resultados de la evaluación de la métrica TOC, la definición del estándar de codificación que será utilizado durante la implementación y el tratamiento de los errores en el sistema.

CONCLUSIONES

Al finalizar este trabajo se lograron cumplir los objetivos planteados de manera satisfactoria, obteniendo como resultado el módulo Documentos para el subsistema Depósitos de Aduana.

El estudio del estado del arte de las soluciones informáticas existentes para la gestión de los Depósitos de Aduana brindó como resultado que ninguno de los sistemas internacionales analizados es factible a utilizar en la Aduana General de la República de Cuba teniendo en cuenta las especificaciones definidas para el sistema Gestión Integral de Aduanas y en el caso del sistema cubano SUA no presenta un módulo o subsistema para la gestión y control de las operaciones que se realizan en los Depósitos de Aduana.

Se determinaron los requerimientos funcionales del módulo a partir de reuniones con los clientes y sobre la base de las características del negocio, y aplicando las técnicas pertinentes se comprobó la validez de la propuesta. Se obtuvieron los artefactos correspondientes al flujo de trabajo de RUP "Análisis y Diseño", establecidos en el Departamento de Soluciones para la Aduana, lo cual se verificó mediante métricas asociadas al software orientado a objetos. Se analizaron los patrones de diseño utilizados en la solución planteada y se diseñaron los diagramas establecidos en el flujo de implementación que dieron paso a la programación de los componentes que forman parte del módulo.

El desarrollo del módulo Documentos aportará eficiencia en la gestión y control de la documentación de los Depósitos de Aduana, contribuyendo notablemente a la toma de decisiones oportuna y acciones correctivas necesarias.

RECOMENDACIONES

Los objetivos propuestos en este trabajo fueron alcanzados satisfactoriamente; sin embargo a lo largo del desarrollo de los mismos surgieron nuevas ideas que serían recomendables tener en cuenta:

- Realizar las pruebas a la solución propuesta con el objetivo de certificar la calidad que presenta.
- Continuar el estudio de los procesos asociados a la gestión de los documentos en los Depósitos de Aduana y el desarrollo del subsistema con el propósito de detectar nuevas necesidades e incorporarle funcionalidades a la solución.
- Extender la búsqueda de nuevas tecnologías informáticas para perfeccionar el módulo Documentos del subsistema Depósitos de Aduana en futuras versiones.

REFERENCIAS BIBLIOGRÁFICAS

1. **Casto Ruz, Fidel. 2008.** Gaceta Oficial No. 18 Ordinaria de 27 de mayo de 1996. *Gaceta Oficial de la República de Cuba.* [Online] 2008. [Cited: Diciembre 5, 2010.] http://www.gacetaoficial.cu/. ISSN 1682-7511.

2. **Aduana General de la República de Cuba.** Resolución No. 33/96 Glosario de Términos. *Sitio Web Sistema de Órganos Aduaneros Aduana General de la República de Cuba.* [Online] 1996. [Cited: Septiembre 10, 2010.] http://www.aduana.co.cu/glosa1.htm.

3. **Gaceta Oficial de la República de Cuba.** Gaceta Oficial No. 18 Ordinaria de 27 de mayo de 1996. *Decreto Ley No. 162.* [Online] 2008. [Cited: Diciembre 5, 2010.] http://www.gacetaoficial.cu/. ISSN 1682-7511.

4. **Comunidad Andina. 2003.** CONVENIO INTERNACIONAL PARA LA SIMPLIFICACIÓN Y ARMONIZACIÓN DE LOS REGÍMENES ADUANEROS. [En línea] 13 de Marzo de 2003. [Citado el: 14 de Diciembre de 2010.] http://www.google.com.cu/url?sa=t&source=web&cd=3&ved=0CCYQFjAC&url=http%3A%2F%2Fintranet.comunidadandina.org%2FDocumentos%2FDInformativos%2FSGdi498.doc&rct=j&q=CONVENIO%20INTERNACIONAL%20PARA%20LA%20SIMPLIFICACI%C3%93N%20Y%20ARMONIZACI%C3%93N%20DE%20LOS%.

5. **Gaceta Oficial de la República de Cuba.** Gaceta Oficial No. 37 Ordinaria de 12 de julio de 2002. *Aduana General de la República. Resolución No. 12/2002.* [En línea] 2008. [Citado el: 15 de Diciembre de 2010.] http://www.gacetaoficial.cu/. ISSN 1682-7511.

6. **IEEE. 1990.** 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology. [En línea] 1990. [Citado el: 23 de Enero de 2011.] http://standards.ieee.org/findstds/standard/610.12-1990.html.

7. **Sommerville, Ian. 2005.** *Ingeniería del Software (séptima edición).* Madrid (España) : Pearson Educación, SA, 2005. 84-7829-074-5.

8. *La ingeniería de requerimientos y su importancia en el desarrollo de proyectos de software.* **Arias Chaves, Michael. 2005.** 10, Universidad de Costa Rica : Revista InterSedes, 2005, Vol. VI. ISSN 1409-4746.

9. **Pressman, Roger S. 2006.** *Ingeniería de Software: Un enfoque práctico.* México DF : MacGraw Hill, 2006. Sexta Edición.

10. **José Escalona, María and Koch, Nora. 2002.** *Ingeniería de Requisitos en Aplicaciones para la Web - Un estudio comparativo.* [PDF] Universidad de Sevilla : Departamento de Lenguajes y Sistemas Informáticos, 2002.

11. *Introducción a la Arquitectura de Software.* **Reynoso Billy, Carlos. 2004.** Buenos Aires : Universidad de Buenos Aires, 2004, Vol. I.

12. *Software Architecture.* **Clements, Paul C. 2002.** Estados Unidos : Software Engineering Institute, 2002. ISBN: 15213-3890.

13. *Uso de Patrones de Arquitectura.* **Veloso Hernández, Pedro. 2009.** s.l. : UMP, 2009.

14. *El patrón de diseño modelo -vista - controlador (MVC) y su implementación el Java Swing.* **Bascón Pantoja, Ernesto. 2004.** 4, 2004, Vol. II.

15. *Introducción a los Patrones de Software.* **Gloria, Cortés and Casallas, Rubby. 2008.** s.l. : Universidad de los Andes, 2008.

16. **Ing. Rodríguez Andrés, Rolando. 2008.** Lenguajes, notaciones y herramientas para el modelado y análisis de procesos. *GestioPolis.* [Online] Junio 16, 2008. [Cited: Marzo 5, 2011.] http://www.gestiopolis.com/administracion-estrategia/lenguajes-notaciones-y-herramientas-en-analisis-de-procesos.htm.

17. **Ferré Grau, Xavier and Sánchez Segura, María Isabel. 2008.** *Desarrollo Orientado a Objetos con UML.* s.l. : Facultad de Informática – UPM, 2008.

18. **2009.** Visual Paradigm for UML. *Sitio de descarga de software.* [En línea] 2009. [Citado el: 22 de Marzo de 2011.] http://www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%29_14720_p/.

19. **Flores, Antonio. 2008.** *Introducción al Lenguaje de Programación PHP.* Madrid : s.n., 2008.

20. **Potencier, Fabien and Zaninotto, François. 2008.** *Symfony 1.2, la guía definitiva.* 2008.

21. **Bullock, Christian. 2002.** Ext JS. *Comunidad en Español.* [Online] 2002. [Cited: Marzo 20, 2011.] http://extjses.com/.

22. **Ing. Manso Martínez, Jenni. 2010.** *Procedimiento para la Ingeniería de Requisitos en el Departamento de Desarrollo de Soluciones para la Aduana del CEIGE.* [PDF] La Habana : Universidad de las Ciencias Informáticas, 2010.

23. **Rumbaugh, James, Jacobson, Ivar and Booch, Grady.** *El Lenguaje Unificado de Modelado. Manual de Referencia.* s.l. : Addison-Wesley.

24. **Departamento de Soluciones para la Aduana, CEIGE. 2011.** *Propuesta de un Estándar de Codiicación.* 2011.

GLOSARIO

Aduana: es la institución de un país encargada de aplicar la legislación aduanera y de recaudar los derechos e impuestos que se aplican a la importación, a la exportación, al movimiento o al almacenaje de mercancías.

Reempaque: se realiza con frecuencia en los Depósitos de Aduana, es la repetición de la acción y efecto de empacar la mercancía.

Consolidación: se realiza con frecuencia en los Depósitos de Aduana, es la acción y efecto de de reunir y unificar la mercancía.

Desconsolidación: es la negación o inversión del significado de consolidación de la mercancía.

AJAX: se trata de varias tecnologías independientes que se unen de formas nuevas y sorprendentes en aplicaciones interactivas para la web. Para ello se utiliza XHTML y CSS (*del inglés, Cascading Style Sheets*) para formatear la información; DOM para interactuar y visualizar dinámicamente la información; se apoya en XML (*del inglés, Extensible Markup Language*) y XSTL para manipular la información mostrada, para la recuperación de datos asincrónica usa el objeto XMLHttpRequest, y JavaScript para actualizar los datos sin necesidad de refrescar la página y para manipular todas estas tecnologías.

Artefacto: pieza de información tangible que es creada, modificada y usada por los trabajadores al realizar actividades.

CASE: *Computer Aided Software Engineering*, en su traducción al español significa Ingeniería de Software Asistida por Computación, es la aplicación de tecnología informática a las actividades, las técnicas y las metodologías propias de desarrollo de software, su objetivo es acelerar el proceso de automatizar o apoyar una o más fases del ciclo de vida del desarrollo de sistemas.

CSS: es un lenguaje de hojas de estilos en cascada, creado para controlar el aspecto o presentación de los documentos electrónicos definidos con HTML y XHTML. Es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Clase: descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, relaciones y semántica.

Declaración de Mercancías: es aquella declaración que se realiza del modo prescrito por la aduana, por la cual las personas interesadas indican qué régimen aduanero pretenden aplicar a las mercancías y suministran los detalles informativos que la Aduana requiere para la aplicación del régimen elegido.

Elicitación: Es el proceso que obtiene todo el conocimiento y la información sobre las operaciones que se realizan en una organización, también es conocido como: proceso de captura de los requisitos.

Diagrama: representación gráfica de un conjunto de elementos. Visualizan un sistema desde diferentes perspectivas.

DOM: el DOM es la estructura de objetos que genera el navegador cuando se carga un documento y se puede alterar mediante JavaScript para cambiar dinámicamente los contenidos y aspecto de la página.

Especificación de requisitos: captura los requerimientos de software para el sistema completo o una porción del mismo.

Framework: un framework, es una estructura de soporte definida mediante la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

JDBC: (*del inglés, Java Database Connectivity*) es una API que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java, independientemente del sistema operativo donde se ejecute o de la base de datos a la cual se accede, utilizando el dialecto SQL del modelo de base de datos que se utilice.

Mercancías: todos los bienes corporales muebles de comercio o no, con la sola excepción de los Efectos Personales de los viajeros.

Metodología: es un proceso de software detallado que define con precisión los artefactos, roles y actividades involucradas.

Régimen Aduanero: tratamiento aplicable a las mercaderías sometidas al control de la aduana, de acuerdo con las leyes y reglamentos aduaneros, según la naturaleza y objetivos de la operación.

RUP: Proceso Unificado del Rational, es un proceso de desarrollo de software, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Scripts: conjunto de comandos u órdenes en un fichero que ordenados producen una salida concreta. El uso habitual de los scripts es realizar diversas tareas como combinar componentes, interactuar con el sistema operativo o con el usuario.

Software: se refiere a los programas y datos almacenados en un ordenador.

Transferencia: régimen aduanero bajo el cual son transportadas las mercancías de un depósito a otro de la misma aduana en territorio nacional bajo control aduanero.

Tránsito: régimen aduanero bajo el cual son transportadas las mercancías de una aduana a otra en territorio nacional bajo control aduanero.

UML: es el Lenguaje de Modelado Unificado para detallar, construir, visualizar y documentar las partes o artefactos de un software.

XHTML: (*en español, Lenguaje de Marcado de Hipertexto Extensible*), es una versión más estricta y limpia de HTML, que nace precisamente con el objetivo de reemplazar a HTML ante su limitación de uso con las cada vez más abundantes herramientas basadas en XML. XHTML extiende HTML 4.0 combinando la sintaxis de HTML, diseñado para mostrar datos, con la de XML, diseñado para describir los datos.

XML: es la abreviatura de lenguaje de marcas extensible, está diseñado para estructurar, transportar y almacenar información. XML no es un reemplazo de HTML sino un complemento a este.

ORM: (*del inglés, Object Relational Mapping*) Mapeo de Objetos Relacional es una técnica de programación para convertir tipos de datos incompatibles entre sistemas de bases de datos y lenguajes orientados a objetos.